Linear transformations of semantic spaces for word-sense discrimination and collocation compositionality grading

Alfredo Maldonado Guerra

Doctor of Philosophy
The University of Dublin, Trinity College

2015

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work. I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Alfredo Maldonado Guerra

Abstract

Latent Semantic Analysis (LSA) and Word Space are two semantic models derived from the vector space model of distributional semantics that have been used successfully in word-sense disambiguation and discrimination. LSA can represent word types and word tokens in context by means of a single matrix factorised by Singular Value Decomposition (SVD). Word Space is able to represent types via word vectors and tokens through two separate kinds of context vectors: direct vectors that count first-order word co-occurrence and indirect vectors that capture second-order co-occurrence. Word Space objects are optionally reduced by SVD. Whilst being regarded as related, little has been discussed about the specific relationship between Word Space and LSA or the benefits of one model over the other, especially with regard to their capability of representing word tokens. This thesis aims to address this both theoretically and empirically.

Within the theoretical focus, the definitions of Word Space and LSA as presented in the literature are studied. A formalisation of these two semantic models is presented and their theoretical properties and relationships are discussed. A fundamental insight from this theoretical analysis is that indirect (second-order) vectors can be computed from direct (first-order) vectors through a linear transformation involving a matrix of word vectors (a word matrix), an operation that can itself be seen as a method of dimensionality reduction alternative to SVD. Another finding is that in their unreduced form, LSA vectors and the Word Space direct (first-order) context vectors define approximately the same objects and their difference can be exactly calculated. It is also found that the SVD spaces produced by LSA and the Word Space word vectors are also similar and their difference, which can also be precisely calculated, ultimately stems from the original difference between unreduced LSA vectors and Word Space direct vectors. It is also observed that the indirect "second-order" method of token representation from Word Space is also available to LSA, in a version of the representation that has remained largely unexplored. And given the analysis of the SVD spaces produced by both models, it is hypothesised that, when exploited in comparable ways, Word Space and LSA should perform similarly in actual word-sense disambiguation and discrimination experiments.

In the empirical focus, performance comparisons between different configurations of LSA and Word Space are conducted in actual word-sense disambiguation and discrimination experiments. It is found that some indirect configurations of LSA and Word Space do indeed perform similarly, but other LSA and Word Space indirect configurations as well as their direct representations perform more differently. So, whilst the two models define approximately the same spaces, their differences are large enough to impact performance. Word Space's simpler, unreduced direct (first-order) context vectors are found to offer the best overall trade off between accuracy and computational expense. Another empirical exercise involves comparisons of geometric properties of Word Space's two token vector representations aimed at testing their similarity and predicting their performance in means-based word-sense disambiguation and discrimination experiments. It is found that they are not geometrically similar and that sense vectors computed from direct vectors are more spread than those computed from indirect vectors. Word-sense disambiguation and discrimination experiments performed on these vectors largely reflect the geometric comparisons as the more spread direct vectors perform better than indirect vectors in supervised disambiguation experiments, although in unsupervised discrimination experiments, no clear winner emerges. The role of the Word Space word matrix as a dimensionality reduction operator is also explored. Instead of simply truncating the word matrix, a method in which dimensions representing statistically associated word pairs are summed and merged, called word matrix consolidation, is proposed. The method achieves modest but promising results comparable to SVD. Finally, the word vectors from Word Space are tested empirically in a task designed to grade (measure) the compositionality (or degree of "literalness") of multi-word expressions (MWEs). Cosine similarity measures are taken between a word vector representing the full MWE, and word vectors representing each of its individual member words in order to measure the deviation in co-occurrence distribution between the MWE and its individual members. It is found that this deviation in co-occurrence distributions does correlate with human compositionality judgements of MWEs.

Acknowledgements

The research presented in this thesis was supported by Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Trinity College Dublin. Some calculations were performed on the Lonsdale cluster maintained by the Trinity Centre for High Performance Computing. This cluster was also funded through grants from Science Foundation Ireland.

I would like to express my sincere thanks to my supervisor Dr Martin Emms, not just for his hands-on involvement in this research and for his support throughout the course of my studies, but for teaching me a whole new way of deep analytical thinking without which this thesis would have not taken shape. In fact, some of the most important contributions included herein, such as the linear transformation formulation in Chapter 4 and the difference between the R_1 and R_2 SVD projections in Chapter 3, are based on original ideas by him. I only followed up on his big good ideas with lots of little good ideas. Credit and gratitude are also owed to the examiners of this thesis, Dr Saturnino Luz and Dr Anna Korhonen, whose feedback and advice strengthened this work significantly.

I would also like to thank Dr Carl Vogel for his continued help and support during my studies as well as for initially having me admitted to the Ph.D. programme in Trinity. Similarly, I would like to thank Elda Quiroga from Tecnológico de Monterrey and Masaki Itagaki from Microsoft for their support in the preceding stages of my Ph.D. This thesis is in many ways the product of their guidance and support.

Many thanks also go to the other Ph.D. students and post-docs for the deep technical discussions and their spirit of camaraderie: Liliana, Héctor, Gerard, Martin, Erwan, Derek, Anne, Roman, Stephan, Francesca, Oscar, Baoli, Nikiforos and Ielka, as well as to the "new generation": Akira, Grace, Kevin, Carmen, Arun and Shane. I would also like to thank the DU Archaeological Society for providing me with a space on campus for intellectual discussions that did not involve computers but dusty old bones, and in particular I wish to thank Mary, Ciarán, Jenny, Deirdre, Pablo, Karl, Aoife, Sean, Alice, Michael, Alex, Victoria and John Tighe for their friendship. Thank you guys, I had a blast!

Heel veel dank aan Wynzen de Vries, for his patience, encouragement and support during my studies and for his understanding when the writing of this thesis soaked up most of my time. *Finalmente, me gustaría agradecer a mis padres*, Beatriz Guerra Treviño and Alfredo Maldonado Osorno for all their care, education and support during the first 20-something years of my life.

Contents

Typographical conventions

- **Bold** will be used for important linguistic and technical terms when defined. E.g. a **word token** is a specific instance of a particular word in a text.
- *Italics* will be used for lexical linguistic units such as words, multi-word expressions, ngrams, lexemes, morphemes, etc. when mentioned in the exposition of the text rather than when they are used in a linguistic example (unless the term is italicised for emphasis in the linguistic example). E.g. In the example, *bank* is used in its 'financial institution' sense.
- SMALL CAPS are used for lexemes (word roots) and as features in componential analysis examples and features of vector examples. E.g. *Tired* is a word form of the lexeme TIRE. The dimensions of vector **w** are [MONEY, ACCOUNT, RIVER, WATER].
- Text surrounded in "double quotes" will usually be direct quotations or linguistic examples embedded in the running text.
- Text surrounded in 'single quotes' will usually be senses or meanings of words. They will also be used as glosses for non-English language words. E.g. In the example, *bank* is used in its 'financial institution' sense. The correct Spanish term is *vino tinto* 'red wine' rather than the more literal **vino rojo*.
- Ungrammatical and semantically anomalous phrases and examples will be preceded by a star. E.g. "*He kicked the pail", "*Colourless green ideas sleep furiously", **vino rojo* (lit. 'red wine').
- Linguistic variables found in text will be represented by lower-case Greek letters. For example, word type variables will be commonly represented by the letters τ (tau) and υ (upsilon), word token variables will be represented by letters such as κ (kappa) and λ (lambda) and sense variables (i.e. a variable holding the sense or "semantic value" of a word) by σ (sigma). Greek letters are not exclusively reserved for linguistic variables however, and so in a few instances Greek letters will be employed as numerical variables in conventional formulae (e.g. the usage of λ as a variable denoting an eigenvalue of a matrix).
- All vectors are represented with bold lower-case letters like **a** or **w** or words like **bank**. Normally, these vectors will represent word types in the Latent Semantic Analysis and

Word Space approaches. Some of these word type vectors will be used in function form, taking a variable as an argument: e.g. $\mathbf{w}(\tau)$ is the word (type) vector of type τ.

- A context vector will be represented by a bold lower-case letter $\mathbf{c}$, with its order indicated as a superscript. For example, a direct (first-order) context vector will be represented by $\mathbf{c}^1$ and an indirect (second-order) context vector by $\mathbf{c}^2$. In some cases, the matrix from which a context vector is derived will be used as its superscript instead: e.g. $\mathbf{c}^{\mathbf{H}}$ is an indirect context vector derived from matrix $\mathbf{H}$. When discussing the context vector of a particular token κ, the vector will be used in function form: e.g. $\mathbf{c}^1(\kappa)$ will be the direct (first-order) context vector of a specific token κ in a corpus or document and $\mathbf{c}^{\mathbf{H}}(\kappa)$ will be the indirect context vector, of κ, derived from matrix $\mathbf{H}$.

- Sense vectors are represented by a lower-case Gothic $\mathfrak{s}$ and a superscript indicating their order. E.g.: $\mathfrak{s}^1$, $\mathfrak{s}^2$

- All matrices are represented with bold upper-case letters. For example, word matrices will be represented as $\mathbf{W}$ whereas matrices consisting of context vectors will be represented by $\mathbf{C}$.

- Non-bold, subscripted, lower-case letters are used as scalars belonging to the vectors or matrices introduced above. For example, c_i would be the i-th element in a context vector $\mathbf{c}^1(\kappa)$ and w_{ij} would be the number of times the i-th word co-occurs with the j-th word in word matrix $\mathbf{W}$. An alternative index notation to refer to the i-th element in a vector will be $[\mathbf{c}^1(\kappa)]_i$.

- Sets are represented as upper case letters in calligraphic script, e.g.: $\mathcal{D}$. The cardinality of a set is indicated by surrounding the set in bars: $|\mathcal{D}|$. A scalar expression surrounded in bars indicates the absolute value of the scalar expression: $|x_i - \mu_i|$.

1 Introduction

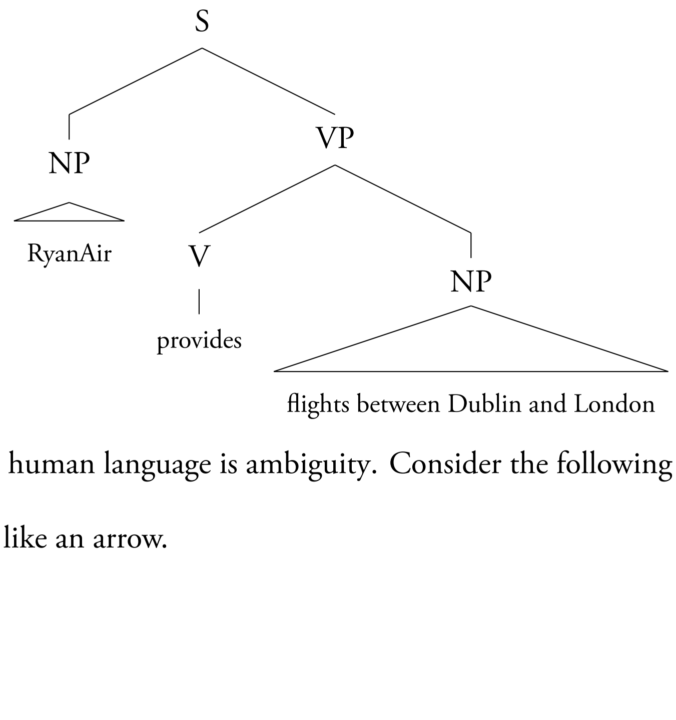
1.1 Motivation

As most aspects of human life settle into an interconnected on-line world, the amount of textual information generated each day accumulates at an unprecedented speed. As a consequence, new computer tools are created daily in order to access such information (search engines), to channel the information into coherent categories (blogs, Twitter, news readers, email classifiers), to make the information available to people around the world (machine translation), etc. Whilst all of these tools might look very different from each other, they all handle the same type of data: human language manifesting in text form.

From a structuralist perspective (de Saussure, 1916), human language can be seen as a communication system based on *signs* consisting of a *signifier* (sounds, letters, words) that represent a *signified* (meaning, concept) and on *rules* (syntactic, semantic, etc.) that dictate how those signs are combined to form meaningful utterances. For example, in (1.1.1) below, the signifier *flight* signifies a journey made through air by means of aeroplanes while *RyanAir* refers to a commercial airline and *Dublin* and *London* to two European cities:

(1.1.1) RyanAir provides flights between Dublin and London.

A syntactic rule tells us that the verb *to provide* takes one noun phrase as a subject and another noun phrase as a direct object, and therefore an English speaker will parse this sentence in this manner:

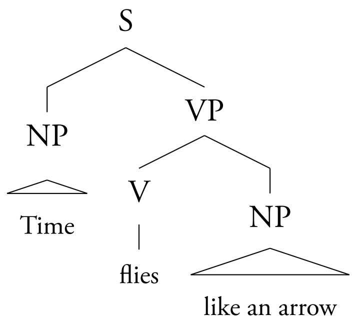


(1.1.2)

A problem with human language is ambiguity. Consider the following example:

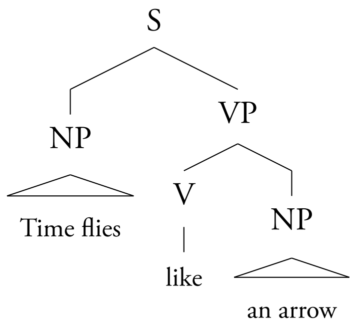
(1.1.3) Time flies like an arrow.

The signifier *flies* is a verb that normally means moving through space. However, most English speakers will interpret the sentence as a figure of speech, a simile, relating the speed of the passage of time with the speed in which an arrow flies (i.e. very fast). This figurative reading of the sentence produces parse tree (1.1.4):



(1.1.4)

However, an alternative though improbable reading, interpreting *like* as the verb meaning to enjoy or to take pleasure from, forces the whole sentence to state that there is a particular type of fly, called "time flies", that are fond of a particular arrow. The parse tree for this interpretation is (1.1.5):



(1.1.5)

While contrived, this example illustrates that language can present ambiguities at different levels. In this case, the ability to perceive a figure of speech or not can cause two distinct parse structures (syntactic ambiguity), each assigning a different part of speech (morpho-syntactic ambiguity) and meaning (lexical semantic ambiguity) to the elements *flies* and *like*. The fact that the interpretation depicted by (1.1.5) is easily perceived as forced or improbable, illustrates that (human) language users are good at dealing with language ambiguity. If emerging software tools are to succeed in helping people make sense of the vast information on the Internet, they will also have to deal with these ambiguities. It is this type of lexical semantic ambiguity that this thesis is concerned with.

As a specific example of what is meant by lexical semantic ambiguity, consider the word *rock*. It could mean a 'natural solid consisting of one or more minerals', a 'genre of popular music usually involving the use of electric guitars' or it could even refer to other less frequent concepts, such as the 'stick-shaped sweet usually sold in sea-side resorts in the UK and Ireland', or even people names like Rock Hudson. This potential semantic ambiguity in words might not be a characteristic immediately acknowledged, but it is pervasive in the lexicon. For example, it has been estimated that any of the 121 most frequent nouns in English have 7.8 meanings on average (Ng and Beng Lee, 1996).

When confronted with this fact, most people agree that usually the context of a word will help in disambiguating its meaning. And in fact that is what people do in every day language since words do not occur in isolation, but in specific conversational or textual situations. So when dealing with lexical semantic ambiguity computationally, the attention usually shifts to the role of context as well. As an example of the power of context to disambiguate words, consider the two book titles in (1.1.6) and (1.1.7) below, illustrating how the correct sense of *rock* can be easily determined by taking into account the other words in the title (i.e. the context to the ambiguous word *rock*):

(1.1.6) *Rock* fractures in geological processes

(1.1.7) *Rock* and popular music in Ireland: before and after U2

In fact the problem of ambiguity at the lexical level was identified early in the first efforts of machine translation in the 1940s and 1950s. Weaver (1955) recognised in those early days that translating a text word by word was difficult without taking into account the context in which those words occurred in since it is impossible to tell the intended meaning of each word in isolation. But if the translation of each word was decided based on the other words surrounding it (its context), then it should be possible to determine their meaning with more certainty. This insight led to the research programme known today as **word-sense disambiguation (WSD)** (Ide and Véronis, 1998; Agirre and Edmonds, 2007, pp. 4-7; Navigli, 2009), which is the computational task of determining the sense of a given word as it occurs in a given context. Normally, the sense assigned to the word in context comes from a pre-defined list of possible senses for that word. A successful early example is the seminal Lesk Algorithm (Lesk, 1986), which disambiguates an instance of polysemous word in context by measuring the overlap (the number of words in common) between its context (the words surrounding the polysemous word) and the words used in each of the definitions for every sense of that word in an electronic dictionary. The sense that maximises such overlap is selected as the meaning of the word instance. The computerisation of dictionaries and the curation of electronic lexical resources such as WordNet (Miller, 1995; Fellbaum, 1998) in the 1980s and 1990s led to a surge in WSD research that continues to this day. Today, WSD is treated as a corpus-based classification problem: given an occurrence of a polysemous word in text and a list of senses for that word, classify the occurrence at hand based on its context. For this classification to take place, a machine learning classifier is usually trained on a corpus that feature the polysemous word(s) that have been already manually disambiguated. Because the required list of word senses has to be manually curated and because a training corpus has to be manually sense-tagged, it is said that this task is **supervised**. The major criticism about traditional word-sense disambiguation is that it is time consuming and expensive to both maintain the list of word senses and to manually tag high volumes of text. This is often termed the "information bottleneck". Besides, it is not easy (or even possible!) to list every sense for a word as that word may modulate or adapt its sense depending on the nature, domain or register of the corpus (Kilgarriff, 1997). Sometimes it is easy to distinguish between the senses of a polysemous word, but phenomena such as regular polysemy and metaphorical extension make the task of listing every single potential sense of a word a virtually impossible task. This has led to the development of **unsupervised** methods such as **word-sense discrimination** (**WSDisc**) or **word-sense induction** (**WSI**)[1].

Word-sense discrimination (Schütze, 1998) is a relaxation of the original word-sense disambiguation task that does not rely on a pre-defined list of word senses is available and instead seeks to automatically induce the senses of a word by clustering the contexts in which it appears. The assumption is that each of such clusters represents a different sense of the word in question. The clusters are then used to classify new instances, as done in WSD. It is considered an unsupervised approach because of its lack of a sense list and because there is no need to manually tag any training corpus. However, it is very common during experimentation to use a manually sense-tagged test corpus for evaluation purposes. The terms *word-sense induction* and *word-sense discrimination* are often used interchangeably. However, one could argue that the initial clustering step in word-sense discrimination could be described as a word-sense induction phase. This thesis takes this view and so will consider word-sense discrimination as a two-step process: a sense induction step in which context clustering takes place and a discrimination proper step in which individual contexts are assigned to the cluster deemed more appropriate by some classification algorithm.

There are also hybrid, semi-supervised approaches to WSD. One group of such approaches use an unsupervised algorithm to learn senses from untagged text based on a sense inventory. For example, in Buitelaar et al. (2001) co-occurrence statistics in untagged text of words that are represented in a WordNet-like taxonomy are computed first. Then, the occurrence of an ambiguous word is tagged with the WordNet sense that maximises a score computed by traversing the taxonomy of each candidate sense and by contrasting each transversal's co-occurrence statistic average with the co-occurrence statistics of the ambiguous word and the words co-occurring with it. Another group of semi-supervised approaches are those that automatically annotate senses in untagged corpora based on a small set of sense-tagged examples. Yarowsky (1995) describes an example of these bootstrapping methods.

As we have seen, words convey meaning; a meaning that varies as a function of context. As words combine with other words in that context to form larger constructions such as phrases and sentences, a "larger" meaning emerges from such a construction. Language is a productive communication system as it allows the production and understanding of new combinations of units (such as words) that have never been experienced by language users (Lyons, 1981, Sec. 1.5). And whilst this productivity is not random but governed by the rules of grammar, language users are free to creatively produce utterances in a potentially

[1]A note on abbreviations: When discussing one of these word-sense-related tasks, the task full name or its abbreviation will be used (WSD, WSDisc, WSI). However, if a discussion applies to all three tasks or if the specific task is not that relevant for the present discussion, the collective abbreviation WSX will be used.

infinite number of ways without the need of ever breaking free from the constraints posed by these rules (Chomsky, 1972, p. 100). Despite the ability that human beings have to use language productively and creatively, there is strong evidence to suggest that language users often recycle prefabricated chunks of word combinations (Cowie, 1988; Sinclair, 1991) which Firth (1957) calls **collocations**. Such collocations range from idiomatic expressions like *saved by the bell* and *kick the bucket* to multi-word terminological units like *weapons of mass destruction* and *operating system*, passing through clichés and simple recurring phrases like a *at this moment in time*, *all that jazz*, *boys will be boys*, *at the speed of light*, *opposites attract*, *moving forward*, etc.

There is a category of collocations that, in addition to showing frequent recurrence, seem to form a self-contained and opaque semantic unit. For example, the idiomatic expression *to throw in the towel* cannot be correctly interpreted as 'to give up' without recurring to knowledge of boxing, from where the expression originated. These collocations are called **non-compositional** since they cannot be interpreted using the standard mechanisms of composition of language, that is, by following the traditional rules of syntax and semantics without recurring to knowledge outside the usual senses of each constituent word. Non-compositional collocations pose a significant problem in many natural language processing problems such as machine translation, where we do want to avoid providing literal translations to idiomatic expressions. This issue poses another layer of lexical ambiguity to consider. Notice however, that context can also help in disambiguating whether a collocation is compositional or not. When used in its metaphorical, idiomatic sense, *to throw in the towel* will be likely surrounded by words and phrases suggesting competition and surrender, and will be unlikely surrounded by words suggesting the actual physical hurling of absorbent cloths to the floor.

Whilst the study of word senses and collocations is of great interest to lexical semantics and linguistics in general, they have practical implications for different natural language processing tasks. As already mentioned, WSD was first identified and defined within the context of machine translation. Indeed, explicit WSD is necessary in traditional, rule-based machine translation at least from a theoretical point of view. Many actual rule-based machine translation systems however do not include an explicit WSD module, although the most successful commercial rule-based system, Systran, does include a WSD module (Resnik, 2007). Statistical machine translation systems generally do not include such a WSD module. Since these systems face a lexical choice comparable to the one faced by WSD components, it is often said that they perform their own implicit WSD as the source language context provides clues as to the correct translation of ambiguous words (Och, 2002, p. 53). Nevertheless, there have been efforts to integrate explicit WSD modules in statistical machine translation. Carpuat and Wu (2007) is a successful example that showed a consistent improvement of phrase-based statistical machine translation from Chinese to English by disambiguating ambiguous source words and phrases by assigning them their corresponding target translation directly, instead of assigning their dictionary or WordNet sense to such ambiguous word and phrases.

Besides machine translation, WSD is being applied to many NLP tasks like information extraction, named-entity classification, co-reference determination, acronym expansion, among others (Agirre and Edmonds, 2005, p. 11).

Unsupervised WSDisc methods have also been applied to NLP tasks such as information retrieval. For instance, Véronis (2004) developed a system that produced graphs of word co-occurrences from text corpora that was able to discriminate the linguistic usages (senses) of highly ambiguous search queries, whilst Navigli and Crisafulli (2010) performed clustering of the results returned by the Yahoo! web search engine based on the induced senses of the words contained in the search result snippets.

WSDisc and compositionality measurement of MWEs can be used in lexicography and terminography (terminology management). For example, WSDisc can be used in a concordancing system that attempts to induce the senses of the word being searched and clusters the concordances based on these word senses and presents them as distinct groups to the lexicographer or terminologist. A MWE compositionality measuring tool could help a terminologist determine whether a multi-word term can be split into smaller sub terms or not (presumably a low compositional multi-word term will be more atomic and will not tend to have sub terms). A low compositionality grade on a multi-word term can also indicate translators not to translate such a term literally and to instead look for a suitable equivalent idiomatic expression in their target language.

1.2 Operationalising context computationally

Lexical ambiguity can manifest in at least two ways: in the way that words can acquire different senses depending on use and in the way that some words will form partnerships that have non-transparent or idiomatic meanings. Context provides language users with clues as to how to resolve such ambiguities. But if we are to use context to help computers resolve lexical ambiguities we need a way to operationalise our intuitions computationally.

First, let us take a closer look the notion of context. At one level, context can be understood as the situation in which a linguistic expression is uttered. This notion of **situational context** is emphasised by Malinowski (1935) by stating that language has to be researched in its context of situation in anthropological studies. He even argues that language plays an active role in human behaviour. A notion that is further refined by Austin (1962) with his distinction between performative and declarative sentences, a notion that later developed into the speech act theory of what eventually became linguistic pragmatics (Lyons, 1981, Sec. 5.6; Geeraerts, 2010, Sec. 4.2.3). Similarly, Wittgenstein (1968) explains that a word's meaning is determined by its use, an idea that brings us to a more immediate and linguistic characterisation of context. Given a particular word or expression of interest, its **syntagmatic context** is formed by the words and expressions that surround it in an instance of its usage (sec. 2.2.3, p. 46). When dealing with text corpora, often this is the only type of context readily available. Even without access to a situational context, syntagmatic context alone can be quite informative with regard to lexical meaning. For example, Firth (1957) observed that much of the meaning of the word *cows* can be deduced from the words it collocates with in expressions such as "they are milking the cows" or "cows give milk". The words co-occurring with a word of interest are indicative of its properties. This intuition can be extended to help in the resolution of word-sense ambiguities. For example, contexts using the word *bank* in its financial sense would likely involve words such as *account*, *balance*, *money*, *deposit*, *steal*, etc. whereas contexts using the same word in its 'edge of body of water' sense will likely include words such as *river*, *sand*, *water*, *canal*, *fish*, and so on (Stubbs, 2002, p. 15). Each sense of the word *bank* are likely to produce mutually exclusive (or nearly mutually exclusive) sets of context words. The theoretical underpinnings of this intuition is Harris' (1954) **distributional hypothesis**, which states that words occurring in similar contexts will tend to have similar meanings. The assumption is that a difference in distribution entails a difference in meaning.

This is the basic intuition behind most methods of word-sense disambiguation and discrimination. In fact, Lesk's algorithm directly operationalises the distributional hypothesis. As we shall see in Section 3.2 (p. 67) this intuition can be formalised and eventually expanded into what is now known as the **vector space model** (**VSM**) (Salton, 1971; Salton et al., 1975) used in information retrieval and eventually adopted in distributional lexical semantics. The VSM encodes a context of an ambiguous word as an occurrence vector, i.e. a multidimensional array in which each dimension represents a word in some pre-defined vocabulary and keeps count of the number of times that each word occurs in the context. The core component of the VSM is a word-document or word-context matrix **A**. The rows of such a matrix represent **word types** and the columns **documents** or shorter text **segments** such as paragraphs or sentences. An element a_{ij} in this matrix, in its most basic form, effectively counts occurrences of a word type τ_i in a document or segment (context) δ_j.

Notice that these representations only take into account the words present (or absent) in a context to estimate semantic similarity or relatedness. They do not take into account other information such as syntax (word order). Because of this feature, these representations, which are at the core of the VSM, are usually called **bag of words** models. Despite this simplicity, the VSM has been quite successful in its native application domain, information retrieval, specifically in the task of ranking the documents in a text collection according to their relevance to a user query. This property of finding documents relevant to a user query has been interpreted to be a proxy for semantic similarity. That is, if the words used in a document are similar in meaning to the words in a user query, and the VSM considers the document to be relevant to the query, then perhaps the same VSM-based relevance metric can be applied to natural language processing problems where measuring semantic similarity between words is the solution or part of the solution. For example, Landauer and Dumais (1997), Turney (2001) and Rapp (2003) tested different variants of the VSM in the synonym section of the Test of English as a Foreign Language (TOEFL). In this section of the test, candidates are asked to select the correct synonym from a list of candidate synonyms. The VSM-based systems developed by these researchers obtained scores between 65% to 93%, similar to those achieved by skilled L2 English speakers and native speakers of English. Also, a VSM system developed by Turney (2006) was tested on the multiple-choice analogy section of the SAT exam and achieved a score of 56%, similar to what human test-takers achieve who on average score 57%.

The link between the VSM and lexical semantics is so strong that, some variants of the VSM, such as LSA, are often seen as plausible models of a major component of human lexical semantic learning (Bullinaria and Levy, 2007; Landauer, 2007). But VSMs are also applied in language tasks beyond semantics. For example, within information retrieval itself, VSMs find applications beyond the traditional ranking of documents based on queries. An extension of this basic application is cross-lingual information retrieval, in which documents written in one language are returned for queries written in another language (Dumais et al., 1997). Another information retrieval area where VSMs have been applied to are text categorisation (Sebastiani, 2001) and document clustering (Manning et al., 2008), important tasks in organising large text collections. Automatic essay grading, document segmentation by subtopics, question answering and call routing are some of the other natural language processing tasks where the VSM has found application (Turney and Pantel, 2010).

Given its success in lexical semantics and other areas of natural language processing, it is not surprising to find that many, if not most, WSX methods are based on a variant of the VSM (Agirre and Stevenson, 2007). This thesis focuses on the VSM in part because of this success and in part because it is virtually language independent as it requires very little linguistic pre-processing, making it available to languages for which computational language resources (such as parsers or part-of-speech taggers) are not available or underdeveloped. This language neutrality property can, in addition, help shed some light on how far we can get with as little resources as possible and by only deriving statistics from the text itself. In particular, this thesis focuses on two instances of the VSM: Latent Semantic Analysis (Deerwester et al., 1990; Landauer and Dumais, 1997) and Word Space (Schütze, 1992, 1998).

Latent Semantic Analysis (**LSA**) is a direct adaptation of Salton's VSM, which uses **A** to represent word types and text segments. The characteristic that distinguishes LSA from Salton's VSM is the usage of **Singular Value Decomposition** (**SVD**) (Golub and Van Loan, 1989; Berry, 1992) in order to project the word type and segment vectors in **A** to a lower dimensionality space. Levin et al. (2006) applied LSA to WSD with a view to applying it to WSDisc at a future stage. They used the text segments containing occurrences of the ambiguous word as proxies for token representations of that ambiguous word.

Word Space constructs a **word-word co-occurrence matrix** (or **word matrix**), denoted here as **W**, in which an element w_{ij} counts the instances of word type τ_i occurring in the vicinity of word type τ_j. The rows and columns in **W** are **word vectors** that represent **word types** and are used to compute indirect **word token** representations called **second-order context vectors**. Given a target word token in context, a second-order context vector is computed by aggregating (by summing or averaging) the word vectors corresponding to the words co-occurring with the target word in a particular context. Before computing second-order context vectors, Word Space optionally reduces the dimensionality of these word vectors via SVD in order to obtain reduced context vectors. Strictly speaking, context vectors computed from SVD-reduced word vectors do not count second-order co-occurrence *per se* since they are aggregating vectors in a different (orthogonal) vector space that do not hold word frequencies as such any more. Also, Kontostathis and Pottenger (2006) demonstrated that SVD-reduced spaces potentially capture co-occurrence information of an order higher than second-order. Because of this, we shall prefer the term **indirect context vectors** when discussing second-order context vectors in general. There is also a **first-order context vector** variant (Pedersen and Bruce, 1997), which represents an instance of a target word in context by counting the words that co-occur directly with the target word within the word window. To contrast them with their indirect (second-order) counterparts, we shall refer to them as **direct context vectors**. While not traditionally included within the Word Space framework, direct context vectors are related to word vectors and in consequence to indirect context vectors: the word vector of a word type τ can be computed by summing the direct context vectors for every token κ of τ in the corpus (see Def. 3.5.3, p. 101; Maldonado-Guerra and Emms, 2012). For our purposes, we shall consider Word Space to have three inter-related components: a word matrix of word vectors representing types and two token representations: direct (first-order) and indirect (second-order) context vectors. Contrast this three-component system with LSA's matrix **A** which is able to simultaneously represent word types and tokens in the same matrix.

This thesis is concerned primarily with the competing token representations given by Word Space and LSA. In spite of the attention given by the WSX literature to these two specific representations, very few works have attempted to compare one with the other. And those few works that attempt this comparison usually do not fully analyse their properties or the exact mathematical relationship between the two models. For example, Pedersen (2010) compares the performance of variants of LSA and Word Space in WSDisc experiments, but he assumes a straightforward relationship between LSA and Word Space which he does not test (see Section 4.3, p. 122). Sahlgren (2006), Utsumi and Suzuki (2006) and Utsumi (2010) also performed empirical comparisons between LSA and Word Space based on identifying semantic relations (such as synonymy, hyponymy, collocation, etc.) but again did not perform an analytical comparison between the spaces represented by both models. This thesis aims to fill this gap in a systematic manner by first analysing the mathematical relationships between Word Space and LSA analytically and then by performing WSX benchmarks in order to attempt to establish which model is best suited for WSX.

Whilst the main focus of the thesis is the study of the spaces produced by LSA and Word Space and their impact on the disambiguation and discrimination of word senses, the thesis also explores, as a secondary topic, the problem of automatically assessing the compositionality of multi-word expressions (see Section 2.1.2, p. 33 and Chapter 8). This topic is included in this thesis as the method presented to perform this assessment is based on Word Space.

The specific line of enquiry is made more explicit in the research questions that shape the structure of this thesis in Section 1.3.

1.3 Research questions and thesis structure

The topic of research for this thesis centres on vector space representations useful for resolving lexical ambiguities in an unsupervised way. The primary vector space representation studied is Schütze's Word Space, with LSA as a secondary model to compare it against. Both models represent the same objects (types and tokens) in their own way. Since both models are based on word counts, use SVD but construct different matrices, it seems reasonable to ask whether the two models are equivalent, approximate or completely different. Little has been discussed about their exact relationship or the benefits of one model over the other for word senses. This is one of the questions that this thesis seeks to address. Also, Word Space itself has two separate token representations (direct or first-order context vectors and indirect or second-order context vectors), so another valid question to pursue is whether they are similar or different, and whether they perform similarly or differently in WSX. This leads to the following research questions:

1. What is the relationship between type representations in LSA's matrix **A** and Word Space matrix **W**?
2. What is the relationship between the token vectors defined in LSA's **A** and Word Space first-order context vectors and second-order context vectors?
3. Are the SVD-projected type and token representations in LSA and Word Space equivalent, similar or different?
4. For WSX experiments, which token representations are better? LSA-based or Word Space-based representations?
5. What is the relationship between the two token representations (direct/first-order and indirect/second-order) offered by Word Space? Are they both approximations of each other or are they different?
6. Which Word Space token representation is better for WSX?

During the analysis of Word Space token representations (Chapter 4), it is observed that second-order context vectors can be computed from first-order context vectors via a matrix multiplication by the word matrix. It is shown that this operation can be regarded as a linear transformation or projection not unlike the projections usually performed via SVD as a means of dimensionality reduction. As a consequence, dimensionality reductions based on this linear transformation are explored. In order to be used as a dimensionality-reducing linear map, $\mathbf{W} \in \mathbb{R}^{m \times n}$ must meet $n < m$ (see Sec. 4.1, p. 108). That is, $\mathbf{W}$ must map vectors in $\mathbb{R}^m$ to $\mathbb{R}^n$, with $\mathbb{R}^n$ being a lower dimensional (and potentially denser) vector space. This means that the features selected for the "output" dimensionality $\mathbb{R}^n$ must be different to the "input" dimensionality $\mathbb{R}^m$. The strategy to produce this "output" dimensionality presented in this thesis is a novel one, and is termed **word matrix consolidation**. In word matrix consolidation, the columns of a word matrix are merged and summed together (or consolidated) based on syntagmatic similarity. An algorithm is proposed to perform these consolidations and several criteria for consolidation are explored. This brings us to the next research questions:

7. Is word matrix consolidation a viable method of dimensionality reduction?
8. Do indirect (second-order) context vectors computed from consolidated matrices perform better than from non-consolidated matrices in WSX experiments? That is, does the consolidation process produce better performing vectors?
9. Is the word matrix consolidation of dimensionality reduction better, the same or worse than SVD for WSX?

The questions presented so far explore only one type of lexical ambiguity: word sense ambiguity. The other type of lexical ambiguity explored by this thesis is the compositionality of multi-word expressions such as collocations. The distributional hypothesis implies that the syntagmatic distribution of semantically unrelated words will be different (i.e. words with unrelated meanings will co-occur with different words) whereas the syntagmatic distribution of semantically related words will be similar (i.e. words with related meanings will co-occur with more or less the same words). A non-compositional collocation such as *red tape* in its bureaucratic sense will co-occur with words relating to government, bureaucracy, politics, etc., whereas the predominant senses of the individual words *red* and *tape* will perhaps have little to do with government, bureaucracy or politics. This intuition is operationalised via the computation of cosines between Word Space word vectors computed from instances of the collocation and word vectors computed from instances of the collocation's individual words. A low cosine value is interpreted to indicate non- or low-compositionality whereas as higher values as medium- or even high-compositionality. This brings us to the last research question of the thesis:

10. Can cosine measures between collocation vectors and individual word vectors be used as a reliable and unsupervised method of collocation compositionality measure?

The overall thesis structure and the main outcomes can be outlined as follows.

Chapter 1 is this introductory chapter.

Chapter 2 introduces basic concepts of linguistic theories that will be used throughout the thesis. The concepts of word, type, token, collocation and word sense are discussed in detail. The chapter is drawn against a structuralist/neo-structuralist backdrop from which most of the intuitions regarding the distributional hypothesis of lexical semantics emerge. It presents the methodological lexicographic concepts of Onomasiology and Semasiology, which roughly correspond to the Supervised and Unsupervised approaches in word-sense disambiguation and discrimination. It also presents the two main types of lexical semantics relations between words: syntagmatic relations and paradigmatic relations.

Chapter 3 fleshes out some of the concepts studied in Chapter 2 in computational terms. For example, it defines word types and word tokens in terms of the Word Space and LSA vector models. The chapter serves mostly as a background and literature review for the comparisons and experiments that will be presented in subsequent chapters. The chapter also makes linguistic interpretations from the syntagmatic and paradigmatic point of views of methods of semantic similarity computation via cosine. Some aspects of this chapter were published in Maldonado-Guerra and Emms (2012) and Emms and Maldonado-Guerra (2013).

Chapter 4 is in some ways a continuation of Chapter 3 (e.g. it describes how SVD can be applied to Word Space objects) but also makes the main theoretical contributions of the thesis, the most fundamental of which is an equivalent formulation of Word Space indirect (second-order) context vectors as the product of a linear transformation of direct (first-order) context vectors via the word matrix (question 5). The case is made to view indirect (second-order) context vectors as projections in an alternative vector space that could also be used for dimensionality reduction in a way reminiscent of the application of SVD to project vectors in lower dimensionality spaces. This linear transformation view also makes the application of SVD to Word Space objects more transparent and comparable to the way it is applied in LSA. The chapter also makes analytical and numerical comparisons between Word Space and LSA (questions 1, 2, 3) and finds that the token and type representations offered by both models, as normally formulated in the literature, are approximations of each other and that their difference can be exactly calculated. This chapter also shows that the same mechanism of indirect context vector construction available to Word Space is also available to LSA, a mechanism that has not been widely exploited or studied. Some aspects of this chapter were published in Maldonado-Guerra and Emms (2012).

Chapter 5 looks closer at the relationship between the two token representations offered by Word Space: direct (first-order) and indirect (second-order) context vectors (questions 5, 6). This is done via geometric and WSX experiments. The geometric experiments attempt to measure the rough similarity of both vector representations and predict their performance in WSX experiments by looking into two geometrical properties: parallelism and angular spread. In **parallelism**, direct and indirect sense vectors (computed by taking the centroid of context vectors of a target word for a particular sense) are compared by means of cosine measures. If the cosine measures are high, it is assumed that the two sense vectors are approximations of each other (i.e. roughly "parallel") whereas if they are low then the two sense vectors are not approximations of each other (i.e. not "parallel"). It is found that direct and indirect sense vectors are not parallel or approximate to each other. So they are indeed different types of vectors. The **angular spread** is the measure of how far apart, according to the cosine measure, sense vectors of a given order (first or second) for different senses of a target word are from each other on average. A low cosine average is interpreted to mean that the sense vectors are far apart from each other and therefore a classification or clustering algorithm based on centroids (like Rocchio, K-Means, Expectation-Maximisation for Gaussian mixtures, etc.) will be able to discriminate senses of a word more easily and therefore achieve good performance. Conversely, it is assumed that a high cosine average will make it more difficult for a classification or clustering algorithm to distinguish among the sense of a particular word token, translating in poorer performance. The angular spread, therefore can be a performance predictor for each kind of token vector. In these experiments, pairwise cosine comparisons between direct sense vectors of a target word for different senses are computed and the resulting cosine measures are averaged. This number is taken to be the angular spread for direct context vectors for that particular target word. The same exercise is repeated for indirect sense vectors, and the resulting average of pairwise cosine measures is taken to be the angular spread for the indirect context vectors. It is found that direct context vectors have a lower cosine average (are more spread) and that indirect context vectors have a higher cosine average (are less spread), so the prediction is that direct context vectors will perform better than indirect context vectors. After the geometric experiment results are analysed, supervised WSD and unsupervised WSDisc experiments are performed. For the supervised WSD experiments, the direct context vectors comfortably outperform the indirect context vectors, largely reflecting the predictions from the angular spread experiments. However, for the unsupervised WSDisc experiments, no clear winner emerges, with the difference in performance between both vector types coming much closer together. However, it has to be stressed that the overall performance scores of the unsupervised experiments are much lower (between 12% and 25% lower) than the supervised experiments, which is congruent with other results in the literature highlighting that unsupervised methods generally perform worse than supervised methods (Pedersen, 2007). The experimental results in this chapter were published in Maldonado-Guerra and Emms (2012).

Chapter 6 reports the results of an empirical comparison between LSA direct and indirect token vectors and Word Space direct and indirect context vectors in supervised WSD and unsupervised WSDisc experiments. Chapter 4 establishes that the LSA-based and the Word Space-based token representations are approximations of each other. The experiments reported in this chapter seek to demonstrate whether they also perform similarly in WSX settings or whether one performs better than the other (question 4). The experiments in this chapter are largely based on the supervised and unsupervised experiments performed in Chapter 5. However, the comparison between LSA-based and Word Space-based token representations is slightly more complicated since Word Space is capable of producing two indirect vector representations (based in $\mathbf{C}$ and $\mathbf{W}$), whereas LSA can only produce one indirect vector representation (based on $\mathbf{A}$). In addition, experiments were performed using unreduced versions of context vectors as well as SVD-reduced versions of context vectors. For the SVD-reduced versions, experiments were conducted at different truncation levels (e.g. at 50, 100, 150, ..., 1000 dimensions). Overall, it was found that indirect Word Space vectors based on $\mathbf{C}$ do perform similarly to indirect LSA vectors based on $\mathbf{A}$. However, only one variant of SVD-reduced (called "R_2 projections") direct Word Space vectors based on **C** only perform similarly to the same variant of SVD-reduction to direct LSA vectors based on **A** in supervised experiments, whilst all other dimensionalities (R_1 projections and unreduced) and supervised experiments perform differently across the two kinds of direct token vectors. Also, indirect Word Space token representations based on **W** perform differently to both indirect Word Space token representations based on **C** and indirect LSA vectors based on **A**. So, in sum, despite the approximations found in Chapter 4, not all LSA vector configurations will perform similarly to their Word Space counterparts in all circumstances. It can be seen that these configurations somewhat offset the performance that they will achieve in supervised or unsupervised experiments. Other interesting observations made by this chapter include that for most configurations SVD-reduction does not help performance, and that the best trade-off between computational expense and performance in the supervised case is given by unreduced direct Word Space context vectors, and in the unsupervised case by unreduced indirect Word Space context vectors, although other unreduced context vectors perform similarly, a result somewhat echoing Chapter 5, where no clear winner was found for the unsupervised experiments.

Chapter 7 goes back to the definition of indirect context vectors as a transformation of direct context vectors via a word matrix introduced in Chapter 4 and attempts to use this insight as an alternative method of dimensionality reduction. The specific method used is word matrix consolidation (question 7). Matrix consolidation is compared with unreduced and SVD-reduced direct and indirect Word Space vectors in supervised and unsupervised experiments (questions 8, 9). Whilst word matrix consolidation can perform similarly and sometimes slightly better to SVD, it still does not provide an increase in performance when compared to unreduced vectors. An advantage over SVD though is that matrix consolidation does not require the user to provide the number of dimensions to keep in advance. While not being an overwhelmingly better alternative to dimensionality reduction, matrix consolidation does show that viewing the indirect context vector construction method as a linear transformation opens the door to the exploration of novel methods of token representation.

Chapter 8 reports the adaptation of the Word Space word matrix for measuring the compositionality of multi-word expressions (MWEs) in a fully unsupervised manner (question 10). The method developed works on bigrams only (i.e. 2-word MWEs) and assumes that one of the words in the bigram is the node (or headword) and the other one is the collocate (or modifier) (Geeraerts, 2010, p. 170). It directly compares the semantic similarity between the node and the full MWE and between the collocate and the full MWE by computing a cosine similarity score between the word vectors representing the node, the collocate and the full MWE. A system implementing three variations of this method participated in the Distributional Semantics and Compositionality (DISCO 2011) shared task (Biemann and Giesbrecht, 2011). Systems were given a set of bigrams that had different degrees of compositionality. Their compositionality or non-compositionality was assessed by native speakers based on “how literal” each bigram sounded to them. These judgements were then averaged and converted to a scale between 0 and 100, where 0 means non-compositional and 100 is fully compositional. Systems were evaluated on their ability to predict this average numeric score and to classify bigrams as either low compositional, medium compositional or highly compositional. The shared task organisers refer to this categorisation as the “coarse” prediction task. One of the variations of the system reported in this thesis achieved a fifth place (out of 15) in the overall numerical prediction task, whereas another of the variations also achieved fifth place in the overall coarse prediction task. However, it achieved first place in the coarse verb-object subset of bigrams. The system description and the experimental results were published in Maldonado-Guerra and Emms (2011).

Chapter 9 presents the main concluding points of the thesis and discusses areas for future work.

The main contributions of the thesis can be summarised as follows. SVD is usually taken as a blackbox/cookbook method that is supposed to have certain semantic properties that are rarely explained or justified. There is some veil of mysticism as to its inner workings with researchers keeping a distance with the mathematical definitions and implications of the method. This thesis attempts to lift this veil and make SVD more transparent by showing that LSA and Word Space have some clear relationships. Researchers have intuitively acknowledged a link between the two models, with some making simplistic assumptions in respect to this link. This is the first time these assumptions are tested and we show that whilst there is indeed a clear link between the two models, their relationship is not as straightforward as one might initially think. Based on these observations, this thesis constructs several variants of both models and tests them empirically in WSD and WSDisc experiments, showing in general that SVD in either model does not tend to help very much in these tasks. In addition, the experimental results reflect the non-trivial relationship between the models, indicating that success, or lack thereof, in one model is not necessarily indicative of the other model’s performance. However, given that the relationship between the two models is clearly known, it is possible to use a system implementing one model in order to make representations in the other model by modifying the values appropriately in the system’s matrices.

2 Linguistic Background

This chapter introduces and defines several aspects of linguistic theory used or referred to throughout this thesis. In doing so, relevant literature on the topic is also reviewed or referred to. The interpretation of linguistic terminology often depends on the linguistic school or branch one chooses to use as a reference. This chapter discusses several of these terms, sometimes considering differing philosophical views, and defines the terms as they pertain to the scope of this thesis. The aim of this process is to reduce ambiguity and increase the precision of the interpretation of the different linguistic terms and concepts employed. Since the thesis subject matter is word-sense disambiguation and discrimination, it seems appropriate to focus the discussion with what we understand by the concepts of *word* and *word sense*.

The **word** can be considered to be the smallest linguistic unit that carries meaning. It is from the way in which words combine in larger linguistic units, such as phrases or sentences, that the larger unit's overall meaning emerge. Given its importance, most empirical research in word-sense disambiguation and discrimination use some computational representation of words as features in their systems (Navigli, 2009). Section 2.1 attempts to define the concept of word mostly from a written language perspective, since the experimental work reported in this thesis is done on written language corpora. It also defines the linguistic concept of *collocations*, recurring multi-word partnerships that often carry a special or non-literal meaning. This section also introduces the concept of *ngrams*, which have resulted convenient and practical from a computational point of view even if they do not necessarily emerge from any theoretical linguistic tradition.

If we are ready to admit that a word can carry meaning, we need to understand how it is that words come to mean something. Do words have their own natural, true meaning or do language users arbitrarily assign meanings to words? How does context help a language user determine the meaning of a word? Section 2.2 introduces some essential concepts from lexical semantics that address these questions. The approach taken by this section is from a structuralist/neo-structuralist point of view. The specific sequence of theories exhibited in this section culminates with the distributional hypothesis of lexical semantics which is operationalised in Chapter 3 and is exploited in all the models described in the thesis. However, this sequence of lexical semantics theories is far from exhaustive. There are many other theories not covered here simply because they do not directly feed into the distributional hypothesis or because they are not amenable to be operationalised computationally.

2.1 What is a word?

2.1.1 Word tokens and word types

When dealing with written text, it is convenient to conceive words as clusters of contiguous characters bound by white space or punctuation marks, such as full stops, commas, quotation marks, etc. From a computational point of view, this definition is quite useful as it enables the employment of simple regular expressions for identifying words automatically. However, this definition is somewhat problematic. For example, many Asian languages such as Chinese and Japanese do not use white space as a word boundary. Instead all words appear contiguously in the text making it difficult to determine where a word starts and where a word ends without knowledge of the language in question. Another problem is that many languages, including English, also employ the full stop to indicate an abbreviation. Different tokenisation solutions have been proposed and their discussion is beyond the focus of this thesis, so the interested reader is directed to Manning and Schütze (1999, sec. 4.2.2) for a comprehensive discussion.

Assuming that there is a mechanism to correctly split a string of text into separate words, we come to the question of what to count as a word. Consider this famous quote by Samuel Johnson:

(2.1.1) When a man is tired of London, he is tired of life, for there is in London all that life can afford.

In total there are 22 words, of which 17 are distinct words. In lexicography, a **word token** is a specific instance of a particular word in a text, whilst a **word type** can be seen as the class to which a word token belongs to. In Example (2.1.1), the word type *is* has 3 word tokens and the word types *tired*, *of* and *London* have 2 word tokens each. We are starting to see that the term *word* is slightly imprecise as it could be used to mean word type or word token, depending on context. And as we shall see in Section 3.5, this distinction will be very important in the mathematical definitions of context vectors (word tokens) and word vectors (word types).

A further way to conceive a word is by considering its canonical or "dictionary" form. For instance, it is possible to remark that this quote is an example of the usage of the words *be* and *tire* even if they do not appear in their canonical form at all, but in their inflected forms *is* and *tired*, respectively. More specifically, we say that the words *is* and *tired* are **word forms** or **realisations** of the **lexemes** BE and TIRE, respectively. This shows again that there are some more precise terms for different conceptions of the term *word* and this thesis will prefer to use these when this precision is important but will freely use the less technical term *word* as an umbrella term covering word forms and lexemes or when this precise distinction is not needed.

Based on their meaning, words can be classified into two major groups: **content words** (also known as **open-class words**, **full words** or **lexical words**) and **function words** (also called **closed-class words**, **stop words**, **grammatical words**, **empty words** or **form words**). Content words are considered to carry most of the meaning (semantic content) in any given sentence or phrase. Verbs, nouns, adverbs and adjectives are usually considered to be content words. They are so-called open class because new words in these part of speech categories are regularly added as new inventions, discoveries and definitions are made. *London*, *afford* and *man* are examples of content words in (2.1.1). By contrast, function words are words like articles, prepositions and pronouns, to which new members are rarely added and whose purpose is mostly to interact with content words and other function words in order to form grammatical and meaningful phrases and sentences. Function words are often perceived to carry little semantic content. In (2.1.1) *of*, *in* and *that* are examples of function words. In natural language systems (notably in information retrieval), function words are specifically called **stop words** and are usually filtered out or ignored during processing due to the belief in their low semantic currency and in order to make savings on technical aspects such as storage requirements, and in order to reduce potential noise. In fact, there are some content words that sometimes are included in stop word lists, such as forms of the verb be (*is*, *are*, *was*, etc.) and auxiliaries[1] (*can*, *do*, *would*, etc.) As a consequence, this thesis will prefer the term **stop word** to refer to those words usually included in a typical information retrieval system's stop word list and the term **content word** for any word not belonging to this list.

2.1.2 Multi-word expressions and collocations

The concept of *word* can be further expanded if we consider so-called **multi-word expressions** or **units**, i.e. recurring sequences of individual words that form a unit, such as *prime minister*, *public house* or *Republic of Ireland*. These word combinations form a whole unit and yet each member in the combination can be seen as unit in itself: *prime* and *minister*, *public* and *house*, *Republic*, *of* and *Ireland*. The notion of the usage of recurring multi-word expressions is termed **collocation** in the linguistics literature (Firth, 1951; Halliday, 1961; McIntosh, 1966; Lyons, 1966; Sinclair, 1966). There is a wide range of collocation types. Idioms such as *once in a blue moon*, *red tape* or *red herring* are the ones that more readily come to mind. But other types of collocations include figurative expressions and clichés (*talking to a brick wall*, *the crack of dawn*, *strike a balance, safe and sound, law and order, to be honest*), multi-word terminology (*collateral damage*, *support vector machine, machine readable passport*), named entities (*United Nations*, *European Economic Area*, *Barack Obama*), phrasal verbs and complex verbal expressions (*look for*, *make up someone's mind*), etc. Unfortunately there is no single definition for the concept of collocation and there tends to be a slightly different treatment of the term between computational linguistics and general linguistics. For example, many natural language processing works often assume collocations are contiguous strings of words,

[1]Notice that the set of verbs that are considered auxiliaries in English is quite small and rarely new members are added to it. Based on this, they should perhaps be considered function words and not content words. However, since verbs are considered content words then, for consistency, auxiliaries should also be considered content words by their virtue of being verbs. This issue shows that these terms are contentious and so a practical definition, rather than a theoretical definition, is usually adopted.

whereas linguistic works will also investigate collocations that are not consecutive but whose components co-occur within a short distance between each other (see e.g. Sinclair, 1991). For instance, *make* and *remark* can collocate in phrases such as *make a remark, make an inappropriate remark, make an unhelpful and distasteful remark*, etc. Apart from the requirement of frequent and near co-occurrence, Manning and Schütze (1999, p. 184) enumerate three other typical requirements or characteristics of collocations[2]:

- **Non-substitutability** – A word making up a collocation often cannot be substituted by another word, not even by a similar word or even a synonym. For example, in *once in a blue moon* the adjective *blue* cannot be substituted by a similar word like *green* or *red*, as the idiom is fixed and cannot be changed. Similarly, translating literally *red wine* into Spanish would render the anomalous **vino rojo* which, while being an accurate description of the wine's colour, is not the usual *vino tinto* that any native speaker would instantly recognise. The non-substitutability characteristic is not an absolute requirement however, as some collocations do allow it in a limited manner. For example, *heavy smoker* can take the form *chain smoker* and *to know one's onions* is equivalent to *to know one's stuff.*

- **Non-modifiability** – Collocations cannot be modified by adding words or by changing grammatical characteristics. For example *to get a frog in one's throat* cannot become **to get an ugly frog in one's throat.* This is especially important in idioms such as *kick the bucket.* English speakers would find anomalous the sentence **she was about to kick her bucket*, even if there is nothing ungrammatical about the substitution of a definite article with a possessive determiner. The non-modifiability criterion is also optional, however, as some collocations do allow a limited degree of modification. For example *seeing is believing* can become *seeing really is believing.*

- **Non-compositionality** – A collocation is not necessarily the "sum of its parts", that is, it cannot always be interpreted as a literal composition of its constituents. The meaning of *x kicked the bucket* cannot be understood from a standard open-choice analysis as this would arrive at interpreting that *x* kicked an actual bucket, rather than interpreting that *x* died. Similarly, *red tape* in its government bureaucracy sense require the language user to have knowledge additional to the language's grammar in order to correctly interpret the phrase. The property of non-compositionality is not confined to idiomatic expressions: *public house* (from which the word *pub* is derived) which usually designates an establishment that provides patrons with alcoholic drinks to be consumed on the premises, is somewhat non-compositional. And again, not all collocations are non-compositional: *orange juice, apple juice* and many technical terms (*machine readable passport*, *internet service provider*) are rather compositional common collocations.

[2]Some examples here are borrowed from Singleton (2000)

Not all these three characteristics are required for every single collocation and perhaps we should consider as a collocation any recurrent word combination that fulfils at least one of these three characteristics. From a lexicographical perspective, Benson (1989) puts forward a similar argument even if he does not employ the same terminology. He considers that frequent word combinations should be listed as lexical entries in a dictionary when they fulfil at least one of the following two criteria: 1) have "peculiar semantics" and 2) be arbitrary. A word combination has peculiar semantics when its meaning cannot be understood by literally interpreting the combination, i.e. when it is non-compositional. If a word combination can be interpreted literally, then it should not be listed as a lexical entry, i.e. it should not be considered a collocation. For example, the following adjectives should not be considered collocates of *evidence*: *absolute, additional, certain, considerable, conspicuous, gratifying, hopeful*, even if they frequently modify this noun. This is because they are compositional, i.e. they form normal and literal "adjective plus noun" combinations that any (competent) speaker of English would understand by just applying this grammar rule without recurring to any external knowledge. The inclusion of such compositional combinations will rarely if ever be consulted by the dictionary's potential users, making the dictionary bigger and not necessarily more useful. By contrast, a combination with peculiar semantics such as *heavy smoker*, which of course does not literally describe a smoker who is overweight or simply physically heavier than the average smoker, would be more useful, for example to an L2 learner of English, to be listed in a dictionary as an entry. The criterion of arbitrariness is a combination of the non-substitutability and non-modifiability characteristics. For example, Benson considers that *make an estimate* should be a dictionary entry because it is not a free word combination as it is not possible to say **make an estimation*. Similarly, he considers *commit treason* (not **commit treachery*) and *warmest greetings* (not **hot greetings*) to also be arbitrary non-free word combinations. Benson's arguments are grounded in good lexicographic practice: the ultimate decision on what constitutes a lexical entry in a dictionary is based on a trade-off between its potential usefulness for a dictionary user and the cost of making the dictionary bigger by adding that entry. It is more of an engineering principle than a linguistic insight, even if the requirements he lists are based on interesting linguistic properties. A similar pragmatic view is usually also taken in computational linguistics and so the study on non-compositional multi-word expressions is increasing in the field. As an example of the importance of processing multi-word expressions, consider a machine translation engine. It would be desirable for such an engine to be able to translate the French *prendre une décision* as *make a decision* and not literally as **take a decision* while there is no reason to stop it from translating *preuve supplémentaire* literally as *additional evidence*.

It has recently been found that it is difficult to categorically distinguish every collocation in a binary, mutually exclusive manner as either compositional or non-compositional, and that instead shades or degrees of compositionality should be considered for a given collocation (Bannard et al., 2003; Katz and Giesbrecht, 2006). For example, while it is clear that *red herring* is completely non-compositional and *orange juice* is very much compositional, *heavy* *smoker* lies somewhere in between: it does describe a type of smoker, one who smokes too much but not an overweight one as we have seen. So, we could say that *heavy smoker* has some sort of medium-to-low compositionality but it is by no means fully non-compositional. *Red light* is a collocation that, depending on meaning or use, will have a different degree of compositionality. Consider these three sentences:

(2.1.2) Many LEDs emit red light.

(2.1.3) The car stopped at a red light.

(2.1.4) Disappointingly, the project got the red light.

It is clear that (2.1.2) is clearly compositional since it literally describes light that is red in colour. Whilst (2.1.3) describes a scenario in which a light that is red in colour is involved, it can be argued that *red light* has acquired the more abstract meaning of a traffic control device signalling the car to stop. This usage involves some literal usage (the signal is in the form of a light that is red in colour) but it also carries other meanings associated to traffic conventions as well as electronic devices designed to enforce these conventions in actual streets. Since (2.1.4) involves decoding a metaphor by exploiting additional knowledge (i.e. a notion of traffic conventions mapped into a business situation), this usage of *red light* exhibits very low compositionality.

The present discussion on collocations has touched informally on some aspects of semantics, especially when describing non-compositionality. This informal introduction to semantic notions should suffice for the purposes of this discussion. However, Section 2.2 addresses in general the topic of lexical semantics more rigorously within a structuralist and distributionalist setting and, in particular, Section 2.2.5 revisits some aspects of the semantic notions of collocations discussed here, against the same structuralist and distributionalist backdrop.

2.1.3 Ngrams

Lexical units as we have discussed so far can be singleton words or multi-word expressions like collocations. In the computational linguistics literature, a sequence of characters with no space in between is usually termed a **unigram**, whilst sequences that include one space in between are termed **bigrams**, two spaces, **trigrams**, etc. The numeric prefix attached to the *-gram* suffix keeps count of the number of contiguous non-space characters members (unigram members) present. They are collectively called **ngrams** or **n-grams** (Manning and Schütze, 1999, p. 193). Notice that these definitions completely disregard whether or not a unigram is itself composed of two identifiable words like in the cases of *teapot* or *website*. Also, it is important to mention that it is somewhat flawed to define lexical units in terms of the written form given that written language is only one manifestation of spoken language and there is evidence, for example, that young children as well as non-literate people are able to identify and isolate single words in speech (Singleton, 2000, ch. 1). This implies that human intuitions as to what a word is are not a consequence of literacy and therefore we need not base our definitions in terms of the written form. However, it can be difficult to operationalise such human intuitions in a natural language processing system that deals primarily or exclusively with text and so a theoretical compromise must be reached. While crude, these definitions are useful for computational operationalisation and because they are divorced of any particular linguistic or philosophical consideration of what a word or a lexical unit is, they can work inside more complex linguistic concepts. For example, certain ngrams are collocations (e.g. *public house*, *machine readable passport*). This thesis will therefore make extensive use of the terms *unigram*, *bigram*, *trigram*, *ngram*, etc. as defined so far. Furthermore, it extends the concepts of word token and word types to ngrams. That is, it is possible to consider a particular instance of an ngram a token, and that ngram a lexicographic type in its own right.

2.2 What is a word sense?

At the core of any discussion about word senses resides the question of how is it that a word carries meaning. The link between linguistic form and meaning has been discussed since antiquity. Perhaps its oldest incarnation is Plato's Cratylus dialogue between Hermogenes, Cratylus and Socrates[3]. This dialogue discusses whether there is an inherent correctness in the names given to things or whether names are given to things by convention. Cratylus believes the names things receive are natural and therefore come from the things themselves. Socrates explains that the sounds (phonemes) used in a word should be appropriate to the thing they are the subject of. Hermogenes opposes this view stating that the naming of things follows custom and convention, that words are not related in any way to the essence of the things they name. Socrates rejects this idea explaining that words have one true meaning, even if sometimes the meaning of a word changes. Socrates cites a few etymological examples but also struggles to find the origin of some words, blaming the difficulty on what we today would call the processes of language change: "Saying, if there is a word we do not know about, that it is of foreign origin. Now this may be true of some of them, and also on account of the lapse of time it may be impossible to find out about the earliest words; for since words get twisted in all sorts of ways, it would not be in the least wonderful if the ancient Greek word should be identical with the modern foreign one." (Plat. Crat. 421e-d). The discussion is not conclusive but the idea that words have a true and natural meaning was extremely influential until the end of the 18th Century, when serious objective research gave birth to the discipline of historical linguistics. Up until that point, etymologies were constructed based on whatever ideas of word purity were in vogue at any given time. By contrast, based on objective formal and semantic analysis of cognates, i.e. versions of semantically-related words in different languages, historical linguistics was able to reconstruct the historical relationships between languages and the reconstruction of theorised ancient languages such as Proto-Germanic or Proto-Indo-

[3]An English translation by Fowler (1921) is freely available on-line at http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.01.0172:text=Crat.

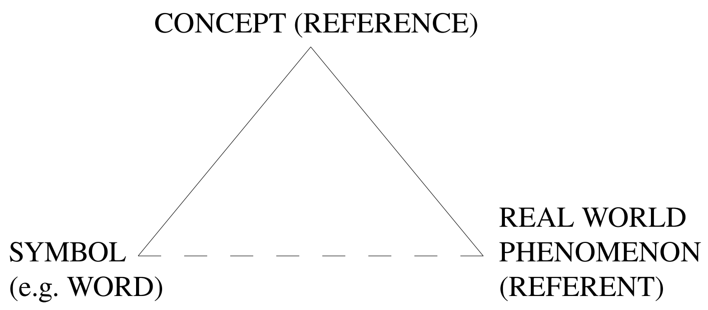


Figure 2.2.1: The semiotic triangle – A mental concept can be seen as a pointer to a phenomenon in the real world and a linguistic symbol, in turn, as a pointer to the mental concept, producing an indirect association between the linguistic symbol and the real world phenomenon, represented here as a dotted line.

European (Geeraerts, 2010, ch. 1).

By the early 20th Century, this shift towards an analytical framework in linguistics began favouring Hermogenes' idea of meaning by custom and convention against Socrates' ideals of true meaning. An influential conceptualisation of this notion is the denotational or referential mechanism between words and real world entities. Through their famous semiotic triangle, adapted in Figure 2.2.1, Ogden and Richards (1936, p. 11) posit that a linguistic symbol like a word or phrase is linked to a concept (a reference) which itself is some mental representation of a real world phenomenon (a referent) such as an object, an animal, a place, an action, an abstract entity, etc. The only connection between the symbol and the actual real world phenomenon is indirect, via its mental representation (the concept)[4].

This model presents several shortcomings. First, it does not account for situations in which more than one word or expression is associated with the same meaning (synonymy) as well as cases where one word or expression is associated with more than one meaning (homonymy or polysemy). Furthermore, not all words are easily associated to real world phenomena. Function words like *and*, *if* and *that* as well as expressions relating to mythological entities (*tooth fairy*, *unicorn*, etc.) clearly illustrate this issue. These words do not have a real world referent, however we can agree that they all have a meaning (in the case of the function words this would be some sort of grammatical, functional meaning). Likewise, there could be one single real world phenomenon having several meanings, as in the commonly cited case of the *morning star* and the *evening star*: both relate to the physical planet Venus, but the two stars are considered to have different meaning because they appear in the sky at different times of the day.

[4]Notice, however that in some cases such as that of onomatopoeic words like *cuckoo*, *meow* and *crash*, there is a clear connection between the form and the sound it refers to. This is the exception rather than the rule however, for the majority of words in a given language are non-onomatopoeic, making no obvious connection between form and referent.

This is not to say that the semiotic triangle model is wrong and we should take Socrates' position. It simply is incomplete. A more modern incarnation of this model is studied under the heading of a more general property of human language: its arbitrariness. This property considers that the connection between linguistic form and meaning is arbitrary as it is not possible to determine the meaning of a form from the form alone as well as it is impossible to predict the form that a meaning will take (Lyons, 1981, p. 19).

2.2.1 Structuralist lexical semantics

Structuralism in linguistics originated with Ferdinand de Saussure's work at the turn of the 20th Century. One of its central ideas is that language is a symbolic system that has properties and principles governing how the linguistic sign functions as a sign. It considers that as a system, language is rather autonomous from psychological processes. That is, even if language is evidence of human cognition, it is regular enough that it can be studied on its own, independently of any considerations to human cognition.

In structuralist semantics, Ferdinand de Saussure recognises that there is indeed a relationship between a lexical unit (*signifiant*) and meaning (*signifié*) as in the referential/denotational model illustrated by the semiotic triangle. However, this relationship is downplayed, and in the case of the American Bloomfieldan school of structuralism, this referential/denotational aspect of meaning is rejected completely (Singleton, 2000, p. 67). The reason why the referential/denotational aspect of meaning is downplayed or rejected is that it depicts meaning as a psychological concept or a mental representation, something that lies outside the linguistic system, something that cannot be observed directly. As mentioned previously, the goal of structuralism is to study language in its own right, based on its own principles and rules. So, anything that is considered to be outside the linguistic system cannot be studied by structuralist linguistics. If we are to study semantics from a structuralist linguistics perspective, we need to bring it into the linguistic structure. De Saussure attempts to do this by stressing that there is a relationship between a lexical unit and other structures in the language and that this has to be considered as part of the meaning of the lexical unit. He uses a monetary analogy comparing coins with words: just as the value of a coin is determined both by the goods it can buy as well as by its relationship with with coins and banknotes of other denominations in the same monetary system, the (semantic) value of a word is determined by the meanings it can be exchanged for as well as by its relationships with other words and phrases in the linguistic system (de Saussure, 1916, part II, ch. 4).

For example, a €5 banknote can be exchanged for certain goods or services of certain value (i.e. those marked with a price of €5) so we say that the value of that banknote is defined in terms of goods and services priced at €5. Likewise, the word *cat* can be "exchanged" for the concept of 'domesticated feline *felis catus*' in the referential/denotational system, so we say that the semantic value of *cat* is that of that concept. But in addition, the value of the €5 banknote can be defined in terms of other banknotes and coins in the Euro currency system:

a €5 banknote is equivalent to five €1 coins or ten 50c coins, etc. Likewise, we can define *cat* in terms of other words and expressions such as *pet*, *feline*, *animal*, *companion*, *kills mice*, etc. Indeed, structuralism developed three ideas to define words in terms of other words and expressions in the linguistic system: lexical field theory, componential analysis and relational semantics.

Lexical field theory (Trier, 1934; Weisgerber, 1954) considers that the semantic value of a linguistic expression is determined by the total set (field) of other linguistic expressions that are semantically related to it. Field theory considers that there are "areas of meaning" that can be covered by linguistic expressions like words or phrases. An "area of meaning" can be conceived as an n-dimensional continuum on which words (or more generally linguistic expressions) can be placed, with each word covering a different sub-area of the continuum. As an illustration, if we visualise this "area of meaning" continuum as a 2-D plane upon which each word covers a rectangular sub-area of meaning, we would end up forming a mosaic of words. The position of a word in the mosaic and the sub-area of the mosaic the word covers determines its semantic value.

Lexical field theory terminology is not standardised and some authors use terms like *semantic field*, *conceptual field* and *lexical field* as synonyms whereas others treat them as quite different concepts. In this thesis, we follow the terminology defined by Lyons (1977, ch. 8): The n-dimensional "area of meaning" that we described above is a **conceptual field**. A 1-dimensional example that Lyons presents is the continuum of colour which is a physical reality that exists independent of language but that we can perceive as human beings. Different individual languages will then structure this continuum differently. English for example will divide up this continuum into distinct ranges within the continuum and call them *red*, *orange*, *yellow*, *green*, *blue*, etc. The word *blue* for example covers a range of the colour continuum, which is distinct and separate for example from the range covered by *red*. The set of words that cover this conceptual field is called a **lexical field**. So, the full set of English words identifying ranges and points along the colour conceptual field is the colour lexical field. Notice that the members of lexical fields need not be mutually exclusive. Some will cover the same or nearly the same range like *violet* and *purple*, giving rise to true or partial synonyms. Some words will also cover a sub-range covered by another word. For example, *azure* can be seen as a sub-range (a hyponym) of *blue*. Conceptual fields can also be discrete, an example is that of *vehicles* whose lexical field would feature members such as: *car*, *bus*, *train*, *motorcycle*, *bicycle*, *unicycle*, etc. The notion of lexical fields given here is very much the initial conceptualisation put forward by Trier. However, it is not Lyon's final refined definition, which will be detailed in Section 2.2.2. But until then and in the remainder of this section, we will continue to work with Trier's original notion of a lexical field as conceptualised so far.

Componential analysis is an extension of lexical field theory. Given the items in a lexical field, componential analysis identifies the features or components that discriminate each member of the lexical field from each other. Consider the previous *vehicle* conceptual field example. We could decompose each member of the lexical field as in Table 2.2.1. By con-

Table 2.2.1: Example of a componential decomposition of the vehicle lexical field

	Vehicle	Engine-powered	Multi-passenger	Private	On Rail	Wheels
car	+	+	+	+	–	4
bus	+	+	+	–	–	4-6
train	+	+	+	–	+	6+
motorcycle	+	+	–	+	–	2
bicycle	+	–	–	+	–	2
unicycle	+	–	–	+	–	1

trasting the items in the lexical field, we can discover the characteristics that differentiate each item form the rest. If an item presents a characteristic we use a plus sign (+) in that characteristic for that item. Conversely, a minus sign (–) is used to indicate that an item lacks certain characteristic. For example, by reading Table 2.2.1 we learn that a *car* is a 4-wheeled, multi-passenger, privately-owned vehicle that is powered by an engine. We also learn that *car*s are not driven on rail tracks.

Both lexical field theory and componential analysis have shortcomings, however. For example, not all conceptual fields can be objectively charted as continuous or even discrete areas of meaning. Even the colour and the vehicle fields could be organised in very different ways by different speakers. It is difficult if not impossible to define lexical fields or perform componential analyses in an uncontroversially universal manner. Even if we ignore this for a moment and agree by convention on a set of lexical fields, we will find anomalies in these lexical fields. For instance, it will be difficult to avoid overlaps or gaps. In the *vehicle* conceptual field above we could argue that an electric bicycle is an overlap between a moped (a type of motorcycle) and a traditional bicycle: it is both manually- and engine-powered. In the colour example, we said that *azure* is a hyponym of *blue*. But what about *cyan*, which is a combination of two distinct hues: *green* and *blue*? As an example of a gap, Lehrer (1974, p. 100) describes a 2-dimensional discrete conceptual field of *cooking* with one dimension describing the nature of the cooking heat —by Convection in an oven, by Conduction in a pan or Radiation from an open flame— and the other dimension describing the cooking medium —Oil, liquid Water or water Vapour—. In her analysis, Lehrer identifies that for example, *frying* is a cooking method with the properties +Oil and +Conduction (pan) whereas *boiling* is +Water and can be done either by +Convection or +Conduction, and at least theoretically also by +Radiation. But she also finds that there are no terms for some cooking heat and cooking medium combinations like cooking in a pan without water, steam or oil (+Conduction, –Water, –Vapour –Oil). There is also no term for cooking with oil on a flame (+Oil, +Radiation on an open flame). Some of these combinations might not be meaningful or desirable from a culinary point of view, but the combinatorial nature of componential analysis allows their specification. Likewise, the componential analysis of our *vehicle* conceptual field allows us to conceive nonsensical vehicles like a multi-passenger, engine-powered unicycle running on a rail track and intended for public transportation. Additionally, it is possible to argue that the components chosen in Table 2.2.1 to decompose the *vehicle* conceptual field or those selected by Lehrer in the *cooking* conceptual field are arbitrary and not necessarily the only ones we can consider. The PRIVATE component in Table 2.2.1, for example, is open to debate. For example, some companies can own buses privately and use them for transporting their own employees and not members of the general public. Similarly, many motorcycles will comfortably accommodate a passenger in addition to the one steering the vehicle, etc.

More importantly however, lexical field theory and componential analysis are rather brittle at dealing with context-specific senses of words. For example, in the New York subway system a *car* is a wagon that forms part of a train in the subway. Similarly, depending on context the term *bike* could mean bicycle or motorcycle. We will discuss these context-specific issues in more detail in Section 2.2.2.

Relational semantics is yet a further refinement on structuralist lexical semantics theory which considers that the meaning of a lexical unit can be defined as the total set of semantic relations it enters with other lexical units (Lyons, 1963). A semantic relation could be synonymy, antonymy, hyponymy, etc. However, Lyons explains that a semantic relation, like synonymy, between any two lexical units is not a consequence of the meanings of those two lexical items, rather that the meanings of the two lexical units are a result of this relation of synonymy in addition to other semantic relations each of them have with other lexical units.

2.2.2 Word senses and the role of context

There are two phenomena that can cause the meaning of a word to be ambiguous: homonymy and polysemy.

Homonymy occurs when two words have the same form (phonetic or orthographic) but have different meaning. For example, *bank*^{1} (meaning 'financial institution') is homonymous with *bank*^{2} ('raised edge of a body of water'). *Bank*^{1} and *bank*^{2} are both **homophones** and **homographs** since both are pronounced and written identically. The fish called *bass*^{1} pronounced /bæs/ is a non-homophonic, homographic homonym of the musical instrument *bass*^{2}, pronounced /beɪs/. Conversely, the pair *read* ('scan/study written material') and *reed* ('water plant') both pronounced /riːd/ are homophonic homonyms, but not homographic. Since the experiments discussed in this thesis handle text exclusively, we shall only be preoccupied with homographic homonyms and assume that no phonetic information or annotation is available to the algorithms processing the text.

Polysemy occurs when a word acquires a different but usually related meaning depending on the context of use, as is the case of the word *mouth* in these two examples:

(2.2.1) Open your *mouth*.

(2.2.2) The East Link bridge is near the *mouth* of the River Liffey.

Table 2.2.2: Examples of homonymous and polysemous words

Homonymous examples	**Polysemous examples**
left: opposite of *right* (n) vs. past tense of *to leave* (v).	*left*: opposite of *right* vs. a political/ideological affiliation.
pole: one of the extreme points on the axis along which Earth turns (n) vs. a person from Poland (n).	*pole*: the Earth poles vs. its magnetic poles and also the poles of a magnet, battery, etc.
bass: a type of fish (n) pronounced /bæs/, a type of musical instrument producing low spectrum sounds (n) or a singer able to sing in a low spectrum (n), both pronounced /beɪs/. A homographic pair.	*bass*: the musical instrument vs. the singer.
deer: animal (n) vs. *dear*: loved (a) or a lovable person (n). A homophonic pair.	*wood*: material extracted from tree vs. a forest.
bark: outer layer of a tree (n) vs. the sound that a dog makes (n) or the act of making this sound (v).	

Table 2.2.2 presents further examples of homonymous and polysemous pairs. The difference between homonymy and polysemy lies in that in homonymy two or more different and unrelated words collide through their form, whereas in polysemy the meaning of a word acquires different but related meanings depending on the context in which the word is used. However, in many cases this difference is not crystal-clear, making it difficult to determine in practice which of the two phenomena is being manifested in a given case of semantic ambiguity. Because of this, the difference between homonymy and polysemy is normally blurred or even completely ignored in computational linguistics works, despite that there is a clear conceptual difference between the two phenomena. For the purposes of this thesis, this theoretical difference between polysemy and homonymy will also be ignored. However, notice that some computational linguistics works such as Boleda et al. (2012) are starting to exploit the regularities of polysemy when processing word senses.

Traditional structuralist lexical semantics ignores or downplays polysemy and homonymy. Lexical field theory and componential analysis define the meaning of a word as it is demarcated by other words. But they pay no attention as to how the meaning of a word can be modified by its context. Some earlier studies in historical linguistics did acknowledge the difference in meaning of a word that can be realised in function of its context. For example, Paul (1920) explains polysemy by stating that any given word has a usual meaning (*usuelle Bedeutung*) and an occasional meaning (*okkasionelle Bedeutung*). The usual meaning of a word is the meaning as understood by most members of a linguistic community, its most established sense. Whereas the occasional meaning(s) of a word are modulations of the usual meaning in actual speech. This view is supported by the observation that the different senses of a polysemous word are often related. Paul also accounts for a language change mechanism in which occasional meanings can become usual meanings: if an occasional meaning of a word is used very frequently, it can overtime become decontextualised, i.e. it becomes the sense that first comes to mind when the word is uttered out of context.

The approach in the semantic analysis presented by Paul differs from that used in structuralism. Paul's analysis consists in observing the word and the contexts it appears in and from these observations induce the word's senses. The traditional structuralist approach, by contrast, first defines a conceptual field and then tries to find the best words —the lexical field— to fill that conceptual field. It is therefore said that structuralist approaches follow an onomasiological approach to lexical semantics whereas an analysis like the one postulated by Paul follows a semasiological approach (Geeraerts, 2010, ch 2).

Onomasiology and semasiology (Baldinger, 1980, p. 278) can be seen as two opposing and complementary methodological ways of describing lexical semantics in disciplines such as lexicography and specialised terminology management. **Onomasiology** assumes that we have a concept or a definition and that the task at hand is to find a name for that concept or definition. It is equivalent to asking questions like "what do you call a domestic animal that meows and purrs?" (a *cat*) or "what do you call going quickly by moving the legs more rapidly than when walking in such a manner that for an instant in each step both feet are off the ground?[5]" (to *run*). **Semasiology** on the other hand starts by looking at a word as it appears in an actual linguistic expression (usually within a corpus) and the assumed task is to determine the word's meaning or sense in that linguistic expression.

By defining lexical fields from pre-defined conceptual fields, traditional structuralist semantics essentially engage in a naming exercise, an onomasiological approach. This is why it is so easy for structuralism to "forget" about polysemy and context: it is impossible to predict by mere introspection every single sense that a word will acquire in all domains or areas of knowledge. Lyons (1977, ch. 8) has made an attempt at correcting this by acknowledging that the meaning of a word depends in great deal on its network of semantic relations with other words and expressions. Lyons explains that the senses of the word in question can be derived or induced from this network of semantic relations with other words or expressions. The advantage of this approach is that it does not require to define a conceptual field a priori, but instead lexical fields emerge from observable semantic relations between words and expressions within particular contexts. In other words, the cliques formed by different words within specific contexts are themselves lexical fields. With this approach, Lyons effectively gives a semasiological twist to structuralist lexical semantics, which had hitherto been insensitive to the role of context in modulating word senses.

Computationally, the distinction between an onomasiological versus a semasiological approach is important as it has direct consequences to the implementation of a system. An onomasiological approach would require a human annotator to manually encode semantic information for a large vocabulary. For example, it would require the manual creation and

[5]Question adapted from first definition of *run* in http://dictionary.reference.com

curation of conceptual fields and their lexical fields or of a vast network of semantic relations between words, like WordNet (Miller, 1995; Fellbaum, 1998). Whereas a semasiological approach is more amenable to the unsupervised methods of sense induction and discrimination.

The types of semantic relations between expressions that Lyons considers correspond to the Saussurean distinction between the syntagmatic and the paradigmatic axes of language. Two or more linguistic units of the same level enter into a **syntagmatic relation** when they are combined in a particular construction (syntagma). At the lexical level, when words are combined within the same phrase or sentence they all are syntagmatically related to each other. For example, in the phrase "the big building", *big* is syntagmatically related with both *the* and *building*. A linguistic unit in a particular construction holds **paradigmatic relations** with other units of the same level when it can be substituted by any of these other units in that particular construction. For example, in the phrase "the big building", *big* could be substituted by paradigmatically related words such as *large*, *small* or even *old*. Notice that this substitution is not necessarily meaning-preserving. The substitution however does preserve the well-formedness of the construction. This implies that the units that are paradigmatically related can be semantically related in some way. Many of the possible semantic relations that can hold between any two paradigmatically related words are those studied by Lyon's relational semantics, some of which were mentioned in the previous section but are briefly expanded upon here:

- **Synonymy**: relationship held by words that have the same or similar meanings. Notice that true synonymy is rare and many synonymous pairs are only synonymous depending on context. E.g. *buy* vs. *purchase*.

- **Antonymy** and **incompatibility**: antonyms are word pairs that mean the exact opposite. Incompatibility in the context of antonymy means that one member of the antonym pair entails that it is not the other member of the same pair. E.g. *hot* vs. *cold*, *big* vs. *small*, etc.

- **Hyponymy** and **hypernymy**: In a taxonomy in which two words have a hierarchical relationship to is other, i.e. one concept refers to a class and the other concept is a member of this class are said to have a hyponymy or hypernymy relationship. The class is said to be the hypernym of the member, and the member the hyponym of the class. E.g. *feline* (hypernym) vs. *cat* (hyponym).

- **Meronymy**: This is a "part-of" relationship between two words. For example, a *house* has *windows*, therefore the windows are the meronyms of the house.

We are now in a position to rework our definition of a lexical field based on Lyons' refinement of the concept. First, we postulate that there exists a **semantic relation** between any two linguistic units if these two units are paradigmatically and/or syntagmatically related. Furthermore, we assume that the **meaning** or **semantic value** of a linguistic unit is determined by the set of semantic relations held by that unit with other linguistic units synchronically within a particular language. Linguistic units of any level that are semantically related belong to the same **semantic field**. A semantic field whose members are exclusively lexemes (or words) is a **lexical field**.

To reiterate, if we define a lexical field based on the semantic relations (paradigmatic and syntagmatic) that a word holds with other words, we are able to discover the meaning or sense of a word in the lexical field by studying these relations (i.e. in a semasiological manner). Since syntagmatic relations are directly observable from corpora, there is no need to first define conceptual fields like Trier and others attempted to do in order to study lexical semantics. It is indeed obvious that statistical, and as we will see also geometric, computational methods can be used to exploit these observable syntagmatic relations in order to discover word senses.

This section concludes with an observation making another case for a semasiological approach over an onomasiological one. There exists a syntagmatic relation between all of the words occurring in a syntagma or construction. However, some of the words in the construction hold a direct syntactic relationship. Consider the opening line of Shakespeare's famous sonnet 18, which reads:

(2.2.3) Shall I compare thee to a summer's day?

All of the words in this line have a syntagmatic relation between each other but some words have direct syntactic relations with each other. For example *I*, *thee* and the phrase *a summer's day* are the subject, direct object and indirect object of *compare*, respectively. However, deeper semantic insights can be inferred from this example: since *thee* is a personal pronoun, we can assume that it is possible to make comparisons between a person and a summer's day, at least in a poetic context. The rest of the poem (context) reveals that the poet derives as much pleasure from the company of a beautiful young person as from a fine summer's day, revealing in the process the creative capacity of language through the mechanism of metaphoric extension (e.g. one line even reads "but thy eternal summer shall not fade"). The point to highlight here is that there is potentially a richer amount of semantic information that can be gained from a semasiological analysis of syntagmatic relations than by a traditional onomasiological brainstorm of concepts. In fact, capturing figurative or metaphorical usages or senses of words cannot be done via an onomasiological process at all since potentially every word in a language can be used figuratively or metaphorically in practically unlimited ways. It would be impossible for any lexicographer to predict a priori every possible metaphorical use that poets in the future will give to any word.

2.2.3 Characterising context

Although not explicitly defined, the notion of context discussed in Section 2.2.2 was what we can call the syntagmatic context of a target linguistic unit token. That linguistic unit token can be a word token, a phrase token, etc. More formally, the **syntagmatic context** of a target linguistic unit token κ_i occurring within a specific syntagma token δ —δ being a unit of a higher linguistic level than κ_i so that it can fully contain it— is the set of syntagmatic relations $\mathcal{R}$ that κ_i holds with other linguistic unit tokens κ_j ($i \neq j$) within δ, with all κ_j being of any unit of a lower linguistic level than δ, but not necessarily of the same linguistic level as κ_i. For instance, in Example (2.2.2), we can define δ as the whole sentence and κ_i as the token *mouth*. The set of κ_j tokens having a relation with κ_i in $\mathcal{R}$ would include individual words like *the*, *near*, *River*, *bridge*, etc. but also phrases or multi-word expressions like *East Link bridge* and *the River Liffey*, as well as sublexical units like any non-free standing morphemes such as roots, inflectional or derivational suffixes or prefixes, etc.

It is well possible to extend the concept of context beyond the confines of the syntagma and indeed venture into sources of non-linguistic information. In a conversation, for example, there will be many situational sources of information that complement and give context to the linguistic utterances being exchanged, like when a buyer says "that one" to a seller in a cake shop while pointing to a particular cake being displayed. The syntagmatic context of "that one" is practically void, leaving the seller to rely largely on the situational context to disambiguate the deictic phrase. Indeed, Firth's (1957) contextual theory of meaning conceives context as a multilevel phenomenon, with each higher level serving as context for its immediate lower level. Therefore, in a conversation, the individual phonemes being uttered lie at the lowest level, the words being formed by the phonemes serve as context for the phonemes, the emerging syntactic constructions in turn serve as context to the words, and so on until we reach the pragmatic and extralinguistic levels serving as situational context for the whole conversation taking place.

Since the experiments described in this thesis exclusively work on textual data, this kind of situational context is not available. However, there are elements of non-linguistic context that are relevant to text corpora, the most important perhaps being the one of domain or genre. The kind of syntagmas we are likely to encounter in a particular corpus largely depend on the domain or genre of the texts collected in the corpus. For example, the shop situation described above could likely feature in a corpus of narrative works but it would be unlikely to be found in a corpus of Physics academic papers. The domain or genre will also determine many grammatical features of the text (likely tense of the sentences, use of active voice vs. passive voice, etc.), as well as the likely senses that a polysemous word or term will have (e.g. a corpus of specialised text will use the more specialised senses of technical terms). This is the reason why the domain or genre should be kept as constant as possible in a corpus used to train any algorithm that deals with lexical senses (Kilgarriff, 1997; Buitelaar et al., 2007). There are other non-linguistic context markers for text, like the position of a target linguistic unit token within the text: at the beginning, in the middle, at the end, as a footnote, as the title or part of the title of a section or even of the whole work, etc. But also typographical cues like bold or italic emphasis, and in the case of word-wide web text, whether a particular term is a hyperlink or not, can be sources of extralinguistic contextual information.

In the present thesis however only two types of contexts will be taken into account: the domain or genre of a corpus, to be simply called **domain**, and the syntagmatic context as currently defined, which will be almost exclusively be referred to simply as **context**. Narrowing our conception of context in this way is motivated by two reasons. The most obvious one is of a practical nature since we will be working with unannotated written corpora available electronically as simple text. The other, more important reason is that this sense of syntagmatic context is directly exploited by the distributional hypothesis of lexical semantics, also called the distributional theory of meaning (Lyons, 1977, sec. 14.4), which is introduced in the following section.

Syntagmatic context of a target word is the primary type of context the present work deals with. The scope of this context however can be rather arbitrary. In Example (2.2.2) the syntagmatic context of the target word *mouth* can be specified to be whole sentence, the noun phrase within which it occurs ('the mouth of the River Liffey') or perhaps just its most immediate neighbouring words *the* and *of*, etc. Different works define context differently and so we would like to introduce a few definitions of context that are more or less precise depending on a particular situation or application (computational or otherwise):

- **syntagma**: a syntactic or textual unit, which in the case of the present work, is larger than the word (phrase, clause, sentence, paragraph, section, chapter, document). When describing an actual system or methodology employing syntagmas as contexts, the specific unit the system or methodology employs will be specified (i.e. "system X uses the sentence a word appears in as context"). When talking at an abstract level or when this distinction is not necessary, the term *syntagma* will be used.

- **text segment** or **segment**: a sequential portion of running text whose length is an arbitrary but fixed number of tokens. I.e. if a document is to be segmented into segments of length l, the first context in the document is the first l words, the second is the second set of l words, and so on.

- **sliding word window**, **word window** or **window** around a target word: a sequential portion of running text of fixed token length, centred at a target word token, not including said target as part of the context. If the window length is say $l = 8$, then the word window context for the word *mouth* in Example (2.2.2) would be "bridge is near the" and "of the River Liffey". That is, the whole window will be of at most 8 words, with 4 words to the left of the target and 4 words to its right.

The sliding word window is markedly different from the text segment and the choice of one over the other has several methodological consequences. For example, a word token can be counted by more than one sliding window, whereas a word token will only be counted by a single word segment. Let us clarify this through an example. Consider the sentence in Example (2.2.4) below and a length $l = 2$.

(2.2.4) The quick brown fox jumps over the lazy dog

This sentence can be divided into 5 segments of (at most) this length: "The quick", "brown fox", "jumps over", "the lazy" and "dog". But it can be divided into 9 sliding word windows (each centred at a token) of at most length 2. The target token at the centre of each window is underlined in the following window enumeration, however note that the underlined target is NOT part of the window context itself: "The quick", "The quick brown", "quick brown fox", "brown fox jumps", "fox jumps over", "jumps over the", "over the lazy", "the lazy dog", "lazy dog". Notice that the first and the last windows are of length 1 each (again, notice that we do not count the target token as being part of the context). This is because each window has to be centred at a token. The first and last tokens will always lack a left and right neighbour, respectively, so they have to have a size less than the fixed window length. Notice however that a text segment or any of the syntagma contexts (paragraph, sentence, etc.) and a sliding word window can interplay while dividing up the corpus into contexts. For example, we can define a word segment of length 100 and then build sliding word windows of size 10 within each word segments. More commonly, the larger segment tends to be a sentence, paragraph or a whole document and is used to avoid building sliding word windows that include words from different documents or paragraphs. I.e. they are used to prevent sliding word windows from crossing the *natural* boundaries of a sentence, a paragraph or a document.

As will be seen in Chapter 3, different approaches use one or the other types of context. For example, LSA-based approaches tend to use segments or syntagmas whilst methods based on Word Space tend to use sliding word windows.

2.2.4 The distributional hypothesis of lexical semantics

It was mentioned in Section 1.2 (p. 20) that the intuition that meaning depends on context can trace its origins to anthropology and philosophy (Malinowski, 1935; Wittgenstein, 1968). Lexicographers have long been using corpora as evidence for word senses. For example, the 1884 edition of the Oxford English Dictionary started the tradition, which continues to this day, of including textual quotations to exemplify word senses (Geeraerts, 2010, Sec. 4.2.3), breaking away from the prescriptivist approach that Samuel Johnson's pioneering Dictionary of the English Language of 1755 started. Firth's (1957) contextual theory of meaning, however, constitutes a methodological starting point for studying word senses from a linguistics point of view: by analysing the words collocating with a target word, it is possible to derive the semantic properties of the target word, or rather its referent. For Firth, the meaning of a linguistic unit is a function of its syntagmatic and situational contexts. "Meaning ... is to be regarded as a complex of contextual relations, and components of the complex in its appropriate context" (Firth, 1957, p. 19).

Harris (1954) described this relationship between meaning and context as a distributional structure. That is, linguistic units co-occur with each other and the co-occurrence patterns between classes of linguistic units would form some sort of distributional structure. He mentions that the relationship or similarity between two words (or other linguistic units) can be determined by the similarity of the environments shared by the two words. For example, if the words occur in almost identical environments but do not co-occur together in the same environment, then we can assume that the words are synonymous (e.g. *oculist* and *eye-doctor*). If the words share some environments but not all, then the words mean different things but their meaning could still be related (e.g. *oculist* and *lawyer*, both being occupations), with their difference in meaning proportional to their difference in shared environments. There could also be some regular ways in which the environments of two words vary. For example, he predicts that *ain't* and *am not* will have a certain regular environmental difference that corresponds to their dialectal difference. The insight that two words are similar to the extent that their contexts (or environments) are similar is termed the **distributional hypothesis of lexical semantics**.

Miller and Charles (1991) articulated a strong and a weak version of the distributional hypothesis in what they called the strong contextual hypothesis and the weak contextual hypothesis:

> **Strong Contextual Hypothesis**: "Two words are semantically similar to the extent that their contextual representations are similar" (p. 8).
>
> **Weak Contextual Hypothesis**: "The similarity of the contextual representations of the two words contributes to the semantic similarity of those words" (p. 9).

The main difference between the two hypotheses is that the weak version implicitly recognises that external information such as a language user's linguistic and general "world" knowledge contribute to his/her judgement of semantic similarity between any two words.

The two hypotheses refer to the notion of semantic similarity but do not elaborate on the type of semantic relation held between the two words deemed to be semantically similar. Miller and Charles consider that the heavy dependence on context similarity in the strong version of the hypothesis is unrealistic in some cases. A case they point out is that of some word pairs that are very similar semantically that will necessarily occur in different syntactic contexts. For example, *department* and *departmental* are very similar but they will not occur in the same contexts because they have different parts of speech. This is why Miller and Charles formulate the weak contextual hypothesis. Yet, this explanation does not clarify what is intended by semantic similarity in these hypotheses.

In the natural language processing literature the concept of semantic semantic similarity is usually a broad one and works often do not distinguish between syntagmatic or paradigmatic relations, let alone between the subtypes of paradigmatic relations such as synonymy, hyponymy, meronymy, etc. Often, the term **semantic relatedness** is preferred over semantic similarity (e.g. Budanitsky and Hirst, 2001). Several empirical works have attempted to construct systems based on different operationalisations of the contextual hypothesis and, through inspection or through their performance in an NLP-based task, determine the type of semantic relations held between word pairs deemed to be similar by each operationalisation. For example, Sahlgren (2006) and Utsumi (2010) found that word-context vector space models are better suited for measuring syntagmatic relations between word pairs whereas word-word vector space models measure paradigmatic relations better (see Section 3.6 on p. 102 for a more detailed discussion), but both argue that both models have a degree of overlap. Turney and Pantel (2010) survey the vector space model literature and link different models with some of the different types of semantic similarity:

- **attributional similarity** is the measure to which two words share properties. A word pair that has high attributional similarity is *dog* and *wolf*. The traditional paradigmatic relations mentioned earlier (synonymy, antonymy, hyponymy, etc.) are all attributional relations. Turney and Pantel consider that word-context models measure well this type of similarity (compare this to Sahlgren's and Utsumi's findings).

- **relational similarity** is the measure to which two word pairs share relations. Two word pairs that have high relational similarity are *dog:bark* and *cat:meow*. Turney and Pantel consider pair-pattern matrix models, like that introduced by Lin and Pantel (2001), to help in measuring relational similarity.

- **taxonomical similarity** is a specific type of attributional similarity that co-occurs between co-hyponyms, like *car* and *bicycle*, both hyponyms of *vehicle*.

- There is a **semantic association** between two words that tend to co-occur, like *bee* and *honey*. In the terminology adopted in this thesis, we would say that these words are syntagmatically related.

This thesis uses the terms *semantic similarity* and *semantic relatedness* as synonymous, referring to the broadest sense of semantic relationships between words as commonly done in the NLP literature. However, when a finer distinction between semantic relations is needed, the terms *paradigmatic relation* and *syntagmatic relation* will be used.

In Section 3.1.1.1 (p. 61) we shall see that knowledge-based, supervised word-sense disambiguation could be seen as a way of operationalising the weak distributional (contextual) hypothesis as it exploits knowledge sources foreign to the target word's own syntagmatic context. However, some of these external knowledge sources include information that is contextual in nature, such as selectional preferences, syntactic behaviour, etc. By contrast, unsupervised word-sense discrimination could be seen as a total commitment to the strong contextual hypothesis (Section 3.1.1.2, p. 64) as it only works with the linguistic information contained in the contexts of the target word. Whilst this can be seen as a weakness, it can also seen as a strength as it can be employed in situations where external knowledge sources are not available. In addition, it has been observed that knowledge sources are often specific to a domain and when used to disambiguate words in another domain, performance decreases (Kilgarriff, 1997; Agirre and Edmonds, 2007).

Perhaps the strongest version of the distributional hypothesis is what Turney and Pantel (2010) call the **bag of words hypothesis**, which states that "frequencies of words in a document tend to indicate the relevance of the document to a query" (p. 153), an information retrieval interpretation of the distributional hypothesis that takes into account no information other than word frequencies and/or statistics derived from such word frequencies. Chapter 3 describes how the operationalisation of this hypothesis in information retrieval in the form of the vector space model (Salton, 1971; Salton et al., 1975) can be adapted for word senses. This operationalisation is the backbone holding the rest of this thesis together.

2.2.5 Meaning beyond context

The preceding sections describe lexical semantics from a structuralist and distributionalist perspective. Section 2.2.2 culminates with Lyons' (1977) definition of meaning which states that the semantic value or meaning of a word is determined by the set of semantic relations it holds with other linguistic units. The same section also classifies these semantic relations into syntagmatic and paradigmatic relations. Then, Sections 2.2.3 and 2.2.4 focus on the syntagmatic relations of words to describe the distributional theory of lexical semantics. As presented, none of these concepts make mention of the referential model presented in the introduction to this chapter. Geeraerts (2010, Sec. 2.4) attempts to interpret several works by Lyons at different points of his career and concludes that apart from the syntagmatic-paradigmatic dimension of semantic relations presented so far, Lyons also recognises another dimension of semantic relations: sense relations and meaning relations. **Sense relations** are relations that words have between each other within the linguistic system and these tend to be primarily the paradigmatic and syntagmatic relations that we have studied. The **meaning relations** between words, on the other hand, have a broader nature than sense relations and tend to lie at the referential/encyclopaedic level, outside the linguistic system. For example, the paradigmatic relations of hyponymy (*car* is a type of *vehicle*) and synonymy (*quickly* $\approx$ *speedily*) presented in Section 2.2.2 are sense relations and would be considered to be part of the linguistic structure since they involve concepts such as entailment and word senses. But *encyclopaedic* type of relations such as maker-product relations (*composer* and *music*, *cook* and *meal*, etc.) would be meaning relations, belonging to the referential realm as they require real-world knowledge beyond pure linguistic knowledge to recognise and realise these relations. Geeraerts also explains that this distinction between the linguistic and referential levels (and as a consequence between sense and meaning relations) is contentious. He cites the case of meronymy (whole-part) relations (like in *hand* and *finger*) which are sense relations that depend on real-world knowledge. Whilst a discussion on these theoretical issues lies beyond the scope of this thesis, they show that both the distributional and the referential models play a part in the way we understand the meaning of words.

The study of collocations (Section 2.1.2 and Chapter 8) is one area touched by this thesis where the distributional and the referential models interact in interesting ways. Sinclair (1991, ch. 8) considers that there are two competing principles in which one can explain how people construct linguistic expressions: the **open-choice principle** and the **idiom principle**. In the open-choice principle, language users construct expressions from words freely guided only by the syntactic and semantic rules of the language whereas in the idiom principle, language users build their sentences from pre-fabricated chunks, i.e. idiomatic expressions or fixed phrases. Sinclair considers that in reality speakers must use a mixture of the two principles in their daily language use with a bias towards one or the other, based on necessity, context, intent, etc. For example, a poet will perhaps be more inclined towards the open-choice principle in order to craft an original poem whereas a writer in a gossip magazine will perhaps tend to use more idioms and clichés as well as puns based on these for some comedic effect.

In later work, Sinclair (2004, p. 29) links the open choice principle with the referential model ("the terminological tendency") and the idiom principle with the distributional model ("the phraseological tendency"). The rationale here is that the more we move towards the open choice model, the more we need to refer to the encyclopaedic or referential meaning of each word to select the precise meaning of the actual message we want to convey. Whilst, as we move more towards the idiom principle, the more the meaning of each word can be explained by the way in which it is combined with other words. This reasoning is motivated by Firth's (1957) observation that the context of a word gives an indication to the meaning of the word, as in the meaning of *cows* in clauses like *they are milking the cows* and *cows give milk*. This is in fact the motivation used in the disambiguation of senses in words like *hard* 'difficult' vs. *hard* 'solid/firm' in many of the experiments in this thesis, as it is assumed that a change in sense reflects a change in co-occurrence distribution. For example, we can expect words such as *exam*, *test*, *job* to co-occur with *hard* in its 'difficult' sense, whereas words like ice, surface, wood, etc. in the neighbourhood of *hard* in its 'solid/firm' sense.

There is an observation to be made here regarding the concept of MWE compositionality. Recall that it is possible to grade a collocation or MWE from completely non-compositional (or non-literal) to fully compositional (or fully literal). It seems natural to correlate this non-compositional-compositional continuum with the trade-off between the open choice and idiom principles. After all, fully non-compositional expressions are often idiomatic expressions. It thus seems appropriate to remark that this theory considers idiomatic and other non-compositional expressions to be fully distributional/syntagmatic and that no recourse to the referential model is warranted. In one sense, this is the case since we can expect the contextual distribution of say *red tape* in its idiomatic (non-compositional) sense to be very different from that of the same collocation when used in its literal (compositional) sense (i.e. an actual piece of tape that is red in colour). The selection of words surrounding an occurrence of *red tape* will be indicative of its meaning and its degree of compositionality.

However, one cannot completely disregard the referential model in this situation. Consider a reader unfamiliar with the idiomatic sense of *red tape* (e.g. an L2 speaker of English). Upon encountering it in a newspaper article about tax declarations, the reader will perhaps find anomalous the occurrence of the bigram in that context. This apparent anomaly will signal the reader that it is being used in another sense. Depending on the size of the context, the number of times the collocation has been encountered and other factors, the reader may or may not be able to induce the meaning of the expression. If the reader is not able to induce the meaning from the context, he/she might recourse to a dictionary or ask another speaker. That is, the reader might fall back on the referential model if context is deemed insufficient to determine the meaning of an expression[6].

These observations show that despite structuralist and distributional semantics to downplay or ignore the referential or mental representations, they need to be considered in the study of lexical semantics. Recall that one of structuralism's original reasons to downplay or completely ignore such mental representations is that they cannot be observed directly, especially at a time when MRI scanners and other tools available to psycholinguists and neurolinguists today had not yet been invented. Since language can, on the other hand, be directly observed through linguistic elicitations and corpora, there exists a motivation to develop methods that can derive as much semantic information as possible from actual linguistic samples. This thesis focuses on a small family of such methods, namely those that take text as input and attempt to derive semantic information from the distributional behaviour of words in such textual data. By their nature, these methods have limited or no access to mental processes. Structural and distributional semantic theories are thus attractive models for the development of these methods for they shed a theoretical light on what can potentially be achieved with what can be observed. This is why they are presented prominently in this chapter.

As mentioned in the introduction to this chapter, the specific lexical semantics theories presented here were selected because they feed into the distributional/contextual hypothesis, which is operationalised computationally in Chapter 3, but this selection is far from being exhaustive. There are many more semantic theories, some of which have been operationalised in natural language processing systems as well. One example is the generativist semantics theory introduced by Katz and Fodor (1963) which transformed structuralism's componential analysis by adding an explicit system of formal description based on generative grammar principles and by using a referential mechanism to describe meaning based on the properties and attributes of the objects being described. This theory has evolved in many ways and has given rise to frameworks like the formal semantics theories introduced by Davidson (1967) and Montague (1974), as well as theories such as frame semantics (Fillmore, 1982), which paved the way for many computational linguistic approaches such as the semantic role la-

[6]It is worth mentioning that full encyclopaedic knowledge is not needed to satisfactorily understand and correctly use idiomatic expressions. For example, many people do not know that the origin of the bureaucratic sense of *red tape* is the red ribbons (or tapes) used to bind official and other State documents in government offices, and yet use and understand the expression successfully. For many other idiomatic expressions, their origins are obscure or not known and this does not preclude their correct interpretation. Besides, when we ask another speaker for the meaning of a word we do not know, usually the reply we get consists of a synonym or near synonym, sometimes with a example sentence illustrating its use or explanations as to its appropriate use, etc. Rarely, if ever, will we get a dictionary definition or encyclopaedic note as a reply to our question from another speaker. This could be seen as giving support to the intuition that people learn a great deal of new words from examples of word use in context (syntagmatic relations) and from the paradigmatic relationships (like synonymy) these new words hold with other words.

belling research programme (Carreras and Màrquez, 2004; Erk, 2007). This latter approach is interesting as it also combines elements of the distributional hypothesis. However, it falls beyond the scope of this thesis and is thus not described.

There are yet other alternative lexical semantics theories that are not fully amenable to computational operationalisation and thus fall outside the scope of this thesis naturally. A clear example is Wierzbicka's natural semantic metalanguage which, like Katzian semantics, attempts to incorporate referential and world knowledge in componential semantic descriptions but using an inventory of universal primitives written in simple, everyday natural language instead of logic primitives (Goddard and Wierzbicka, 2002).

3 Computational Background

Chapter 2 presented a survey of lexical semantics theory and concepts that culminated with the distributional or contextual hypothesis. These theoretical constructs can serve as a guideline to construct natural language processing systems. For instance, Weaver (1955) pointed out that for a machine translation system to properly resolve lexical ambiguity, it should exploit co-occurrence patterns of the words serving as context for a given ambiguous word. Whilst this is an important invocation of the distributional hypothesis (and thus of lexical semantics theory) in a computational setting, the application of lexical semantics theory to natural language processing tasks does not end there. It has indeed been applied in many other computational tasks such as natural language understanding, computational lexicography, lexical acquisition, information retrieval, etc.

However, for this application of lexical semantics theory to NLP to actually take place, it is necessary to first define exactly how it can be operationalised computationally. This chapter seeks to describe computational models that operationalise the distributional hypothesis and some of the other linguistic concepts presented in the previous chapter. In particular, this chapter (Sec. 3.3) is concerned with descriptions of two models of distributional lexical semantics that derive from Salton's vector space model (VSM) (Salton, 1971; Salton et al., 1975): Word Space (Schütze, 1992, 1998) and Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer and Dumais, 1997). The common theme in all three models is that they represent words in context mathematically via high-dimensional vectors. This vector representation can be interpreted as an operationalisation of the distributional hypothesis, and more specifically, as an operationalisation of what Turney and Pantel (2010) call the bag of words hypothesis.

This chapter does not attempt a direct comparison between LSA and Word Space. That is left for Chapters 4 and 6. Rather, it describes the mathematical and semantic properties of each vector space. For example, Section 3.4 presents details of LSA's SVD projections as well as a brief numerical review exploring the often-claimed capabilities of the model to capture semantic relations between words whilst being sensitive to polysemy. And Section 3.6 attempts to perform a linguistic interpretation of the vector spaces defined by Salton's VSM, LSA and Word Space. This analysis concludes that Salton's VSM is an implementation of an approximation of De Saussure's syntagmatic relations whereas non-SVD reduced Word Space in turn can be seen as capturing somewhat De Saussure's paradigmatic relations (Sahlgren, 2006). LSA seems to mix syntagmatic and paradigmatic characteristics in the same vector space.

The chapter starts by documenting how LSA and Word Space have been successfully applied to several natural language processing tasks in Section 3.1. It focuses on word-sense disambiguation, word-sense discrimination/induction and multi-word expression compositionality grading. Then, Section 3.2 introduces Salton's VSM in its native information retrieval context while highlighting the properties that will be exploited once it is adapted for lexical semantics. Section 3.3 presents actual adaptations of the VSM to lexical semantics, whilst Sections 3.4 and 3.5 introduce LSA and Word Space, respectively. Finally, Section 3.6 presents the discussion on syntagmatic and paradigmatic spaces mentioned before and makes the case to consider Salton's VSM as a syntagmatic space, Word Space as a paradigmatic space and LSA as a combination of syntagmatic and paradigmatic spaces.

Some of the definitions presented in this and subsequent chapters introduce token representations. In order to avoid confusing the different token representations, each definition is marked with a three-part label of the form X-X-XX, which summarises the properties of a token representation. For instance, D-A-UR describes a direct "D" token representation based on Salton's "**A**" matrix, unreduced "UR", whilst I-W-R2 describes an indirect "I" token representation based on Schütze's word matrix "**W**" and reduced by SVD's "R2" projection. The specific meaning of these descriptions will be explained in this and the following chapter. The label for a token representation is presented as part of its definition. Also, Table 4.2.1 (p. 111) is a catalogue of each of these token representations pointing to the exact places in the thesis where they are defined.

3.1 Natural language processing tasks

Lexical semantics theory in general, and the distributional hypothesis in particular, provides insights as to what aspects of text are more informative to capture and exploit the semantic properties of words for a particular purpose. Academic and applied fields that make use of the insights provided by lexical semantics are as diverse as philosophy, lexicology, lexicography, corpus linguistics, syntax, pragmatics, child language acquisition, psychology, etc. Computationally, lexical semantics inform and/or enable natural language processing tasks such as natural language understanding, lexical acquisition, knowledge representation, computational lexicography, machine translation, information retrieval, etc. Within these tasks, there are several subtasks that aim to directly exploit these lexical semantic insights and are often designed as (explicit or implicit) components within larger NLP systems. These subtasks include word-sense disambiguation and discrimination, multi-word expression compositionality grading, ontology construction, semantic role labelling, etc.

As already mentioned, machine translation's dependence on resolving the ambiguity of words in context was recognised early on (Weaver, 1955) and whilst many systems (rule-based or statistical) do not include an explicit word-sense disambiguation module (Och, 2002; Resnik, 2007) there are studies indicating that a form of word-sense disambiguation/discrimination based on assigning target translations to ambiguous source words, instead of assigning senses, can indeed help phrase-based statistical machine translation systems (Carpuat and Wu, 2007).

Besides machine translation, the supervised WSD subtask can be applied to other NLP tasks like information extraction, named-entity classification, co-reference determination, acronym expansion, among others (Agirre and Edmonds, 2005, p. 11). Its unsupervised counterpart, WSDisc has also been applied to NLP tasks such as information retrieval. For instance, Véronis (2004) developed a system that produced graphs of word co-occurrences from text corpora that was able to discriminate the linguistic usages (senses) of highly ambiguous search queries, whilst Navigli and Crisafulli (2010) performed clustering of the results returned by the Yahoo! web search engine based on the induced senses of the words contained in the search result snippets.

WSDisc and compositionality measurement of MWEs can be used in lexicography and terminography (terminology management). For example, WSDisc can be used in a concordancing system that attempts to induce the senses of the word being searched and clusters the concordances based on these word senses and presents them as distinct groups to the lexicographer or terminologist. A MWE compositionality measuring tool could help a terminologist determine whether a multi-word term can be split into smaller sub terms or not (presumably a low compositional multi-word term will be more atomic and will not tend to have sub terms). A low compositionality grade on a multi-word term can also indicate translators not to translate such a term literally and to instead look for a suitable equivalent idiomatic expression in their target language. In any case, MWEs in general constitute an important problem that NLP systems need to solve to fully succeed in many tasks (Sag et al., 2002).

Beyond word senses and MWEs, lexical semantic theories can be applied to many other natural language tasks. One is the interface between syntax and semantics. For instance, the semantic attributes of a word can be used as predictors of its correct inflection (Bond, 2005), useful e.g. in natural language generation. But also, the syntactic environment of a word can give many clues to its semantic properties (Levin, 1993), an insight that is exploited in semantic role labelling (Fillmore, 1982; Carreras and Màrquez, 2004; Erk, 2007).

As discussed in Chapter 2, lexical semantics also studies the semantic relations between words. Learning and/or correctly handling these semantic relations are of great importance to ontology construction, lexical acquisition, computational lexicography and terminography, etc. Sahlgren (2006), Utsumi and Suzuki (2006) and Utsumi (2010) compare several computational representational mechanisms derived from the VSM in their efficacy to represent syntagmatic or paradigmatic relations between words (Sec. 2.2.2, p. 42). Works that study paradigmatic relations from a computational point of view include Widdows (2004), who gives an introduction to distributional methods in general and covers these types of semantic relations, as well as Miller and Charles (1991), Miller (1995) and Fellbaum (1998) who study these semantic relations against the backdrop of the distributional hypothesis with the view of developing the WordNet lexical database.

There have been many ways in which the distributional hypothesis and other lexical semantic theories have been operationalised. An early example for WSD is Lesk's (1986) algorithm (see Section 3.1.1.1) which disambiguates polysemous words based on their sense definitions in machine-readable dictionaries. An alternative example of an operationalisation of the distributional hypothesis can be seen in the statistical association measures for collocations (Dunning, 1993; Banerjee and Pedersen, 2003; Pecina, 2005). These measures can determine the degree to which two words found in the vicinity of each other in text form a collocation by performing certain statistical tests based on the relative frequency of one word on its own against its frequency when it is accompanied by the other word. This thesis however is concerned with a different family of such operationalisations: the group of vector spaces derived from Salton's vector space model (VSM) (Salton, 1971; Salton et al., 1975). The VSM family of methods have the advantage over other methods of being cemented on a clear mathematical footing: that of vectors and geometry. If words can be represented in vector space, it is possible to take advantage of the numerous linear algebra tools and methods developed for vectors in many fields of engineering and science (Widdows, 2004). In addition, VSMs allow the incorporation of non-VSM methods. For example, the collocation statistical association measures mentioned before can be used as vector features that represent a word type or a token (Purandare and Pedersen, 2004) and even matrices computed from raw word counts can be transformed to such statistical features with relative ease (Section 7.1, p. 154).

In addition to information retrieval, and cross-lingual information retrieval (Dumais et al., 1997), the VSM and some of its derivatives, such as LSA, have been applied to areas such as document clustering and classification (Sebastiani, 2001; Manning et al., 2008), essay grading (Wolfe et al., 1998; Foltz et al., 1999), answering TOEFL synonym questions (Landauer and Dumais, 1997; Turney, 2001; Rapp, 2003), answering analogy SAT questions (Turney, 2006), semantic role labelling (Erk, 2007), just to mention a few.

Of course, VSMs have also been applied to word-sense disambiguation and discrimination/induction (Schütze, 1998; Purandare and Pedersen, 2004; de Marneffe and Dupont, 2004; Agirre and Stevenson, 2007; Navigli, 2009; Navigli and Crisafulli, 2010). The rest of this section describes in more detail these tasks, as well as the task of multi-word expression compositionality measurement. As part of this description, a brief survey of ways in which VSMs have been applied to these tasks is provided.

3.1.1 WSX: Word-sense disambiguation, discrimination and induction

Vector space models have been widely applied to the problem of resolving lexical semantic ambiguities. Leacock et al. (1993) was perhaps the first such application in a classic word-sense disambiguation experiment. Schütze (1998) introduced the whole concept of unsupervised word-sense discrimination and adapted his type-based Word Space model (Schütze, 1992) to represent tokens via indirect (second-order) context vectors. He also showed the usage of

SVD to reduce the dimensionality of Word Space vectors. Levin et al. (2006) adapted LSA for word-sense disambiguation by using its direct context vectors (D-A-*) as proxies for token representations of the words to be disambiguated. Direct (first-order) context vectors (D-C-*) were used by Pedersen and Bruce (1997) in a word-sense discrimination setting. The following subsections discuss the tasks of word-sense disambiguation and discrimination/induction and describes the application of vector space models to these tasks.

3.1.1.1 Word-sense disambiguation

Word-sense disambiguation (WSD) is the task of determining the sense of an ambiguous word in context. The task was first conceived as a computational task by Weaver (1955) as a component for a machine translation system. It can be summarised as follows: given a token of a word occurring in some syntagma and given a list of possible senses for such a word, determine the token's correct sense from the list. The task is therefore seen as a classification task, in which each sense of the ambiguous word is seen as a class into which each instance (token) of the word has to be assigned. It is assumed that every possible sense of the word has been defined and that that list is finite and relatively short.

This assumed list of senses is WSD's connection to lexicography as many historical and current systems employ machine-readable dictionaries as sense inventories for large vocabularies. For example, Lesk (1986) used the sense definitions in the Oxford Advanced Learner's Dictionary of Current English (OLAD). Lesk's algorithm works by computing the overlap (see eq. 3.2.1 on p. 68) between the context of an ambiguous word token and each OLAD sense definition for that ambiguous word, and selecting the sense for which the overlap is maximised. This basic algorithm was expanded by subsequent works. For example, Guthrie et al. (1991) and Yarowsky (1992) adapted Lesk's algorithm to use the Longman Dictionary of Contemporary English and Roget's thesaurus, respectively. However, most modern dictionary-based methods exploit the WordNet lexical resource (Miller, 1995; Fellbaum, 1998), which organises a set of words hierarchically into sets of cognitive synonyms called "synsets". These WordNet synsets, which can be used as sense labels for ambiguous words, are the most widely used sense inventory in WSD (Agirre and Edmonds, 2007, p. 7).

WSD methods based on a vector space model often do not employ such lexical resources directly. Instead, they rely on a sense-tagged corpus. That is, a corpus on which the occurrences of a target word of interest have been annotated. Often, the senses employed in the annotation come from a lexical resource. For example, Leacock et al. (1993) directly employed Salton's VSM segment vectors representing sentences containing occurrences of the polysemous word *line* (see Section 5.1, p. 136). Each of the instances of *line* was manually sense-tagged according to WordNet synsets. Then, a classifier was trained on a randomly selected subset of the segment vectors by computing sense vectors based on a weighted aggregation of the individual segment vectors belonging to each sense. Performance was computed via the accuracy in classifying the remaining segment vectors not used in the training phase (i.e. the test vectors) to their correct sense via the dot product (3.2.6). Similarly, Levin et al. (2006) adapted LSA's segment vectors to train a means-based Rocchio classifier on the hard-interest-line-serve (HILS) dataset (Sec. 5.1, p. 136). They also randomly partitioned each sense-tagged corpus into mutually exclusive training and test portions. Performance was measured by computing the classifier's accuracy in assigning each test segment vector to the correct sense class. We conducted a very similar experiment in Maldonado-Guerra and Emms (2012) (see Sec. 5.3, p. 140) but using Word Space direct (first-order) and indirect (second-order) context vectors. Chapter 6 repeats the experiment more systematically with unreduced and SVD-reduced versions of direct LSA segment vectors, Word Space direct (first-order) context vectors, indirect LSA segment vectors, Word Space indirect (second-order) context vectors computed from a word matrix and from a matrix of direct context vectors. Finally, Section 7.3 (p. 157) repeats the experiment using our proposed method of dimensionality reduction for Word Space based on word matrix consolidation.

Modern supervised WSD can be seen as a machine learning approach that exploits two knowledge sources: lexical semantic information coming from a dictionary or lexical resource and contextual information coming from sense-annotated occurrences of ambiguous words in actual corpora. WSD can thus be thought of being an interpretation of the weak distributional hypothesis, relying on two sources of information rather than on context alone. There are however a few inherent weaknesses in the dependence of external lexical resources. One is that different dictionaries organise and group the senses of a word in different ways. Also, dictionaries need to balance trade-offs between concept demarcation precision, space, brevity, and ease of use. So, dictionaries cannot be extremely detailed with every single shade of meaning that a word can acquire. Because of this, it is unlikely that a general purpose dictionary or lexical resource such as WordNet will be informative of important sense distinctions for terminology in every single technical or specialised domain. We could however make the executive decision of defining word or term senses for a specific domain and develop a system based on a generic algorithm that will work well provided that sense lists are available for that domain. Applying the system to a new domain would only be a matter of compiling the relevant word or term sense list. This however, brings us to the other issue of supervised WSD: the expense in terms of both time and money in creating and maintaining in the long run such a sense list as well as the effort in maintaining the sense annotation in training text corpora. This has been dubbed the **information bottleneck** that plagues supervised WSD.

Kilgarriff (1997) argued that what we understand as a word sense depends on the purpose of the task that requires word senses and that word senses themselves do not exist independent of this purpose. For example, a task could be analysing the senses of the word *bank* in a particular newspaper corpus. The sense distinctions we are likely to obtain from such a task will be those that are important for that particular corpus in that particular task. This echoes the onomasiological/semasiological dichotomy discussed in Chapter 2, in that supervised WSD takes an onomasiological standpoint and assumes that an exhaustive list of senses for every ambiguous word will be available, whereas in reality the senses a word can take are modulated according to each particular domain in which it occurs, and a semasiological/unsupervised approach has better chances of learning these senses as they manifest in each domain in question. These observations have motivated the research on unsupervised approaches to word senses such as word-sense discrimination and induction (Sec. 3.1.1.2), which decrease or eliminate their dependence on manually crafted sense lists and rely more on the information that can be directly derived from the context of ambiguous words themselves, effectively embracing a stronger version of the distributional hypothesis.

An interesting recent trend might help overcome (at least partially) the information bottleneck problem. This trend is the exploitation of user-generated collaborative semi-structured content (Hovy et al., 2013). Wikipedia is a prime example of such content. It is user-generated and collaborative in nature but is also semi-structured, in the sense that articles are inter-linked and are organised in hierarchies or taxonomies. Projects such as BabelNet (Navigli and Ponzetto, 2012) aim to take advantage of these connections in order to build a fully-structured lexical and semantic database. BabelNet in particular combines lexicographic and encyclopaedic knowledge from WordNet and Wikipedia, respectively, in a structured manner in the form of a graph termed a semantic network. Moro et al. (2014) introduce a graph-based approach to WSD and named-entity disambiguation that takes advantage of BabelNet. This approach, called Babelfy, attempts to disambiguate all the words and terms in a document that are also captured in BabelNet by traversing the sense graphs of each of these words and terms and selecting the most dense sense subgraphs formed. Since there are no sense labels *per se* in this approach and because disambiguation is performed by processing the semantic networks formed by word types, it could perhaps be regarded as a type-based (Pedersen, 2007) word-sense discrimination approach.

An important point of discussion in WSD research is the issue of evaluation. There are two general ways in which an NLP module can be evaluated: intrinsically and extrinsically. Intrinsic evaluation is a direct and isolated type of evaluation in which the NLP module is tested in a lab setting, without regard to a particular application. An extrinsic evaluation is a task-based evaluation that tests the NLP module in question as part of a bigger system and, all other settings being equal, evaluates the difference in performance of the bigger system brought in by the addition of the NLP module that is being tested. The early Senseval WSD evaluation competitions focused on intrinsic evaluation (Palmer et al., 2007) whilst new Semeval competitions have tended to also incorporate extrinsic evaluations[1], in addition to traditional and improved intrinsic evaluations. Both intrinsic and extrinsic evaluations have advantages and disadvantages. On the one hand, intrinsic evaluations are easy to perform, can be automated and allow researchers and practitioners to compare two or more competing modules against the exact same criteria. However, they do not necessarily predict accurately the performance of the module in a real production system. On the other hand, extrinsic evaluations do compare

[1]For example, the latest iteration of the competition, Semeval-2014, included extrinsic evaluations based on systems that required word-sense disambiguation, such as computer-assisted writing and composition for L2 users/second-language learners (Van Gompel et al., 2014), as well as many other semantics-related evaluations that did not centre on word senses.

the performance of modules in more realistic settings. However, sometimes the difference in performance of a larger system could be due to factors external to the modules being tested, so extra care should be taken in performing these evaluations. Another disadvantage of extrinsic evaluations is that they require more time and effort to put together and also, the performance of a module can be system- and task-dependent. That is, the performance of a module in a particular system for a particular task is not necessarily predictive of that same module's performance in another task or even in the same task but in a different system. For robustness, it is recommended that both intrinsic and extrinsic evaluations be performed, where possible. Since these evaluation issues are present in NLP systems in general, the discussion presented here is also applicable to unsupervised WSDisc as well as automatic compositionality grading.

3.1.1.2 Word-sense discrimination

As mentioned in the previous section, relying on manually curated lists of word senses brings limitations and philosophical challenges to the supervised word-sense disambiguation approach. Schütze (1998) introduced an alternative approach: **word-sense discrimination** (WSDisc) or **induction** (WSI). Word-sense discrimination differs from word-sense disambiguation in that it does not require manually created sense lists and instead automatically induces the senses of an ambiguous word from its occurrences in an unannotated corpus. WSDisc however has a theoretical limitation over WSD. Since the word senses in a WSD setting are defined and known a priori, a WSD system is able to tell which sense each target word token has. By contrast, in a WSDisc setting, word senses are not known a priori and so a WSDisc system cannot assign a sense to a particular token of a target word. It could assign a symbolic ID or label to each of the induced senses, but this ID or label cannot be directly interpreted by a language user. In practice this might not be problematic, as there are applications that are not concerned with human-interpretable sense labels but with determining whether two or more instances of some target word have the same sense or not, i.e. with discriminating instances of a word semantically.

Let us define the task of word-sense discrimination as follows. Given an untagged corpus $\mathcal{C}$, an ambiguous word τ, and a set of tokens of τ, $Tokens = \{\kappa_i : \tau = \text{type}(\kappa_i)\}$ (where $\text{type}(\kappa)$ returns the word type of word token κ), cluster the vector representations $\mathbf{r}(\kappa_i)$ of instances (tokens) κ_i of τ in C. Each cluster $\mathcal{K}_j$ induced represents a sense σ_j of τ, for which there exists some vector representation $\mathbf{s}(\sigma_j)$. Given a new instance κ' of τ, produce a context representation $\mathbf{r}(\kappa')$ and assign it to the sense representation $\mathbf{s}(\sigma_j)$ most similar to $\mathbf{r}(\kappa')$. I.e. the sense for κ' is $\arg_{\sigma_j} \max(\text{sim}(\mathbf{r}(\kappa'), \mathbf{s}(\sigma_j)))$, where $\text{sim}(\mathbf{u}, \mathbf{v})$ is some similarity measure between $\mathbf{u}$ and $\mathbf{v}$. All of this is done without recurring to a previously sense inventory such as a machine-readable dictionary or a thesaurus.

Word Space was very much born inside WSDisc. Indirect (second-order) context vectors were introduced by Schütze (1998), the same paper that introduced WSDisc. These context vectors are computed from a word co-occurrence matrix, introduced in Schütze's earlier work

(1992). Direct (first-order) context vectors were also applied in WSDisc around the same time by Pedersen and Bruce (1997). Word Space context vectors have been used for WSDisc in many works such as Purandare (2004), de Marneffe and Dupont (2004), Charoenporn et al. (2007) and Maldonado-Guerra and Emms (2012).

There have not been many works applying LSA to WSDisc. Despite the title of the Levin et al. (2006) paper, the work does not use LSA's segment vectors for WSDisc but for WSD, although the WSD experiment is admittedly assumed to predict a ceiling for a hypothetical WSDisc experiment. Pino and Eskenazi (2009) by contrast used LSA segment vectors in a WSDisc component used in an interactive reading comprehension/vocabulary learning educational system. This system presents students with a reading passage containing a number of words they need to learn. After the reading time is up, the system presents the students with automatically generated fill-in-the-blank questions in which they have to select the word from a list that best fills the blank. The questions are sentences automatically extracted from the web containing the word(s) the student is to learn. The WSDisc component is used to select sentences that use the word in the same sense as they were used in the passage presented to the student in the first part of the exercise. Pedersen (2010) attempts a comparison between LSA and Word Space in a WSDisc setting. However, as will be discussed in Section 4.3 (p. 122), this work does not capture a subtle difference in counting between each model's direct token representations, and as a result he ends up comparing several Word Space variants instead.

This thesis also presents adaptations of Word Space and LSA to the WSDisc problem. Section 5.4 (p. 141) compares the performance of direct (first-order) and indirect (second-order) Word Space context vectors in word-sense discrimination experiments. Then, Chapter 6 repeats the experiment more systematically with unreduced and SVD-reduced versions of direct LSA segment vectors, Word Space direct (first-order) context vectors, indirect LSA segment vectors, Word Space indirect (second-order) context vectors computed from a word matrix and from a matrix of direct context vectors. Finally, Section 7.3 (p. 157) repeats the experiment using our proposed method of dimensionality reduction for Word Space based on word matrix consolidation.

3.1.2 Measuring the compositionality of multi-word expressions

Lexicalised multi-word expressions (also termed collocations) are often opaque or non-literal in that their overall meaning cannot be readily determined by the way in which the meaning of their elements are combined according to the rules of grammar. For example, if we directly parse syntactically and semantically the MWE *heavy metal*, we would arrive at the conclusion that it refers to a metal that is heavy in weight (like mercury, lead, etc.) If however we see this MWE in the context of the topic of music (e.g. in a newspaper review of a recently released album), we as human beings would not literally analyse the expression word by word, rule by rule but instead interpret the whole expression as a music genre, unless, of course, we did not know of the existence of a music genre called heavy metal. It is said that such opaque, non-literal MWEs are non-compositional as they do not necessarily follow the standard rules of meaning composition. By contrast, the expression *orange juice*, despite being lexicalised, is indeed compositional. That is because if we know what *orange* (the fruit) and what *juice* both refer to, we can easily understand, by applying syntactic and semantic rules, the expression *orange juice*, even in the (unlikely) case that we had never seen a glass of orange juice. There are also some contentious cases that could lie somewhere in between a non-compositional–compositional continuum. An example is *soft drink*. The literal interpretation alluding to the idea that a drink, which has to be in liquid form to be drinkable, can be soft is simply semantically anomalous as softness/hardness are characteristics normally associated with solid objects. It has a non-literal interpretation (that of a non-alcoholic drink) and because of that we could decide to classify *soft drink* as a non-compositional MWE. However, as just mentioned, a soft drink is still a drink, a liquid intended for human consumption, and even if we were not familiar with the metaphorical/non-literal use of *soft*, we would still be able to understand that *soft drink* must refer to a liquid of some unknown and/or strange description which is nevertheless intended for human consumption. Because *soft drink* has characteristics of both compositional and non-compositional MWEs, it is considered to be of medium compositionality.

An intuition that enters into play in this discussion is that the semantics of the individual constituent words forming an MWE behave very differently in the fully compositional case from the non-compositional case. In *orange juice* for example, *orange* refers to a fruit (or a flavour) whereas *juice* refers to a liquid extracted from a fruit. Whether *juice* or *orange* occur alone or forming collocations with other words (e.g. *apple juice*, *grape juice*, *orange pulp*, *orange pips*), they will more or less remain semantically related to *orange juice*. In *heavy metal* on the other hand, the usual meanings of *heavy* and *metal* are not semantically related to *heavy metal* in the music genre sense[2]. Since the notions of semantic relatedness and contexts enter into play in this intuition, it is interesting to consider whether the vector space model of lexical semantics can be used to try to measure the compositionality of multi-word expressions.

McCarthy et al. (2003) first exploited this intuition by measuring the similarity between a phrasal verb (a main verb and a preposition like *blow up*) and its main verb (*blow*) by comparing the words that are closely semantically related to each, and use this similarity as an indicator of compositionality. Katz and Giesbrecht (2006) also exploited the same intuition but they used an SVD-reduced word matrix. They first produced word vectors for the 20,000 most frequent words in a corpus as well as for the candidate MWEs. They used the 1,000 most frequent words in the corpus as features for those word vectors. Then, the word vectors were arranged as the rows for a word matrix which was then reduced to 100 columns by SVD. They then used the cosine measure between the SVD-reduced word vector for a given MWE and the sum of the individual MWE members' word vectors as a proxy to grade the compositionality of the MWE. If the cosine measure between the MWE vector and summed

[2]Notice however that *metal* can form other collocations related to the music genre sense such as *metal concert*, *metal fan*, etc.

word vectors of each of its members is close to zero, then it is assumed that the MWE is either non-compositional or has very low compositionality. Conversely, if the cosine measure is high, the MWE is deemed to be highly compositional.

In our method, we assume that each bigram in the dataset is made up of a node (or head-word) and a collocate (or modifier) (see Sec. 8.3, p. 167). Our method compares the semantic similarity between the node and the bigram and between the collocate and the bigram by computing a cosine similarity score (see Eq. 3.2.9 on page 72) between word vectors that represent the node, the collocate and the bigram. Three variations (configurations) of this basic method are explored, all defined in Section 8.3. A system based on this method was submitted to the Distributional Semantics and Compositionality (DISCO 2011) shared task (Biemann and Giesbrecht, 2011). Systems were given a set of bigrams that had different degrees of compositionality. Their compositionality or non-compositionality was assessed by native speakers based on "how literal" each bigram sounded to them. These judgements were then averaged and converted to a scale between 0 and 100, where 0 means non-compositional and 100 is fully compositional. Systems were evaluated on their ability to predict this average numeric score and to classify bigrams as either low compositional, medium compositional or highly compositional. The shared task organisers refer to this categorisation as the "coarse" prediction task. One of the variations of the system reported in this thesis achieved a fifth place (out of 15) in the overall numerical prediction task, whereas another of the variations also achieved fifth place in the overall coarse prediction task. However, it achieved first place in the coarse verb-object subset of bigrams.

3.2 The vector space model of information retrieval

As previously mentioned, the VSM has been applied to NLP tasks that fall well beyond its native information retrieval environment. As a lexical semantics tool it has been used to find word synonyms (Landauer and Dumais, 1997; Turney, 2001; Rapp, 2003) and other types of semantic and ontological relations between words (Sahlgren, 2006; Utsumi and Suzuki, 2006; Utsumi, 2010), as well as to solve analogy questions (Turney, 2006), to label semantic roles (Erk, 2007) and of course to disambiguate and discriminate word senses (Schütze, 1998; Purandare and Pedersen, 2004; de Marneffe and Dupont, 2004; Agirre and Stevenson, 2007; Navigli, 2009; Navigli and Crisafulli, 2010). In fact, the link between the VSM and lexical semantics is so strong that it is often seen as a plausible model of a major component of human lexical semantic learning (Bullinaria and Levy, 2007; Landauer, 2007). Other NLP tasks that have incorporated VSMs successfully include essay grading (Wolfe et al., 1998; Foltz et al., 1999), document segmentation by subtopics, question answering and call routing (Turney and Pantel, 2010), as well tasks more related to information retrieval such as document clustering and classification (Sebastiani, 2001; Manning et al., 2008) and cross-lingual information retrieval (Dumais et al., 1997).

This section introduces the vector space model within its own native context of information retrieval, paying attention to the properties and operations that will be used later when it is extended and adapted for lexical semantics (Section 3.3).

Information retrieval seeks to address the problem of accessing relevant information from digital but unstructured document collections (like the Web) via two computational tasks: document search and document classification. Document search (or Web search) identifies a set of documents (a subset from the document collection) that are relevant to a user query, whilst document classification consists in classifying the documents in the collection according to a set of predefined topics or themes. A variant of the document classification task is document clustering, which organises the documents in the collections into topical subgroups automatically without the need of supplying a list of predefined topics.

These tasks all involve comparing the contents of each document with something. In document search, the contents of each document in the collection are compared with a user query. In document classification, the contents of the documents are compared with the contents of documents that have already been classified manually and thus assigned to their most appropriate class. In document clustering, the contents of each document is directly compared with the contents of every other document in the collection in order to induce common topics and organise the documents according to these induced topics. Both document classification and document clustering involve comparing the contents of one document with another document. In document search, a user query is regarded to be an approximation to a document, albeit a very short one, and so it is usually called a **pseudodocument**. This makes the task of comparing one document with a user query essentially identical to comparing one document with another document, just as it is done in document classification and clustering. We can therefore state that a fundamental task in information retrieval is to compare the contents of one document against the contents of one or more other documents.

The "contents" of a document can be defined in terms of any of the units making up the documents in the collection. A common unit used in information retrieval is the word. And so the comparison of two documents often involves comparing whether the two documents share the same words or share words with similar meaning. Documents that significantly overlap lexically are deemed to be similar or relevant to each other. One way to compute this lexical overlap is by modelling documents as sets. If two documents δ_i and δ_j are respectively modelled as the sets $\mathcal{D}_i$ and $\mathcal{D}_j$, whose members are the word types τ that occur in each δ_i and δ_j respectively, we could *estimate* the degree to which the two documents are related by counting the number of terms in common, i.e. by counting the number elements in the intersection between $\mathcal{D}_i$ and $\mathcal{D}_j$:

$$relevance(\delta_i, \delta_j) = |\mathcal{D}_i \cap \mathcal{D}_j| \tag{3.2.1}$$

This set representation of a document is effectively a formalisation of the the **bag of words** concept or hypothesis, which models a whole document solely in terms of its word types and ignoring any other linguistic feature such as word order. This is also the formalisation of the

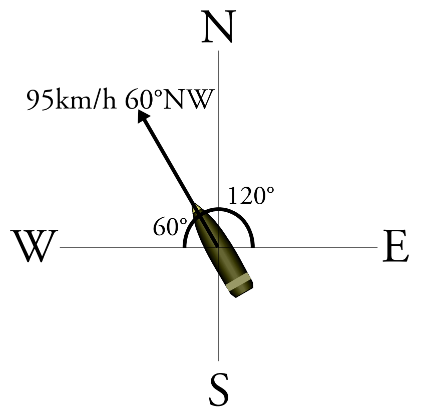


Figure 3.2.1: A graphic representation of a velocity vector

word overlap function used in Lesk's (1986) algorithm for word-sense disambiguation. In an information retrieval system, the super set $\mathcal{A}$ whose members are these set representations of documents would form the system's **index**, i.e. the record of what words occur in which documents.

A limitation of this simple set-based representation is that it does not take into account the frequency of the words occurring in each document. Intuitively, we can expect words related to the main topic of the document to occur more often than words unrelated to this main topic. For example, a document about linguistics would feature words such as *language*, *syntax*, *speaker*, *morpheme* relatively more frequently that medical terms such as *cancer*, *virus*, *infection*, *morphine*. Effectively, (3.2.1) gives the same weight to all words occurring in both documents, regardless of their frequency in each document. It is therefore common to weight the occurrence of a word by its frequency (i.e. **term frequency**). But since a set representation does not normally allow to represent members more than once, an alternative representation is usually employed: the Euclidean vector.

A **Euclidean vector** (or simply a **vector**) is a geometric quantity that has a **length** (also called **norm** or **magnitude**) that points towards a **direction**. Vectors find application in numerous scientific endeavours. For example, in Physics it is common to express velocities as vectors, where one could say that a projectile moves on a plane at a velocity of 95 km/h 60° north west. This vector is graphically represented in Figure 3.2.1. The speed quantity 95 km/h is the magnitude of that velocity, and 60° north west is its direction. A vector $\mathbf{v}$ can be expressed in terms of its dimensions as a **tuple** $[v_1, \ldots, v_n]$, i.e. an ordered list of elements. The tuple representation of our velocity example can be given in terms of its x and y components on the plane along which the projectile is travelling and would be

$[x, y] = [95\cos 120°, 95\sin 120°] = [-47.5, 82.272]$, with the first element being interpreted as the west-east speed component of the projectile (in this case a negative quantity because of its westward direction) and the second element being interpreted as the south-north speed component of the projectile (a positive quantity since the projectile is moving northwards). The units of the components still are kilometres per hour. More formally, it is said that the vector is decomposed with respect to the mutually perpendicular (orthogonal) reference axes (also called basis vectors) $\overrightarrow{SN}$ and $\overrightarrow{WE}$ in Figure 3.2.1.

It is this tuple expression that lends itself particularly well as a document representation in terms of the words it contains (Salton et al., 1975): if each of the vector's dimensions represents a word type that exists in the language, each element in the tuple can hold the frequency of its corresponding word type in the document. More formally and more generally, we can define a document vector **d** as follows:

Definition 3.2.1 (Document vector D-A-UR). Given a set of documents (i.e. a corpus) $\mathcal{C}$ and a vocabulary set $\mathcal{V}$ containing all of the word types occurring in $\mathcal{C}$, a document $\delta \in \mathcal{C}$ is represented by a vector $\mathbf{d} \in \mathbb{R}^m$ (where $m = |\mathcal{V}|$) with each dimension d_i relating to either the absence or occurrence of a word type $\tau_i \in \mathcal{V}$ in δ:

$$\mathbf{d} = \begin{array}{c} \tau_1 \\ \vdots \\ \tau_m \end{array} \begin{bmatrix} d_1 \\ \vdots \\ d_m \end{bmatrix} \tag{3.2.2}$$

where each d_i in **d** holds a real value that records whether τ_i occurs in δ and, if it does occur, also measures the importance of this occurrence through:

$$d_i = f_\delta(\tau_i) \tag{3.2.3}$$

where $f_\delta(\tau_i)$ is a **feature function** that maps word types τ_i occurring in a document δ to real values.

As it will be explained in Section 3.3, a document vector can be used to represent shorter text segments, such as paragraphs, sentences, etc. A document vector thus applied receives the name of segment vector. Those segment vectors are often used in WSX as proxies to represent word tokens (as in Levin et al., 2006), a usage that will be studied more carefully especially with regard to LSA where they are normally reduced by SVD. Over the course of this and the following chapter, we shall be discussing several competing kinds or configurations of token vectors. In order to avoid confusion, they can all be quickly referred to in Table 4.2.1 (p. 111). The vector configuration corresponding to the document or segment vector of Definition 3.2.1 is D-A-UR. "D" stands for direct as it is a direct or first-order representation.

“A” stands for the matrix from which this vector is derived, in this case the word-document or word-segment matrix **A** from Definition 3.2.2 below. “UR” stands for “unreduced”, that is a vector representation that has not been reduced by SVD or any other dimensionality reduction method.

In its most basic form, the feature function $f_\delta(\tau_i)$ is the **frequency** of the word type τ_i in the document δ, i.e. the count of occurrences of τ_i in δ:

$$f_\delta(\tau) = n_\delta(\tau) = \sum_{\forall \kappa \epsilon \delta} \begin{cases} 1 & \textit{if } \tau = \text{type}(\kappa) \\ 0 & \textit{otherwise} \end{cases} \tag{3.2.4}$$

where κ is each word token contained in document δ and type(κ) is a function or hash map that returns the word type τ of word token κ.

A value of 0 would indicate that τ_i does not occur in δ and therefore τ_i is considered to be not relevant for retrieving δ. If on the other hand, τ_i does occur in this document, then this value counts the number of times this word occurs in the document. The frequency feature function thus ascribes more relevance to words that appear more frequently.

Notice that vectors are flexible enough in that they allow us to formulate a representation equivalent to the set-based representation $\mathcal{D}$ we had originally by using a binary feature function such as:

$$f_\delta(\tau) = \begin{cases} 1 & \textit{if } n_\delta(\tau) > 0 \\ 0 & \textit{otherwise} \end{cases} \tag{3.2.5}$$

In our original set representation we compared two sets $\mathcal{D}_i$ and $\mathcal{D}_j$ through the cardinality of their intersection (3.2.1). The corresponding vector comparison of two vectors $\mathbf{d_i}$ and $\mathbf{d_j}$ is the sum of each element of one vector times the same element of the other vector, that is, the **dot product** or **inner product** of the two vectors:

$$\mathbf{d_i} \cdot \mathbf{d_j} = \sum_{k=1}^{m} d_{ik} d_{jk} \tag{3.2.6}$$

Observe that when dealing with binary features (3.2.5) the dot product is equivalent to the intersection cardinality (3.2.1). The dot product is considered to be a **vector similarity function** in information retrieval as well as in distributional lexical semantics as it measures the degree to which two vectors are similar to each other. A value of 0 indicates that the two vectors have no features (words) in common whereas a large positive value indicates a high degree of words shared.

Notice, however, that it is well possible to have two documents that are closely related but that vary in word count considerably, as for example in the case of a user query and a 100-page long document. In this situation, the smaller (pseudo)document will not feature relevant terms in high numbers. To alleviate this situation it is common to normalise vectors, that is, to have all of its dimensions divided by its norm (length):

$$\overline{\mathbf{d}} = \frac{\mathbf{d}}{\|\mathbf{d}\|_l} \tag{3.2.7}$$

In general, the norm of a vector is calculated by:

$$\|\mathbf{d}\|_l = \left(\sum_{i=1}^{m} |d_i|^l\right)^{1/l} \tag{3.2.8}$$

In the specific case when $l = 2$, this norm is the **Euclidean norm**[3] or **L2-norm**. An important result is that when a vector is normalised by the Euclidean norm (when it is L2-normalised), its length will always be 1, unless it is a zero-vector, in which case its length will be 0. Another popular norm is the **L1-norm**, i.e. where $l = 1$. A vector of frequencies that is L1-normalised can be interpreted as a probability distribution of the words in the document.

The dot product of L2-normalised vectors has an important geometric interpretation. If θ_{ij} is the angle formed between vectors $\mathbf{d}_\mathrm{i}$ and $\mathbf{d}_\mathrm{j}$, then the dot product of the L2-normalised versions of these vectors is the **cosine** of θ_{ij}:

$$\cos\theta_{ij} = \cos\left(\mathbf{d}_\mathrm{i}, \mathbf{d}_\mathrm{j}\right) = \frac{\mathbf{d}_\mathrm{i} \cdot \mathbf{d}_\mathrm{j}}{\|\mathbf{d}_\mathrm{i}\|\,\|\mathbf{d}_\mathrm{j}\|} \tag{3.2.9}$$

Intuitively, the cosine measures the degree of dimension overlap between the two vectors whilst, via the normalisation involved, reducing the importance of the actual raw, absolute frequencies. Geometrically, the cosine compares the direction of the two vectors whilst ignoring their individual lengths (norms) through the same normalisation. If the vectors point roughly in the same direction, the cosine value will be high, otherwise it will be low. The image of the cosine is $-1 \leq \cos\theta \leq 1$ for $-\infty < \theta < \infty$. Notice however that since we are dealing with non-negative values (word counts), the effective range of values will be $0 \leq \cos\theta \leq 1$ as the angle range will be $0 \leq \theta \leq \pi/2$. A value of 1 indicates complete similarity (even if the individual vector lengths are different) since $\theta = 0$, whilst a value of 0 indicates complete dissimilarity as the two vectors are orthogonal, i.e. the two vectors have no dimensions in common and so $\theta = \pi/2 = 90°$. Because of this intuitive interpretation and due to its closed range of values (unlike the dot product which is only bound for complete dissimilarity at 0), the cosine is a very important similarity function in information retrieval and distributional lexical semantics, where it is known as the **cosine similarity function** or **score** (often simply called **cosine measure**) and is normally denoted as $\cos(\mathbf{u}, \mathbf{v})$.

It has become such an established convention to work directly with cosine values that the actual angle value is rarely if ever computed. This established convention seems to stem from the intuitiveness associated with the interpretation of a similarity measure in a scale from 0 to 1, especially when contrasted with the perceived awkwardness in interpreting similarity as an inverse scale of $\pi/2$ to 0. Besides, one could argue that expressing an angle in terms of cosine

[3] In this thesis, the Euclidean norm of a vector, $\|\mathrm{v}\|_2$, will always be written simply as $\|\mathrm{v}\|$.

values, degrees or radians is a mere matter of choosing one's preferred measurement unit[4], and converting from cosine to degrees or radians engages us in a unit conversion exercise that sheds no additional information and in fact only negatively impacts the running time of an actual system and even risks loosing some precision by making additional unnecessary computations.

Just as there are vector similarity functions, there are also **vector distance functions**, the most common of which is the Euclidean distance. The **Euclidean distance** or **L2-distance** of two vectors $\mathbf{d_i}$ and $\mathbf{d_j}$ is defined as

$$\operatorname{dist}\left(\mathbf{d_i}, \mathbf{d_j}\right) = \left\|\mathbf{d_i} - \mathbf{d_j}\right\| = \sqrt{\left(d_{i1} - d_{j1}\right)^2 + \cdots + \left(d_{iN} - d_{jN}\right)^2} \tag{3.2.10}$$

The Euclidean distance can be seen as a **dissimilarity function** because the larger this amount is the more distant the two vectors are from each other. As the distance between the two vectors increases, the similarity of their corresponding documents decreases. Therefore, it is common to convert the Euclidean distance into a similarity function. One way is by using its inverse:

$$\operatorname{sim}_{\text{dist}}\left(\mathbf{d_i}, \mathbf{d_j}\right) = \frac{\lambda}{\left\|\mathbf{d_i} - \mathbf{d_j}\right\| + \lambda} \tag{3.2.11}$$

where λ is a manually chosen constant $0 < \lambda \leq 1$ used to avoid a division by zero when $\mathbf{d_i} = \mathbf{d_j}$.

If the two vectors are L2-normalised ($\overline{\mathbf{d}} = {}^{\mathbf{d}}/_{\|\mathbf{d}\|}$), then it is also possible to use the complement of the Euclidean distance as a similarity function:

$$\operatorname{sim}_{\text{dist}}\left(\overline{\mathbf{d_i}}, \overline{\mathbf{d_j}}\right) = 1 - \left\|\overline{\mathbf{d_i}} - \overline{\mathbf{d_j}}\right\| \tag{3.2.12}$$

In the cases of both (3.2.11) and (3.2.12) a value of 1 will indicate identity (complete similarity) whereas a value of 0 (or near zero for the inverse version) will indicate complete dissimilarity.

The raw frequency values that we have been using as features thus far are somewhat flawed as indicators of relevance, however. On the one hand, a word appearing three times in a document is not necessarily three times more relevant than a word appearing only once in the same document. And on the other hand, there are many common words in the language that will tend to have high frequencies across all documents, something that will make cosine values for unrelated documents be artificially high. The former issue is often mitigated by **dampening** raw frequencies (3.2.4) through square roots or logs (Manning and Schütze, 1999, p. 542-4):

[4]Notice though that conceptually the radian is not considered a measurements unit but a "pure number" computed from the ratio of two lengths. Similarly, we cannot argue that a cosine value is expressed in terms of some measurement unit. However, from a computational point of view, calculating the inverse of the cosine to obtain a degree or radian value is equivalent to converting quantities expressed in one traditional measurement unit to another in the sense that both cases involve an arithmetic manipulation of some sort.

$$f_\delta(\tau_i) = \sqrt{n_\delta(\tau_i)} \tag{3.2.13}$$

$$f_\delta(\tau_i) = \begin{cases} 1 + \log n_\delta(\tau_i) & n_\delta(\tau_i) \neq 0 \\ 0 & \textit{otherwise} \end{cases} \tag{3.2.14}$$

Whilst the latter issue is addressed by employing some weighting function like the well-known **inverse-document frequency** (IDF) function (Spärck Jones, 1972):

$$\text{idf}(\tau) = \log \frac{|\mathcal{C}|}{n_\tau(\mathcal{C})} \tag{3.2.15}$$

where $|\mathcal{C}|$ is the total number of documents in the corpus $\mathcal{C}$ and $n_\tau(\mathcal{C})$ is the number of documents in $\mathcal{C}$ that contain τ. That is:

$$n_\tau(\mathcal{C}) = \sum_{\forall \delta \in \mathcal{C}} \begin{cases} 1 & n_\delta(\tau) > 0 \\ 0 & \textit{otherwise} \end{cases} \tag{3.2.16}$$

IDF has been shown (Salton et al., 1975) to give more weight to words that only occur in a relatively small document subset and less weight to words uniformly distributed across documents. The intuition behind this is that words that only occur in a few documents discriminate topics better than words that appear in nearly every single document. Thus by giving more weight to words that concentrate in a few documents that focus around the same or related topics, it is possible to better rank documents by relevance and/or more accurately categorise them by topic.

A word frequency weighted by the word's IDF is normally called **TF-IDF**[5] and is a very popular feature function used in vectors representing documents:

$$f_\delta(\tau_i) = n_\delta(\tau_i) \times \text{idf}(\tau_i) \tag{3.2.17}$$

A common practice to reduce even further the effects of words such as articles and prepositions that are extremely common and extremely unlikely to be clues for topical relevance, is to omit them altogether from the vocabulary $\mathcal{V}$. This is equivalent to give them a weight of 0, making their feature function value to always be 0. This practice is called **stop word filtering** (see p. 32 for more details on function/stop words).

Recall that we previously defined the index of an information retrieval system as the super set $\mathcal{A}$ whose members are the $\mathcal{D}_i$ set representations corresponding to each document δ_i in the corpus $\mathcal{C}$. We can redefine such an **index** in terms of the $\mathbf{d_i}$ vectors corresponding to the same documents as follows:

[5]TF stands for "term frequency". In information retrieval terminology, singleton words (unigrams) are called terms. This thesis does not adopt this usage and understands *term* more traditionally as a lexical unit designating a specific specialised or technical concept. In this sense, a term could end up being an ngram of any length.

Definition 3.2.2 (Document matrix or index). Given a corpus $\mathcal{C}$ of n documents (i.e. $n = |\mathcal{C}|$) and given n vectors $\mathbf{d_i} \in \mathbb{R}^m$ each representing each document $\delta_i \in \mathcal{C}$, it is possible to arrange these vectors as the columns of the matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$:

$$\mathbf{A} = \begin{array}{c} \\ \mathbf{t_1} \\ \vdots \\ \mathbf{t_m} \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{d_1} & \cdots & \mathbf{d_n} \end{array} \\ \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \end{array} \tag{3.2.18}$$

Each row of this matrix is thus a word type vector $\mathbf{t_i}$ corresponding to a word type τ_i in the vocabulary $\mathcal{V}$ in $\mathcal{C}$ whilst its n columns correspond to the documents in $\mathcal{C}$. This matrix is often called a **word-document matrix** or a **term-document matrix** in information retrieval. If using basic raw counts as features (3.2.4), each cell a_{ij} in $\mathbf{A}$ counts the number of times word type τ_i occurs in document δ_j.

As previously mentioned, the VSM effectively reduces the representation of a document to a bag of words that makes no consideration to the finer linguistic rules governing the syntax and the grammatical relationships between the words in the document, or to its overall structure and organisation. Whilst this approach seems to be rather crude at the outset, it has achieved very good results in information retrieval and other natural language processing tasks. Intuitively, we can explain its success in expecting that the words chosen by an author writing a document are largely determined by the document's topic. The VSM's capability of determining the similarity of any two documents has been adapted to determine the similarity of any two words. This adaptation is the concern of the following section.

3.3 The VSM as a distributional lexical semantics model

The previous section described Salton's VSM in its most basic original form intended for information retrieval tasks such as document search, document classification and document clustering. We saw that the VSM is very well suited to measure similarities and dissimilarities between documents via their abstraction and representation as vectors in high-dimensional space. This section describes an adaptation of the VSM to lexical semantics. In particular, we shall see how we can also measure similarities and dissimilarities between words also represented in high-dimensional space.

The document matrix (index) $\mathbf{A}$ from Definition 3.2.2 describes two types of vectors, i.e. two vector spaces. In its columns, $\mathbf{A}$ defines a document vector space in which each vector $\mathbf{d_i}$ corresponds to each document $\delta_i \in \mathcal{C}$ (Definition 3.2.1). Each dimension in $\mathbf{d_i}$ corresponds to a word type $\tau_j \in \mathcal{V}$. The other vector space in $\mathbf{A}$ is formed in its rows. We shall call each row vector of $\mathbf{A}$ a **word type vector** or a **word vector** (also called a **term vector** in information retrieval) $\mathbf{t_j}$ corresponding to each word type $\tau_j \in \mathcal{V}$. Each dimension in $\mathbf{t_j}$ corresponds to a document $\delta_i \in \mathcal{C}$. We can formally define a word vector as follows:

Definition 3.3.1 (Word vector (document features)). Given a set of documents (i.e. a corpus) $\mathcal{C}$ and a vocabulary set $\mathcal{V}$ containing all of the word types occurring in $\mathcal{C}$, a word type $\tau \in \mathcal{V}$ is represented by a vector $\mathbf{t} \in \mathbb{R}^{1\times n}$ (where $n = |\mathcal{C}|$) with each dimension t_i relating to either the absence or occurrence of τ in the document $\delta_i \in \mathcal{C}$:

$$\mathbf{t} = \begin{array}{c} \begin{matrix} \delta_1 & \cdots & \delta_n \end{matrix} \\ \left[\begin{matrix} t_1 & \cdots & t_n \end{matrix} \right] \end{array} \tag{3.3.1}$$

where each t_i in $\mathbf{t}$ holds a real value that records whether τ occurs in δ_i and, if it does occur, also measures the importance of this occurrence through a feature function $t_i = f_{\delta_i}(\tau)$ (See (3.2.3)).

Section 3.2 showed how it is possible to estimate the similarity of two documents via a similarity measure, such as the cosine similarity score (3.2.9), that computes a similarity score on vectors that represent those two documents. Since we also have a vector representation of any word type in the corpus via Definition 3.3.1 there is no reason to expect that a vector similarity function on any two of these word vectors cannot be interpreted as a proxy of the actual semantic similarity of their two corresponding word types. And in fact, this is exactly what researchers in distributional semantics do when they apply the VSM to words (Turney and Pantel, 2010). By looking into the rows of $\mathbf{A}$ instead of its columns, we essentially shift the focus of the VSM from documents to word types.

Let us however examine more closely the intuitiveness and the linguistic interpretation of comparing two word type vectors. As previously discussed, a similarity function such as the cosine will measure the degree of overlap in dimensions between two vectors. Any given dimension in this vector space represents a document, so effectively the cosine score measures the degree to which two words tend to co-occur in the same documents. More generally, since documents can be simply seen as (large) units of text (see Sec. 2.2.4), i.e. syntagmas, we can state that the cosine score on two word vectors effectively measures the degree to which two words co-occur the same syntagmatic contexts. The document can be a rather large unit of text and while in a sense all of the words in a document are related because they are all contribute to the document's main topic(s), it can be difficult to affirm intuitively that any two isolated words in a document that occur at a considerable distance from each other, as in different paragraphs or perhaps different chapters, are clearly related. For this reason, distributional lexical semantics uses a modified version of the vector space model that works with alternative text units smaller than the document. This smaller alternative text unit can be a paragraph, a sentence or even a text segment of fixed length. In the distributional lexical semantics literature this unit is usually called a **context**. From the point of view of the linguistic analysis we discussed in the previous chapter, we could say that these contexts either match the notion of a syntagma (as when we work in terms of paragraphs, sentences or phrases) or approximate it (when we deal with text segments of fixed length). The experiments in this thesis work primarily with text segments of fixed length and so the terms *contexts*, *syntagmas*, and ***segments*** are technically interchangeable. However, as a convention the term *segment vector* will be used for vectors representing segments/contexts in Salton's VSM as well as in LSA whereas *context vector* will be used to designate vectors used representing segments/contexts in Word Space. There is two reasons for this. One reason being that Salton's VSM and LSA segment vectors represent discrete, non-overlapping text segments, whereas in Word Space context vectors represent word windows centred at a target word that may or may not overlap with other word windows (see p. 48). The other reason is that both types of vectors can be used to represent word tokens, but their representations are different. Word Space context vectors are specifically designed to represent tokens in context, but VSM/LSA segment vectors are designed to represent full contexts and not tokens directly. VSM/LSA segment vectors however have been used successfully to represent tokens in word-sense disambiguation experiments (e.g. Levin et al., 2006) by using a segment vector containing an instance (token) of a word as a proxy to represent that instance. In Section 4.3.1 (p. 123) we shall see that this subtlety makes a small but important difference between a VSM segment vector and a direct (first-order) Word Space context vector.

Given this modified VSM, we can view the cosine as a measure of the tendency between any two word types to co-occur within the same syntagma, i.e. a measure of their tendency to collocate (see Sec. 2.1.2). Recall that if two words co-occur within the same syntagma, they engage in a syntagmatic relationship (Sec. 2.2.2, p. 42). We can therefore state that the cosine similarity score (as well as other similarity scores like the dot product, the inverse and complement of the Euclidean distance, etc.) of the row vectors of **A** is a measure of syntagmatic similarity (Sahlgren, 2006). More generally, we can state that the vector space formed by the word type vectors (row vectors) in **A** can be interpreted, through a Saussurean lens, as a **syntagmatic vector space** (or simply a **syntagmatic space**). It is important to stress that being a syntagmatic space is not an inherent property of the word vectors of **A** *per se*, but rather a characteristic that only arises when any two such vectors are compared with each other (see Sec. 3.6).

There are however two major shortcoming with the vector-space representations of documents and word types given by **A**. One of them has to do with synonymy. If two documents (or two smaller contexts) express the same meaning or content but happen to use different words (e.g. by using synonyms or paraphrases), the cosine similarity between them will be very low. Likewise, synonymous words are often alternated in written works in order to avoid repetition and so if a model uses relatively short contexts like sentences, not many synonymous pairs will be captured within the same context. The other shortcoming is related to polysemy/homonymy. If two documents employ the same words but in different senses, then cosine similarities would be higher than they should. There are two main strategies that aim to mitigate this shortcoming: Latent Semantic Indexing/Analysis and Word Space.

Latent Semantic Indexing, LSI, (Deerwester et al., 1990) is a technique that aims to exploit the higher-order co-occurrence of words in order to implicitly detect synonymous pairs and thus find similarities of documents (or contexts) that represent similar contents even if they use different words. **Latent Semantic Analysis**, LSA, is basically the same process but usually applied to finding similarities of words rather than documents (or contexts) or when applied outside information retrieval. LSI/A works by projecting word or document vectors into an alternative vector space that represents latent semantic dimensions. Roughly speaking, co-occurring words end up being merged into the same dimensions whilst non co-occurring words remain separate. This alternative vector space is also usually of lower dimensionality than the original vector space. This alternative vector space is obtained by using Singular Value Decomposition (SVD). Apart from finding these latent semantic dimensions, SVD also has the effect of reducing noise by smoothing the term-document matrix since it contains many unreliable small counts. More details on LSA and SVD will be given in Section 3.4.

Word Space (Schütze, 1992) is an alternative way of representing words in vector space altogether. Instead of employing a word-document (or word-context) matrix as in Salton's original vector space model or as the input to LSA, word space instead directly constructs a square, symmetric word-word co-occurrence matrix in which each entry w_{ij} counts how many times w_i co-occurs with w_j within the same context. Context here is usually a sentence, a paragraph or a word-window of a predefined word length, usually centred at the word w_i. This word-word co-occurrence matrix is termed a **word matrix** in this thesis. Each row vector in the matrix (or column vector since the matrix is symmetric) $\mathbf{w_i}$ is a **word vector**. Whilst the word matrix can be exploited as is, it can be optionally transformed using SVD. Word Space is discussed in detail in Section 3.5.

3.4 Latent Semantic Analysis

Introduced by Furnas et al. (1988) and Deerwester et al. (1990), Latent Semantic Analysis (LSA) started life as **Latent Semantic Indexing (LSI)**, an information retrieval method that sought to address the shortcomings caused by synonymy and polysemy/homonymy that we mentioned towards the end of the previous section. Synonymy (and other semantic relations such as hypernymy, hyponymy, co-hyponymy, etc.) can reduce the recall of information retrieval systems when users use words in their queries that are different to those used in the relevant documents themselves, even if the words in the queries are relevant to the topic of documents users wish to retrieve. Polysemy and homonymy on the other hand cause a reduction in precision of information retrieval systems by making them retrieve documents that use one or more of the words in the user query, but in a different and potentially unrelated sense to that intended by the user.

Latent Semantic Indexing works by reducing the dimensionality of the index, that is, the word-document matrix **A** introduced in Definition 3.2.2, via truncated Singular Value Decomposition (SVD). The application of truncated SVD to the index "uncovers" hidden semantic relationships between documents via the words they contain, as well as simultaneously "uncovering" hidden semantic relationships between words based on the documents they occur in. The nature of these "uncovered hidden semantic relationships" will become more evident and transparent after explaining the mathematical framework behind LSA (Section 3.4.1). And as we shall see in Section 3.4.4, they have the effect of inducing semantic similarities between words that do not necessarily co-occur directly but have higher-order co-occurrence. These semantic relationships also have the effect of "modulating the correct sense" of a word in terms of the words it directly co-occurs with in a given syntagma or context, becoming a form of "polysemy detection". In addition, it is expected that the dimensionality reduction involved in SVD acts as a noise reduction technique that eliminates chance or spurious co-occurrences of words.

Whilst LSI is primarily an information retrieval tool, it has been applied to many other computational tasks that benefit from its lexical semantic properties. In the literature, when LSI is applied outside information retrieval, it receives the moniker **Latent Semantic Analysis** (**LSA**). Just like Salton's VSM can be used for information retrieval or general lexical semantics problems depending on whether we decide to look at the columns of the VSM matrix (documents, information retrieval) or the rows of the same matrix (words, lexical semantics), we shall see that the difference between LSI and LSA also boils down very much to whether we decide to look at the columns (documents, information retrieval, LSI) or the rows (words, lexical semantics, LSA) of versions of the SVD-reduced matrix. This is a very subtle distinction and since the focus of this thesis is on lexical semantic tasks such as word-sense disambiguation and discrimination, we shall use the term LSA more extensively and only refer to LSI in the context of information retrieval.

3.4.1 SVD: the mathematical foundation of LSA

At the heart of Latent Semantic Analysis (LSA) lies **Singular Value Decomposition** (**SVD**) (Golub and Van Loan, 1989; Berry, 1992), a mathematical technique used to decompose an arbitrary matrix of real values into the product of three matrices. SVD is based on the eigendecomposition of diagonalisable matrices (Anton and Rorres, 2000, Sec. 7.3).

Theorem 3.4.1 (Eigendecomposition or spectral decomposition of a diagonalisable matrix). *If $\mathbf{M} \in \mathbb{R}^{m\times m}$ is a diagonalisable matrix (i.e. if there exists a matrix $\mathbf{R}$ such that $\mathbf{R}^{-1}\mathbf{M}\mathbf{R}$ is a diagonal matrix), then $\mathbf{M}$ can be factorised in terms of its eigenvalues and eigenvectors as*

$$\mathbf{M} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1} \tag{3.4.1}$$

where the columns of $\mathbf{Q} \in \mathbb{R}^{m\times m}$ are the eigenvectors of $\mathbf{M}$ and $\boldsymbol{\Lambda}$ is a diagonal matrix whose *diagonal elements are the corresponding eigenvalues of* $\mathbf{M}$. *In addition, if* $\mathbf{M}$ *is symmetric, then* $\mathbf{Q}$ *is an orthogonal matrix.*

This eigendecomposition can only be applied to square and diagonalisable matrices. The word-document or word-context matrix $\mathbf{A}$ is unlikely to be square and even more unlikely to be diagonalisable. SVD can be applied to such arbitrary rectangular matrices because it internally converts the matrix into a symmetric matrix (i.e. a matrix $\mathbf{M}$ such that $\mathbf{M} = \mathbf{M}^T$) by squaring it (i.e. by computing $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$) and because all symmetric matrices are diagonalisable.

Theorem 3.4.2 (Singular Value Decomposition). *An arbitrary rectangular matrix* $\mathbf{M} \in \mathbb{R}^{m\times n}$, *with* $m \geq n$ *and rank*$(\mathbf{M}) = r$, *can be factorised as*

$$\mathbf{M} = \mathbf{U\Sigma V}^T \tag{3.4.2}$$

where:

$\mathbf{U} \in \mathbb{R}^{m\times r}$ *is an orthogonal matrix whose column vectors are the left-singular vectors of* $\mathbf{M}$, *i.e. these vectors are the eigenvectors of* $\mathbf{MM}^T$. $\mathbf{U}$ *is orthogonal because* $\mathbf{MM}^T$ *is symmetric.*

$\mathbf{\Sigma} \in \mathbb{R}^{r\times r}$ *is a diagonal matrix whose values along the diagonal* $(\sigma_{11}, \sigma_{22} \ldots, \sigma_{rr})$ *are the singular values of* $\mathbf{M}$. *These non-negative values are the square roots of the non-zero eigenvalues of both* $\mathbf{MM}^T$ *and* $\mathbf{M}^T\mathbf{M}$. *Also, these values are sorted in descending order, i.e.* $\sigma_{11} \geq \sigma_{22} \geq \ldots \geq \sigma_{rr} \geq 0$.

$\mathbf{V} \in \mathbb{R}^{n\times r}$ *is an orthogonal matrix whose columns are the right-singular vectors of* $\mathbf{M}$, *i.e. these vectors are the eigenvectors of* $\mathbf{M}^T\mathbf{M}$. $\mathbf{V}$ *is orthogonal because* $\mathbf{M}^T\mathbf{M}$ *is symmetric.*

The rank r of $\mathbf{M}$ *is the number of linearly independent rows and columns in* $\mathbf{M}$, *which is the same as the number of eigenvalues in the diagonal in* $\mathbf{\Sigma}$. *Notice that* $r \leq \min(m, n)$.

In LSA though the full SVD from above is not employed and instead a truncated, low-rank approximation is used:

Theorem 3.4.3 (Low rank approximation). *If* $\mathbf{U\Sigma V}^T$ *is the SVD of* $\mathbf{M}$ *whose rank is r, then*

$$\widehat{\mathbf{M}} = \mathbf{M}_\mathrm{k} = \mathbf{U}_\mathrm{k}\mathbf{\Sigma}_\mathrm{k}\mathbf{V}_\mathrm{k}{}^T \tag{3.4.3}$$

is an optimal approximation of rank k (with $k \ll r$*) of* $\mathbf{M}$ *where*

$\mathbf{U}_\mathrm{k}$ *is the first k columns of* $\mathbf{U}$.

$\mathbf{\Sigma}_\mathrm{k}$ *is a diagonal matrix with the top k highest values from* $\mathbf{\Sigma}$ *along its diagonal.*

$\mathbf{V}_\mathrm{k}$ *is the first k columns of* $\mathbf{V}$.

$\widehat{\mathbf{M}}$ (i.e. $\mathbf{M_k}$) is the best approximation of rank k of $\mathbf{M}$ with minimum distance to the original $\mathbf{M}$. This minimum distance is taken to be the sum of squares of corresponding matrix positions, i.e. the Frobenius norm of the matrix difference $\mathbf{M} - \mathbf{M_k}$ (Manning et al., 2008, pp. 376-8). That is:

$$\min_{\mathbf{Z}|rank(\mathbf{Z})=k} \|\mathbf{M} - \mathbf{Z}\|_F = \|\mathbf{M} - \mathbf{M_k}\|_F = \left(\sum_{i=k+1}^{r} \sigma_i^2 \right)^{1/2} \tag{3.4.4}$$

Actual software implementations of SVD tend to compute this decomposition incrementally from 1 dimension till k. This allows these implementations to decompose very large matrices into a relatively small number of dimensions k (usually $50 \leq k \leq 500$ approx) efficiently. So rarely the full decomposition from (3.4.2) is ever computed.

In LSA, the truncated matrices $\mathbf{U_k}$, $\mathbf{\Sigma_k}$ and $\mathbf{V_k}$ are used to transform document/segment vectors or word type vectors of dimensionality m or n, respectively into new co-ordinates of dimensionality k. Once vector objects have been transformed into the new reduced dimensionality, comparisons between them can be performed. Just like in the original vector space model described in Sections 3.2 and 3.3, comparisons between any two reduced vectors can done by using cosine similarity, Euclidean distance or other measure. Here we shall consider cosine similarity. LSA permits comparing any two word type vectors, any two segment vectors and even a word type vector and a segment vector.

Objects in this new, lower dimensionality are expected to have been reduced in "noise" since variability generated by chance or spurious word co-occurrences gets eliminated. In addition, this truncation seems to capture an underlying semantic structure between word types and segments. Basically, word type vectors of semantically similar words come nearer each other in the reduced vector space even when they do not co-occur in the same documents. As well, vectors representing topically related segments (or documents) become closer in this reduced space, even if they do not share words in common (Berry et al., 1995; Martin and Berry, 2007). These semantic properties are explored in Section 3.4.4 whilst details on the projections themselves are given in the next section.

Before moving on, notice that LSA relies on one parameter, k, which specifies the number of dimensions to keep. Unfortunately, finding the optimal value for k has to be determined empirically as it depends on the corpus used and the application. However, most applications seem to use values in the low hundreds (i.e. $100 \leq k \leq 500$).

3.4.2 Projecting word and segment vectors into the reduced space

Before we can perform cosine similarity comparisons, we need to give details on how segment vectors and word type vectors are to be transformed or projected into their reduced dimensionality versions using the matrices produced by SVD. In Emms and Maldonado-Guerra

(2013) we found that there are two contending transformations used in the literature, which we called $R_1(\mathbf{v})$ and $R_2(\mathbf{v})$, respectively. They will both be covered here and a brief survey of their use in the literature is given in Section 3.4.3.

Definition 3.4.1 (Segment vector projections D-A-R1 and D-A-R2)**.** Given a segment δ represented by column vector $\mathbf{d} \in \mathbb{R}^m$, we can obtain its SVD-reduced version $\widehat{\mathbf{d}}$ through one of the two following projections:

$$\widehat{\mathbf{d}} = R_1(\mathbf{d}) = \mathbf{U_k}^T\mathbf{d} \tag{3.4.5}$$

$$\widehat{\mathbf{d}} = R_2(\mathbf{d}) = \mathbf{\Sigma_k^{-1}}\mathbf{U_k}^T\mathbf{d} = \mathbf{\Sigma_k^{-1}}R_1(\mathbf{d}) \tag{3.4.6}$$

If $\mathbf{d}$ is actually a column vector in $\mathbf{A}$ and therefore is also represented by row vector $\mathbf{v} \in \mathbb{R}^{1\times k}$ in $\mathbf{V_k}$, then these projections become:

$$\widehat{\mathbf{d}} = R_1(\mathbf{d}) = \mathbf{\Sigma_k}\mathbf{v}^T \tag{3.4.7}$$

$$\widehat{\mathbf{d}} = R_2(\mathbf{d}) = \mathbf{v}^T \tag{3.4.8}$$

Notice that with any of these projections $\widehat{\mathbf{d}} \in \mathbb{R}^k$, i.e. we obtain column vectors of dimensionality k.

These are the D-A-R1 and D-A-R2 token vector representation configurations presented in Table 4.2.1 (p. 111).

The equivalence between the two R_1 projections (3.4.5) and (3.4.7) can be shown by an algebraic re-arrangement of the original SVD definition in Theorem 3.4.2:

$$\begin{aligned} \mathbf{A} &= \mathbf{U\Sigma V}^T \\ \mathbf{U}^T\mathbf{A} &= \mathbf{U}^T\mathbf{U\Sigma V}^T \end{aligned}$$

Since $\mathbf{U}$ is orthogonal, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$:

$$\begin{aligned} \mathbf{U}^T\mathbf{A} &= \mathbf{\Sigma V}^T \\ \mathbf{U_k}^T\mathbf{d} &= \mathbf{\Sigma_k}\mathbf{v}^T \end{aligned}$$

A similar algebraic re-arrangement of the same SVD definition can be performed to show the equivalence between the R_2 projections:

$$\begin{aligned}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{A} &= \mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ \mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{A} &= \mathbf{\Sigma}^{-1}\mathbf{\Sigma}\mathbf{V}^T\end{aligned}$$

Since $\mathbf{\Sigma}$ is a diagonal matrix, $\mathbf{\Sigma}^{-1}\mathbf{\Sigma} = \mathbf{I}$:

$$\begin{aligned}\mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{A} &= \mathbf{V}^T \\ \mathbf{\Sigma}_\mathbf{k}^{-1}\mathbf{U_k}^T\mathbf{d} &= \mathbf{v}^T\end{aligned}$$

Definition 3.4.2 (Word type vector projections)**.** Given a word type τ represented by row vector $\mathbf{t} \in \mathbb{R}^{1\times n}$, we can obtain its SVD-reduced version $\widehat{\mathbf{t}}$ via one of the following two projections:

$$\widehat{\mathbf{t}} = R_1(\mathbf{t}) = \mathbf{t}\mathbf{V_k} \tag{3.4.9}$$

$$\widehat{\mathbf{t}} = R_2(\mathbf{t}) = \mathbf{t}\mathbf{V_k}\mathbf{\Sigma}_\mathbf{k}^{-1} = R_1(\mathbf{t})\mathbf{\Sigma}_\mathbf{k}^{-1} \tag{3.4.10}$$

If $\mathbf{t}$ is actually a row vector in $\mathbf{A}$ and therefore is also represented in $\mathbf{U_k}$ by row vector $\mathbf{u}$, then these projections become:

$$\widehat{\mathbf{t}} = R_1(\mathbf{t}) = \mathbf{u}\mathbf{\Sigma_k} \tag{3.4.11}$$

$$\widehat{\mathbf{t}} = R_2(\mathbf{t}) = \mathbf{u} \tag{3.4.12}$$

Notice that with any of these projections $\widehat{\mathbf{t}} \in \mathbb{R}^{1\times k}$, i.e. we obtain row vectors of dimensionality k.

Again, we can use algebraic re-arrangements of the original SVD definition to show the equivalence of the two versions of each projection in Definition 3.4.2. For R_1 projections: $\mathbf{AV} = \mathbf{U\Sigma V}^T\mathbf{V} = \mathbf{U\Sigma}$. Therefore, $\mathbf{tV_k} = \mathbf{u\Sigma_k}$. For R_2 projections: $\mathbf{AV\Sigma}^{-1} = \mathbf{U\Sigma V}^T\mathbf{V\Sigma}^{-1} = \mathbf{AV\Sigma}^{-1} = \mathbf{U\Sigma\Sigma}^{-1} = \mathbf{U}$. Therefore, $\mathbf{tV_k\Sigma_k}^{-1} = \mathbf{u}$.

We can also show intuitively that the R_1 projections are the more appropriate ones for the type of comparisons that we want to perform, i.e. between any two (reduced) word type vectors or between any two (reduced) document/segment vectors. The cosine similarity can be seen as a dot product between two vectors that have been L2-normalised. If we ignore the normalisation carried out by the cosine, we can consider the dot product to also be a similarity measure. If we wanted to compare every word type vector with each other in $\mathbf{A}$ using the dot product we could simply compute $\mathbf{A}\mathbf{A}^T$, which would produce an $m \times m$ matrix in which every cell x_{ij} will contain the dot product value of word type vector $\mathbf{t_i}$ and word type vector $\mathbf{t_j}$. If instead we wanted to obtain the dot products between all word type vectors but in the reduced space, we would compute $\mathbf{A_k}\mathbf{A_k}^T$. By looking at the original SVD definition again, this matrix multiplication yields (see Martin and Berry, 2007):

$$\mathbf{A_k}\mathbf{A_k}^T = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^T \left(\mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^T\right)^T = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^T\mathbf{V_k}\mathbf{\Sigma_k}\mathbf{U_k}^T = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{\Sigma_k}\mathbf{U_k}^T \qquad (3.4.13)$$

This result tells us that the dot products between every two row vectors from $\mathbf{A_k}$ is the same as the dot products between every two row vectors from $\mathbf{U_k}\mathbf{\Sigma_k}$, which is the matrix version of the R_1 formulation in (3.4.11).

The same analysis can be performed to show that the R_1 projections of document/segment vectors are more intuitively motivated than the R_2 projections. We can compute all pairwise comparisons between each pair of document/segment vectors in the reduced space by computing $\mathbf{A_k}^T\mathbf{A_k}$, which following the original SVD definition, is equivalent to:

$$\mathbf{A_k}^T\mathbf{A_k} = \left(\mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^T\right)^T\mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^T = \mathbf{V_k}\mathbf{\Sigma_k}\mathbf{U_k}^T\mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^T = \mathbf{V_k}\mathbf{\Sigma_k}\mathbf{\Sigma_k}\mathbf{V_k}^T \qquad (3.4.14)$$

Similar to what we had before, $\mathbf{\Sigma_k}\mathbf{V_k}^T$ is very much the matrix version of the R_1 formulation in (3.4.7).

A further insight into the better suitability of R_1 over R_2 can be given. If $k = r = rank(\mathbf{A})$, i.e. if we perform SVD without removing any dimensions, then $\mathbf{A}\mathbf{A}^T = \mathbf{A_k}\mathbf{A_k}^T$ and $\mathbf{A}^T\mathbf{A} = \mathbf{A_k}^T\mathbf{A_k}$; that is, the dot product of any two word type vector pair or segment pair in the non-decomposed space will be identical to that of the decomposed space. All distances and angles are preserved after SVD has been applied to $\mathbf{A}$. And as it was shown in (3.4.13) and (3.4.14), these operations correspond to computing dot products of any two R_1-projected vectors. R_2 on the other hand, can be seen as a scaling of R_1 by $\mathbf{\Sigma_k^{-1}}$. Because of this, there is no reason to expect that R_2 will preserve distances and angles of vectors present in $\mathbf{A}$ after decomposition.

There is an intuitive relationship between the R_1 and R_2 projections however that is worth mentioning. $R_2(\mathbf{t})$ in particular can be seen as the projection of an isolated token of type τ in the lower dimensionality space. That is, $R_2(\mathbf{t})$ represents an occurrence of τ in isolation, i.e. out of context. An isolated token of τ_i can be represented as a segment vector $\mathbf{d}$ in which there is a 1 in the dimension t_i corresponding to τ_i and zeroes everywhere else, i.e.:

$$\mathbf{d}^T = \begin{array}{c} \begin{array}{ccccc} t_0 & \cdots & t_i & \cdots & t_n \end{array} \\ \left[\begin{array}{ccccc} 0 & \cdots & 1 & \cdots & 0 \end{array}\right] \end{array} \qquad (3.4.15)$$

And more importantly, it can be shown that the R_1 projection of this segment vector is equivalent to the R_2 projection of type vector $\mathbf{t_i}$. This is stated more formally as follows.

Theorem 3.4.4 (R_2 as the projection of an isolated token). *If* $\mathbf{d}$ *is a (column) vector of dimensions* $t_1, \ldots, t_m$ *for which* $t_i = 1$ *and* $t_j = 0$ *for every* $0 \leq j \leq m$ *such that* $j \neq i$ *and* $\mathbf{t_i}$ *is a (row) vector in the unreduced matrix* $\mathbf{A}$*, then*

$$R_1(\mathbf{d}) = R_2(\mathbf{t_i})^T \tag{3.4.16}$$

Proof. We know from Definition 3.4.1 that $R_1(\mathbf{d}) = \mathbf{U_k}^T\mathbf{d}$ (3.4.5). The matrix multiplication operation in this equation can be seen as arranging the results of dot products between the row vectors in $\mathbf{U_k}^T$ and $\mathbf{d}$ into a column vector $R_1(\mathbf{d})$. Since all dimensions in $\mathbf{d}$ are 0 with the exception of t_i, then only the values in the corresponding column from $\mathbf{U_k}^T$ will be carried over to $R_1(\mathbf{d})$. Compare the following equation with (3.4.12):

$$R_1(\mathbf{d}) = \mathbf{U_k}^T\mathbf{d} = \begin{array}{c} \begin{array}{ccccc} \mathbf{u_1}^T & \cdots & \mathbf{u_i}^T & \cdots & \mathbf{u_m}^T \end{array} \\ \begin{bmatrix} u_{11} & \cdots & u_{1i} & \cdots & u_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ u_{k1} & \cdots & u_{ki} & \cdots & u_{km} \end{bmatrix} \end{array} \times \begin{array}{c} d_1 \\ \vdots \\ d_i \\ \vdots \\ d_m \end{array} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{array}{c} \mathbf{u_i}^T \\ \begin{bmatrix} u_{1i} \\ \vdots \\ u_{ki} \end{bmatrix} \end{array} = \mathbf{u_i}^T = R_2(\mathbf{t_i})^T \tag{3.4.17}$$

□

There is an analogous relationship between the R_1 projection of a type vector and the R_2 projection of a segment vector. It has a less straight-forward interpretation, but it could be understood as relating to a word that only appears in one single segment.

Theorem 3.4.5 (R_2 as the projection of a single-segment type). *If* $\mathbf{t}$ *is a (row) vector of dimensions* $d_1, \ldots, d_n$ *for which* $d_i = 1$ *and* $d_j = 0$ *for every* $0 \leq j \leq m$ *such that* $j \neq i$ *and* $\mathbf{d_i}$ *is a (column) vector in the unreduced matrix* $\mathbf{A}$*, then*

$$R_1(\mathbf{t}) = R_2(\mathbf{d_i})^T \tag{3.4.18}$$

Proof. The proof is similar to that of Theorem 3.4.4. Recall that $R_1(\mathbf{t}) = \mathbf{tV_k}$ (3.4.9). Since only the value d_i in $\mathbf{t}$ is 1 and all others are 0, only the i-th row of $\mathbf{V_k}$, which corresponds to vector $\mathbf{v_i}$, will be preserved after the multiplication. In sum, $R_1(\mathbf{t}) = \mathbf{tV_k} = \mathbf{v_i} = R_2(\mathbf{d_i})^T$. See (3.4.8). □

3.4.3 The R_1 and R_2 projections in the literature

In Emms and Maldonado-Guerra (2013) we argued that the difference between the R_1 and R_2 SVD projections has been overlooked in the literature. This section considers the work of a number of authors, arguing that some are adhering to the R_1 formulation and some to the R_2 formulation.

The R_2 formulation is presented in many, fairly widely cited, publications. For example, in the notation of Rosario (2000), the reduced rank SVD of the word-document matrix is $\mathbf{T}_\mathrm{k}\mathbf{S}_\mathrm{k}(\mathbf{D}_\mathrm{k})^T$, with $\mathbf{T}_\mathrm{k}$, $\mathbf{S}_\mathrm{k}$ and $\mathbf{D}_\mathrm{k}$ used in place of $\mathbf{U}_\mathrm{k}$, $\boldsymbol{\Sigma}_\mathrm{k}$ and $\mathbf{V}_\mathrm{k}$, respectively. This is described as providing a representation in an alternative space whereby "the matrices **T** and **D** represent terms [word types] and documents in this new space" (p. 3) and additionally the representation of a query is given (p. 4) as $\mathbf{q}^T\mathbf{T}_\mathrm{k}\mathbf{S}_\mathrm{k}{}^{-1}$, which matches, modulo notational switches, the R_2 formulation given in (3.4.6). Similarly, in the notation of Zelikovitz and Hirsh (2001), the SVD of a word-document matrix is $\mathbf{TSD}^T$: "a query is represented in the same new small space that the document collection is represented in. This is done by multiplying the transpose of the term vector of the query with matrices **T** and $\mathbf{S}^{-1}$" (p. 114). Again modulo notational switches, this is the R_2 formulation of (3.4.6). And finally, in the notation of Gong and Liu (2001), the SVD of a word-segment matrix is $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, and the SVD is described as defining a mapping which "projects each column vector i in matrix **A** [...] to column vector $\psi_i = \begin{bmatrix} v_{i1} & v_{i2} & \cdots & v_{ir} \end{bmatrix}^T$ of matrix $\mathbf{V}^T$" (p. 21). Thus the i-th column of **A** is represented by the i-th row of **V**, which is the R_2 formulation given in (3.4.8).

On the other hand, the R_1 formulation of LSA is also presented in other many, fairly widely cited, publications. For example, in the notation of Bartell et al. (1992) the reduced rank SVD of a word-document matrix is $\mathbf{U}_\mathrm{k}\mathbf{L}_\mathrm{k}\mathbf{A}_\mathrm{k}{}^T$, and their definitions of document and query representations are "row i of $\mathbf{A}_\mathrm{k}\mathbf{L}_\mathrm{k}$ gives the representation of document i in k-space. [...] Let the query be encoded as a row vector **q** in $\mathbb{R}^t$. Then the query in k-space would be $\mathbf{qU}_\mathrm{k}$" (p. 162). These coincide, modulo notational differences, with the R_1 formulations of (3.4.7) and (3.4.5), respectively. Similarly, in the notation of Papadimitriou et al. (1998), the reduced rank SVD of a word-document matrix is $\mathbf{U}_\mathrm{k}\mathbf{D}_\mathrm{k}\mathbf{V}_\mathrm{k}{}^T$. Then concerning document representation they say that "[t]he rows of $\mathbf{V}_\mathrm{k}\mathbf{D}_\mathrm{k}$ above are then used to represent the documents. In other words, the column vectors of **A** (documents) are projected to the k-dimensional space spanned by the column vectors of $\mathbf{U}_\mathrm{k}$" (p. 220). This coincides, modulo notation, with the R_1 formulations in (3.4.7) and (3.4.5). Finally, in the notation of Kontostathis and Pottenger (2006), the reduced-rank SVD of a word-document matrix is $\mathbf{T}_\mathrm{k}\mathbf{S}_\mathrm{k}(\mathbf{D}_\mathrm{k})^T$. They represent queries "in the reduced space by $\mathbf{T}_\mathrm{k}{}^T\mathbf{q}$. [...] Queries are compared to the reduced document vectors, scaled by the singular values $(\mathbf{S}_\mathrm{k}\mathbf{D}_\mathrm{k}{}^T)$" (p. 3). These column vector formulations would be a row vector formulation $\mathbf{qT}_\mathrm{k}$ and $\mathbf{D}_\mathrm{k}\mathbf{S}_\mathrm{k}$, which, modulo notational differences are the R_1 formulations of (3.4.5) and (3.4.7), respectively. On the basis of these works, there would appear to be an R_1-vs-R_2 ambiguity in the formulation of LSA, possibly a fairly widespread one.

Table 3.4.1: Word type-segment matrix **A**

A	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1			1					
interface	1		1						
computer	1	1							
user		1	1		1				
system		1	1	2					
response		1			1				
time		1			1				
EPS			1	1					
survey		1							1
trees						1	1	1	
graph							1	1	1
minors								1	1

3.4.4 Semantic properties of LSA

It has been mentioned that the truncated SVD operation involved in LSA has the effects of reducing the noise in the reduced-dimensionality vector space as well as uncovering a hidden semantic structure that identifies words that are paradigmatically related (synonyms, co-hyponyms, etc.) and that is sensitive to the polysemy of words. This section explores these semantic properties of LSA in more detail. This exploration will be done intuitively using a toy corpus used for illustration purposes in publications by the creators of LSA such as Furnas et al. (1988) and Deerwester et al. (1990). Despite being such a small sample, this toy corpus is useful for visualising the operations involved in LSA easily and transparently. It also has the advantage of being used in several other works such as Kontostathis and Pottenger (2006) to illustrate LSA and SVD concepts, so it can be useful to also compare explanations and definitions between works. The toy corpus is a selection of article titles on human computer interaction and graph theory. They are used here to demonstrate intuitively the aforementioned semantic properties of LSA. The toy corpus of article titles is as follows:

- c1: Human machine interface for ABC computer applications
- c2: A survey of user opinion of computer system response time
- c3: The EPS user interface management system
- c4: System and human system engineering testing of EPS
- c5: Relation of user perceived response time to error measurement
- m1: The generation of random, binary, ordered trees
- m2: The intersection graph of paths in trees

Table 3.4.2: SVD matrices of **A** truncated to $k = 2$ dimensions

$\mathbf{U_k}$	k1	k2
human	-0.22	-0.11
interface	-0.20	-0.07
computer	-0.24	0.04
user	-0.40	0.06
system	-0.64	-0.17
response	-0.27	0.11
time	-0.27	0.11
EPS	-0.30	-0.14
survey	-0.21	0.27
trees	-0.01	0.49
graph	-0.04	0.62
minors	-0.03	0.45

$\mathbf{\Sigma_k}$	k1	k2
k1	3.34	0.00
k2	0.00	2.54

$\mathbf{V_k}$	k1	k2
c1	-0.20	-0.06
c2	-0.61	0.17
c3	-0.46	-0.13
c4	-0.54	-0.23
c5	-0.28	0.11
m1	0.00	0.19
m2	-0.01	0.44
m3	-0.02	0.62
m4	-0.08	0.53

- m3: Graph minors IV: Widths of trees and well-quasi-ordering
- m4: Graph minors: a survey

Article titles on human-computer interaction are c1-c5 and those on graph theory are m1-m4. For our purposes one text segment will be a whole article title. In order to build the actual word-segment matrix, we need to do a vocabulary (feature) selection. The selected vocabulary from this dataset is: {*human*, *interface*, *computer*, *user*, *system*, *response*, *time*, *EPS*, *survey*, *trees*, *graph*, *minor*}. All other word types are ignored. This allows us to construct a type-segment matrix $\mathbf{A} \in \mathbb{R}^{12\times 9}$ which can be seen in Table 3.4.1. Table 3.4.2 shows the SVD matrices of **A** reduced to $k = 2$ dimensions.

3.4.4.1 Semantic relations

As explained in Section 3.3, the vector space introduced by Salton's original VSM, exemplified by the type-segment/document matrix **A**, is a syntagmatic space. That is, it is able to give an indication on whether two words tend to co-occur or not in a corpus through the cosine measure of the vectors representing these words in the vector space. As we shall see shortly, type vectors projected into the SVD-reduced space of **A** forms a combination of a syntagmatic and paradigmatic space, capable of predicting, via the same cosine operation, whether two word types are both syntagmatically and paradigmatically related. A few examples from the toy corpus will help shed some light on this. Some of the observations made here are based and expanded upon those made by Kontostathis and Pottenger (2006).

Table 3.4.3 shows pairwise cosines between all type vectors from the toy corpus. The table is divided in two sections: the right-upper part (white cells) shows the pairwise cosines between word type vectors from the unreduced matrix **A** of Table 3.4.1, whilst the left-lower part

Table 3.4.3: Pairwise cosines of word type vectors from toy corpus – Cosine values between unreduced word type vectors (white cells) and cosine values of SVD-reduced word type vectors (shaded cells)

	human	interface	computer	user	system	response	time	EPS	survey	trees	graph	minors
human		0.50	0.50	0.00	0.58	0.00	0.00	0.50	0.00	0.00	0.00	0.00
interface	0.99		0.50	0.41	0.29	0.00	0.00	0.50	0.00	0.00	0.00	0.00
computer	0.88	0.93		0.41	0.29	0.50	0.50	0.00	0.50	0.00	0.00	0.00
user	0.89	0.93	1.00		0.47	0.82	0.82	0.41	0.41	0.00	0.00	0.00
system	0.99	1.00	1.00	0.95		0.29	0.29	0.87	0.29	0.00	0.00	0.00
response	0.79	1.00	0.98	0.98	0.88		1.00	0.00	0.50	0.00	0.00	0.00
time	0.79	0.85	0.98	0.98	0.88	1.00		0.00	0.50	0.00	0.00	0.00
EPS	1.00	1.00	0.89	0.90	0.99	0.80	0.80		0.00	0.00	0.00	0.00
survey	0.42	0.51	0.80	0.79	0.56	0.89	0.89	0.44		0.00	0.41	0.50
trees	-0.33	-0.23	0.15	0.14	-0.17	0.32	0.32	-0.31	0.72		0.67	0.41
graph	-0.28	-0.17	0.21	0.20	-0.11	0.38	0.38	-0.25	0.76	1.00		0.82
minors	-0.27	-0.17	0.21	0.20	-0.11	0.38	0.38	-0.25	0.76	1.00	1.00	

(shaded cells) shows the pairwise cosines between R_1-projected word type vectors of dimensionality 2, computed using the matrices of Table 3.4.2. As an example of how to read the table, say that we would like to compare the cosine score between *computer* and *system* in the unreduced space, so we seek this value from the cell at the intersection of the *computer* row and the *system* column, which is 0.29. If instead we wish to obtain the cosine value for the same pair but in the reduced space, we refer to the cell at the *system* row and the *computer* column, which reads 1.00. The value of 0.29 is a measurement of the degree to which bot *computer* and *system* tend to collocate. And indeed, we see that the two words only co-occur once. As we previously mentioned, the vectors in the reduced space are expected to to better capture the semantic relations between any two words. A value of 1 could be interpreted as indicating that *computer* and *system* are identical concepts. In a way, this is the case since by examining the article titles above we can tell that every mention of *system* seems to be in relation to a computer system and since all titles are from human-computer interaction and graph theory, both sub-disciplines within computer science, we can safely assume that every title is computer related. The value of 1 given by the LSA model is usually interpreted as indicating that *system* and *computer* are synonyms. So, not only is LSA telling us that these two words are syntagmatically associated, but that there is also a paradigmatic relationship between them, i.e. it is possible to substitute one word for the other in any context for this corpus. Of course, this interpretation can only be derived in the context of this toy corpus and cannot be generalised as applying to the English language as a whole.

Table 3.4.4 presents pairwise cosine values in a similar manner to Table 3.4.3 but the vectors used in its cosine computations were constructed from about two years worth of articles (1998-2000) from The New York Times (NYT). Just as before, Table 3.4.4 shows cosines from unreduced word type vectors (white cells) and cosines from R_1-projected word type vectors (shaded cells). This time the space was reduced to 300 dimensions. Although much bigger than the toy corpus, the sample of The New York Times is still too small to be considered an unbiased sample of the English language. It only covers journalistic language, which is a rather contained and specific type of language with very specific style and characteristics. It cannot be even be considered as a representative sample of journalistic English since it is tied to one single newspaper at a very specific time frame and in a very specific geographical location. But for the purposes of this example, it should be enough for illustrative purposes. In this NYT corpus, the cosine value between *system* and *computer* in the unreduced space is 0.055 whilst in the reduced space it is slightly reduced to 0.052. This indicates that there is little syntagmatic association between the two words and that there is very little evidence to suggest that one word could substitute the other in any given context in the NYT. This is perhaps because *system* in the NYT is used in much broader senses such as weather systems, political systems, the civil justice system, financial systems, etc. and computer systems do not feature as predominantly in the selected articles.

Another interesting word pair is *human* and *user*. In the toy corpus unreduced space (Table 3.4.3) their cosine is 0 since they do not co-occur within any title. But in the reduced space their cosine is 0.89. This is explained because both words co-occur separately with *interface*. That is, *human* and *user* have second-order co-occurrence. It is tempting to interpret this result as LSA somehow understanding that a user is a human being. While LSA does recognise a second-order co-occurrence relationship, there is no evidence to believe that this hyponymy relation is being captured by LSA explicitly. In fact, notice that the cosine between *human* and *computer* shoots from 0.50 in the unreduced space to 0.88 in the unreduced space and *user* and *computer* from 0.41 in the unreduced space to 1 in the reduced space. In the LSA literature, this is interpreted as evidence for the topical relationship between these words. That is, we would say that *human*, *user* and *computer* are topically related. And indeed, all three words feature in the human computer interaction titles from the toy corpus. In the NYT corpus (Table 3.4.4) the cosine between *human* and *user* in the unreduced space is 0.002 and 0.036 in the reduced space, which while being 18 times larger it still is quite small. Relatively speaking, it is a promising result since the cosine between *human* and *computer* only doubles when going from the unreduced (0.010) to the reduced space (0.021). Notice however that the cosine between *user* and *computer* is also quite powerful, being 0.051 in the unreduced space and 0.626 in the reduced space, 12 times larger.

This topical relationship can also be confirmed by contrasting the cosines of these three words with the cosines of the words that belong to the graph theory titles: *trees*, *graph* and *minors*. In the toy corpus (Table 3.4.3) the cosine between any of these three graph theory words and *human*, *user* or *computer* in the unreduced space is 0. In the reduced space, the cosine between *human* and any of the three graph theory words is negative. A negative result can be interpreted very much like a 0 result: complete non-relatedness. Both *computer* and *user* hold positive cosine values ranging from 0.15 to 0.21 with the three graph theory words.

Table 3.4.4: Pairwise cosines of word type vectors from full articles of The New York Times – Cosine values between unreduced word type vectors (white cells) and cosine values of SVD-reduced word type vectors (shaded cells)

	human	interface	computer	user	system	response	time	EPS	survey	trees	graph	minors
human		0.003	0.010	0.002	0.013	0.008	0.015	0.000	0.006	0.003	0.001	0.002
interface	0.078		0.022	0.079	0.012	0.001	0.002	0.000	0.000	0.001	0.000	0.001
computer	0.021	0.617		0.051	0.055	0.007	0.022	0.000	0.007	0.001	0.003	0.004
user	0.036	0.837	0.626		0.020	0.002	0.007	0.000	0.001	0.000	0.000	0.000
system	0.020	0.333	0.052	0.224		0.009	0.022	0.001	0.004	0.002	0.001	0.007
response	0.150	0.210	0.102	0.250	0.124		0.013	0.000	0.005	0.001	0.000	0.002
time	0.021	0.059	0.023	0.073	0.022	0.153		0.000	0.009	0.008	0.001	0.009
EPS	0.012	0.122	-0.001	0.087	0.054	0.121	0.000		0.000	0.000	0.000	0.000
survey	0.057	0.091	0.054	0.125	0.023	0.293	0.059	0.308		0.001	0.000	0.001
trees	0.077	0.112	0.007	0.078	0.020	0.193	0.107	0.136	0.135		0.000	0.001
graph	0.074	0.354	0.268	0.362	0.119	0.323	0.069	0.111	0.279	0.267		0.000
minors	0.031	0.118	0.048	0.113	0.087	0.272	0.105	0.092	0.102	0.165	0.130	

These positive values can be explained via higher-order co-occurrence via a word that bridges the two sets of titles: *survey* since it appears in a human computer interaction title (c2) and in a graph theory title (m4). Both *graph* and *minors* have a second-order co-occurrence relationship with both *computer* and *user*. But *trees* holds a third-order co-occurrence relationship with these two words. It is clear in the toy corpus that words belong to one or the other topic in varying degrees, depending on their higher-order co-occurrence behaviour. Obviously, this is not observed in the NYT corpus (Table 3.4.4) where the topical specialised human-computer-interface/graph-theory dichotomy is unlikely to play a role in such a newspaper that reports on broader, more mainstream news topics.

The size of a training corpus is an important engineering decision in any NLP project. Usually, the more training data available to a system, the better chance it will have to find and generalise semantic relations. However, another (and perhaps more important) decision is that of deciding the domain of the training corpus. An NLP system designer should select as training corpora those texts belonging to the same domain that the NLP system will be applied upon. In Table 3.4.4 the training and the application corpora were from different domains. This explains why specialised term such as *graph* and *minors* receive low values both in the unreduced and reduced spaces even if they are truly related within its own specialised domain.

Nevertheless, having access to a bigger and less specialised corpus such as the NYT does open up the possibility of exploring word pairs that hold more obvious or conventional semantic relations. Table 3.4.5 presents cosine scores in unreduced and reduced space from the NYT for a few of the following word pairs: *buy* and *purchase* (synonyms), *cheap* and *expensive* (antonyms), *cat*, *feline*, *mammal* and *animal* (hypernyms – i.e. a cat is a feline, all felines are

Table 3.4.5: Pairwise cosine scores between a few words holding obvious semantic relationships – Cosine values between unreduced word type vectors (white cells) and cosine values of SVD-reduced word type vectors (shaded cells) constructed from The New York Times corpus

	buy	purchase	cheap	expensive	bird	cat	feline	mammal	animal	house	room	window
buy		0.035	0.017	0.023	0.002	0.002	0.001	0.000	0.002	0.015	0.006	
purchase	0.691		0.003	0.007	0.001	0.000	0.000	0.000	0.000	0.006	0.003	0.002
cheap	0.601	0.504		0.019	0.001	0.000	0.001	0.000	0.001	0.003	0.003	0.002
expensive	0.623	0.569	0.821		0.001	0.001	0.000	0.000	0.002	0.007	0.006	0.003
bird	0.126	0.139	0.362	0.287		0.007	0.000	0.008	0.009	0.003	0.004	0.004
cat	0.250	0.186	0.460	0.398	0.491		0.080	0.001	0.016	0.007	0.005	0.003
feline	0.118	0.153	0.349	0.274	0.382	0.574		0.000	0.006	0.000	0.002	0.000
mammal	0.059	0.087	0.277	0.298	0.334	0.348	0.371		0.007	0.000	0.000	0.000
animal	0.180	0.190	0.371	0.372	0.385	0.535	0.460	0.648		0.007	0.002	0.002
house	0.062	0.056	0.068	0.097	0.060	0.199	0.019	0.028	0.125		0.030	0.012
room	0.007	0.036	0.096	0.121	0.108	0.176	0.235	0.013	0.048	0.029		0.018
window	0.311	0.228	0.488	0.463	0.420	0.567	0.428	0.263	0.371	0.181	0.350	

mammals, all mammals are animals) vs. *bird* (not a cat, not a feline and not a mammal, but an animal), *house*, *room*, *window* (meronyms – i.e. a house has rooms, a house has windows and a room has windows).

The reduced space is able to capture relatively high values for the synonym and antonym pairs, as expected, as well as between conceptually related words such as *buy* and *cheap* and *expensive* and between the latter two words and *purchases*. It is a bit surprising that the pair *buy* and *cat* has a higher value (0.250) than *buy* and *house* (0.062). As expected, there are relatively high values for *cat* and *feline* (0.574) and *cat* and *animal* (0.535), but not as high for *cat* and *mammal* (0.348) or *feline* and *mammal* (0.371), even if the value is high for *mammal* and *animal* (0.648). *Bird* and *cat* have a medium value, perhaps due to a tendency to co-occur (0.491) and while the cosines for *bird* and *feline* (0.382) and *mammal* (0.334) are relatively low, disappointingly the cosine between *bird* and *animal* scores at around the same value (0.385). The meronymy set (*house*, *room*, *window*) all score low, with the exception of the combination *room* and *window* which does not do too bad (0.350), showing again that many times the interpretation of these scores as high or low can be relative.

As we have seen here, LSA can tell whether two words are semantically related, whether this relation is syntagmatic, paradigmatic or both but it cannot tell what is the proportion of syntagmatic vs. paradigmatic or even what subtype of relation is operating (synonymy, antonymy, hyponymy, etc.)

3.4.4.2 Polysemy

LSA is also expected to capture polysemy or be sensitive to polysemous words. The word type vector for any given word as explored in Section 3.4.4.1, both in reduced and unreduced form, is representative of all of the occurrences of that word in the corpus irrespective of any difference of sense or usage in each of those occurrences. In a way, such a type vector conflates senses and it is not easy to determine which segment dimensions (in the unreduced form) or which reduced dimensions relate to which sense of the word. The segment vector, on the other hand, could be used to disambiguate or discriminate senses. For example, Levin et al. (2006) employed SVD-reduced segment vectors to discriminate senses of polysemous words achieving good results (this experiment is further discussed in Section 4.2, p. 110). In fact, it can be shown that such SVD-reduced segment vectors can modulate the different senses of a word. An example is given by Landauer (2007, p. 17) in which he computes the cosine of the segment "a circle's diameter" with "radius of spheres" (0.55) and with "music of the spheres" (0.03), showing that LSA is sensitive to the polysemy of the word *sphere* and is seemingly able to select (or modulate) the correct sense of the word given the words that surround it (its context), i.e. either *radius* or *music*. In the SVD-reduced space of the NYT corpus introduced in Section 3.4.4.1 the values these cosines yield are 0.441 for the first pair and 0.0121 for the second.

This sense modulation can be explained intuitively by projecting segment vectors through the SVD matrices from the toy corpus introduced in the previous section. As discussed in that section, there are two main topics in the corpus: human-computer interaction and graph theory. There is only one word that is present in titles from both topics: *survey*. Because of this, this word can be considered to be polysemous. In fact, within the confines of this toy corpus it indeed is polysemous. In the only human-computer interaction title using it, *survey* is used in the 'questionnaire' sense, whereas 'literature review' is the sense used in the one graph theory title this word occurs. Since LSA is only able to capture polysemy in segment vectors, let us make up the segments "system survey", "EPS survey", "graph survey" and "minors survey", the former two relating to the human-computer interaction topic and the latter two to the graph theory topic. In LSA/information retrieval parlance these made up segments would be called pseudodocuments, since they do not form part of the collection and can indeed be viewed as queries, although that is not how we see them here. The pairwise cosines of the R_1-projected (3.4.5) versions of these segment vectors can be seen in the shaded cells of Table 3.4.6. As before, the white cells in this table are the cosines of unreduced vectors. Notice that in the unreduced space every vector for these segments would only have two non-zero values (two ones to be precise). One of which will always be in the *survey* dimension and the other non-zero value in some other dimension, which will never be matched by another vector. Because of this, the cosine value of any two of these particular segment vectors will always be 0.5. As expected, in the reduced space the cosines for pairs belonging to the same topic are quite high (0.99 for "system survey" and "EPS survey", 1 for "graph survey" and "minors survey") and

Table 3.4.6: Pairwise cosines of segment (pseudodocument) vectors using the toy corpus – Cosine values between unreduced word type vectors (white cells) and cosine values of SVD-reduced word type vectors (shaded cells)

	system survey	EPS survey	graph survey	minors survey
system survey		0.50	0.50	0.50
EPS survey	0.99		0.50	0.50
graph survey	0.38	0.50		0.50
minors survey	0.43	0.54	1.00	

relatively low for cross-topic cosines with values in the 0.38 to 0.54 range.

This behaviour can be easily explained by tracing the way the R_1 projection is computed. Recall that the projection is performed via $R_1(\mathbf{d}) = \mathbf{U_k}^T\mathbf{d}$. If we see this matrix multiplication as the dot products between the row vectors in $\mathbf{U_k}^T$ and the **d** column vector, we can see that only those columns in $\mathbf{U_k}^T$ corresponding to non-zero values in **d** will be preserved. In addition, all of those preserved columns will be summed in order to generate the final, reduced vector $R_1(\mathbf{d})$, with the values in **d** acting as weights in this sum. In other words, the R_1 projection could be seen as a weighted sum of the column vectors of $\mathbf{U_k}^T$ (or equivalently the row vectors of $\mathbf{U_k}$) in which the weights are specified by the segment vector **d** itself.

Table 3.4.7 shows the R_1 projections for the four segments introduced above, in row vector format. It can easily be verified that the dimensions for these vectors are in fact the sums of the rows in $\mathbf{U_k}$ corresponding to the non-zero dimensions in the unreduced segment vector, i.e. those corresponding to the words appearing in the segment itself. For example, for "system survey", the resulting reduced vector from 3.4.7 can be seen as a sum of the row vectors for *system* and *survey* from $\mathbf{U_k}$ (see Table 3.4.2): $R_1^T(\mathbf{system}, \mathbf{survey}) = d_{system}\,\mathbf{u_{system}} + d_{survey}\,\mathbf{u_{survey}} = [-0.64, -0.17] + [-0.21, 0.27] = [-0.85, 0.10]$.

It can be seen by inspecting the $\mathbf{U_k}$ matrix from Table 3.4.2, that the values of the reduced dimensions (k_1 and k_2) for words occurring in titles for one topic fall within a particular range while the corresponding dimensional values for words occurring in titles for the other topic fall in another range. For the human-computer interaction words, the ranges are $-0.64 \leq k_1 \leq -0.21$ and $-0.17 \leq k_2 \leq 0.27$, whereas the rages for the graph theory words are $-0.21 \leq k_1 \leq -0.01$ and $0.27 \leq k_2 \leq 0.62$. The effect that the vector sum has on word vectors from the same topic is that the dimensional values get deeper inside the ranges typical of one topic and move away from the ranges of the other topic. As will be seen in Section 4.1 (p. 108), a similar idea serves as the inspiration in the construction of Schütze's indirect (second-order) context vectors, a token-based representation derived from word type vectors in Word Space.

Table 3.4.7: R_1-reduced segment vectors

$R_1^T(\mathbf{d})$	k1	k2
system survey	-0.85	0.10
EPS survey	-0.51	0.13
graph survey	-0.25	0.89
minors survey	-0.24	0.72

3.4.4.3 Noise reduction

The noise reduction property is inherent to the actual dimensionality reduction process. As mentioned in Section 3.4.1, $\mathbf{A_k}$ is the best approximation of rank k of $\mathbf{A}$ and the error of this approximation can be measured by the Frobenius norm of $\mathbf{A} - \mathbf{A_k}$ (see Eq. 3.4.4). Recall from Theorem 3.4.2 that singular values in $\mathbf{\Sigma}$ are arranged in decreasing order in the diagonal, i.e. $\sigma_{11} \geq \sigma_{22} \geq \ldots \geq \sigma_{rr} \geq 0$. Therefore, removing the higher order singular values $(\sigma_{k+1,k+1}, \ldots, \sigma_{rr})$ only incurs in a small error. Of course, the larger k is, the smaller this error is. But recall as well that singular values can be interpreted as an indication of the importance of a dimension in the vector space produced by SVD. So by eliminating the higher and therefore less important dimensions, we also remove less important information, which we interpret to be mostly noise such as chance or irrelevant word co-occurrences.

3.5 Word Space

For the purposes of this thesis, the term **Word Space** refers to the word matrix introduced by Schütze (1992), the indirect (second-order) context vectors derived from this word matrix introduced by Schütze (1998) as well as the direct (first-order) context vectors used in word-sense discrimination experiments by Pedersen and Bruce (1997) and Purandare and Pedersen (2004). Whilst these context vectors do not form part of the core Word Space matrix, they are considered here to be part of Word Space because they are computed from this matrix. Also, recall that the matrices involved in both Salton's VSM as well as LSA are capable of representing word types as well as word tokens via segment/document vectors. Second-order context vectors, which represent tokens, complement the main Word Space matrix because it is only capable of representing word types. So, second-order context vectors are added as a mechanism that make Word Space a more complete counterpart to Satlon's VSM and LSA. Although strictly speaking not part of Word Space, first-order context vectors, another token-based vector representation, will also be introduced here. This is because, as it will be shown in Section 3.5.2, first-order context vectors can be used to define the word vectors that make up the word matrix at the core of Word Space, so they naturally fit as a component of the model.

This section mostly introduces Word Space as commonly described in the literature. Chapter

4 redefines it as a linear transformation, explores its properties and makes detailed comparisons with LSA.

3.5.1 The word matrix: representing word types

The core component of Word Space is a word-word co-occurrence matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, also called a **word co-occurrence matrix**, often simply called a **word matrix**. In its simplest incarnation, a cell w_{ij} in $\mathbf{W}$ counts how many times word type τ_i co-occurs with word type τ_j. More specifically, w_{ij} counts how many times τ_i and τ_j both occur within the same context. Just like in LSA, a "context" here can be any textual unit such as a paragraph, a sentence or a word segment of fixed length. However, a sliding word window of fixed length centred at a token (see Section 2.2.3) is commonly used.

Provided that the vocabulary represented by the rows and columns is the same and is represented in the same order in both rows and columns, then $\mathbf{W}$ will always be a square, symmetric matrix. There are variations of this matrix though that violate this symmetry. For example, some word matrices in Schütze (1998) represent the top 20,000 most frequent words in the corpus as the rows of the matrix, but only the top 2,000 most frequent words as the columns of the matrix, producing a $20,000 \times 2,000$ rectangular word matrix, which is only symmetric in its first 2,000 rows. Purandare and Pedersen (2004) use a word matrix variant in some experiments that only takes into account the co-occurrence of a word with words co-occurring to its right (the so-called bigram features, i.e. ordered co-occurrence). A word matrix of "bigram features" will not be symmetric even if it is square. Besides, the feature values w_{ij} can again be weighted by different functions so that the value w_{ij} reflects the degree of importance of the co-occurrence of τ_i and τ_j.

Each row vector of the word matrix $\mathbf{W}$ is called a **word vector $\mathbf{t}$**, which represents a word type τ. If $\mathbf{W}$ is symmetric, then its column vectors are also word vectors. By convention in this thesis, however, when the term *word vector* is used, normally a reference to the row vectors of the word matrix is implied. Word vectors are formally defined in Definition 3.5.3 in terms of first-order context vectors. A formal definition of a word matrix can be given in terms of these word vectors:

Definition 3.5.1 (Word Matrix). Let $\mathbf{W}$ be an $m \times n$ matrix, with m the size of some chosen vocabulary $\mathcal{V}_m$ and n the size of the feature set (a vocabulary $\mathcal{V}_n$) used by the word vectors. The i^{th} row of $\mathbf{W}$ is the word vector for the i^{th} word in $\mathcal{V}_m$.

The row vectors of both $\mathbf{A}$ and $\mathbf{W}$ represent the same thing: word types. So, cosines between any two row vectors (from the same matrix) reflect a degree of similarity between the two words represented by each vector. However, the cells a_{ij} and w_{ij} in each matrix represent rather different things and, as a consequence, the type of similarity measured by the cosines of any two row vectors is different for each matrix. In **A** a cell represents occurrence in a context. In **W** a cell represents a co-occurrence of two words in several contexts in a corpus. Therefore, if $\mathbf{a_i}$ and $\mathbf{a_j}$ are row vectors (type vectors) from **A** representing τ_1 and τ_2, respectively, then $\cos(\mathbf{a_i}, \mathbf{a_j})$ will measure the degree to which words types τ_i and τ_j tend to co-occur. As established in Section 3.3, this makes **A** to be a syntagmatic vector space. On the other hand, if $\mathbf{w_i}$ and $\mathbf{w_j}$ are row vectors (word vectors) from **W** representing τ_1 and τ_2, respectively, then $\cos(\mathbf{w_i}, \mathbf{w_j})$ will measure the degree to which words types τ_i and τ_j tend to co-occur with the same words. That is, a high cosine score indicates that τ_i and τ_j tend to collocate with the same words but not necessarily with each other. We assume that if two words co-occur with the same words, i.e. they occur in similar environments, then one of these words could potentially be substituted by the other. We thus say that these words hold a paradigmatic relation (see Section 2.2.2). As a consequence, we consider **W** to be a **paradigmatic vector space** (or simply a **paradigmatic space**).

As a highly dimensional matrix, **W** is often decomposed by truncated SVD, in a similar way as it is done in LSA. In contrast with its use in LSA, Schütze (1992) justifies his use of SVD mostly as a dimensionality reduction tool and not so much as a noise reduction technique or as a method for finding higher order co-occurrences. Basically he uses SVD to be able to make comparisons and computations between high-dimensional vectors tractable given the computational restrictions of the time when that first paper was written. However, Schütze (1998) seems to have changed opinion slightly and concedes that his motivation for using SVD in his Word Space matrix is similar to the use of SVD in LSA. He notes in this later paper that SVD finds the major dimensions of variation in Word Space and as a consequence expects cosine similarities to be a better similarity measure between SVD-reduced vectors than between unreduced vectors. In addition, Purandare and Pedersen (2004) justify their use of truncated SVD because of its expected abilities to capture synonyms. All of these authors intuitively expect to obtain, from an SVD-reduced word matrix, similar properties to those usually obtained from an SVD-reduced type-segment matrix. However, since the two matrices being reduced are very different from each other, it is reasonable to ask whether the two matrices have similar or different properties. We shall see in Section 4.3 (p. 122), however, that there are many similarities to the SVD structure from both matrices and the expectations of these authors are therefore warranted. However, in Chapter 6 we shall see that despite these similarities, context vectors derived from one or the other model can still perform differently in WSX experiments.

3.5.2 Context vectors: representing word tokens

As previously mentioned, the word matrix from Definition 3.5.1 can only represent word types and not word tokens. In order to represent word tokens, we turn to context vectors. In this thesis, two basic types of context vectors are studied: direct or first-order context vectors and indirect second-order context vectors. **Direct (first-order) context vectors** record the words that a word token co-occurs with, whilst **indirect (second-order) context vectors** represent a word token by counting the words that the words occurring in the token's context tend to co-occur with elsewhere in the corpus. In general, two words τ and υ are considered to have second-order co-occurrence if a third word ϕ tends to co-occur with both τ and υ, but not necessarily at the same time. The arguments in favour of second-order context representations against first-order representations somewhat echo the arguments given for LSA and word space against Salton's original VSM, in that the first-order context vectors of two tokens of the same word type used in the same sense will only have a high cosine similarity value if both tokens appear in the vicinity of mostly the same words. Second-order context vectors will suffer less from this problem because even if the two tokens are surrounded by different words, since the token is used in the same sense, the meaning of these surrounding words will have to be related somehow (they will probably belong to the same semantic field) so they will have to co-occur together at some point in the corpus. To illustrate this reasoning, consider these sample contexts using the word *line* in two senses: TELEPHONE LINE and QUEUE (taken from the HILS corpus/Wall Street Journal, see Sec. 5.1, p. 136).

(3.5.1) He made another call and came back on the *line* with the news. [TELEPHONE]

(3.5.2) ... he said while his lawyers held on another telephone *line*. [TELEPHONE]

(3.5.3) In this society, if someone gets hurt, somebody, anybody must pay – and a *line* of hungry trial lawyers will be waiting outside the hospital room to get a share of the action. [QUEUE]

While *line* is used in the same sense (TELEPHONE) in examples (3.5.1) and (3.5.2), the cosine similarity value of their first-order vector representations (assuming that each dimension counts co-occurrence with words like *call*, *came*, *news*, *lawyers*, *held*, *telephone*, etc.) will be 0. In fact, under such representation, the cosine similarity value between a first-order context vector representing (3.5.2) and (3.5.3) will be more than 0, since both contexts share the occurrence of *lawyers*.

Under a second-order vector representation, it would be expected that words neighbouring *line* like *call* and *telephone* will co-occur with more or less the same words, i.e. they appear in similar contexts, which is an indication that they are not only semantically related, but paradigmatically related. Notice as well that it is very likely for *call* and *telephone* to co-occur directly in many contexts, which would make them syntagmatically related as well.

Second-order context vectors represent tokens, as previously mentioned. However, since they need to capture co-occurrence with words in other parts of the corpus, they are commonly computed from word vectors, which represent the co-occurrence patterns of word types in the corpus. The second-order context vector of a target token is computed by summing (or averaging) the word vector's of the token's neighbours co-occurring with the target token. In (3.5.1), for example, the second-order context vector would be the sum (or average) of the

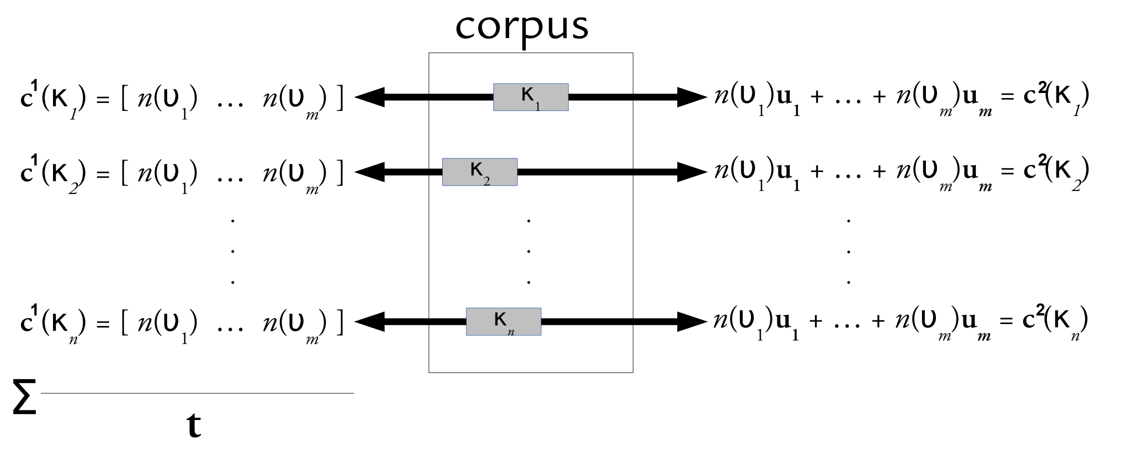

corpus
$\mathbf{c}^1(\kappa_1) = [\ n(\upsilon_1)\ \dots\ n(\upsilon_m)\ ]$
κ_1
$n(\upsilon_1)\mathbf{u}_1 + \dots + n(\upsilon_m)\mathbf{u}_m = \mathbf{c}^2(\kappa_1)$
$\mathbf{c}^1(\kappa_2) = [\ n(\upsilon_1)\ \dots\ n(\upsilon_m)\ ]$
κ_2
$n(\upsilon_1)\mathbf{u}_1 + \dots + n(\upsilon_m)\mathbf{u}_m = \mathbf{c}^2(\kappa_2)$
$\mathbf{c}^1(\kappa_n) = [\ n(\upsilon_1)\ \dots\ n(\upsilon_m)\ ]$
κ_n
$n(\upsilon_1)\mathbf{u}_1 + \dots + n(\upsilon_m)\mathbf{u}_m = \mathbf{c}^2(\kappa_n)$
Σ
t

Figure 3.5.1: Conceptual relationship between direct (first-order) context vectors, word vectors and indirect (second-order) context vectors. – The centre of the figure represents a corpus in which the instances of a target word τ have been found as tokens κ_i (a total of n tokens). Each context instance constitutes a word window centred at each token. On the left-hand side of the figure, each of these context instances are represented by direct context vectors $\mathbf{c}^1(\kappa_i)$ of m dimensions. Each dimension $n(\upsilon_j)$ is the count of υ_j unigrams occurring in the window. The sum of all $\mathbf{c}^1(\kappa_i)$ for τ produce its word vector $\mathbf{t}$. In a similar manner, it is possible to compute a word vector for any word occurring in the corpus, including the u_j unigrams. On the right-hand side of the figure, the word vectors for each u_j are used to compute indirect context vectors $\mathbf{c}^2(\kappa_i)$ for each context instance centred at κ_i by multiplying the frequency of each unigram in the word window $n(u_j)$ by that unigram's word vector $\mathbf{u}_\mathrm{j}$.

word vectors for *call*, *came*, *news*, etc. Figure 3.5.1 depicts conceptually the computation of first-order context vectors, word vectors and second-order context vectors.

So far intuitive descriptions of the first-order and second-order context and word vectors have been provided. In what follows, a formal framework defining these objects is set out.

Let $\mathcal{C}$ be a corpus, and let $\mathrm{win}^l(\kappa)$ be a function returning a set[6] of tokens κ' around κ (i.e. the **window** around κ) where l is the window width and typically $\kappa' \in \mathrm{win}^l(\kappa)$ iff $\mathrm{pos}(\kappa) - l/2 \leq \mathrm{pos}(\kappa') \leq \mathrm{pos}(\kappa) + l/2$, where $\mathrm{pos}(\kappa)$ is a function that returns the position of a token κ in the document or corpus. In other words, $\mathrm{win}^l(\kappa)$ is a function that returns the set of the $l/2$ tokens to the left of κ and the $l/2$ tokens to the right of κ, and does not include κ itself. Then in general a *feature* can be identified with a function that maps windows to

[6]A set is a collection of distinct objects that does not allow member repetition. It could be argued that a set representation of a word window (or text segment for that matter) such as "she loves you yeah yeah yeah" should only contain one instance of *yeah* since sets do not allow repetition. Whilst this would be true for a set representation of the word *types* of this example, it is certainly not true for a set representation of the word *tokens* of the same example. Recall from Section 2.1.1 (p. 32) that a word token is an actual instance of a word type. As a consequence, each of the *yeah* tokens is unique as it is, in its own right, a separate instance of the (same) type . The set returned by $\mathrm{win}^l(\kappa)$ is a set of tokens (not types) around κ.

real numbers. For a particular token κ in $\mathcal{C}$, the **direct** or **first-order context vector**, $\mathbf{c}^1(\kappa)$, is a vector giving the values of the features in the window around κ. In most cases, features are equated with *unigrams*, one for each member of $\mathcal{V}_m$, some chosen subset of the unigrams of the corpus $\mathcal{C}$[7], and the value of a unigram feature υ on the window $\text{win}^l(\kappa)$ is simply the count of the tokens of υ in the window:

Definition 3.5.2 (Direct or first-order context vector D-C-UR)**.** For window width l, and a choice of dimensionality m corresponding to a restriction to some unigram vocabulary $\mathcal{V}_m$, the first-order context vector for token κ, $\mathbf{c}^1(\kappa)$, is the vector of dimensionality m such that for every $\upsilon_i \in \mathcal{V}_m$ the i-th dimension u_i in $\mathbf{c}^1(\kappa)$ is

$$u_i = n(\upsilon_i, \kappa) \tag{3.5.4}$$

where $n(\upsilon_i, \kappa)$ is the frequency of the tokens of unigram υ_i in word window $\text{win}^l(\kappa)$:

$$n(\upsilon, \kappa) = \sum_{\forall \lambda \in \text{win}^l(\kappa)} \begin{cases} 1 & \textit{if } \upsilon = \text{type}(\lambda) \\ 0 & \textit{otherwise} \end{cases} \tag{3.5.5}$$

In this thesis, $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$.

This is the token vector configuration D-C-UR summarised in Table 4.2.1 (p. 111). The "C" indicates that it is a vector coming from the matrix of direct context vectors $\mathbf{C}$ (see below).

Observe that $\mathbf{c}^1(\kappa)$ is defined whether or not the token κ in $\mathcal{C}$ is one of the word dimensions of the vector. In other words, $\mathbf{c}^1(\kappa)$ is defined even if $\tau \notin \mathcal{V}_m$, where τ is the corresponding type for κ.

Context vectors can be thought of as either row or column vectors. In previous work, we have considered them as row vectors (Maldonado-Guerra and Emms, 2012). However, in this thesis, direct and indirect context vectors will be seen as column vectors, mostly to keep their relation to their LSA counterparts, segment vectors, more transparent. This also allows us to reserve the row vector configuration for word vectors coming either from $\mathbf{A}$, $\mathbf{W}$ or any other appropriate matrix.

This brings us to the matrix of direct context vectors. For a vocabulary choice $\mathcal{V}$, an entire corpus can be represented by a matrix $\mathbf{C} \in \mathbb{R}^{m \times K}$, in which each first-order context vector for every token κ in the corpus is arranged as a column of the matrix. This way, K is the number of tokens for which a first-order context $\mathbf{c}^1(\kappa)$ is defined, and $m = |\mathcal{V}|$. The row vectors in this matrix can be regarded as word vectors. However, these are not the word vectors traditionally used in Word Space. These are word vectors of context or token features. For the remainder

[7]This subset of unigrams $\mathcal{V}_m$ is a selection of m words from the corpus that fulfil some criteria. Following Schütze (1998), experiments in this work select the top m most frequent words in the corpus, excluding function (stop) words, but many other selection criteria are possible.

of this chapter the matrix of direct context vectors **C** and the word vectors defined on its rows will not be further considered. We shall return to their discussion in Section 4.2.1 (p. 111).

A traditional Word Space **word vector t** is a vector that represents a particular word type τ in a corpus, and is based on the aggregation of the set of first-order context vectors of its tokens κ_i. The simplest possibility is just to sum these.

Definition 3.5.3 (Word vector). If the direct (first-order) context vectors $\mathbf{c}^1(\kappa_i) \in \mathbb{R}^n$ represent each word token κ_i of word type τ in the corpus, the word vector of word type τ, $\mathbf{t} = \mathbf{w}(\tau)$, is the sum of all $\mathbf{c}^1(\kappa_i)$:

$$\mathbf{t} = \mathbf{w}(\tau) = \sum_{\forall \kappa_i : \text{type}(\kappa_i) = \tau} \left(\mathbf{c}^1(\kappa_i)\right)^T \tag{3.5.6}$$

where type(κ) is a function or hash table mapping a word token κ in the corpus or document with its corresponding word type. Notice that word vectors $\mathbf{t} \in \mathbb{R}^{1\times n}$ and that they form the rows of $\mathbf{W} \in \mathbb{R}^{m\times n}$.

This definition implies that in the word vector for τ, $\mathbf{t} = [u_1, \ldots, u_m]$ with each unigram feature u_i corresponding to word type υ_i, each u_i is simply the count of how often τ and υ_i co-occur within $l/2$ words of each other in the corpus, where l is the chosen window-width for the $\mathbf{c}^1$ vectors.

An **indirect** or **second-order context vector**, $\mathbf{c}^2(\kappa)$, for a particular word token κ of word type τ is, in its simplest incarnation, the sum of the word vectors of words occurring in the context window.

Definition 3.5.4 (Indirect or second-order context vector as a sum of word vectors I-W-UR). If $\mathbf{w}(\upsilon_i)$ is the word vector of word type υ_i whose token λ_i co-occurs with token κ of target word τ (i.e. type(κ) $= \tau$ and type(λ_i) $= \upsilon_i$) within a word window centred at κ, then the indirect or second-order context vector of κ is

$$\mathbf{c}^2(\kappa) = \sum_{\forall \lambda_i \in \text{win}^l(\kappa)} \left(\mathbf{w}(\text{type}(\lambda_i))\right)^T \tag{3.5.7}$$

Notice that $\mathbf{c}^2(\kappa) \in \mathbb{R}^n$ because each $\mathbf{w}(\text{type}(\lambda_i)) \in \mathbb{R}^{1\times n}$.

This is the token vector configuration I-W-UR summarised in Table 4.2.1 (p. 111). "I" indicates indirect or second-order and "W" is the matrix from which this vector is derived, the word matrix **W**. Effectively, this nomenclature indicates that this context vector is "I"ndirectly derived from "W".

This is the standard definition found in works such as Schütze (1998) and Purandare and Pedersen (2004). An alternative but equivalent definition of second-order context vectors as a linear transformation is given in Section 4.1 (p. 108).

In word-sense disambiguation and discrimination works, often the notion of a sense vector is employed. A **sense vector** is an aggregation of first or second-order context vectors that correspond to a single sense of the word they represent. That is, a sense vector represents geometrically the sense of a target word. Suppose τ has k senses $\sigma_1 \dots \sigma_k$. If $\mathcal{P}_i$ is the set of τ's occurrences manifesting sense σ_i, then by summing (or averaging) the vectors $\mathbf{c}^1$ that represent these occurrences (i.e. $\forall \kappa \in \mathcal{P}_i$), we obtain what can be termed the **first-order sense vector**, $\mathfrak{s}^1{}_i$. Such vectors have been used for word-sense disambiguation (Oh and Choi, 2002; Sugiyama and Okumura, 2009; Martinez and Baldwin, 2011). Notice that the word vector can be seen as the special case in which *all* occurrences representing all k senses of the word are counted.

In unsupervised word-sense induction or discrimination tasks, the aim is to induce a sense-reflecting partition over the tokens of an ambiguous word. There are a variety of ways to evaluate such a partitioning, but one is basically to derive sense vectors from the induced partitions, and categorise a test set of items by their nearness to the first-order context vectors of test set items.

Just as a $\mathfrak{s}^1$ vector can be defined from $\mathbf{c}^1$ vectors for a particular set of occurrences, a **second-order sense vector** $\mathfrak{s}^2$ can be defined by aggregating the $\mathbf{c}^2$ vectors for a particular set of occurrences. A number of authors have since worked with essentially these second-order representations (Schütze, 1998; Purandare and Pedersen, 2004; de Marneffe and Dupont, 2004; de Marneffe et al., 2005; Wang and Hirst, 2010; Sagi et al., 2011).

We now define first- and second-order sense vectors formally. If τ is an polysemous word and $\mathcal{S}_i$ is a set of τ's tokens exhibiting a particular sense σ_i, then the centroid of the context-vectors for the occurrences in $\mathcal{S}_i$ is a candidate representation of the sense:

Definition 3.5.5 (First- and second-order sense vector, $\mathfrak{s}^1$, $\mathfrak{s}^2$). If $\mathcal{S}_i$ is a set of tokens, the first-order sense and second-order sense vectors based on $\mathcal{S}_i$ are

$$\mathfrak{s}^1{}_i = \frac{1}{|\mathcal{S}_i|} \sum_{\kappa \in \mathcal{S}_i} \mathrm{c}^1(\kappa) \text{ and } \mathfrak{s}^2{}_i = \frac{1}{|\mathcal{S}_i|} \sum_{\kappa \in \mathcal{S}_i} \mathrm{c}^2(\kappa) \qquad (3.5.8)$$

3.6 Syntagmatic space and paradigmatic space

Section 3.3 described Salton's vector space model adapted to distributional lexical semantics and Section 3.5 described Schütze's modification of this vector space model into what has become known as Word Space. These sections also stated that the word vectors in the former semantic space constitute a syntagmatic vector space whilst the word vectors in the latter form a paradigmatic vector space, without too much justification other than an intuitive one. This section intends to expand on this justification.

The syntagmatic or paradigmatic property emerging from comparisons of word vectors from these two semantic spaces was first pointed out by Sahlgren (2006). The justification given for these properties in this work was quite intuitive, not unlike the one given in the present thesis so far and which can be summarised as follows as a recap. Recall that Saussurean structuralism states that any two word types sustain a syntagmatic relation if they have the potential to co-occur within the same syntagma (or unit of text), whereas if those two word types can instead be substituted for one another in a particular syntagma without changing its overall semantics (even if the actual meaning of the syntagma changes), then Saussurean structuralism considers them to be engaged in a paradigmatic relation. We saw that a word type can be represented as a high-dimensional vector with (at least) two types of dimensional features: syntagma/document features (word-by-document vector/matrix) or word type features (word-by-word vector/matrix). The cosine function (and other vector similarity functions) measures the degree of dimensional overlap between any two vectors. If the dimensions overlapped by the cosine of word vectors are syntagmas or documents, we say that the vector space formed by those vectors is a syntagmatic (vector) space. If, on the other hand, the dimensions overlapped by word vectors are other word types, we say that the vector space formed by those vectors is a paradigmatic (vector) space, since it measures the degree to which those two words can substitute one another in environments where roughly the same words appear.

These paradigmatic and syntagmatic relation types are not necessarily mutually exclusive, however. Two words, like *vehicle* and its hyponym *car*, do have a paradigmatic relation (hypernymy/hyponymy) and yet hold a syntagmatic relation in a clause such as "cars and other vehicles must not park on a double yellow line". However, this degree of overlap will depend somewhat on the type of paradigmatic relation involved. Whilst a hypernymy/hyponymy will allow some overlap as we have seen, other paradigmatic relations, such as synonymy will perhaps allow less overlap since it is expected that (near) synonyms are unlikely to co-occur.

Sahlgren attempted to measure this degree of overlap between syntagmatic and paradigmatic spaces with a series of experiments. In one of such experiments, he directly estimated this overlap by counting the number of common closest neighbours of words between syntagmatic and paradigmatic spaces: given a corpus, he produced word vectors in syntagmatic space (i.e. word-by-syntagma vectors) and word vectors in paradigmatic space (i.e. word-by-word vectors). He then selected word types that had a frequency of at least 20 in the corpus and computed pairwise cosine scores among the vectors representing these word types, within each vector space. Then, for each word, he obtained the closest neighbour set in the paradigmatic space and the closest neighbour set in the syntagmatic space and counted the degree of overlap between the two sets. That is, for a word type τ_i, the closest paradigmatic neighbour set $\mathcal{P}_i$ is the top 10 words whose vectors have the highest cosine value with $\mathbf{t_i}$ in paradigmatic space, whereas the closest syntagmatic neighbour set $\mathcal{S}_i$ is the top 10 words whose vectors have the highest cosine value with $\mathbf{t_i}$ in syntagmatic space. The degree of overlap between both spaces for word type τ_i is $overlap_i = |\mathcal{P}_i \cap \mathcal{S}_i|$. Sahlgren found that when using short context windows the average degree of overlap between both vector spaces was very low (around 1%). However, as the context window in the paradigmatic vector space increased, the overlap with the syntagmatic vector space also increased. He did notice though that this overlap was still relatively small (around 10%) when the window size was set to the maximum he experimented with (40 words). We shall comment on the interpretation of this increase in overlap in larger windows shortly.

In another experiment Sahlgren contrasted the performance of syntagmatic spaces with the performance of paradigmatic spaces in a synonym selection test, a primarily paradigmatic task as we have seen. The synonym selection test is the same TOEFL synonym multiple choice test performed by Landauer and Dumais (1997) with LSA. In this test, a stimulus word is given and the candidate must choose its correct synonym from four possible answers given. In Sahlgren's reproduction of the test, he achieved a maximum score of 75% using a paradigmatic space and 68% with a syntagmatic space, giving support to the hypothesis that paradigmatic spaces indeed do represent paradigmatic relations. However, Sahlgren himself as well as Schütze (1998) and Utsumi (2010) observe that paradigmatic spaces conflate the different senses of the word that they represent and as a consequence it can become difficult to tell the distribution of each sense apart in a word-by-word vector. Utsumi performs an additional synonym test only using words that are monosemous in his corpus and notices a further increase of performance on paradigmatic spaces and a decrease in performance on syntagmatic spaces. This result seems to reflect that synonym selection depends somewhat on selecting the correct sense of the word. To illustrate this intuitively, consider the sample synonym test question in Table 3.6.1. The correct synonym for *spot*, among the options given, is *location*. However, there are many other words that could be correct synonyms for *spot* such as *blemish*, *jot*, *particle*, etc., each of them depending on a different sense of *spot*. Sahlgren observed that this issue manifested more obviously when working with large corpora such as the British National Corpus (BNC) and when using highly frequent word features. Filtering out highly frequent word features from paradigmatic word vectors computed from the BNC, increased their performance on the synonym test. The assumption is that more frequent words tend to exhibit more senses in a corpus than less frequent words.

As previously mentioned, Sahlgren also observed that as he increased the context window in paradigmatic spaces their overlap with syntagmatic spaces also increased. Utsumi also observes a similar trend by observing that paradigmatic relationships such as synonymy are better represented by paradigmatic spaces using short contexts. Similarly, Peirsman et al. (2008a) experimented with Word Space word vectors in predicting word associations made by humans when presented with a cue and observed that some cue-associate pairs are more paradigmatic in nature, whilst others are more syntagmatic. They noticed that word vectors produced from short contexts predicted paradigmatic cue-associate pairs relatively more accur-

Table 3.6.1: Sample TOEFL synonym test question with the correct answer highlighted. Taken from Sahlgren (2006, p. 101)

Stimulus	Choices
spot	sea
	location
	latitude
	climate

ately whilst word vectors using longer contexts predicted the syntagmatic pairs relatively more accurately. For example, they found short contexts to better predict paradigmatic pairs such as *melancholy-sad* and *rapidly-quickly* which are synonymous pairs, as well as *cormorant-bird* which have a hypernym-hyponym relation and *new-old*, an antonymous relation. Longer contexts, by contrast were better at predicting syntagmatic/co-occurrence pairs such as *sill-window*, *riding-horse*, *reflection-mirror*, *spend-money*, etc. These results seem to point that as context windows become larger, paradigmatic spaces start approximating syntagmatic spaces. This can also be explained intuitively. As the context window in a paradigmatic space gets bigger, it begins to cover a bigger fraction of the document. So, the cosine measure between two word vectors in a paradigmatic space that uses context windows sufficiently large to almost cover the size of an average document, will be a measure of the overlap of documents in which both words occur, which is similar to our interpretation of the cosine of two vectors in syntagmatic space. Despite of this overlap on larger context windows, we shall continue to designate word-word spaces as paradigmatic spaces and word-segment spaces as syntagmatic spaces and will leave their respective context window sizes as parameters. But we should remain aware that as context windows on paradigmatic spaces become wider, their difference with syntagmatic spaces becomes more blurred.

There is an alternative vector space formulation modelling paradigmatic and syntagmatic spaces that is not studied in this thesis but is worth mentioning for the sake of completeness: the directional word-by-word matrix **C** introduced by Schütze and Pedersen (1993). This word-by-word matrix is similar to **W** from Word Space, except that is not symmetric. The j-th dimension in row vector $\mathbf{l_i}$ in **C** counts the number of ordered co-occurrences $\tau_i\tau_j$ in the corpus, whereas the j-th dimension in column vector $\mathbf{r_i}$ in **C** counts the number of the ordered co-occurrences $\tau_j\tau_i$. That is, the row vectors count right co-occurrences whilst column vectors count left co-occurrences. Schütze and Pedersen then decompose **C** by SVD. For the purposes of the present discussion, suffice it to say that the authors discovered some interesting properties in this SVD-decomposed version of **C**. First, this decomposition produced a left-hand side vector space and a right-hand side vector space. Then, the left-hand side vector space could be called a left-hand paradigmatic space in that the cosine between any two left-hand side vectors $\cos(\mathbf{l_i}, \mathbf{l_j})$ measures the degree in which corresponding words τ_i and τ_j shared the same neighbours on the left. In other words, $\cos(\mathbf{l_i}, \mathbf{l_j})$ measures the degree to which words τ_i

and τ_j tend to have the same left-neighbours. Similarly, they found that $\cos(\mathbf{r_i}, \mathbf{r_j})$ measures the degree to which τ_i and τ_j tend to have the same right-neighbours, forming a right-hand side paradigmatic space. Interestingly, they also found that $\cos(\mathbf{l_i}, \mathbf{r_j})$ measures the degree to which τ_j can appear to the right of τ_i, i.e. the degree to which $\tau_i\tau_j$ is a valid collocation, whilst $\cos(\mathbf{r_i}, \mathbf{l_j})$ measures the degree to which $\tau_j\tau_i$ is a valid collocation. That is, the left-hand side vectors and the right-hand side vectors form, when compared between each other, a directional (left-to-right or right-to-left) syntagmatic space.

As a final point, it is worth mentioning that the theoretical insight given by this discussion of paradigmatic vs syntagmatic spaces can have practical implications for tasks such as word-sense disambiguation and discrimination. For example, it is unlikely that two different senses of a word will be called upon within the same syntagma (unless of course some sort of pun is taking place). That is, if a word occurs more than once in a syntagma, both occurrences are likely to use the word in the same sense. In addition, two unrelated senses (or homonyms) like *bank*$_1$ 'financial institution' and *bank*$_2$ 'edge of body of water' will co-occur with different sets of words (i.e. a paradigmatic "cosine" between *bank*$_1$ and *bank*$_2$ will be low), making them non-substitutable by each other.

4 Linear Transformations in Word Space and LSA

Section 3.5 (p. 95) gave the basic definitions for Word Space as commonly found in the literature. This chapter presents a continuation of that section in that it advocates a particular equivalent formulation of indirect (second-order) context vectors of dimensionality n in which they are derived from direct (first-order) context vectors of dimensionality m by multiplication by a $m \times n$ word matrix $\mathbf{W}$. In essence, the word matrix is regarded as a linear transformation that maps token-based representations in $\mathbb{R}^m$ to $\mathbb{R}^n$, where n can potentially (but not necessarily) be much smaller than m. This formulation makes more transparent the relation of the second-order context vector construction to standard methods for dimensionality reduction, such as SVD. This linear transformation formulation is presented in Section 4.1, where it is observed that $\mathbf{W}$ can indeed be substituted for any other suitable matrix, and in fact, the direct context vector matrix $\mathbf{C}$ and the LSA word-segment matrix $\mathbf{A}$ are suitable candidates.

This linear transformation does constitute a generic method of indirect context vector construction, and Section 4.2 further explores this. It presents how indirect context vectors can be produced from unreduced and SVD-reduced word vectors coming from $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{W}$. This section also reviews SVD projection of Word Space direct context vectors and LSA segment vectors, the latter of which were already covered in Chapter 3.

Finally, Section 4.3 offers an analytical comparison between the direct and indirect token vector representations provided by Word Space and LSA. First the unreduced direct token representations from Word Space and LSA are compared, noticing trivial similarities between them. It is shown however, that this trivial similarity translates into a more complicated relationship between the word vectors in Word Space and in LSA. However, due to a mathematical equivalence, it is shown that the SVD projections of LSA word vectors and Word Space word vectors are approximate to each other. The consequence of this is that the indirect context vector representations derived from $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{W}$ studied in Section 4.2 must also be similar. This is shown both analytically and by producing some numerical toy examples. A systematic and empirical comparison between the direct and indirect LSA and Word Space in word-sense disambiguation and discrimination experiments is conducted in Chapter 6.

4.1 W as a linear map

Definition 3.5.4 (p. 101) characterises the construction of a second-order context vector for a specific token κ as a sum of a selection of word vectors taken from the rows of the word matrix **W**. A word vector is selected to be summed if a token of the word type it represents co-occurs with κ within a word window of length l centred at κ. In other words, the corresponding word vector for every token that appears in the token set $\text{win}^l(\kappa)$ gets selected and summed to construct $\mathbf{c}^2(\kappa)$, the second-order context vector of κ. Recall from Section 3.2 (p. 67) that sets can be represented as vectors, provided that an appropriate feature (dimension) mapping function is provided. The definition for first-order context vectors (Definition 3.5.2 on page 100) can be seen as such a mapping function capable of translating the token set $\text{win}^l(\kappa)$ into a vector format $\mathbf{c}^1(\kappa)$ that contains zeroes in the dimensions corresponding to words not present in $\text{win}^l(\kappa)$ and non-zero values in dimensions corresponding to words present in that set. If the feature set (vocabulary) of $\mathbf{c}^1(\kappa)$ is $\mathcal{V}_m$ and the word matrix **W** defines a word vector for every word in that same set $\mathcal{V}_m$ and the order in which the dimensions of $\mathbf{c}^1(\kappa)$ are arranged is the same order in which the rows of **W** are arranged, then the matrix multiplication $\mathbf{W}^T\mathbf{c}^1(\kappa)$ selects the word vectors (row vectors) in **W** that correspond to the non-zero dimensions in $\mathbf{c}^1(\kappa)$ and sums those word vectors. Thus, this matrix multiplication is equivalent to the summation of word (type) vectors in **W** of the tokens present in $\text{win}^l(\kappa)$ from Definition 3.5.4. This observation was previously made in Maldonado-Guerra and Emms (2012).

Proof. (3.5.7) sums the word vectors of the words co-occurring with κ in context. Word vectors are summed the same number of times as the frequency of the words they represent in that context. Since $[\mathbf{c}^1(\kappa)]_i$ holds the frequency of τ_i in the context we could equivalently multiply every row vector $\mathbf{r_i}$ in **W** by $c_i = [\mathbf{c}^1(\kappa)]_i$ and sum the resulting vectors, i.e.

$$\mathbf{c}^2(\kappa) = \left(\sum_{i=1}^{m} c_i\mathbf{r_i}\right)^T = \left(\sum_{i=1}^{m} c_i\left[w_{i1},\ldots,w_{ij},\ldots,w_{in}\right]\right)^T \tag{4.1.1}$$

$$= \left[\sum_{i=1}^{m} c_i w_{i1},\ldots,\sum_{i=1}^{m} c_i w_{ij},\ldots,\sum_{i=1}^{m} c_i w_{in}\right]^T \tag{4.1.2}$$

If a word is not present in $\text{win}^l(\kappa)$, it will have a c_i value of zero, preventing the sum of its corresponding word vector $\mathbf{r_i}$. If a word appears more than once in $\text{win}^l(\kappa)$ (i.e. if there is more than one token for the same type), then its corresponding word vector would be summed the same number of times it appears in the window by using either the above equation or the original Definition 3.5.4. The number of times a word appears in the window is also recorded by its corresponding dimension c_i in $\mathbf{c}^1(\kappa)$.

Let us now consider the matrix multiplication $\mathbf{W}^T\mathbf{c}^1(\kappa)$. This multiplication can be seen as computing the dot product of every column vector in **W** with $\mathbf{c}^1(\kappa)$ and placing each result in each dimension of $\mathbf{c}^2(\kappa)$. This matrix multiplication can be expanded in terms of these dot products for the j-th row of $\mathbf{W}^T$ (using the w_{ij} indexation for **W**) as $[c_1,\ldots,c_m]\cdot$

$\left[w_{1j}, \ldots, w_{mj}\right] = \sum_{i=1}^{m} c_i w_{ij}$, which is the same as the central element in (4.1.2). □

We can now re-define direct (second-order) context vectors as the result of a matrix multiplication:

Definition 4.1.1 (Indirect (second-order) context vector with parameter **W** — I-W-UR). Given a word matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ whose rows are the word vectors for words in some chosen vocabulary $\mathcal{V}_m$ with $m = |\mathcal{V}_m|$, if $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ is the direct (first-order) context vector representation (using $\mathcal{V}_m$ for features) of token κ, then $\mathbf{c}^2(\kappa) \in \mathbb{R}^n$ is the second-order context vector representation (with features $\mathcal{V}_n$) of token κ and is defined by

$$\mathbf{c}^2(\kappa) = \mathbf{c}^{\mathbf{W}}(\kappa) = \mathbf{W}^T \mathbf{c}^1(\kappa) \tag{4.1.3}$$

This configuration is also I-W-UR from Table 4.2.1 just like in Definition 3.5.4 (p. 101), as they are equivalent. This formulation of the $\mathbf{c}^2$ construction is not the customary one, but a useful one. Although (4.1.3) and (3.5.7) look different, (4.1.3) is also defining $\mathbf{c}^2(\kappa)$ to be a sum of multiples of relevant rows of **W**, as demonstrated before. Moreover, it is not hard to show that the relation between $\mathbf{c}^1$ and $\mathbf{c}^2$ vectors exemplified in (4.1.3) lifts to a relation between $\mathfrak{s}^1$ and $\mathfrak{s}^2$, given the definition of sense vectors as averages over $\mathbf{c}^1$ and $\mathbf{c}^2$ vectors:

$$\mathfrak{s}^2{}_i = \frac{1}{|\mathcal{S}_i|} \mathbf{W}^T \mathfrak{s}^1{}_i \tag{4.1.4}$$

Because the word vectors on the rows of **W** are sometimes shorter than the height of **W** i.e. $n < m$, equation (4.1.3) typically represents a dimensionality lowering transformation. It is interesting to note, that formulating the $\mathbf{c}^2$ construction as we have, (4.1.3) is strikingly similar to the projection equations for dimensionality reduction via SVD. As discussed in Section 3.4.1, if **M** is a $m \times n$ matrix, it has a so-called reduced-rank SVD $\widehat{\mathbf{M}} = \mathbf{U_k} \mathbf{\Sigma_k} \mathbf{V_k}^T$. Applied to any **M** whose rows have the same dimensionality as that of $\mathbf{c}^1$ vectors, the SVD dimensionality reduction procedure would lead via (3.4.5) or (3.4.6) (p. 82) to a transformation of a $\mathbf{c}^1 \in \mathbb{R}^m$ vector to a vector in $\mathbb{R}^k$. In an analogous manner, the $\mathbf{c}^2$ construction as defined in (4.1.3) transforms a $\mathbf{c}^1 \in \mathbb{R}^m$ vector to a $\mathbf{c}^2 \in \mathbb{R}^n$ vector.

In order to encompass all variants of the $\mathbf{c}^2$ vector notion that have been proposed in the literature, the formulation in (4.1.3) must be developed a little further, addressing *weighting* and *normalisations*. If $\boldsymbol{\alpha} \in \mathbb{R}^m$ is a weight vector, let diag($\boldsymbol{\alpha}$) $\in \mathbb{R}^{m \times m}$ be a diagonal matrix with $\boldsymbol{\alpha}$ on the diagonal and zeros elsewhere. Pre-multiplying **W** by diag($\boldsymbol{\alpha}$) corresponds to weighting its rows by the values in $\boldsymbol{\alpha}$. Let **nm** be a function on vectors $\mathbf{v} \in \mathbb{R}^m$ giving a *normalising factor*. Possible settings for such a function are the L1-norm, which simply sums the values, the L2-norm, which gives the Euclidean length, and a constant setting **I**, mapping all vectors to 1.

Definition 4.1.2 (Second-order context vector with parameter $\mathbf{W}$, α, $\mathbf{nm}$ — I-W-UR). Given a word matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, a weights vector $\boldsymbol{\alpha} \in \mathbb{R}^m$, and a function $\mathbf{nm}$, from a first-order context vector $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ representing token κ, the corresponding second-order context vectors $\mathbf{c}^2(\kappa) \in \mathbb{R}^n$ is defined by

$$\mathbf{c}^2(\kappa) = \frac{1}{\mathbf{nm}(\mathbf{c}^1(\kappa))} (\mathrm{diag}(\boldsymbol{\alpha})\mathbf{W})^T \mathbf{c}^1(\kappa) \tag{4.1.5}$$

Note that with $\mathbf{nm} = \mathbf{I}$, and $\boldsymbol{\alpha}$ consisting of all 1's, then this reduces to (4.1.3). The notion of $\mathbf{c}^2$ in Schütze (1998) is an instance of (4.1.5) using $\mathbf{nm} = \mathbf{I}$ and a non-trivial weights vectors $\boldsymbol{\alpha}$, based on TF-IDF values. The notion of $\mathbf{c}^2$ in Purandare and Pedersen (2004) is an instance of the definition, with trivial weight vectors, and with $\mathbf{nm}$ set to be L1-norm.

For simplicity, however, for the remainder of this thesis we shall focus on the simpler formulation from Definition 4.1.1. It will be used in the next section to demonstrate the construction of indirect context vectors by using matrices other than $\mathbf{W}$, such as the direct context vector matrix $\mathbf{C}$ or the LSA word-segment matrix $\mathbf{A}$. When used with these alternative matrices, the resulting indirect context vectors will receive the labels I-C-UR and I-A-UR from Table 4.2.1, respectively.

4.2 Direct and indirect token representations

Chapter 3 introduced, among other things, VSM/LSA segment vectors and Word Space direct (first-order) context vectors. It was then mentioned that Word Space direct context vectors were specifically designed to represent tokens, whilst VSM/LSA segment vectors have been used by a number of works as proxies to represent tokens. Basically, if a text segment contains a token of a target word of interest, its segment vector can be used as a vector representation for that token. This will be examined in more detail in Section 4.3, but for now let us assume that both LSA and Word Space have their own, competing mechanism of (unreduced) *direct* token representation: segment vectors and direct (first-order) context vectors, respectively. Both vector types are column vectors and as such they can form the columns of one of two matrices: $\mathbf{A}$, a matrix whose columns are LSA's segment vectors, and $\mathbf{C}$, a matrix that has Word Space direct context vectors for columns. Both matrices are equally amenable to SVD reduction, and as a consequence both token representations have an SVD-reduced version.

Word Space also defines an indirect token vector representation: indirect or second-order context vectors. These can be created by aggregating word vectors (Def. 3.5.4, p. 101), or equivalently, transforming direct context vectors via a word matrix $\mathbf{W}$ containing such word vectors along its rows, as defined in Section 4.1. And as mentioned in that same section, the rows in both $\mathbf{A}$ and $\mathbf{C}$ also define word vectors, albeit of different features. So, they too can be used in the aggregation or linear transformation operations mentioned above (I-A-UR, I-C-UR). Alternatively, the SVD-reduced versions of these matrices can be used in order to obtain SVD-reduced indirect token representations (I-A-R1/2, I-C-R1/2, I-W-R1/2). This section presents all of these token vector representation configurations, making some observations along the way. Table 4.2.1 summarises them all.

Table 4.2.1: Token vector representation configurations from LSA and Word Space. Configurations are named as a three-part label X-X-XX. The values for the first part are "D" (direct) and "I" (indirect). The values in the middle part refer to the matrix from which the representation is derived (**A**, **C** or **W**). The values on the last part represent the projection type if SVD is used ("R1" or "R2") or "UR" (unreduced) to indicate that SVD was not used.

Direct token representations				
Conf.	**Description**	**Also known as**	**Def.**	**Equation**
D-A-UR	LSA direct context vector	LSA doc. vec.	3.2.1	$\mathbf{d}$, column in $\mathbf{A}$
D-A-R1	SVD R_1 LSA direct context vector	SVD R_1 LSA doc. vec.	3.4.1	$R_1(\mathbf{d}) = \mathbf{U_k}^T\mathbf{d}$
D-A-R2	SVD R_2 LSA direct context vector	SVD R_2 LSA doc. vec.	3.4.1	$R_2(\mathbf{d}) = \mathbf{\Sigma}_\mathbf{k}^{-1}\mathbf{U_k}^T\mathbf{d}$
D-C-UR	WS direct context vector	WS first-order	3.5.2	$[\mathbf{c}^1(\kappa)]_i = n(\upsilon, \kappa)$
D-C-R1	SVD R_1 WS direct context vector	SVD R_1 WS first-order	4.2.1	$R_1(\mathbf{c}^1(\kappa)) = \mathbf{U_k}^T\mathbf{c}^1(\kappa)$
D-C-R2	SVD R_2 WS direct context vector	SVD R_2 WS first-order	4.2.1	$R_2(\mathbf{c}^1(\kappa)) = \mathbf{\Sigma}_\mathbf{k}^{-1}\mathbf{U_k}^T\mathbf{c}^1(\kappa)$

Indirect token representations				
Conf.	**Description**	**Also known as**	**Def.**	**Equation**
I-A-UR	LSA indirect context vector via $\mathbf{A}$		4.2.6	$\mathbf{c}^\mathbf{A}(\kappa) = \mathbf{A}^T\mathbf{c}^1(\kappa)$
I-A-R1	SVD R_1 LSA indirect context vector via $\mathbf{A}$		4.2.7	$\mathbf{c}^{R_1(\mathbf{A})}(\kappa) = \mathbf{\Sigma_k}\mathbf{U_k}^T\mathbf{c}^1(\kappa)$
I-A-R2	SVD R_2 LSA indirect context vector via $\mathbf{A}$		4.2.7	$\mathbf{c}^{R_2(\mathbf{A})}(\kappa) = \mathbf{U_k}^T\mathbf{c}^1(\kappa)$
I-C-UR	WS indirect context vector via $\mathbf{C}$		4.2.3	$\mathbf{c}^\mathbf{C}(\kappa) = \mathbf{C}^T\mathbf{c}^1(\kappa)$
I-C-R1	SVD R_1 WS indirect context vector via $\mathbf{C}$		4.2.4	$\mathbf{c}^{R_1(\mathbf{C})}(\kappa) = \mathbf{\Sigma_k}\mathbf{U_k}^T\mathbf{c}^1(\kappa)$
I-C-R2	SVD R_2 WS indirect context vector via $\mathbf{C}$		4.2.4	$\mathbf{c}^{R_2(\mathbf{C})}(\kappa) = \mathbf{U_k}^T\mathbf{c}^1(\kappa)$
I-W-UR	WS indirect context vector via $\mathbf{W}$	WS second-order	4.1.1	$\mathbf{c}^\mathbf{W}(\kappa) = \mathbf{W}^T\mathbf{c}^1(\kappa)$
I-W-R1	SVD R_1 WS indirect context vector via $\mathbf{W}$	SVD R_1 WS second-order	4.2.5	$\mathbf{c}^{R_1(\mathbf{W})}(\kappa) = \mathbf{\Sigma_k}\mathbf{U_k}^T\mathbf{c}^1(\kappa)$
I-W-R2	SVD R_2 WS indirect context vector via $\mathbf{W}$	SVD R_2 WS second-order	4.2.5	$\mathbf{c}^{R_2(\mathbf{W})}(\kappa) = \mathbf{U_k}^T\mathbf{c}^1(\kappa)$

4.2.1 Token representations via C

As previously mentioned, direct (first-order) context vectors arranged as the columns of a matrix **C** can be used to compute indirect context vectors and/or transformed into objects of lower dimensionality via SVD. There are several ways in which first-order context vectors can be arranged as the column vectors of a matrix. These are three basic options:

1. For every token κ in the corpus, produce n direct context vectors $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ with a window $\text{win}^l(\kappa)$ of length l centred at κ and arrange each of these n direct context vectors as the columns of a matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$.

2. For every token κ of a specific target word type τ in the corpus, produce p direct context vectors $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ with a window $\mathrm{win}^l(\kappa)$ of length l centred at κ and arrange each of these p direct context vectors as the columns of a matrix $\mathbf{C} \in \mathbb{R}^{m \times p}$.

3. For each segment δ in the corpus, choose one token κ in occurring in δ, produce q direct context vectors $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ with a window $\mathrm{win}^l(\kappa)$ centred at κ of sufficient length to cover the whole segment, and arrange each of these q direct context vectors as the columns of a matrix $\mathbf{C} \in \mathbb{R}^{m \times q}$.

Option 1 effectively represents every token in the corpus in each column in $\mathbf{C}$. As such, $n \geq m$, as every corpus usually has far more tokens than types. So, an indirect representation based on this version of $\mathbf{C}$ will not result in a dimensionality reduction. Also, notice that tokens that lie close to each other in the corpus will have very similar vectors (as they co-occur with almost the same tokens). Whilst this might look like storing redundant information, an SVD-reduction might be able to properly benefit from such information. Option 2 is an attractive choice for a WSX experiment in which all of the tokens of a target or ambiguous word have been identified. Option 3, however, is the option that more closely resembles the LSA segment vector in which every discrete segment of the corpus is captured. This is the option used in the experiments presented in Chapter 6. In those experiments, LSA segment vectors are created for every segment of the corpus and direct context context vectors are created for every occurrence of the target word, making sure that its word window covers the whole segment. Since every segment in each of the corpora (Sec. 5.1, p. 136) used in the experiments contains an occurrence of the target word, the approach used for constructing $\mathbf{C}$ is really a hybrid between options 2 and 3.

4.2.1.1 D-C-UR: Unreduced direct context vectors

Regardless of the option chosen to create $\mathbf{C}$, direct context vectors can be readily used to represent tokens. Definition 3.5.2 (p. 100) introduces the way they are computed.

4.2.1.2 D-C-R1/2: SVD-reduced direct context vectors

After constructing the first-order context matrix $\mathbf{C}$ in one of the three ways presented earlier or some other similar way, the matrix can then be decomposed by truncated SVD through Theorem 3.4.3 (p. 80) and individual first-order context vectors can be transformed using R_1 and R_2 projections in the following manner:

Definition 4.2.1 (Direct context vector projections D-C-R1/2). Given a word token κ represented by direct context vector $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$, we can obtain its SVD-reduced version $\widehat{\mathbf{c}^1(\kappa)} \in \mathbb{R}^k$

via one of the following two projections:

$$\widehat{\mathbf{c}^1(\kappa)} = R_1(\mathbf{c}^1(\kappa)) = \mathbf{U_k}^T\mathbf{c}^1(\kappa) \tag{4.2.1}$$

$$\widehat{\mathbf{c}^1(\kappa)} = R_2(\mathbf{c}^1(\kappa)) = \mathbf{\Sigma}_\mathbf{k}^{-1}\mathbf{U_k}^T\mathbf{c}^1(\kappa) = \mathbf{\Sigma}_\mathbf{k}^{-1}R_1(\mathbf{c}^1(\kappa)) \tag{4.2.2}$$

If $\mathbf{c}^1(\kappa)$ is actually a column vector in $\mathbf{C}$ and therefore is also represented in $\mathbf{V_k}$ by row vector $\mathbf{v} \in \mathbb{R}^{1\times k}$, then these projections become:

$$\widehat{\mathbf{c}^1(\kappa)} = R_1(\mathbf{c}^1(\kappa)) = \mathbf{\Sigma_k}\mathbf{v}^T \tag{4.2.3}$$

$$\widehat{\mathbf{c}^1(\kappa)} = R_2(\mathbf{c}^1(\kappa)) = \mathbf{v}^T \tag{4.2.4}$$

Notice that with any of these projections $\widehat{\mathbf{c}^1(\kappa)} \in \mathbb{R}^k$, i.e. we obtain row vectors of dimensionality k.

In Emms and Maldonado-Guerra (2013) a dimensionality reduction like the one described in this section was performed on direct context vectors. First, a direct context vector matrix $\mathbf{C}$ was constructed using method 2 above and decomposed by SVD using various truncation values k in the low hundreds. Then, the direct context vectors were transformed using R_1 and R_2 projections from Definition 4.2.1. The resulting SVD-reduced direct context vectors were then employed in word-sense discrimination experiments. It was found that R_1-projected direct context vectors performed better than R_2-projected direct context vectors in said word-sense discrimination experiments. Similar experiments, including supervised word-sense disambiguation experiments, are presented in Chapter 6.

Notice that the projections from Definition 4.2.1 correspond to the segment/document vector projections that were studied in Definition 3.4.1 (p. 82). Recall that the rows of the direct context vector matrix $\mathbf{C}$ represent word types. So, it is possible to use a row vector projection (like that from Definition 3.4.2, p. 83) to produce SVD-reduced word type vectors:

Definition 4.2.2 (Type vector projections from $\mathbf{C}$). Given a word type τ represented by row vector $\mathbf{t} \in \mathbb{R}^{1\times n}$, we can obtain its SVD-reduced version $\widehat{\mathbf{t}}$ through one of the two following projections:

$$\widehat{\mathbf{t}} = R_1(\mathbf{t}) = \mathbf{t}\mathbf{V_k} \tag{4.2.5}$$

$$\widehat{\mathbf{t}} = R_2(\mathbf{t}) = \mathbf{t}\mathbf{V_k}\mathbf{\Sigma}_\mathbf{k}^{-1} = R_1(\mathbf{t})\mathbf{\Sigma}_\mathbf{k}^{-1} \tag{4.2.6}$$

If $\mathbf{t}$ is actually a row vector in $\mathbf{C}$ and therefore is also represented by row vector $\mathbf{u} \in \mathbb{R}^{1\times k}$ in

$\mathbf{U_k}$, then these projections become:

$$\widehat{\mathbf{t}} = R_1(\mathbf{t}) = \mathbf{u}^T\mathbf{\Sigma_k} \tag{4.2.7}$$

$$\widehat{\mathbf{t}} = R_2(\mathbf{t}) = \mathbf{u}^T \tag{4.2.8}$$

Notice that with any of these projections $\widehat{\mathbf{t}} \in \mathbb{R}^{1\times k}$, i.e. we obtain column vectors of dimensionality k.

In Sections 4.3.1 and 4.3.2 it will be discussed that word type representations in **C** are related, if not similar to word type representations in **A** and **W**. Definition 4.3.1 establishes a relationship between direct (first-order) context vectors and the segment vectors along the columns of **A**. Effectively, Salton's VSM and the first-order context matrix are very similar representations, with the only significant difference being described by Definition 4.3.1: the token representations in the first-order context matrix do not count its own type as occurring in the representation.

4.2.1.3 I-C-UR: Unreduced indirect context vectors via C

The indirect context vector version via **C**, which transforms direct context vectors of word type features into indirect context vectors of token features is computed as follows:

Definition 4.2.3 (Indirect context vector with parameter **C** — I-C-UR). Given a direct (first-order) context matrix $\mathbf{C} \in \mathbb{R}^{m\times n}$ whose rows are the word vectors for words types in some chosen vocabulary $\mathcal{V}_m$ with $m = |\mathcal{V}_m|$ and whose columns represent the dimensions (features) of each of those row token vectors corresponding to a selection of word tokens $\mathcal{K}_n$ extracted from some corpus, with $n = |\mathcal{K}_n|$, if $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ is the direct (first-order) context vector representation (using $\mathcal{V}_m$ for features) of token κ, then $\mathbf{c}^{\mathrm{C}}(\kappa) \in \mathbb{R}^n$ is the indirect context vector representation (using the $\mathcal{K}_n$ as its feature set) of token κ, and is defined by

$$\mathbf{c}^{\mathrm{C}}(\kappa) = \mathbf{C}^T\mathbf{c}^1(\kappa) \tag{4.2.9}$$

This definition is virtually identical to Definition 4.1.1. Notice however that whilst the context vectors produced by Definition 4.1.1 are usually regarded as second-order co-occurrence vectors, the indirect context-vectors produced by Definition 4.2.3 cannot be realistically called "second-order". This is the reason why we use the more general term *indirect* over *second-order*. In addition, by using the word vector projections form **C** or **W** in place of those matrices in these definitions, it is possible to use them to produce SVD-reduced indirect context vectors. This is covered by Subsections 4.2.1.4 for the case of SVD-reduced **C** and 4.2.2.2 for that of SVD-reduced **W**.

4.2.1.4 I-C-R1/2: SVD-reduced indirect context vectors via C

The word vector projections from Definition 4.2.2 can be used in Definition 4.2.3 to obtain SVD-reduced indirect context vectors via **C**:

Definition 4.2.4 (Indirect context vector with parameter $\widehat{\mathrm{C}}$ — I-C-R1/2). Given a direct context vector matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$ whose rows are the word vectors for words types in some chosen vocabulary $\mathcal{V}_m$ with $m = |\mathcal{V}_m|$ and whose columns represent the dimensions (features) of each of those row token vectors corresponding to a selection of word tokens $\mathcal{K}_n$ extracted from some corpus, with $n = |\mathcal{K}_n|$, let $\widehat{\mathbf{C}} = \mathbf{U_k \Sigma_k V_k}$ be the singular value decomposition of **C** truncated to $k \leq \min(m, n)$ dimensions. If $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ is the direct (first-order) context vector representation (using $\mathcal{V}_m$ for features) of token κ, then $\mathbf{c}^{\widehat{\mathrm{C}}}(\kappa) \in \mathbb{R}^k$ is the indirect context vector representation (using the orthogonal feature set given by the truncated SVD) of token κ, and is defined by either (4.2.10) or (4.2.11) below depending on whether the R_1 or R_2 projection is used:

$$\mathbf{c}^{\widehat{\mathrm{C}}}(\kappa) = \mathbf{c}^{R_1(\mathrm{C})}(\kappa) = \mathbf{\Sigma_k U_k}^T \mathbf{c}^1(\kappa) \tag{4.2.10}$$

$$\mathbf{c}^{\widehat{\mathrm{C}}}(\kappa) = \mathbf{c}^{R_2(\mathrm{C})}(\kappa) = \mathbf{U_k}^T \mathbf{c}^1(\kappa) \tag{4.2.11}$$

4.2.2 Token representations via W

The Word Space word matrix **W** cannot represent tokens directly. However, it can represent them indirectly, via the the so-called second-order context vectors. Alternatively, these indirect context representations can be produced from SVD-reduced word vectors. This section studies these possibilities.

4.2.2.1 I-W-UR: Unreduced indirect context vectors via W

Unreduced indirect context vectors via **W** are computed through Definition 4.1.1, or equivalently Definition 3.5.4 (p. 101).

Schütze (1998), Purandare and Pedersen (2004) and related works produce indirect (second-order) context vectors via the word matrix **W**. The second-order context vector of a token of a target word in context (or an instance of a target word) is computed by summing or averaging the (row) word vectors from **W** that correspond to the words co-occurring with the target word. So, for example, if *survey* is our ambiguous word in "A survey of user opinion of computer system response time" (title number c2 in the toy corpus of Section 3.4.4, p. 87), a second-order context representation (using the whole title as the context window) based on the matrix **W** from table 4.3.1, based on the sum of the context's word vectors (or equivalently via word matrix multiplication) would be:

$$\mathbf{c}^2(\kappa) = \sum_{w_i \text{ occurs in } \mathrm{win}^l(p)} \mathbf{w_i} = \mathbf{W}^T \mathbf{c}^1(\kappa) \tag{4.2.12}$$

$$\begin{aligned}
\mathbf{c}^2(\mathit{survey}_{c2}) &= \mathbf{w_{user}} + \mathbf{w_{computer}} + \mathbf{w_{system}} + \mathbf{w_{response}} + \mathbf{w_{time}} \qquad (4.2.13)\\
&\quad \begin{bmatrix} 0 & 1 & 1 & 0 & 2 & 2 & 2 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}\\
&\quad + \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}\\
&= + \begin{bmatrix} 2 & 1 & 1 & 2 & 2 & 1 & 1 & 3 & 1 & 0 & 0 & 0 \end{bmatrix}\\
&\quad + \begin{bmatrix} 0 & 0 & 1 & 2 & 1 & 0 & 2 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}\\
&\quad + \begin{bmatrix} 0 & 0 & 1 & 2 & 1 & 2 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}\\
&= \begin{bmatrix} 3 & 3 & 4 & 7 & 7 & 6 & 6 & 4 & 5 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}$$

a vector in which the last three dimensions (relating to the graph theory paper titles) are zeroes and non-zero values on the other dimensions, which pertain to the HCI papers. In a way, *survey* is a polysemous word in this toy dataset. It appears once in a HCI paper title (the one in this example) and once in a graph theory paper title (“Graph minors: a survey”, title m4). Presumably, the sense in which *survey* is used in each paper is different: in the HCI paper it is used in its ‘questionnaire’ sense, whereas in the graph theory paper it is used in its ‘study’ or ‘literature review’ sense. Compare $\mathbf{c}^2(\mathit{survey}_{c2})$ with $\mathbf{c}^2(\mathit{survey}_{m4})$:

$$\begin{aligned}
\mathbf{c}^2(\mathit{survey}_{m4}) &= \mathbf{w_{graph}} + \mathbf{w_{minors}} \qquad (4.2.14)\\
&= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 3 & 2 & 2 \end{bmatrix}
\end{aligned}$$

which has zeroes in the HCI dimensions and non-zero values on the graph theory dimensions. The cosine between these two vectors is $\cos\left(\mathbf{c}^2(\mathit{survey}_{c2}), \mathbf{c}^2(\mathit{survey}_{m4})\right) = 0.14$, a relatively low similarity value.

4.2.2.2 I-W-R1/2: SVD-reduced indirect context vectors via $\mathbf{W}$

Schütze (1998) and Purandare and Pedersen (2004) also compute second-order context vectors using SVD-reduced versions of word vectors and not the word vectors from $\mathbf{W}$ directly. Effectively, they project $\mathbf{W}$ via R_1 or R_2 using equations (3.4.11) or (3.4.12) to form the reduced matrix $\widehat{\mathbf{W}}$ and row vectors $\widehat{\mathbf{w}}_\mathbf{i}$ from this matrix are used in the sum involved in Definition 3.5.4 instead of $\mathbf{w_i}$. We can equivalently say that they transform first-order context context vectors via Definition 4.1.1 by using $\widehat{\mathbf{W}}$ instead of $\mathbf{W}$. These works continue to call the resulting context vectors “second-order context vectors” even when they clearly do not capture second-order co-occurrence information in the same way as the context vectors computed from unreduced word vectors do. To avoid confusion, this terminology is abandoned in this thesis and instead we call these indirect context vectors produced from SVD-reduced word vectors as **SVD-reduced indirect context vectors** and are defined as follows:

Definition 4.2.5 (Indirect context vector with parameter $\widehat{\mathbf{W}}$ — I-W-R1/2). Given a word matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ whose rows are the word vectors for words in some chosen vocabulary $\mathcal{V}_m$ with $m = |\mathcal{V}_m|$ and whose columns represent the dimensions (features) of each of those row word vectors corresponding to some chosen vocabulary $\mathcal{V}_n$ with $n = |\mathcal{V}_n|$, let $\widehat{\mathbf{W}} = \mathbf{U_k \Sigma_k V_k}$ be the singular value decomposition of $\mathbf{W}$ truncated to $k \leq \min(m, n)$ dimensions. If $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ is the first-order context vector representation (using $\mathcal{V}_m$ for features) of token κ, then $\mathbf{c}^{\widehat{\mathbf{W}}}(\kappa) \in \mathbb{R}^k$ is the SVD-reduced indirect context vector representation (using the orthogonal feature set given by the truncated SVD) of token κ, and is defined by either (4.2.15) or (4.2.16) below depending on whether R_1 or R_2 projection is used:

$$\mathbf{c}^{\widehat{\mathbf{W}}}(\kappa) = \mathbf{c}^{R_1(\mathbf{W})}(\kappa) = \mathbf{\Sigma_k U_k}^T \mathbf{c}^1(\kappa) \tag{4.2.15}$$

$$\mathbf{c}^{\widehat{\mathbf{W}}}(\kappa) = \mathbf{c}^{R_2(\mathbf{W})}(\kappa) = \mathbf{U_k}^T \mathbf{c}^1(\kappa) \tag{4.2.16}$$

4.2.3 Token representations via A

The word-segment matrix **A** used in Salton's VSM and LSA represents both word types and text segments. Text segments can be used (and have been used) as proxies to represent tokens. So, matrix **A** is well capable of representing tokens directly, which can in addition be SVD-reduced. Since this matrix also produces word types, it is well possible to use them to compute indirect context vectors, in the same manner customarily done with **W**. This section contemplates these possibilities.

4.2.3.1 D-A-UR: Unreduced segment vectors

Levin et al. (2006) adapted LSA for word-sense disambiguation. In their experiments, they split a sense-tagged corpus into training and test portions from which they construct high-dimensional representations. The representations from the training set are grouped according to the sense tagging of the corpus and the centroids (sense vectors) of the groups formed are used to classify each representation from the test portion of the corpus. Evaluation is done by the accuracy of this classification of the test set. Effectively, Levin et al. are using LSA's segment vectors are proxies for token representations. The unreduced version of these segment vectors is defined formally in Definition 3.2.1 (p. 70).

4.2.3.2 D-A-R1/2: SVD-reduced segment vectors

The vectors used in Levin et al. (2006) were actually SVD-reduced. In this work, the representation of each training instance of w was a projection of a segment vector (column vector) of **A** into reduced space, presumably via R_1 or R_2 (the authors did not specify which) as per equations (3.4.7) or (3.4.8) (p. 82). Then, representations for each test instance of w are constructed by representing each test instance as a column vector that has the same dimensions as rows as **A** and then projecting that vector by R_1 or R_2 (whichever method was used in the training phase) via equations (3.4.5) or (3.4.6) using the SVD matrices produced from the SVD reduction of the matrix **A** containing the training vectors.

This method of projecting vectors on the "segment space" of LSA is the most obvious and natural way to apply LSA for this problem as the segment vector represents an actual instance of the token of the ambiguous word in context. This way of applying LSA is also the most similar to the way it is applied in Information Retrieval, where documents and queries are represented using practically the same segment vectors.

This method is defined formally in Definition 3.4.1 (p. 82).

4.2.3.3 I-A-UR: Unreduced indirect context vectors via A

Pedersen (2010) suggested that an alternative way of using LSA to create a token-based representation is to use the unreduced or reduced word type vectors from **A** and sum them (or alternatively use them to transform first-order context vector) as it is done in the creation of indirect (second-order) context vectors in Word Space. Here we contemplate the method using unreduced word type vectors from the rows of **A**. Subsection 4.2.3.4 does the same but using SVD-reduced versions of those word type vectors instead.

Definition 4.2.6 (Indirect context vector with parameter **A** — I-A-UR)**.** Given a VSM matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ whose rows are the word vectors for words in some chosen vocabulary $\mathcal{V}_m$ with $m = |\mathcal{V}_m|$ and whose columns represent the dimensions (features) of each of those row word vectors corresponding to the segments δ in some corpus $\mathcal{C}_n$ with $n = |\mathcal{C}_n|$ (i.e. the number of segments in the corpus), if $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ is the direct (first-order) context vector representation (using $\mathcal{V}_m$ for features) of token κ, then $\mathbf{c}^{\mathbf{A}}(\kappa) \in \mathbb{R}^n$ is the indirect context vector representation (using the segments in the corpus as its feature set) of token κ, and is defined by:

$$\mathbf{c}^{\mathbf{A}}(\kappa) = \mathbf{A}^T \mathbf{c}^1(\kappa) \qquad (4.2.17)$$

4.2.3.4 I-A-R1/2: SVD-reduced indirect context vectors via A

Given that $\mathbf{A}\mathbf{A}^T$ and **W** are similar, as will be discussed in Section 4.3.2, it can be expected that indirect context vectors created by employing LSA in this way would be similar to the indirect context vectors produced by Word Space. Second-order context vectors computed from word type vectors from LSA's $\widehat{\mathbf{A}}$ (or equivalently from the SVD of $\mathbf{W} + \mathbf{F}$) can be formally stated as follows:

Definition 4.2.7 (Indirect context vector with parameter $\widehat{\mathbf{A}}$ — I-A-R1/2). Given a VSM matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ whose rows are the word vectors for words in some chosen vocabulary $\mathcal{V}_m$ with $m = |\mathcal{V}_m|$ and whose columns represent the dimensions (features) of each of those row word vectors corresponding to the segments δ in some corpus $\mathcal{C}_n$ with $n = |\mathcal{C}_n|$ (i.e. the number of segments in the corpus), let $\widehat{\mathbf{A}} = \mathbf{U_k \Sigma_k V_k}$ be the singular value decomposition of $\mathbf{A}$ truncated to $k \leq \min(m, n)$ dimensions. If $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ is the direct (first-order) context vector representation (using $\mathcal{V}_m$ for features) of token κ, then $\mathbf{c}^{\widehat{\mathbf{A}}}(\kappa) \in \mathbb{R}^k$ is the SVD-reduced indirect context vector representation (using the orthogonal feature set given by the truncated SVD) of token κ, and is defined by either (4.2.18) or (4.2.19) below depending on whether R_1 or R_2 projection is used:

$$\mathbf{c}^{\widehat{\mathbf{A}}}(\kappa) = \mathbf{c}^{R_1(\mathbf{A})}(\kappa) = \mathbf{\Sigma_k U_k}^T \mathbf{c}^1(\kappa) \qquad (4.2.18)$$

$$\mathbf{c}^{\widehat{\mathbf{A}}}(\kappa) = \mathbf{c}^{R_2(\mathbf{A})}(\kappa) = \mathbf{U_k}^T \mathbf{c}^1(\kappa) \qquad (4.2.19)$$

Observe that the R_2 projection in Definition 4.2.7 is equivalent to the R_1 projection in Definition 3.4.1 (p. 82) if $\mathbf{d}$ represents a segment containing κ and if we accept $\mathbf{c}^1(\kappa) \approx \mathbf{d}$ (See Def. 4.3.1).

4.2.4 A (toy) numerical comparison

To summarise, there are two kind of projections that LSA can give: segment projections or word type projections. The natural, obvious way to apply LSA to word-sense discrimination is to use the segment projections, which is what Levin et al. (2006) did. But what Schütze (1998) and Purandare and Pedersen (2004) did in their respective works was to use an indirect approximation of LSA, by computing indirect context vectors based on Word Space's word matrix.

Which one is better? The segment projections option is definitely more natural and obvious. But there is nothing wrong in principle with the method using indirect representations via type projections. Recall the example given by Landauer (2007) which shows that LSA's segment projections are sensitive to the polysemy of the words in the segment and seem to "select" the appropriate sense of each word in the segment (see Section 3.4.4.2 on p. 93). However, the indirect context vectors produced though (4.2.15) or (4.2.16) are also context sensitive. Section 4.3 will perform an analytical comparison between LSA and Word Space and Chapter 6 is dedicated to performing extensive experiments on the effects of these alternative ways of creating token representations from reduced spaces.

We can do toy experiments here with the toy dataset of paper titles and Octave. For all of these "experiments", we shall use $k = 2$ and R_1 in the SVD projections. We will compute cosine measures between different representations of the two tokens of *survey* (*survey*$_{c2}$ and *survey*$_{m4}$) as well as a new token of *survey* which appears in the segment "user survey on the interface quality of the graph minors visualisation system" (*survey*$_x$). Intuitively *survey*$_x$ should be more similar to *survey*$_{c2}$ than *survey*$_{m4}$ since it uses *survey* in its 'questionnaire' sense. Notice however that *graph* and *minors* occurs in this segment, so let us see if this has a negative impact in the representations.

Recall that despite being such a small sample, this toy corpus is useful for visualising the operations involved in unreduced and reduced spaces easily and transparently. Since it has also been used in several other works (Furnas et al., 1988; Deerwester et al., 1990; Kontostathis and Pottenger, 2006), it can be useful to also compare explanations and definitions between works.

Experiment A Let us try first the unreduced token vectors computed from **W** (i.e. "plain vanilla" unreduced second order context vectors):

- $\cos\left(\mathbf{c}^2(\mathit{survey}_{c2}), \mathbf{c}^2(\mathit{survey}_{m4})\right) = 0.14$
- $\cos\left(\mathbf{c}^2(\mathit{survey}_{c2}), \mathbf{c}^2(\mathit{survey}_{x})\right) = 0.90$
- $\cos\left(\mathbf{c}^2(\mathit{survey}_{m4}), \mathbf{c}^2(\mathit{survey}_{x})\right) = 0.47$

$\cos\left(\mathbf{c}^2(\mathit{survey}_{c2}), \mathbf{c}^2(\mathit{survey}_{x})\right)$ is a high value as expected and $\cos\left(\mathbf{c}^2(\mathit{survey}_{m4}), \mathbf{c}^2(\mathit{survey}_{x})\right)$ is not as high but still a considerable value, meaning that some confusion is taking place by the presence of *graph* and *minors*.

Experiment B Let us now see what values we get from the unreduced segment vectors from **A** (column vectors):

- $\cos\left(\mathbf{c2}, \mathbf{m4}\right) = 0.24$
- $\cos\left(\mathbf{c2}, \mathbf{x}\right) = 0.50$
- $\cos\left(\mathbf{m4}, \mathbf{x}\right) = 0.71$

This is a particular bad result because it's saying that the meaning of *survey*$_x$ is more similar to the m4 'literature review' sense than the c2 'questionnaire' sense.

Experiment C Let us now see how LSA through segment projections behave (i.e. à la Levin et al. (2006)):

- $\cos\left(R_1(\mathbf{c2}), R_1(\mathbf{m4})\right) = 0.39$
- $\cos\left(R_1(\mathbf{c2}), R_1(\mathbf{x})\right) = 0.90$
- $\cos\left(R_1(\mathbf{m4}), R_1(\mathbf{x})\right) = 0.75$

This is a bit better. At least the cosine value between *survey*$_x$ and *survey*$_{c2}$ is higher than that between *survey*$_x$ and *survey*$_{m4}$, but a value of 0.75 is still very high.

Experiment D Now, let us see how first-order context vector transformations via SVD-reduced word space ($\mathbf{c}^1(p) \times \mathbf{U}_\mathbf{k}^{(\mathrm{W})}\boldsymbol{\Sigma}_\mathbf{k}^{(\mathrm{W})}$, where $\mathbf{U}_\mathbf{k}^{(\mathrm{W})}$ and $\boldsymbol{\Sigma}_\mathbf{k}^{(\mathrm{W})}$ are the SVD matrices coming from **W**), i.e. pseudo-second-order context vectors (à la Schütze and Purandare and Pedersen) fare:

- $\cos\left(\mathbf{c}^{\widehat{\mathrm{W}}}(\mathit{survey}_{c2}), \mathbf{c}^{\widehat{\mathrm{W}}}(\mathit{survey}_{m4})\right) = 0.23$
- $\cos\left(\mathbf{c}^{\widehat{\mathrm{W}}}(\mathit{survey}_{c2}), \mathbf{c}^{\widehat{\mathrm{W}}}(\mathit{survey}_{x})\right) = 0.98$
- $\cos\left(\mathbf{c}^{\widehat{\mathrm{W}}}(\mathit{survey}_{m4}), \mathbf{c}^{\widehat{\mathrm{W}}}(\mathit{survey}_{x})\right) = 0.42$

This is much better. The cosines between survey_x and survey_{c2} have gone up whilst the cosines between survey_x and survey_{m4} have gone down.

Experiment E As a sanity check, let us compute these cosines using the projection $\mathbf{c}^1(p) \times \mathbf{U}_\mathbf{k}^{(\mathrm{A})}\boldsymbol{\Sigma}_\mathbf{k}^{(\mathrm{A})}$, where $\mathbf{U}_\mathbf{k}^{(\mathrm{A})}$ and $\boldsymbol{\Sigma}_\mathbf{k}^{(\mathrm{A})}$ are the SVD matrices coming from **A**, (i.e. using "the other side of the coin" of LSA). The cosine values should be similar to those in Experiment D, but not quite exactly the same, due to the differences along the diagonal between $\mathbf{AA}^T$ and **W**.

- $\cos\left(\mathbf{c}^{\widehat{\mathrm{A}}}(\mathit{survey}_{c2}), \mathbf{c}^{\widehat{\mathrm{A}}}(\mathit{survey}_{m4})\right) = 0.14$
- $\cos\left(\mathbf{c}^{\widehat{\mathrm{A}}}(\mathit{survey}_{c2}), \mathbf{c}^{\widehat{\mathrm{A}}}(\mathit{survey}_{x})\right) = 0.92$
- $\cos\left(\mathbf{c}^{\widehat{\mathrm{A}}}(\mathit{survey}_{m4}), \mathbf{c}^{\widehat{\mathrm{A}}}(\mathit{survey}_{x})\right) = 0.53$

As expected, these values are not too different from Experiment D. However, notice that these values are actually quite similar to those of Experiment A, (plain vanilla second-order context vectors). So, those different values along the diagonals of the matrices being reduced can play an important part in discrimination.

Experiment F Finally, an experiment that computes the cosines between the first-order context vectors without any reduction whatsoever:

- $\cos\left(\mathbf{c}^1(\mathit{survey}_{c2}), \mathbf{c}^1(\mathit{survey}_{m4})\right) = 0$
- $\cos\left(\mathbf{c}^1(\mathit{survey}_{c2}), \mathbf{c}^1(\mathit{survey}_{x})\right) = 0.40$
- $\cos\left(\mathbf{c}^1(\mathit{survey}_{m4}), \mathbf{c}^1(\mathit{survey}_{x})\right) = 0.63$

As expected, not very good results. However, consider that in a K-Means clustering experiment, context vectors are not computed with each other directly in this way. They are instead compared with a cluster centroid, i.e. a generalised sense vector. So, whilst first-order context vectors might not fare well using direct pairwise comparisons, they can perform much better on a word-sense discrimination experiment based on K-Means. Chapter 5 is indeed devoted to comparing extensively unreduced first-order context vectors with unreduced second-order context vectors and finds that first-order context vectors can perform better than second-order context vectors in WSX experiments using means-based sense vectors.

4.3 A comparison between LSA and Word Space

Both LSA (Deerwester et al., 1990) and Schütze's Word Space (Schütze, 1992, 1998) represent word types as high dimensional vectors. At some point, both LSA and Word Space use SVD as a method of reducing the dimensionality of the high dimensional vectors to a few hundred dimensions. As discussed in Section 3.4.4 (p. 87), SVD in LSA also has the effect of reducing noise and finding higher-order co-occurrences, giving the space some useful semantic qualities such as synonym/semantic relation detection and sensitivity towards polysemous words. Authors such as Schütze (1992, 1998); Purandare and Pedersen (2004) defend their use of SVD on Word Space for word-sense discrimination experiments, based on its semantic benefits when used on LSA.

The matrices being reduced in Word Space and LSA are quite different, however. And so it is reasonable to question whether SVD will be able to replicate the same semantic effects from LSA on Word Space. An analysis of the effects that the application that SVD has on these matrices is presented here. Perhaps the key difference to note is that an unreduced Word Space direct (first-order) context vector counts the words co-occurring with the target token but does not count the target token itself, whilst an unreduced LSA context vector counts every word occurring in a context, including the target token. The reason for this difference is really historical or based in the original applications of the models: Word Space direct context vectors are designed to count co-occurrence with a target word whilst LSA context vectors capture full contexts (or documents), without regard to any particular target word of interest. We also find that in their unreduced form word (type) vectors represented by $\mathbf{A}$ and $\mathbf{W}$ are different and incompatible since LSA word vectors have segment features whilst Word Space word vectors have word features. However, we find that, due to a mathematical equivalence, the SVD-reduced versions of both vector types are approximate to each other, with their difference stemming from the same counting difference we pointed out earlier. Of course, by controlling and manipulating these differences in the early processing stages, it is possible to adapt an LSA system to exactly emulate a WS system and vice versa. Whilst it is relatively easy to demonstrate this equivalence mathematically, it seems that this fact has escaped the literature, causing the two spaces to be treated differently by different authors. Some works treat the models as distinct, sometimes acknowledging a relationship which they often do not explore in detail (e.g. Pedersen, 2007; Cohen and Widdows, 2009). Other works make simplified assumptions on this relationship like Pedersen (2010) and ourselves in Emms and Maldonado-Guerra (2013) by assuming that LSA and Word Space direct context vectors are identical or that their differences are negligible. Yet many other works either are ambiguous on their distinction, assume that the two models are interchangeable or assume that the only difference between them is that LSA uses SVD while Word Space does not (e.g. Levin et al., 2006; Katz and Giesbrecht, 2006; Navigli, 2009).

At the same time, it is noted that many works in the literature are vague when it comes to describing exactly how they apply SVD in experimentation and it is therefore possible that each experimenter has applied SVD in slightly different ways, making even more difficult to see the mathematical equivalence presented here. In the discussion that follows the features in all matrices are raw co-occurrence counts only and, in the case of word matrices from Word Space, only square symmetric matrices that reflect unordered co-occurrences will be considered.

There is a body of works comparing two or more different vector space models. Within the Word Space tradition, there have been works comparing the performance of direct (first-order) vs. indirect (second-order) context vectors in tasks relying on word senses (Purandare and Pedersen, 2004; Peirsman et al., 2008b; Maldonado-Guerra and Emms, 2012), generally concluding that first-order context vectors tend to perform equally or better than second-order context vectors. Pedersen (2010) attempts a comparison between LSA and Word Space, but as mentioned, this work does not capture the subtle difference in counting between each model's direct token representations. Pedersen's work however does compare several Word Space variants on word-sense discrimination experiments, specifically between Word Space direct variants (configurations D-C-* in Table 4.2.1) vs. Word Space indirect variants via word vectors from **W** (I-W-*) vs. Word Space indirect variants via word vectors from **C** (I-C-*), in unreduced and SVD-reduced forms. Pedersen found his unreduced indirect variants to perform best. Sahlgren (2006), Utsumi and Suzuki (2006) and Utsumi (2010) compared the word vectors produced by LSA and Word Space in identifying semantic relations (such as synonymy, hyponymy, collocation, etc.) between word pairs. They found LSA to perform better on collocation prediction and Word Space to perform better on predicting the other types of semantic relations. In one way, the work of Dinu et al. (2012) could be seen as an analogous comparison to the one made here. In this work, the authors compared the mathematical properties and performance of what they call contextualised type vectors, which amount to be different methods of vector composition (Guevara, 2011). By composing (i.e. combining through a linear operation) the type vector of a word τ_1 with the type vector of another word τ_2 found in the vicinity of a token of τ_1, the authors "contextualise" the type vector of τ_1. The authors found that after applying some simplifications, the different methods of semantic composition (and thus contextualisation) turn out to be equivalent or nearly equivalent to each other. Just like this thesis compares two token vector representations (LSA and Word Space) and finds similarities between them, Dinu et al. find near equivalences between diverse methods of vector composition.

4.3.1 A vs W: The difference and relationship between the unreduced spaces of LSA and Word Space

The row vectors of Salton's original VSM matrix (or equivalently, the LSA matrix before SVD) **A** and the unreduced Word Space word matrix **W**, both represent the same thing: word types. So, cosines between any two row vectors from the same matrix reflect a degree of similarity between the two words represented by each vector. As mentioned in Sections 3.3, 3.5 and

3.6 (pages 75, 95 and 102, respectively), cosines in the vector space formed by the rows of **A** measure syntagmatic relations between words whilst cosines in the vector space formed by the rows of **W** (or by its columns, since the matrix is symmetric) measure paradigmatic relations between words. It is that for this reason, **A** is called a syntagmatic vector space and **W** is called a paradigmatic vector space in this thesis.

While the vectors and the values that **A** and **W** hold are different, they are not unrelated. We shall see that they are, in fact, very much related.

Recall that a Word Space direct context vector is a row vector $\mathbf{c}^1 \in \mathbb{R}^m$ in which the i-th dimension $[\mathbf{c}^1]_i$ represents a word type feature $\tau_i \in \mathcal{V}_m$ (see Definition 3.5.2, p. 100). Recall also that the j-th column vector in Salton's VSM matrix **A**, $\mathbf{d_j}$, represents the corpus segment δ_j, with each of its cells (dimensions) a_{ij} counting the frequency of word type τ_i in segment δ_j (see Definition 3.2.2, p. 75). Since both vectors have the same dimensions, we can state that the LSA segment vector $\mathbf{d_j}$ is equivalent to the Word Space direct (first-order) context vector of any word token κ_i occurring in δ_j, adding 1 in the dimension representing the dimension corresponding to the word type τ_i of κ_i as the Word Space direct context vector does not record the occurrence of κ_i but $\mathbf{d_j}$ does. This equivalence holds if the sliding window in $\mathbf{c}^1$ and the segment in $\mathbf{d_j}$ cover the exact same span of text. We can formalise this relationship by defining $\mathbf{d_j}$ vectors in terms of $\mathbf{c}^1$ vectors:

Definition 4.3.1 (Relationship between segment vectors and direct context vectors — D-A-UR and D-C-UR). Given a segment δ_j represented by column vector $\mathbf{d_j}$ in **A**, then we have

$$\mathbf{d_j} = \mathbf{c}^1(\kappa) + \mathbf{g}(\kappa) \tag{4.3.1}$$

where $\mathbf{c}^1(\kappa)$ is the direct (first-order) context vector of a token occurring in δ_j with a sliding window extending to the whole word segment covered by δ_j and $\mathbf{g}(\kappa)$ is a vector of the same dimensions as $\mathbf{c}^1(\kappa)$ containing zeroes in all dimensions, except in the one corresponding to the word type $\tau = \text{type}(\kappa)$ which has the value of 1, i.e. $(\mathbf{g}(\kappa))^T =$

$$\begin{array}{ccccc} g_1 & \dots & g_i{=}\text{type}(\kappa) & \dots & g_m \\ \left[0 \right. & \dots & 1 & \dots & \left. 0 \right] \end{array}.$$

If the corpus is divided in n segments and each column vector $\mathbf{d_j}$ is arranged in such a way as to form the columns of matrix **A**, then **A** is an $m \times n$ matrix (recall that each vector $\mathbf{d_j}$ has m dimensions, one for each word type feature in $\mathcal{V}_m$).

In Word Space, if we have a row word vector $\mathbf{w_i}$ for every word in $\mathcal{V}_m$ and each of these vectors have $\mathcal{V}_m$ for features (i.e. $\mathbf{w_i} \in \mathbb{R}^{1\times m}$ since $m = |\mathcal{V}_m|$) and if each of those row word vectors is arranged in such a way as to form the rows of matrix **W**, then **W** is a square, symmetric matrix of size $m \times m$. Equivalently, we could have arranged every vector $\mathbf{w_i}^T$ as the columns of **W** and we would have arrived at the same matrix.

Notice that in Definition 3.5.3 (p. 101) no reference is made to the segments in which the corpus is divided up. While the segmentation of the corpus is not explicitly used in the definition, it is taking an effect via the first-order context vectors since the segmentation does not allow sliding windows to cross a segmentation boundary. For example, imagine that "No pain. No gain." is a corpus segmented into its two constituent segments ("No pain" is one segment and "No gain" is the other segment). If we define a sliding window of (at most) length 3 centred at each token (ignoring punctuation marks), we would generate the following windows: "$\underline{\text{No}}$ pain", "No $\underline{\text{pain}}$" (from segment 1), "$\underline{\text{No}}$ gain", "No $\underline{\text{gain}}$" (from segment 2). Notice that a window like "pain $\underline{\text{No}}$ gain" cannot be formed, because it would require a segment boundary to be crossed.

There is one further way in which **A** and **W** are related. Recall from Theorem 3.4.2 (p. 80) that when SVD is applied to **A**, the **U** matrix contains the eigenvectors of $\mathbf{AA}^T$ (i.e. the so-called left-singular vectors of **A**) and $\mathbf{\Sigma}$ contains the square roots of its eigenvalues. Conjecture 4.3.2 below describes an equation between this matrix $\mathbf{AA}^T$ and **W**:

Conjecture 4.3.2. *If the sliding word windows in a Word Space are created in such a way as to always be of the same length as Salton's VSM text segments, so that the word window of a token includes all of the other tokens occurring within the same segment (whether this segment is a sentence, paragraph, fixed-length word segment, document, etc.), then the following equation holds:*

$$\mathbf{AA}^T = \mathbf{W} + \mathbf{F} \tag{4.3.2}$$

where F *is an* $m \times m$ *diagonal matrix containing the overall corpus frequency of each word type* τ_i *along its diagonal elements* f_{ii} *and zero everywhere else.*

This equation can be justified intuitively. A (row) word vector $\mathbf{w_i}$ in **W** counts the co-occurrences of τ_i with every other word in the corpus. A row vector $\mathbf{a_i}$ in **A** counts the occurrences of the τ_i in every segment in the corpus. A row vector $\mathbf{m_i}$ in $\mathbf{AA}^T$ (or column vector since the matrix is symmetric) essentially multiplies the occurrences of τ_i with the occurrences of every other word in each document in the corpus, and then sums the resulting products in order to get the total number of co-occurrences of τ_i with every other word in the corpus. Along the diagonal, however, $\mathbf{AA}^T$ counts each occurrence of τ_i as if it were co-occurring with itself, in addition to real co-occurrences. As an example, imagine that a segment has the token sequence "a b c b", producing the segment vector $\mathbf{d}_1{}^T = [1, 2, 1]$, where each dimension respectively corresponds to frequencies of types *a*, *b* and *c*. When computing $\mathbf{d}_1\mathbf{d}_1{}^T$, we obtain the following 3×3 matrix:

$$\mathbf{d}_1\mathbf{d}_1{}^T = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{4.3.3}$$

Let m_{ij} denote an entry in this matrix. As previously noted, m_{ij} is counting the co-occurrences between word τ_i and τ_j in the corpus, which in this simple case consists of just one segment.

For example, m_{12} counts how many times a and b co-ocurr, which is two, since b occurs twice and each occurrence co-occurs with the single occurrence of a. However, notice that the values on the diagonal are $[1, 4, 1]$ and the only self co-occurrence (i.e. the only type with more than one token featured) in the segment is b, and this co-occurrence happens only twice and not four times as suggested by m_{22}. Additionally, both a and c are recorded as co-occurring once with themselves, something that is not possible, since the minimum value for a self co-occurrence is 2. If a word occurs only once in a segment or word window, its self co-occurrence value is 0. If it occurs twice, then its self co-occurrence value is 2, since each occurrence co-occurs with the other co-occurrence, as we have seen in the case of b in the example above[1]. The diagonal in $\mathbf{d}_1\mathbf{d}_1{}^T$ is therefore counting both self co-occurrences as well as the raw frequencies of words in the corpus. By contrast, the diagonal in the word matrix, **W**, only counts self co-occurrences. The word matrix for the example segment above (in which the sliding word window for each token extends to the whole segment) is:

$$\mathbf{W} = \begin{bmatrix} 0 & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & 0 \end{bmatrix} \tag{4.3.4}$$

whose diagonal values are $[0, 2, 0]$, reflecting the true self co-occurrence counts in the example. The **F** matrix that would equate **W** and $\mathbf{d}_1\mathbf{d}_1{}^T$ (or $\mathbf{AA}^T$) via Conjecture 4.3.2 is the diagonal matrix containing the frequencies of each word in the example, $\mathbf{F} = \text{diag}(1, 2, 1)$.

Despite this difference between $\mathbf{AA}^T$ and **W**, it is worth mentioning that row (or column) vectors in both matrices form paradigmatic spaces. That is, a cosine measure between any two row vectors (from the same matrix) measures to what degree the words they represent co-occur with the same words. Whilst **A** on its own is a syntagmatic space, $\mathbf{AA}^T$ is a paradigmatic space, similar to that described by **W**. Recall that we previously mentioned that the SVD of **A** computes eigenvalues and eigenvectors for $\mathbf{AA}^T$. Word type and segment vectors are projected using matrices containing these eigenvalues and eigenvectors (see Section 3.4.2 on p. 81). It is perhaps at least in part due to this that these projected word type and segment vectors manifest some semantic (paradigmatic) properties as discussed in Section 3.4.4 (p. 87).

4.3.2 Decomposition of W

At the core of LSA there is a matrix decomposition process via SVD. A related, but more fundamental way of matrix decomposition is the eigendecomposition presented in Theorem 3.4.1 (p. 79). Its main drawback, however, is that it can only decompose diagonalisable matrices. Because Salton's VSM matrix **A** is not guaranteed to be diagonalisable, SVD is used in LSA as it can be applied to arbitrary matrices, like **A**. Roughly speaking, SVD computes the "square of the matrix", $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$, and decomposes that. Those "matrix squares" are

[1] In general, the number of self co-occurrences for a word is the number of 2-permutations of the word frequency f in the sequence or word window, i.e. $\frac{f!}{(f-2)!}$

Table 4.3.1: Word Space matrix **W**

W	hu.	in.	co.	us.	sy.	re.	ti.	EP.	su.	tr.	gr.	mi.
human		1	1		2			1				
interface	1		1	1	1			1				
computer	1	1		1	1	1	1		1			
user		1	1		2	2	2	1	1			
system	2	1	1	2	2	1	1	3	1			
response			1	2	1		2		1			
time			1	2	1	2			1			
EPS	1	1		1	3							
survey			1	1	1	1	1				1	1
trees											2	1
graph									1	2		2
minors									1	1	2	

symmetric and all symmetric matrices are diagonalisable by orthogonal matrices.

The word matrix **W** is already symmetric and can therefore be decomposed directly via regular eigendecomposition:

$$\mathbf{W} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} \tag{4.3.5}$$

where $\mathbf{Q} \in \mathbb{R}^{m\times m}$ is a matrix whose columns are the eigenvectors of **W** and $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements are the corresponding eigenvalues of **W**.

Recall from Conjecture 4.3.2 that **W** and $\mathbf{AA}^T$ are similar matrices. Because of this, their eigenvectors **Q** (for **W**) and **U** (for $\mathbf{AA}^T$) should also be similar.

One way to test this is by computing some SVDs and eigendecompositions on matrices **A** and **W** derived from a toy dataset like that by Deerwester et al. (1990), which was presented in Section 3.4.4 (p. 87). Assuming that each title itself is one segment and that first-order context vectors will be constructed from this corpus from word windows that cover the full segment (i.e. the title) centred at each target word, this dataset, allows us to construct a type-by-segment matrix $\mathbf{A} \in \mathbb{R}^{12\times 9}$, which can be seen in Table 3.4.1 (p. 87) and a type-by-type matrix $\mathbf{W} \in \mathbb{R}^{12\times 12}$, which is shown in Table 4.3.1.

What follows are matrix dumps and code snippets from GNU Octave[2], a software environment that offers convenient ways of computing eigendecompositions and SVDs.

The full, rank-9 SVD matrices of **A** are:

```
octave:13> [Ua, Sa, Va] = svd(A, 9)
Ua =
  -0.22  -0.11   0.29  -0.41  -0.11  -0.34  -0.52   0.06   0.41
  -0.20  -0.07   0.14  -0.55   0.28   0.50   0.07   0.01   0.11
  -0.24   0.04  -0.16  -0.59  -0.11  -0.25   0.30  -0.06  -0.49
  -0.40   0.06  -0.34   0.10   0.33   0.38  -0.00   0.00  -0.01
```

[2]https://www.gnu.org/software/octave/

```
  -0.64  -0.17   0.36   0.33  -0.16  -0.21   0.17  -0.03  -0.27
  -0.27   0.11  -0.43   0.07   0.08  -0.17  -0.28   0.02   0.05
  -0.27   0.11  -0.43   0.07   0.08  -0.17  -0.28   0.02   0.05
  -0.30  -0.14   0.33   0.19   0.11   0.27  -0.03   0.02   0.17
  -0.21   0.27  -0.18  -0.03  -0.54   0.08   0.47   0.04   0.58
  -0.01   0.49   0.23   0.02   0.59  -0.39   0.29  -0.25   0.23
  -0.04   0.62   0.22   0.00  -0.07   0.11  -0.16   0.68  -0.23
  -0.03   0.45   0.14  -0.01  -0.30   0.28  -0.34  -0.68  -0.18

Sa =
Diagonal Matrix
  3.34     0     0     0     0     0     0     0     0
     0  2.54     0     0     0     0     0     0     0
     0     0  2.35     0     0     0     0     0     0
     0     0     0  1.64     0     0     0     0     0
     0     0     0     0  1.50     0     0     0     0
     0     0     0     0     0  1.31     0     0     0
     0     0     0     0     0     0  0.85     0     0
     0     0     0     0     0     0     0  0.56     0
     0     0     0     0     0     0     0     0  0.36
     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0


Va =
  -0.197  -0.056   0.110  -0.950   0.046  -0.077  -0.177   0.014   0.064
  -0.606   0.166  -0.497  -0.029  -0.206  -0.256   0.433  -0.049  -0.243
  -0.463  -0.127   0.208   0.042   0.378   0.724   0.237  -0.009  -0.024
  -0.542  -0.232   0.570   0.268  -0.206  -0.369  -0.265   0.019   0.084
  -0.279   0.107  -0.505   0.150   0.327   0.035  -0.672   0.058   0.262
  -0.004   0.193   0.098   0.015   0.395  -0.300   0.341  -0.454   0.620
  -0.015   0.438   0.193   0.016   0.349  -0.212   0.152   0.762  -0.018
  -0.024   0.615   0.253   0.010   0.150   0.000  -0.249  -0.450  -0.520
  -0.082   0.530   0.079  -0.025  -0.602   0.362  -0.038   0.070   0.454
```

The squares of the diagonal elements in $\mathbf{\Sigma}$ of **A** (`Sa` in Octave) are:

```
octave:18> SSa = (diag(Sa) .^ 2)'
SSa =
  11.16   6.46   5.54   2.70   2.26   1.71   0.72   0.31   0.13
```

We now compute the eigendecomposition of **W**. Octave's `eig` function performs the eigendecomposition operation. However it does not sort eigenvectors and eigenvalues like `svd` does. So we need to rearrange matrices after computation:

```
octave:17> [Qw, Lw] = eig(W);
octave:18> dlw = diag(Lw);
octave:19> [v, i] = sort(abs(dlw), 'descend');
octave:20> diag(dlw(i, :)) % Lw (eigenvalues) sorted by descending abs value of eigenvalue
ans =
Diagonal Matrix
  8.26     0     0     0     0     0     0     0     0     0     0     0
     0  3.84     0     0     0     0     0     0     0     0     0     0
     0     0  3.11     0     0     0     0     0     0     0     0     0
     0     0     0 -2.66     0     0     0     0     0     0     0     0
     0     0     0     0 -2.50     0     0     0     0     0     0     0
     0     0     0     0     0 -2.41     0     0     0     0     0     0
     0     0     0     0     0     0 -2.00     0     0     0     0     0
     0     0     0     0     0     0     0 -1.89     0     0     0     0
     0     0     0     0     0     0     0     0 -1.43     0     0     0
```

```
       0      0      0      0      0      0      0      0      0  -0.61      0      0
       0      0      0      0      0      0      0      0      0      0   0.49      0
       0      0      0      0      0      0      0      0      0      0      0  -0.20

% We also display Qw with its columns sorted by their corresponding eigenvalue (abs value descending)
octave:21> Qw(:,i)
ans =
   0.23  -0.19   0.31  -0.35  -0.06   0.02  -0.00  -0.35   0.41  -0.47   0.37   0.16
   0.22  -0.13   0.18   0.28   0.17   0.00  -0.00  -0.28   0.08   0.56   0.47  -0.41
   0.27   0.05  -0.13  -0.03  -0.05  -0.13   0.00   0.59  -0.33  -0.15   0.63   0.07
   0.41   0.06  -0.25  -0.50  -0.51   0.30  -0.00   0.01   0.01   0.25  -0.15  -0.28
   0.57  -0.20   0.28   0.54  -0.17   0.28   0.00   0.07  -0.13  -0.15  -0.29   0.18
   0.30   0.16  -0.41   0.14   0.22  -0.19  -0.71  -0.06   0.24  -0.17  -0.08  -0.12
   0.30   0.16  -0.41   0.14   0.22  -0.19   0.71  -0.06   0.24  -0.17  -0.08  -0.12
   0.31  -0.23   0.35  -0.39   0.36  -0.49   0.00   0.22  -0.08   0.17  -0.34  -0.05
   0.23   0.32  -0.11  -0.16   0.25   0.08  -0.00  -0.43  -0.38   0.24   0.07   0.58
   0.01   0.41   0.28  -0.11   0.33   0.34   0.00  -0.06  -0.34  -0.37  -0.04  -0.51
   0.04   0.55   0.31   0.15  -0.50  -0.55  -0.00  -0.15  -0.04  -0.01  -0.01  -0.06
   0.04   0.48   0.26  -0.01   0.17   0.28   0.00   0.42   0.56   0.25   0.00   0.22
```

We cannot say that the eigenvalues of **W** (`Lw` in Octave) are too similar to the squares of the singular values of **A** (the eigenvalues of $\mathbf{AA}^T$), `SSa` in Octave. However, **W**'s eigenvectors in `Qw` are very similar to **A**'s left-singular vectors `Ua`. However, notice that the sign of some columns have been flipped. Observe as well that some columns corresponding to the higher rank eigenvalues appear in different positions in `Ua` and `Qw`. For example, column 4 in `Ua` corresponds to column 11 in `Qw`, column 5 in `Ua` to column 12 in `Qw`, column 6 in `Ua` to column 10 in `Qw`, etc. This difference in column ordering is due to the differences in the eigenvalues associated to each matrix being significant enough to cause this difference in ordering. In practice, this difference in column ordering would have no effect as the order of dimensions in vectors do not affect the cosine measure or any distance measure. Notice as well that columns 6, 7, 8 and 9 in `Ua` become increasingly difficult to match with columns in `Qw`. The approximation seems to drop as we move to higher ranks. This is more likely to have an impact on the performance of context vectors derived from each of these matrices.

W is not completely equal to $\mathbf{AA}^T$ as the matrices differ in their diagonals, as already shown through Conjecture 4.3.2. So, let us now eigendecompose $\mathbf{W} + \mathbf{F}$ instead:

```
octave:45> [Qwf, Lwf] = eig(W + F);
octave:47> dlwf = diag(Lwf);
octave:48> [v, i] = sort(abs(dlwf), 'descend');
octave:49> diag(dlwf(i, :)) % Lwf (eigenvalues) sorted by descending abs value of eigenvalue
ans =
Diagonal Matrix
  11.16      0      0      0      0      0      0      0      0      0      0      0
      0   6.46      0      0      0      0      0      0      0      0      0      0
      0      0   5.54      0      0      0      0      0      0      0      0      0
      0      0      0   2.70      0      0      0      0      0      0      0      0
      0      0      0      0   2.26      0      0      0      0      0      0      0
      0      0      0      0      0   1.71      0      0      0      0      0      0
      0      0      0      0      0      0   0.72      0      0      0      0      0
      0      0      0      0      0      0      0   0.31      0      0      0      0
      0      0      0      0      0      0      0      0   0.13      0      0      0
      0      0      0      0      0      0      0      0      0  -0.00      0      0
      0      0      0      0      0      0      0      0      0      0   0.00      0
      0      0      0      0      0      0      0      0      0      0      0  -0.00
```

```
% We also display Qwf with its columns sorted by their corresponding eigenvalue (abs value descending):
octave:50> Qwf(:,i)
ans =
   0.22 -0.11  0.29 -0.41 -0.11 -0.34 -0.52 -0.06 -0.41  0.23  0.09  0.23
   0.20 -0.07  0.14 -0.55  0.28  0.50  0.07 -0.01 -0.11 -0.17  0.19 -0.47
   0.24  0.04 -0.16 -0.59 -0.11 -0.25  0.30  0.06  0.49 -0.06 -0.28  0.25
   0.40  0.06 -0.34  0.10  0.33  0.38 -0.00 -0.00  0.01  0.46  0.18  0.45
   0.64 -0.17  0.36  0.33 -0.16 -0.21  0.17  0.03  0.27  0.06  0.28 -0.25
   0.27  0.11 -0.43  0.07  0.08 -0.17 -0.28 -0.02 -0.05  0.25 -0.49 -0.56
   0.27  0.11 -0.43  0.07  0.08 -0.17 -0.28 -0.02 -0.05 -0.71  0.31  0.11
   0.30 -0.14  0.33  0.19  0.11  0.27 -0.03 -0.02 -0.17 -0.35 -0.66  0.27
   0.21  0.27 -0.18 -0.03 -0.54  0.08  0.47 -0.04 -0.58  0.00  0.00 -0.00
   0.01  0.49  0.23  0.02  0.59 -0.39  0.29  0.25 -0.23  0.00  0.00  0.00
   0.04  0.62  0.22  0.00 -0.07  0.11 -0.16 -0.68  0.23 -0.00 -0.00 -0.00
   0.03  0.45  0.14 -0.01 -0.30  0.28 -0.34  0.68  0.18  0.00 -0.00  0.00
```

The eigenvalues of $\mathbf{W} + \mathbf{F}$ (`Lwf`) are identical to the squares of the singular values of $\mathbf{A}$ (`SSa`), with the exception of the last three values. Also, the eigenvectors of $\mathbf{W} + \mathbf{F}$ (`Qwf`) are also identical to the left-singular values of $\mathbf{A}$ (`Ua`), with the exception of the last three columns and the flipped signs in some columns.

LSA can be defined as the process in which a word-segment matrix $\mathbf{A}$ is decomposed by an SVD truncated to dimensionality k and the type vectors of $\mathbf{A}$ are projected into the vector space formed by this truncated dimensionality. If we define as "Eigenreduced Word Space" the process of decomposing a word type-by-word type matrix $\mathbf{W}$, which has been computed from sums of first-order token vectors for each token in each segment and consider the whole segment as the context for the token, by an eigendecomposition truncated to dimensionality k and the type vectors of $\mathbf{W}$, then we can say that this Eigenreduced Word Space is equivalent to LSA, given what we have seen so far. Notice however that while LSA can also project segment vectors into the reduced space, Eigenreduced Word Space cannot project segment vectors, simply because they are not represented at all in the original Word Space word matrix.

As mentioned previously, previous work (like Schütze, 1998; Purandare and Pedersen, 2004, just to name two) perform matrix decomposition and dimensionality reduction on $\mathbf{W}$ by SVD and not by eigendecomposition. From what we have seen so far, we know that decomposing such a Word Space word matrix by SVD *should* produce matrices $\mathbf{U} = \mathbf{V}$ containing the eigenvectors of $\mathbf{W}\mathbf{W}^T$ or $\mathbf{W}^T\mathbf{W}$ (since $\mathbf{W}\mathbf{W}^T = \mathbf{W}^T\mathbf{W}$ because $\mathbf{W}$ is symmetric). If we are willing to accept that $\mathbf{A}\mathbf{A}^T \approx \mathbf{W}$, then in terms of $\mathbf{A}$, the eigenvectors that we are obtaining are roughly those of $\mathbf{W}\mathbf{W}^T \approx \mathbf{A}\mathbf{A}^T\mathbf{A}\mathbf{A}^T$. It is therefore reasonable to argue that the SVD being performed to $\mathbf{W}$ has nothing to do with the SVD done to the matrix $\mathbf{A}$ involved in LSA, since both SVDs are computing the eigenvectors of different (although still related) matrices. It turns out, however, that the SVD of $\mathbf{W}$ is actually almost identical to its direct eigendecomposition. The consequence of this is that since $\mathbf{A}\mathbf{A}^T \approx \mathbf{W}$, then the SVD of $\mathbf{W}$ should also be approximate to that of $\mathbf{A}$. This implies that many of the Word Space experiments in the literature that use an SVD-reduced have been performing, perhaps unwittingly, to an approximation of LSA.

Let us state more formally this equivalence of SVD and eigendecomposition for symmetric matrices of real values.

Theorem 4.3.1. *Let* $\mathbf{W} \in \mathbb{R}^{m \times m}$. *If* $\mathbf{W} = \mathbf{W}^T$, *then*

$$\mathbf{W} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{4.3.6}$$

Furthermore,

$$\mathbf{U} = \mathbf{Q} \tag{4.3.7}$$

and $\mathbf{U}$ *and* $\mathbf{V}$ *are almost identical, differing only in the signs of their columns:*

$$\mathbf{v_i} = \text{sign}(\lambda_{ii})\mathbf{u_i} \tag{4.3.8}$$

where λ_{ii} *is the i-th eigenvalue in* $\mathbf{\Lambda}$ *diagonal and* $\mathbf{u_i}$ *and* $\mathbf{v_i}$ *are the corresponding column vectors in* $\mathbf{U}$ *and* $\mathbf{V}$, *respectively. The relationship between the singular values* σ_{ii} *in* $\mathbf{\Sigma}$ *and the eigenvalues* λ_{ii} *in* $\mathbf{\Lambda}$ *is given by*

$$\sigma_{ii} = |\lambda_{ii}| \tag{4.3.9}$$

Proof. Because $\mathbf{W} \in \mathbb{R}^{m \times m}$ is a symmetric matrix, its eigendecomposition (4.3.10) is defined. And since a singular value decomposition exists for arbitrary matrices, (4.3.11) is also defined for $\mathbf{W}$:

$$\begin{aligned} \mathbf{W} &= \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T && (4.3.10) \\ \mathbf{W} &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T && (4.3.11) \end{aligned}$$

This proves the equality (4.3.6). Since $\mathbf{W} = \mathbf{W}^T$, we have $\mathbf{W}\mathbf{W}^T = \mathbf{W}^T\mathbf{W} = \mathbf{W}\mathbf{W} = \mathbf{W}^2$. By inserting the eigendecomposition definition from Theorem 3.4.1 (p. 79) in these equations we have:

$$\begin{aligned} \mathbf{W}\mathbf{W}^T &= \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T)^T && (4.3.12) \\ &= \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \\ &= \mathbf{Q}\mathbf{\Lambda}^2\mathbf{Q}^T \end{aligned}$$

$$\begin{aligned} \mathbf{W}^T\mathbf{W} &= (\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T)^T\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T && (4.3.13) \\ &= \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \\ &= \mathbf{Q}\mathbf{\Lambda}^2\mathbf{Q}^T \end{aligned}$$

Similarly, by inserting the SVD definition from Theorem 3.4.2 (p. 80) we get:

$$\begin{aligned} \mathbf{W}\mathbf{W}^T &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T \\ &= \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T \end{aligned} \tag{4.3.14}$$

$$\begin{aligned} \mathbf{W}^T\mathbf{W} &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{\Sigma}\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \end{aligned} \tag{4.3.15}$$

We therefore have

$$\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T = \mathbf{Q}\mathbf{\Lambda}^2\mathbf{Q}^T = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \tag{4.3.16}$$

One of the properties of eigenvalues is that if $\lambda_1, \lambda_2, ..., \lambda_n$ are the eigenvalues of some matrix $\mathbf{M}$, the eigenvalues of $\mathbf{M}^k$ for any positive integer k are $\lambda_1^k, \lambda_2^k, ..., \lambda_n^k$. Since $\mathbf{\Sigma}$ has the square roots of the eigenvalues of $\mathbf{W}^2$, it would be well possible to equate it with $\mathbf{\Lambda}$, which contains the eigenvalues of $\mathbf{W}$ (i.e. $\mathbf{\Lambda} = \mathbf{\Sigma}$). And from (4.3.16) and the fact that $\mathbf{W}\mathbf{W}^T$ and $\mathbf{W}^T\mathbf{W}$ have the same eigenvectors, we could also equate $\mathbf{Q} = \mathbf{U} = \mathbf{V}$. However, whilst $\mathbf{W}\mathbf{W}^T$ and $\mathbf{W}^T\mathbf{W}$ are guaranteed to be non-negative definite symmetric matrices (see e.g. App. A in Lawson and Hanson, 1974), $\mathbf{W}$ is not guaranteed to be non-negative definite. This means that $\mathbf{W}$ can have negative eigenvalues in $\mathbf{\Lambda}$. As a convention from Theorem 3.4.2, the values in $\mathbf{\Sigma}$ are non-negative. So, instead of having $\mathbf{\Lambda} = \mathbf{\Sigma}$, the relationship between $\mathbf{\Lambda}$ and $\mathbf{\Sigma}$ is given by (4.3.9). In order to preserve the equality in (4.3.16) we must disallow a sign equality between $\mathbf{U}$ and $\mathbf{V}$. One way of doing this is by stating (4.3.7) and establishing (4.3.8) as the relationship between $\mathbf{U}$ and $\mathbf{V}$. □

This can be shown numerically through Octave as well. Recall that the eigenvalues of the toy corpus' $\mathbf{W}$ (`Lw` in Octave above) are

$$\text{diag}(\mathbf{\Lambda}) = [8.26, 3.84, 3.11, -2.66, -2.50, -2.41, -2.00, -1.89, -1.43, -0.61, 0.49, -0.20] \tag{4.3.17}$$

The SVD matrices of $\mathbf{W}$ can be seen in the Octave code dump below. Notice that the absolute values in $\mathbf{U}$ and $\mathbf{V}$ (`Uw` and `Vw`, respectively in the Octave code below) are identical. However, the sign in the columns of these matrices are flipped for all columns except the first three and the 11th column. Observe that the columns with flipped signs correspond to the negative eigenvalues in diag($\mathbf{\Lambda}$) and the columns with identical signs correspond to the positive eigenvalues. This behaviour is consistent with (4.3.8). Notice as well that (4.3.9) also describes the relationship between diag($\mathbf{\Lambda}$) and diag($\mathbf{\Sigma}$) below (diagonal values in `Sw`).

```
octave:54> [Uw, Sw, Vw] = svd(W)
Uw =
  -0.23  -0.19  -0.31  0.35 -0.06  0.02  0.00 -0.35  0.41  0.47  0.37 -0.16
  -0.22  -0.13  -0.18 -0.28  0.17  0.00 -0.00 -0.28  0.08 -0.56  0.47  0.41
  -0.27   0.05   0.13  0.03 -0.05 -0.13  0.00  0.59 -0.33  0.15  0.63 -0.07
  -0.41   0.06   0.25  0.50 -0.51  0.30 -0.00  0.01  0.01 -0.25 -0.15  0.28
  -0.57  -0.20  -0.28 -0.54 -0.17  0.28  0.00  0.07 -0.13  0.15 -0.29 -0.18
  -0.30   0.16   0.41 -0.14  0.22 -0.19  0.71 -0.06  0.24  0.17 -0.08  0.12
  -0.30   0.16   0.41 -0.14  0.22 -0.19 -0.71 -0.06  0.24  0.17 -0.08  0.12
  -0.31  -0.23  -0.35  0.39  0.36 -0.49 -0.00  0.22 -0.08 -0.17 -0.34  0.05
  -0.23   0.32   0.11  0.16  0.25  0.08  0.00 -0.43 -0.38 -0.24  0.07 -0.58
  -0.01   0.41  -0.28  0.11  0.33  0.34  0.00 -0.06 -0.34  0.37 -0.04  0.51
  -0.04   0.55  -0.31 -0.15 -0.50 -0.55 -0.00 -0.15 -0.04  0.01 -0.01  0.06
  -0.04   0.48  -0.26  0.01  0.17  0.28  0.00  0.42  0.56 -0.25  0.00 -0.22

Sw =
Diagonal Matrix
  8.26     0     0     0     0     0     0     0     0     0     0     0
     0  3.84     0     0     0     0     0     0     0     0     0     0
     0     0  3.11     0     0     0     0     0     0     0     0     0
     0     0     0  2.66     0     0     0     0     0     0     0     0
     0     0     0     0  2.50     0     0     0     0     0     0     0
     0     0     0     0     0  2.41     0     0     0     0     0     0
     0     0     0     0     0     0  2.00     0     0     0     0     0
     0     0     0     0     0     0     0  1.89     0     0     0     0
     0     0     0     0     0     0     0     0  1.43     0     0     0
     0     0     0     0     0     0     0     0     0  0.61     0     0
     0     0     0     0     0     0     0     0     0     0  0.49     0
     0     0     0     0     0     0     0     0     0     0     0  0.20

Vw =
  -0.23 -0.19 -0.31 -0.35  0.06 -0.02  0.00  0.35 -0.41 -0.47  0.37  0.16
  -0.22 -0.13 -0.18  0.28 -0.17 -0.00 -0.00  0.28 -0.08  0.56  0.47 -0.41
  -0.27  0.05  0.13 -0.03  0.05  0.13  0.00 -0.59  0.33 -0.15  0.63  0.07
  -0.41  0.06  0.25 -0.50  0.51 -0.30 -0.00 -0.01 -0.01  0.25 -0.15 -0.28
  -0.57 -0.20 -0.28  0.54  0.17 -0.28 -0.00 -0.07  0.13 -0.15 -0.29  0.18
  -0.30  0.16  0.41  0.14 -0.22  0.19 -0.71  0.06 -0.24 -0.17 -0.08 -0.12
  -0.30  0.16  0.41  0.14 -0.22  0.19  0.71  0.06 -0.24 -0.17 -0.08 -0.12
  -0.31 -0.23 -0.35 -0.39 -0.36  0.49  0.00 -0.22  0.08  0.17 -0.34 -0.05
  -0.23  0.32  0.11 -0.16 -0.25 -0.08 -0.00  0.43  0.38  0.24  0.07  0.58
  -0.01  0.41 -0.28 -0.11 -0.33 -0.34 -0.00  0.06  0.34 -0.37 -0.04 -0.51
  -0.04  0.55 -0.31  0.15  0.50  0.55  0.00  0.15  0.04 -0.01 -0.01 -0.06
  -0.04  0.48 -0.26 -0.01 -0.17 -0.28 -0.00 -0.42 -0.56  0.25  0.00  0.22
```

4.4 Summary

This chapter analysed in some detail the relationship (similarities and differences) between LSA and Word Space. It also analysed the relationship between the different components of Word Space (direct context vectors, indirect context vectors and the word matrix) in more detail. It also analysed the properties of the SVD-reduced token representations in LSA and Word Space (direct and indirect vectors). The main points are summarised as follows.

First, the Word Space word matrix can be seen as a linear transformation matrix that can be used to convert direct context vectors into indirect (second-order) context vectors. Since the dimensions of word vectors can be different to the dimensions of direct context vectors, the word matrix can be itself used as a method of dimensionality reduction (or feature transformation), an idea that is further explored in Chapter 7.

Second, The square of Salton's VSM, the matrix used in LSA, is very similar to the word matrix. And this relationship can be explicitly and easily stated. The implication of this is that when Salton's VSM and the word matrix are reduced by SVD, they produce similar word type spaces. Since the relationship between the unreduced VSM and word matrices is known, it is possible to compute an LSA word type space via a word matrix or vice versa.

Third, token representations from VSM/LSA and Word Space/SVD-reduced Word Space via direct and indirect context vectors are also related, but this relationship is somewhat more complicated. Unreduced segment (token) vectors from VSM are similar to unreduced direct context vectors. Again, the exact relationship between the two is known. Because of this, reduced segment vectors (LSA) and directly SVD-reduced direct context vectors are also related. These reduced representations have already been employed in WSX experiments: LSA segment vectors were used by Levin et al. (2006) and SVD-reduced Word Space direct context vectors were used by Emms and Maldonado-Guerra (2013). The reduced indirect context vectors are computed from an SVD-reduced word matrix. Schütze (1998) and Purandare and Pedersen (2004) conducted word-sense discrimination experiments using variants of these reduced indirect context vectors. There is a relationship between indirect context vectors and LSA that has not been explored in detail yet. These are indirect or "second-context" context vectors computed from the word type vectors of LSA's VSM (after SVD reduction). Chapter 6 in this thesis is dedicated to comparing the performance of these different definitions of SVD-reduced direct and indirect context vectors on word-sense disambiguation and discrimination experiments. Chapter 5 explores the performance on unreduced direct and indirect context vectors.

Table 4.2.1 summarises the methods of token vector construction and categorises them. The upper sub-table shows the direct token representations (i.e. first-order token representations) from LSA and Word Space. The lower sub-table shows their indirect token representation counterparts (i.e. the second order representations).

5 WSX experiments: direct vs. indirect Word Space token spaces

Both direct (first-order) and indirect (second-order) context vectors aim to represent the same object: an instance of a target word in context. While direct context vectors seek to operationalise the distributional hypothesis directly by simply counting the words occurring in a particular context, indirect context vectors instead sum the word vectors of the words that occur in that context.

There are diverse intuitions/motivations for using indirect context vectors, rather than their direct counterparts. One intuition is that direct vectors will tend to be sparse, having zeros on all but a small number of dimensions and that indirect vectors will tend to be less so (Schütze, 1998). A related intuition is that by finding indirect co-occurrences, indirect context vectors can help identifying similarities amongst instances that are conceptually related but that do not necessarily share the exact same set of words (Purandare and Pedersen, 2004), an argument similar to that used to justify the use of LSA. Consequently, if the corpus on which word-sense discrimination is done is relatively small, clustering could benefit from indirect context vectors built from word vectors computed on a much larger corpus. These are intuitions only and there has not been a great deal of work systematically comparing the two context representations. Purandare and Pedersen (2004) made a comparison of (a certain variant of) direct context vectors with (a certain variant of) indirect context vectors. Their conclusion coincides with the intuition that indirect vectors out-perform direct vectors when the corpus is small (a few hundred instances). But they also concluded that direct vectors out-perform indirect vectors when there are many examples ($\approx 4,000$).

The easiest way of comparing direct and indirect context vectors is to simply run word-sense disambiguation and/or discrimination experiments and compare results from direct context vectors with results from indirect context vectors, and indeed this has been done (with certain differences) both in Purandare and Pedersen (2004) and Maldonado-Guerra and Emms (2012). However, since the general philosophy of the vector space model is to model meanings with vectors, and to model semantic relations with geometrical notions based on these vectors, it is also reasonable to consider ways to compare the geometry of sets of modelled meanings under the direct and indirect approaches. Given a corpus of sense-labelled occurrences, all tokens instantiating a particular sense σ_i of a particular word τ can grouped in the set $\mathcal{S}_i$, and from this set direct and indirect sense vectors, $\mathfrak{s}^1{}_i$ and $\mathfrak{s}^2{}_i$, can be computed – once all relevant direct context vectors, $\mathbf{c}^1$, and indirect context vectors, $\mathbf{c}^2$, have been computed.

```
<instance id="hard-a.sjm-274_1:">
        <answer instance="hard-a.sjm-274_1:" senseid="HARD1"/>
        <context>
        <s>
                " He may lose all popular support ,
                but someone has to kill him to
                defeat him and that 's
                <head>HARD</head> to do . "
        </s>
        </context>
</instance>

<instance id="hard-a.t12_1:">
        <answer instance="hard-a.t12_1:" senseid="HARD3"/>
        <context>
        <s>
                Water becomes stiff and
                <head>HARD</head> as clear stone .
        </s>
        </context>
</instance>
```

Figure 5.1.1: Hard subcorpus sample

By studying geometric properties of these sense vectors, we should be able to predict their performance in a word-sense disambiguation or discrimination setting.

This chapter is divided into two main sections: Section 5.2 describes the geometric experiments performed and Section 5.3 covers the word-sense disambiguation and discrimination experiments done to give a perspective to the geometric experiments. Section 5.5 concludes and compares our work with that of Purandare and Pedersen (2004). As a preamble, Section 5.1 describes the corpora used in this chapter's experiments.

5.1 Corpora

The geometric and WSX experiments presented in this chapter were conducted using two datasets. One is the *hard-interest-line-serve* dataset (henceforth called "the HILS dataset") widely used in the word-sense disambiguation literature[1]. The second dataset is a larger corpus of untagged articles from the New York Times (NYT) from the 1998-2000 period[2].

For each target word (*hard*, *interest*, *line* and *serve*), the HILS dataset contains sense-tagged

[1]This dataset is in the public domain and can be freely downloaded from http://www.d.umn.edu/~tpederse/data.html

[2]The NYT articles are part of the AQUAINT Corpus: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T31

short context samples from newspaper articles written between 1987 and 1991. Figure 5.1.1 is a sample of two contexts in the *hard* subcorpus. For each word in the corpus there are 3 to 6 senses. Each target-word subcorpus was compiled by different people at different times: *line* by Leacock et al. (1993), *interest* by Bruce and Wiebe (1994) and *hard* and *serve* by Leacock et al. (1998). Each target word subcorpus consists of short context samples from several bigger corpora:

- The 1987-89 Wall Street Journal corpus (WSJ) - *interest, line, serve*
- The corpus from American Printing House for the Blind (APHB) - *line, serve*
- The 1991 San José Mercury News corpus (SJM) - *hard*

See Figure 5.1.2 for the number of instances of each target word and their sense distribution.

The 1998-2000 portion of the NYT was chosen because it would present a journalistic style comparable to that used in the HILS dataset and also because the years covered by HILS and the NYT are distant enough to ensure the same news stories will not feature in both datasets. The NYT dataset consists of 1.92×10^8 tokens distributed in 205990 articles.

The untagged NYT dataset is used in several different ways. We use the **pseudoword** technique (Yarowsky, 1993) whereby occurrences of two unrelated words (for example *banana* and *moon*) are replaced by their concatenation (*banana_moon*), creating a resolution task to revert each pseudoword correctly to the word replaced. Applied carefully, this generates a larger number of training and test instances than in a sense-tagged corpus. We use the pseudowords introduced by de Marneffe and Dupont (2004). They are listed here with their total number of occurrences and their constituent distribution in the NYT:

- *animal_river* – total: 11808, *animal*: 33%, *river*: 67%
- *banana_moon* – total: 3953, *banana*: 24%, *moon*: 76%
- *data_school* – total: 49498, *data*: 21%, *school*: 79%
- *railway_admission* – total: 3733, *railway*: 14%, *admission*: 86%
- *rely_illustration* – total: 4970, *rely*: 78%, *illustration*: 22%

In work connected to the sense disambiguation and discrimination of the ambiguous words in the HILS dataset, the NYT dataset is also used as a source of the word vectors used in the construction of indirect context vectors.

5.2 Geometric experiments

A simple geometric property that can be considered is how similar the direct sense vector of a particular sense of a word is to the indirect sense vector of the same sense of the same word (i.e. how similar ${\mathfrak{s}^1}_i$ and ${\mathfrak{s}^2}_i$ are?) In this work, this property is called **parallelism**. It is measured by

TARGET	*hard*	TARGET	*interest*	TARGET	*line*	TARGET	*serve*
INSTANCES	4333	INSTANCES	2368	INSTANCES	4146	INSTANCES	4378

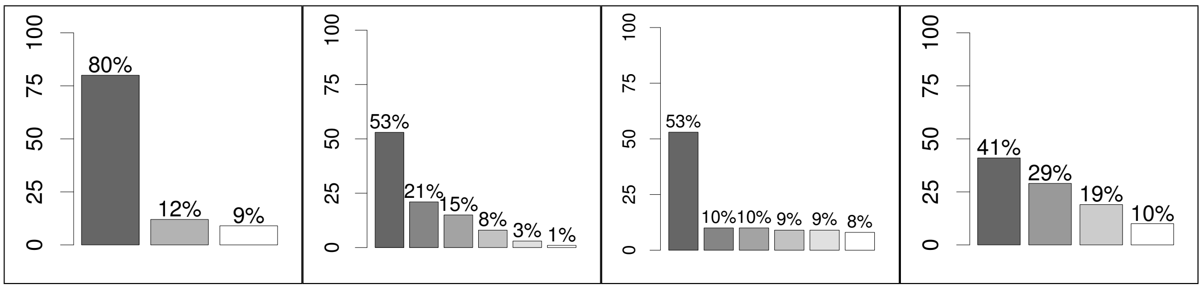


Figure 5.1.2: HILS dataset sense distributions

computing the cosine (Eq. 3.2.9 on p.72) between $\mathfrak{s}^1{}_i$ and $\mathfrak{s}^2{}_i$. The left-hand side of Figure 5.2.1 shows two examples of how parallel an $\mathfrak{s}^1$ vector is to an $\mathfrak{s}^2$ when both represent the same sense 'difficult' of the adjective *hard*. The more parallel (more similar) they are, the more they are approximations of each other.

Another geometric property considered is the angular **spread**, as measured by cosine similarity, amongst the sense vectors for different senses of a given word. The right-hand side of Figure 5.2.1 shows examples of low and high spread of two same-order sense vectors for different senses of the word *hard*. Intuitively, a high angular spread will benefit a sense disambiguation or discrimination algorithm exploiting means-based sense vectors, whereas a low spread will make distinguishing amongst senses more difficult. Spread is measured by taking the average of pairwise cosine measures between sense vectors of the same order that represent different senses of an ambiguous word.

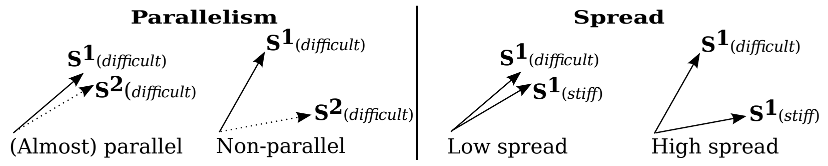


Figure 5.2.1: A depiction of parallelism and spread in sense vectors representing two senses for *hard* – On the left-hand side, we measure how parallel different-order sense representations of the same sense are between each other. On the right-hand side, we measure how same-order sense vectors representing different senses are spread.

Geometric parallelism and spread experiments were conducted on the sense-tagged HILS dataset and on the pseudoword-tagged NYT dataset.

For simplicity, this work ensures that the direct and indirect context vectors (and thereby their derived direct and indirect **sense** vectors) have the same dimensional unigram features, from a vocabulary $\mathcal{V}_m$, and build a square $m \times m$ word matrix (i.e. a word matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ where $n = m$). We can then make m-dimensional direct context vectors, which convert to further m-dimensional indirect context vectors (see Definitions 3.5.4 on p. 101 and 4.1.1

on p. 109). For the HILS data, the word matrix used in the derivation of indirect context vectors was computed once in an **internal** fashion and once in an **external** fashion[3]. In the internal variant, for each word-specific (i.e. *hard*, *interest*, *line* or *serve*) sub-corpus $\mathcal{T}$ of the HILS dataset, the word vectors making up the word matrix are computed from occurrences in $\mathcal{T}$, using all its non-stop (content) words, $NS(\mathcal{T})$, as the features $\mathcal{V}_m$. In the external variant, word vectors are instead computed from the NYT corpus, using $NS(\mathcal{NYT}) \cap NS(\mathcal{T})$ for features of the word vectors, where $NS(\mathcal{NYT})$ are the 20,000 most frequent non-stop words in the NYT corpus. The idea is to have the word vectors determined from a much larger data set, containing word-occurrences that are not constrained to be in the vicinity of one of the words in the HILS dataset. For the pseudoword data, the word vectors were always computed in the internal fashion.

Table 5.2.1: Summary of geometric experiment results

WORD	PARALLELISM				SPREAD				
	INTERNAL		EXTERNAL			INTERNAL		EXTERNAL	
	$\mathfrak{s}^1 \parallel \mathfrak{s}^2$	$\mathfrak{s}^1 \parallel \mathfrak{s}^2_{\widehat{\mathbb{W}}}$	$\mathfrak{s}^1 \parallel \mathfrak{s}^2$	$\mathfrak{s}^1 \parallel \mathfrak{s}^2_{\widehat{\mathbb{W}}}$	$\mathfrak{s}^1$	$\mathfrak{s}^2$	$\mathfrak{s}^2_{\widehat{\mathbb{W}}}$	$\mathfrak{s}^2$	$\mathfrak{s}^2_{\widehat{\mathbb{W}}}$
hard	.14	.18	.48	.49	.23	.99	.99	.98	.98
interest	.27	.33	.36	.38	.13	.85	.85	.91	.95
line	.35	.33	.40	.42	.14	.93	.95	.95	.96
serve	.50	.45	.50	.54	.14	.74	.85	.89	.87
animal_river	.64	.68			.42	.99	.99		
banana_moon	.45	.49			.23	.99	.98		
data_school	.62	.64			.26	.96	.96		
railway_admission	.57	.59			.28	.98	.98		
rely_illustration	.61	.61			.14	.79	.95		

Parallelism results Under PARALLELISM, INTERNAL in Table 5.2.1, column $\mathfrak{s}^1 \parallel \mathfrak{s}^2$ reports the average of cosine measures between direct representations and indirect representations, $\frac{1}{N}\sum_{i=1}^{N} \cos\,[\mathfrak{s}^1{}_i, \mathfrak{s}^2{}_i]$, for all senses σ_i of each target word in the internal variant. Column $\mathfrak{s}^1 \parallel \mathfrak{s}^2_{\widehat{\mathbb{W}}}$ shows outcomes when the rows of the word matrix are L2-normalised (euclidean length), a common practice in word-sense disambiguation and discrimination experiments. EXTERNAL gives the external variant.

For neither the internal nor the external variants would one say that these cosine values indicate that the derived direct and indirect sense vectors are approximately parallel. For most of the HILS items, the average cosine scores increased in moving from the internal to the external calculation of the word vectors, but still did not result in approximately parallel vectors. For the pseudowords, although the parallelism is higher, they still cannot be fruitfully thought of as approximations of each other.

If both vector types were approximations of each other, we could predict that they would

[3]In Maldonado-Guerra and Emms (2012) the internal and external approaches described here are referred to as local and global, respectively.

perform similarly in any given WSX task. But since they are not, it is not possible to predict the performance of one based on the performance of the other.

Spread results In Table 5.2.1, with the same settings used as for the consideration of parallelism under **SPREAD**, **INTERNAL** and **EXTERNAL**, the cosine averages are given for each ambiguous item for the sense vector types $\mathfrak{s}^1$, $\mathfrak{s}^2$ and $\mathfrak{s}^2{}_{\widehat{W}}$. For both the HILS and the pseudoword data, direct sense representations are far more spread (lowest average cosine scores) than their indirect counterparts, with the external variants being still even less spread than the internal.

The interpretation of this result is that means-based sense disambiguation/discrimination algorithms, such as Rocchio classification or K-Means clustering, will be able to discriminate senses better on spaces based on direct context vectors than on spaces based on indirect context vectors. So, direct context vectors could perform better than indirect context vectors in those experiments. However, recall that direct context vectors are far more sparse than indirect vectors. So their actual performance will depend on a trade-off between angular spread and their ability to capture the semantic properties of the contexts they represent.

5.3 Supervised disambiguation

In order to make the supervised experiments as comparable as possible with the unsupervised experiments, a classifier that presented as many of the same characteristics as the K-Means clustering method used in the unsupervised experiments was selected for the supervised experiments. A Rocchio classifier (i.e. a nearest centroid classifier) was thus implemented. In preparation to the classifier's training, the data is separated into a training and a test portion. The classifier is trained by creating sense vectors on the training portion. That is, for each sense in the training portion, a sense vector is computed (direct or indirect). Classification takes place by computing context vectors (direct or indirect) of ambiguous tokens from the test portion and assigning each of these context vectors to the nearest candidate sense vector, as measured by the cosine score. Evaluation is performed via a precision score representing the percentage of test context vectors assigned to the correct senses (according to test set sense tagging).

As previously mentioned, experiments were done both with HILS data and the pseudoword data from the NYT. The corpus for an ambiguous item was randomly split into 60% training and 40% test. To ensure robustness, four independent splits were done and results are reported as averages over these splits. Performance is evaluated via a precision score representing the percentage of test context vectors assigned to their correct senses. In computing indirect context vectors, the internal approach was taken, so with word vectors computed from the sub-corpus of occurrences of the ambiguous item. The outcomes are shown in Table 5.3.1 under the **SUPERVISED** header.

For the HILS and pseudoword data, direct context vectors outperformed indirect (both $\mathbf{c}^2$

and $\mathbf{c}^2_{\widehat{\mathrm{W}}}$) vectors. And in three out of four HILS cases, and all pseudoword cases, $\mathbf{c}^2_{\widehat{\mathrm{W}}}$ vectors outperformed $\mathbf{c}^2$ vectors. These results mirror to a large extent the behaviours observed in the geometric experiments, specifically in the spread experiments: since indirect sense vectors are not very spread out, it is difficult for the Rocchio classifier to correctly assign a sense to an instance.

For the pseudoword data the gap between $\mathbf{c}^1$ and $\mathbf{c}^2_{\widehat{\mathrm{W}}}$ was smaller, and overall the pseudoword results are across the board significantly higher than the HILS target word results. This is consistent with previous research that has found that word-sense disambiguation experiments done on pseudowords tend to report higher results than the same experiments done on real ambiguous words (Gaustad, 2001).

A majority sense baseline can also be seen in Table 5.3.1 under column **M**. $\mathbf{c}^1$ vectors outperform this baseline more often than the other two context vectors, but $\mathbf{c}^2_{\widehat{\mathrm{W}}}$ come at a close second place.

Table 5.3.1: Performance results for Supervised Rocchio word-sense disambiguation experiments and Unsupervised K-Means word-sense discrimination experiments

WORD	M	SUPERVISED			UNSUPERVISED (L2)						UNSUPERVISED (cos)					
		trn & tst: HILS			trn & tst: HILS			trn: NYT, tst:HILS			trn & tst: HILS			trn: NYT, tst:HILS		
		$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$	$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$	$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$	$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$	$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$
hard	80	79	76	69	79	60	52	77	62	45	57	66	71	66	62	51
interest	53	78	60	71	46	32	42	50	47	59	47	55	49	48	49	59
line	53	77	48	69	50	31	36	53	39	51	43	38	44	44	47	52
serve	41	81	63	70	39	38	50	44	50	62	54	54	53	59	47	61
AVERAGE	*57*	*79*	*62*	*70*	*54*	*40*	*45*	*56*	*49*	*55*	*50*	*53*	*54*	*54*	*51*	*56*
		trn & tst: NYT			trn & tst: NYT						trn & tst: NYT					
PSEUDOWORD	M	$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$	$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$				$\mathbf{c}^1$	$\mathbf{c}^2$	$\mathbf{c}^2_{\widehat{\mathrm{W}}}$			
animal_river	67	85	65	79	67	67	63				67	66	57			
banana_moon	76	88	66	82	75	74	75				69	73	65			
data_school	79	83	82	87	73	77	68				58	77	64			
railway_admission	86	90	68	83	83	71	56				70	77	58			
rely_illustration	78	90	85	87	78	79	81				82	83	79			
AVERAGE	*77*	*87*	*73*	*84*	*75*	*74*	*69*				*69*	*75*	*65*			

5.4 Unsupervised discrimination

For the unsupervised experiments a training set of context vectors of an ambiguous item are clustered via the K-Means algorithm[4]. Assuming a 1-to-1 sense/cluster relationship, *k* is set to the number of senses of the ambiguous item. In the standard K-Means formulation, the metric that decides the assignment of a data point to a cluster is the L2 (euclidean length) distance. However, to ensure symmetry with the supervised experiments we also carried out

[4]We used a modified version of Wei Dong's implementation: http://www.cs.princeton.edu/~wdong/kmeans/

experiments using cosine as the assignment metric. Clustering is evaluated according to the Munkres (1957) algorithm: items from a test set are assigned to their nearest cluster centres and for each possible sense-to-cluster mapping, a precision score on the test set is determined, with the maximum of these reported as the final score. Two types of experiments are done using the HILS data. An internal type uses the same training-test splits as in the supervised experiments; results are reported in Table 5.3.1 under (**trn & tst: HILS**). An external type induces clusters on the NYT and then uses the full HILS target word sub-corpora as test set; results are reported under (**trn: NYT, tst: HILS**). Experiments are also done with the NYT pseudoword data. For increased robustness, we run each K-Means experiment 10 times with different sets of randomly chosen initial cluster centres. The evaluation scores of these 10 runs are averaged. For the internal experiments, the 10-run averages for each splitting are in turn averaged again to provide an overall precision score for each target word.

It is clear from the table that supervised results are far superior to the unsupervised results, with the $\mathbf{c}^1$ roughly following the distribution of the predominant sense (column **M**). In general, it is difficult for context-based WSDisc systems to outperform this baseline (McCarthy et al., 2004). It seems that global training tends to benefit all three types of context vectors when clustering by L2 K-Means. But the benefit of global training for cosine K-Means is modest at best. Within the L2 K-Means case, the local $\mathbf{c}^1$ vectors perform better than the other two types of indirect context vectors, largely reflecting the geometry predictions. However, the global $\mathbf{c}^2_{\widehat{\mathrm{W}}}$ turns out somewhat comparable to $\mathbf{c}^1$. Across the cosine K-Means experiments (local and global), results for the two types of second-order context vectors have mixed performance in relation to the baseline but remain comparable to $\mathbf{c}^1$. It can be seen that $\mathbf{c}^2_{\widehat{\mathrm{W}}}$ vectors outperform the baseline 7 times, followed by $\mathbf{c}^1$ at 4 times and then the $\mathbf{c}^2$ at 3 times. The target word where the baseline is outperformed more often is *serve*, which is the word that has a more balanced sense distribution (see figure 5.1.2).

In the pseudowords (bottom part of table 5.3.1), $\mathbf{c}^2$ vectors follow the baseline closely in L2 and cosine K-Means. L2 $\mathbf{c}^1$ scores follow the baseline closer than cosine $\mathbf{c}^1$ scores, while $\mathbf{c}^2_{\widehat{\mathrm{W}}}$ scores tend to stray below the baseline for both L2 and cosine K-Means, with the notable exception of *rely_illustration*, a pseudoword that tends to have good results across the board possibly because it is made up of words with mixed parts of speech (a verb and a noun), making discrimination easier.

Notice that while the pseudoword "sense" distributions tend to be very skewed (see section 5.1 but also pseudoword baselines), the HILS sense distributions tend to be less so, with the clear exception of *hard*. K-Means experiments will in general depend to a great extent on the skewness of the underlying sense distribution. The performance of $\mathbf{c}^1$ vectors will be consistent and predictable as they will tend to follow this skewness. $\mathbf{c}^2$ and $\mathbf{c}^2_{\widehat{\mathrm{W}}}$ vectors might achieve better or similar results to $\mathbf{c}^1$ vectors (under certain circumstances) but it is not guaranteed that this will be consistent.

5.5 Comparisons and conclusions

The evaluation methods for WSD and WSDisc presented here were chosen due to their usage in previous work, notably Schütze (1998), Purandare and Pedersen (2004) and de Marneffe and Dupont (2004). Also, the classification and clustering methods used here are admittedly basic and simple. The reason why they are used is because their properties are well understood and they enable solid comparison between different versions of token representations. For consistency, the same methods will be used in Chapters 6 and 7.

Purandare and Pedersen (2004) report a number of experiments on word-sense discrimination, and contrast direct and indirect outcomes. They evaluate on two separate sense-tagged corpora: a version of the HILS dataset and a smaller dataset derived from the SENSEVAL-2 dataset which has 10 times fewer examples per sense than the HILS dataset. They found that indirect outcomes exceeded direct on the smaller data set, but that direct outcomes exceeded indirect on the larger data set. As we argue below, however, inspection of their definitions reveals that they are not really comparing minimal pairs in the first and second order versions.

In all our experiments the features of vectors have been identifiable with unigrams in some vocabulary $\mathcal{V}$. Purandare and Pedersen (2004) work also with features they term co-occurrences $\mathcal{F}_{co}$ and bigrams $\mathcal{F}_{bi}$. $\mathcal{F}_{co}$ is a chosen set of **co-occurrence features**, each of which is a pair $\{\tau, \upsilon\}$ defined to occur in a window if $\tau\beta\upsilon$ or $\upsilon\beta\tau$ is a subsequence of the context, with β a to-be-specified tolerance of intervening words. The selection of the actual set of features in $\mathcal{F}_{co}$ from all possible such pairs is through a statistical word association test, such as the log-likelihood criterion (Dunning, 1993). $\mathcal{F}_{bi}$ is a chosen set of **bigram features**, each of which is a pair $\{\tau, \upsilon\}$ in which τ and υ must occur in that order, again with a tolerated interval of other words between them.

For their first-order experiments, they construct direct context vectors using $\mathcal{F}_{co}$ and $\mathcal{F}_{bi}$ for features, and these are contrasted with indirect outcomes as follows. For $\mathcal{F}_{co}$, with a tolerance of β, let $\mathcal{V}_{co}$ be all unigrams that are members of the selected co-occurrences in $\mathcal{F}_{co}$, construct direct context vectors using the unigrams $\mathcal{V}_{co}$ for features, with window width of β, and form from these a word matrix $\mathbf{W_{co}}$. Indirect context vectors are then made by transforming unreduced direct context vectors with unigram features from $\mathcal{V}_{co}$ by $\widehat{\mathbf{W}}_{\mathbf{co}}$, an SVD reduction of $\mathbf{W_{co}}$. For $\mathcal{F}_{bi}$, with a tolerance of β, let $\mathcal{V}_{fst}$ (resp. $\mathcal{V}_{snd}$) be all unigrams occurring first (resp. second) in a bigram feature, construct direct context vectors using the unigrams $\mathcal{V}_{snd}$ for features, with a window width of β, going only *right* of a target word, and form from these a word matrix $\mathbf{W_{bi}}$. Indirect context vectors are then made by transforming unreduced direct context vectors with unigram features from $\mathcal{V}_{fst}$ by $\widehat{\mathbf{W}}_{\mathbf{bi}}$, an SVD reduction of $\mathbf{W_{bi}}$.

They ultimately thus present results contrasting

$\langle\ \mathbf{c}^1\colon \mathcal{F}_{co}$ vs. $\mathbf{c}^2\colon \mathbf{c}^1(\mathcal{V}_{co}) \times \widehat{\mathbf{W}}_{\mathbf{co}}\ \rangle$ $\langle \mathbf{c}^1\colon \mathcal{F}_{bi}$ vs. $\mathbf{c}^2\colon \mathbf{c}^1(\mathcal{V}_{fst}) \times \widehat{\mathbf{W}}_{bi}\ \rangle$

These experiments do not therefore contrast outcomes with a particular kind of direct context vector with those that would be obtained simply by transforming *exactly those direct context vectors* by a some kind of word matrix. The natural comparison to make is between outcomes with these indirect context vectors and outcomes with their unigram-features direct counterparts before transformation by the word matrix.

Thus it seems fair to say that the conclusions they draw concerning dependency on the size of the data set are not based on contrasting minimal pairs. Rather than contrasting outcomes with a particular kind of direct context vector with those that would be obtained simply by transforming *exactly those direct context vectors* by some kind of word matrix, they are contrasting co-occurrence based direct context vectors with the transformation of unigram-based direct context vectors. Their findings concerning dependency on the size of the data-set are thus potentially attributable to factors other than the direct vs. indirect contrast. In our experiments we did not systematically vary the size of the data set and it remains for future work to revisit this size dependency issue with more strictly comparable first- and indirect representations.

In our work, we contrasted direct context vectors versus second order-context vectors, both using unigram features. The geometric experiments as well as the supervised word-sense disambiguation experiments suggest that in this simplest configuration, $\mathbf{c}^1$ vectors are better than $\mathbf{c}^2$ vectors. In the supervised WSD experiments, the $\mathbf{c}^1$ vectors beat both variants of $\mathbf{c}^2$ vectors on all of the HILS words and on 4 of the pseudowords, out of a total of 5. On both datasets $\mathbf{c}^1$ vectors beat the M baseline in 8 out of 9 cases and $\mathbf{c}^2{}_{\widehat{\mathbf{W}}}$ does so in 7 out of 9 cases.

In the unsupervised experiments, on the average the $\mathbf{c}^1$ vectors and $\mathbf{c}^2$ vectors outcomes are much closer together, and for the HILS data, best outcomes are about 25% down from the supervised case, whilst for the pseudoword data, the fall is around 12%; thus the multi-way ambiguity of the HILS data versus the 2-way ambiguity of the pseudoword data seems to be particularly challenging to the unsupervised methods. On the pseudowords, the **M** baseline is seldom beaten, and the advantage of $\mathbf{c}^1$ vectors over $\mathbf{c}^2$ vectors from the supervised case is not replicated. On the HILS data, only for *serve* is the **M** baseline often beaten. Across the representations, the external version with clustering on the large NYT corpus performed better than the internal version, which clusters on a subset of the HILS. And again, the advantage of $\mathbf{c}^1$ vectors over $\mathbf{c}^2$ vectors from the supervised case is not replicated, with varying outcomes across the words, and a close final average.

Thus these experiments have shown an advantage of $\mathbf{c}^1$ vectors over $\mathbf{c}^2$ vectors for the supervised case, and no clear winner for the unsupervised case. It has to be stressed that the setting used for $\mathbf{c}^1$ vectors and $\mathbf{c}^2$ vectors were in many respects, the simplest possible, and a different picture might emerge under different settings.

6 WSX experiments: unreduced vs. SVD-reduced token spaces

As mentioned in Chapter 3, the motivation of using Singular Value Decomposition (SVD) is threefold: 1) to reduce the dimensionality of the vector spaces in order to make computations more tractable, 2) to group together features that are deemed to be similar or related via higher-order co-occurrence, and 3) to produce a vector space that is more sensitive to the polysemy of words, a highly desirable effect for methods dealing primarily with word senses. As a consequence, several adaptations of SVD have been involved in word-sense disambiguation, induction and discrimination (WSX) methods based on the vector space model (VSM). As it will be seen in Section 6.1, these adaptations are based on Word Space, LSA or some mix of the two. However, works reporting WSX experiments based on these adaptations do not usually consider the relationships between Word Space and LSA discussed in Chapter 4, so even works that present benchmarks comparing the two approaches might not be completely systematic. This chapter aims to conduct word-sense disambiguation and discrimination experiments more systematically, according to the settings and combinations summarised in Table 4.2.1 (p. 111).

6.1 Background

Perhaps the first work to incorporate SVD in a word-sense discrimination task was Schütze (1998) who found indirect context vectors computed from SVD-reduced word matrices (I-W-R1/2) to perform better than unreduced indirect context vectors (I-W-UR) on word-sense discrimination experiments based on pseudowords and some real polysemous words. Purandare and Pedersen (2004) also conducted word-sense discrimination experiments but they contrasted different variants of direct context vectors (Pedersen and Bruce, 1997) and indirect context vectors similar to Schütze's. All of their indirect context vectors were constructed from SVD-reduced word matrices (I-W-R1/2), whilst their direct context vectors were all used in unreduced form (D-C-UR). So no comparison of the effects of using SVD on same-order context vectors was performed (i.e.: I-W-UR vs. I-W-R1/2 or D-C-UR vs. D-C-R1/2). Emms and Maldonado-Guerra (2013) did work with SVD-reduced direct context vectors (D-C-R1/2). While we did not perform experiments with unreduced direct context vectors in this work, by consulting results in Maldonado-Guerra and Emms (2012) (see also Chapter 5), it can be found that unreduced direct context vectors actually perform very similarly to

SVD-reduced vectors on unsupervised word-sense discrimination experiments. The common thread uniting these works is that they all cluster direct (first-order) or indirect (second-order) context vectors based on Word Space.

An alternative line of work using SVD on word-sense discrimination is Levin et al. (2006). This work disambiguates SVD-projected segment vectors from Salton's original vector space model, essentially a direct token representation based on LSA which this thesis labels as D-A-R1/2. It is reasonable to expect D-A-* and D-C-* to perform similarly, given that $\mathbf{A} \approx \mathbf{C}$ as per Definition 4.3.1 (p. 124). Something similar could be said for I-A-* and I-C-*. Since the SVD projections for word vectors from $\mathbf{A}$ and $\mathbf{W}$ are also approximations of each other if $\mathbf{W}$ is symmetric as shown in the discussions in Section 4.3 (p. 122), we can expect I-W-R1/2 and I-A-R1/2 to perform similarly, even if I-W-UR and I-A-UR turn out to perform differently as the word vectors and their features in each configuration is different and incompatible.

It would be interesting to compare empirically the performance of these token vector configurations in actual WSX experiments. A work that goes some way towards this direction is Pedersen (2010). This work carried out word-sense discrimination experiments based on pseudowords using three main types of context vectors:

- Unreduced direct context vectors (called `o1` in Pedersen's paper). There were two subtypes of direct context vectors features: unigram features (`o1-uni`), which are identical to the vectors defined by Definition 3.5.2 (p. 100), and bigram features (`o1-big`), which use the bigram features used by Purandare and Pedersen (2004) and discussed in Section 5.5 (p. 143).

- Indirect context vectors computed from a word matrix of so-called bigram features. This is the matrix $\mathbf{W_{bi}}$ described in Section 5.5 (`o2SC`). Another subtype of indirect context vector is also produced using $\widehat{\mathbf{W}}_{\mathbf{bi}}$, an SVD-reduced version of $\mathbf{W_{bi}}$ (`o2SC-SVD`). The paper does not make explicit if R_1 or R_2 projections are used in the computation of $\widehat{\mathbf{W}}_{\mathbf{bi}}$.

- LSA-inspired variants of context vectors. The paper mentions that a word-context matrix, which I shall call $\mathbf{X}$, is constructed with direct context vectors of unigram features as its columns. Then, indirect context vectors are formed by (averaged) matrix multiplication of direct context vectors by this matrix, i.e.: $\mathbf{c}^2(\kappa) = \mathbf{X}^T\mathbf{c}^1(\kappa) / \sum_{i=1}^{m}[\mathbf{c}^1(\kappa)]_i$, where $[\mathbf{c}^1(\kappa)]_i$ is one of the m features in $\mathbf{c}^1(\kappa)$. Pedersen refers to this setting as `o2LSA`, even if it does not use SVD. In the paper descriptions, it is somewhat ambiguous whether $\mathbf{X} = \mathbf{C}$, where $\mathbf{C}$ is the matrix of direct context vectors, or $\mathbf{X} = \mathbf{A}$, Salton's original VSM matrix. Recall that the main difference between $\mathbf{A}$ and $\mathbf{C}$ is the fact that $\mathbf{A}$ gives an additional count to the token that would be represented by each type vector in $\mathbf{C}$, since it counts full segments (Def. 4.3.1, p. 124). Given Pedersen team's experience with Word Space methods, it is highly likely that $\mathbf{X} = \mathbf{C}$. But that is only a guess. Also, recall from Section 4.2.1 (p. 111) that there are at least

Table 6.1.1: An attempt to align vector configurations in Pedersen (2010) with vector configurations in this thesis.

Pedersen's configuration	**This thesis configuration**	
	Definition	**Conf. label**
`o1-uni`	3.5.2	D-C-UR
`o1-big`	3.5.2 (bigram)	D-C-UR (bigram)
`o2SC`	4.1.1(bigram)	I-W-UR (bigram)
`o2SC-SVD`	4.2.5(bigram)	I-W-R1? or -R2? (bigram)
`o2LSA`	4.2.3?	I-C-UR?
	4.2.6?	I-A-UR?
`o2LSA-SVD`	4.2.4?	I-C-R1? or -R2?
	4.2.7?	I-A-R1? or -R2?

three ways in which **C** can be constructed. Option 3 is the most similar to Salton's VSM matrix. But it is not clear from Pedersen's paper which of the three options (or even whether another option not contemplated in this list) was used, if he indeed used $\mathbf{X} = \mathbf{C}$. An SVD-reduced version of **X** was also created, $\widehat{\mathbf{X}} = \mathbf{U_k \Sigma_k V_k}$, and used to create reduced indirect context vectors via the usual matrix transformation of first order context vectors: either $\mathbf{c}^{R_1(\mathbf{X})}(\kappa) = \mathbf{\Sigma_k U_k}^T \mathbf{c}^1(\kappa) / \sum_{i=1}^{m} (\mathbf{c}^1(\kappa))_i$ or $\mathbf{c}^{R_2(\mathbf{X})}(\kappa) = \mathbf{U_k}^T \mathbf{c}^1(\kappa) / \sum_{i=1}^{m} (\mathbf{c}^1(\kappa))_i$. This setting is called `o2LSA-SVD` in Pedersen's paper.

Table 6.1.1 summarises the vector configurations used in Pedersen's paper and attempts to pair them with the summary of vector configurations from Table 4.2.1 (p. 111). The main issue with Pedersen's configurations is the ambiguity in the two `o2LSA` configurations. Another issue is the comparison of different configurations using different features (unigrams vs. bigrams), which adds another layer of complexity to the comparison between configurations. Lastly, a significant omission is the direct reduced and unreduced segment vector based on **A** as in Levin et al. (2006), or at least its approximation via **C**.

This chapter aims to repeat these experiments in a more systematic way as informed by the conclusions in Section 4.4 and largely following the methodology used in Maldonado-Guerra and Emms (2012) upon which Chapter 5 is based and plotting the performance of different degrees of truncation on the SVD as was done in Levin et al. (2006) and Emms and Maldonado-Guerra (2013). It is expected that these experiments will show:

1. Which of the two LSA configurations, direct (D-A-*) or indirect (I-A-*), performs better?

2. Are the differences in LSA and Word Space significant enough to impact on their performance on WSX experiments? In particular: D-A-* vs. D-C-* and I-A-* vs. I-W-*.

6.2 Methodology

Maldonado-Guerra and Emms (2012), work upon Chapter 5 is based, contrasted unreduced direct context vectors with unreduced indirect context vectors by comparing two geometric properties of these representations (parallelism and angular spread) as well as by conducting word-sense disambiguation and discrimination experiments. This work found that both vector representations are very different from each other (low parallelism), that sense vectors based on direct context vectors are more spread than indirect context vectors and found a superior performance of direct context vectors on supervised word-sense disambiguation, while finding relatively poor performance in both context representations on unsupervised word-sense discrimination experiments. This chapter will only be concerned with the word-sense disambiguation and discrimination experiments. In order to avoid running many configuration combinations, only experiments on the internal HILS dataset will be conducted. That is, experiments training externally on the NYT and testing on the HILS dataset are not considered. Table 4.2.1 (p. 111) shows the configurations for the experiments presented in this chapter.

We largely follow the methodology used by Levin et al. (2006) and Maldonado-Guerra and Emms (2012) (Chapter 5) and use the HILS dataset (Sec. 5.1, p. 136). Using repeated (4 times) random sub-sampling validation, each target-word corpus was split into a 60% training portion and a 40% test portion in each repetition. Context vectors using the configuration combinations from Table 4.2.1 were constructed. For those combinations requiring SVD, vectors were created using truncations from 50 to 1000 dimensions at 50 dimension intervals. Independent word-sense disambiguation and discrimination experiments were conducted on every configuration and dimension combination for every split of each target-word corpus and the accuracy results were averaged across the splits for each combination.

Supervised word-sense disambiguation experiments are based on a simple Rocchio classifier: the training vectors are grouped by sense as per the dataset's own sense-tagging, and the centroid (average) of every sense-specific group is taken. Each test vector is then classified to the sense whose sense vector is closest according to the cosine measure. Performance is measured by the accuracy of the classifier on the test set.

Unsupervised word-sense discrimination experiments are based on a cosine-based K-Means clusterer. Being an unsupervised experiment, the sense-tagging of the training set is completely ignored. K clusters are induced from the training set, with K being the number of the target word senses. Test vectors are then assigned to their closest cluster based on cosine. Performance is measured according to the Munkres (1957) algorithm: accuracies are computed for each possible mapping of senses with cluster assignments and the maximum of such accuracies is reported as the experiment's performance. A K-Means experiment is repeated 30 times for robustness and the average accuracy is then reported as its overall accuracy.

Table 6.3.1: Accuracy maxima for supervised (top sub-table) and unsupervised (bottom sub-table) experiments. For R_1 and R_2 projections, the number of SVD dimensions kept is indicated in parentheses.

SUP	hard (MS=80)			interest (MS=53)			line (MS=53)			serve (MS=41)		
	R1	R2	UR	R1	R2	UR	R1	R2	UR	R1	R2	UR
D-A	79.14 (1000)	**80.36 (350)**	79.69	62.18 (900)	**75.97 (450)**	62.23	58.99 (1000)	75.11 (950)	59.52	67.77 (1000)	**81.20 (600)**	67.88
D-C	77.53 (950)	73.34 (500)	78.08	72.94 (950)	73.36 (400)	72.94	75.66 (1000)	75.59 (950)	**76.28**	75.23 (950)	80.17 (600)	75.54
I-W	75.38 (950)	76.96 (1000)	75.36	58.96 (500)	72.12 (900)	58.75	48.63 (750)	75.17 (1000)	48.61	64.19 (100)	74.86 (850)	64.22
I-A	72.61 (900)	77.03 (1000)	72.42	64.08 (650)	72.89 (1000)	64.06	67.14 (850)	75.73 (1000)	67.17	67.20 (950)	75.50 (950)	67.25
I-C	74.28 (950)	77.53 (950)	74.25	64.27 (700)	72.94 (950)	64.19	68.58 (1000)	75.66 (1000)	68.58	67.38 (950)	75.23 (950)	67.37

UNS	hard			interest			line			serve		
	R1	R2	UR	R1	R2	UR	R1	R2	UR	R1	R2	UR
D-A	65.15 (650)	51.34 (50)	64.87	45.42 (650)	35.65 (50)	**45.50**	26.07 (850)	25.63 (100)	26.47	39.16 (950)	36.37 (50)	39.60
D-C	55.05 (50)	39.81 (50)	52.77	42.10 (150)	31.02 (50)	41.86	29.18 (900)	29.10 (100)	31.40	53.73 (850)	39.02 (50)	54.44
I-W	**68.28 (1000)**	52.90 (250)	67.28	44.93 (350)	42.31 (450)	44.39	**37.55 (1000)**	29.56 (800)	37.37	**58.98 (350)**	54.69 (50)	58.89
I-A	60.44 (250)	50.11 (900)	60.00	43.11 (850)	42.02 (800)	42.99	32.47 (400)	29.82 (250)	32.97	56.18 (700)	53.91 (150)	56.86
I-C	59.42 (50)	55.12 (50)	57.54	43.17 (950)	42.11 (200)	42.73	32.75 (650)	29.52 (300)	31.80	56.43 (1000)	53.58 (900)	56.29

6.3 Experimental results

Table 6.3.1 shows the best accuracy scores achieved by the R_1 and R_2 projections (R1 and R2 columns, respectively) for each configuration combination, indicating the number of SVD dimensions kept in parentheses. The UR column ("unreduced") reports the accuracy achieved when SVD is not applied. The accuracies in bold are the maximum accuracy obtained from all configuration combinations for a word in the supervised or unsupervised settings. The MS numbers next to the word headings in the top sub-table indicate the majority sense baseline for each word, i.e. the accuracy score obtained when assigning all test instances to the majority sense. Figure 6.3.1 reports these results in more detail as plots of accuracy against number of SVD dimensions kept.

6.3.1 General observations

First, from the graphs it is evident that in general supervised experiments (solid lines) achieve higher accuracy scores than unsupervised ones (dotted lines), which is not surprising and is consistent with previous results (Pedersen, 2007). The supervised scores generally meet or exceed the majority sense baseline (MS number in Table 6.3.1, see also Figure 5.1.2, p. 138) whilst the unsupervised scores struggle to approximate it.

Second, The R_1 vectors (lines with black points) on the supervised (solid) and unsupervised (dotted) cases perform below their respective unreduced vectors (grey lines) in the lower dimensions but as dimensions grow, they increase their performance and meet the unreduced vectors' performance generally quite quickly, with the exception of *hard*-D-A-R1 and *line*-D-C-R1 which increase relatively slowly. However, the R_1-projected vectors never really outperform the unreduced vectors, in neither supervised/unsupervised case. The R_2-projected vectors (lines with white dots) on the other hand behave very differently to the unreduced vectors (grey lines). For the supervised case (solid lines), R_2 vectors meet or outperform the unreduced vectors, but for the unsupervised case (dotted lines), their performance seems to decrease as dimensions grow in the direct configurations (D-A-R2, D-C-R2), whilst in the indirect configurations (I-W-R2, I-A-R2, I-C-R2) they remain roughly constant and almost always below the unreduced and R_1 vectors.

Third, D-A-R2 (standard LSA via R_2) gives the overall best scores for the supervised case. However, observe that the scores for unreduced D-C-UR (first-order) vectors are quite similar. Given the expense of performing SVD, D-C-UR vectors might perhaps offer the best trade-off between computational cost and accuracy. The best unreduced unsupervised score (bold grey dotted line) is generally achieved with the I-W-UR configuration, i.e. using Schütze's standard second-order context vectors, with the exception of *interest*. For the unsupervised case, SVD does not seem to help much, a result consistent with Pedersen (2010), and given the expense of computing SVD and the overall modest benefits it brings to D-A-R2 over D-C-UR, it is perhaps advisable to stick to unreduced vectors (in this case the D-C-UR version) for the supervised case as well.

Section 3.4.2 (p. 81) argued that R_1 projections should be more appropriate than R_2 projections for the type of comparisons that we wanted to perform in WSX experiments in LSA and Word Space. However, the experimental results presented here suggest that R_2 projections can indeed provide much better results than R_1 projections in some supervised experiments (such as supervised D-A-R2, I-A-R2, I-W-R2 and I-C-R2), but the whole picture almost reverses for the unsupervised case (esp. in D-A-R2 and to a lower extent in I-W-R2), where R_2 performs worse or at least not better than R_1, which is more consistent with the original remarks made in Section 3.4.2. Recall as well that in that section an intuitive interpretation of R_2 projections was given in which they could be regarded as segments in which only one token occurs or types that only occur in one segment. Perhaps this is having a noise reduction effect in the supervised case that aids in classification, but perhaps hinders the generalisations needed to successfully cluster in an unsupervised manner. Further investigation into this is left for future work.

6.3.2 Direct vectors: D-A vs. D-C

The unreduced vectors (grey bold lines) do present some variation for both the supervised (solid line) and the unsupervised case (dotted line), indicating that the small difference in count from (4.3.1) has an impact on performance. However, for supervised R_2 (solid lines with white points), the scores and the shapes of the lines between D-A-R2 and D-C-R2 are very similar for all words, although they are a bit dissimilar for *hard*. Similarly, the shapes between supervised D-A-R1 and D-C-R1 (solid lines with black points) are also similar, even if the actual accuracies are not that similar. The overall effects of SVD seem to be the same in all words, but the difference in count between Word Space and LSA will offset the performance of both unreduced vectors and SVD-reduced vectors.

6.3.3 Indirect vectors: I-W vs. I-A and I-C

For I-A-* vs. I-C-*, with the exception of unsupervised *hard*, the supervised and unsupervised lines for all words are almost identical. It seems therefore that the difference in count between Word Space and LSA do not impact the overall performance of unreduced and SVD-reduced vectors when these are computed using the indirect (a.k.a. "second-order") method. Both I-A-* and I-C-* are slightly more different to I-W-*, which comes as a slight surprise, given that from the analysis in Chapter 4, we would have expected I-W-* to be more similar to both I-A-* and I-C-* than they actually are. The difference in the diagonals between $\mathbf{AA}^T$ and $\mathbf{W}$ described in (4.3.2) seems to have a slightly bigger impact on performance.

6.4 Conclusions

Chapter 4 showed that the relationship between Word Space and LSA is more complicated than generally believed and whilst it can be shown that the vectors used to represent word types and tokens in each model are approximations of each other, they are not interchangeable in all circumstances. Despite being approximations, it is possible to state relations of equality between the objects in each model, allowing the adaptation of software tools designed for one model to be used for the other model with some degree of ease. That chapter also demonstrated that Word Space and LSA allow two basic methods of representing tokens via context vectors: a direct (or "first-order") method and an indirect (or "second-order") method, the latter of which can be regarded as the result of a linear transformation of the former via some matrix acting as a linear map. Customarily, this linear map is the Word Space word matrix $\mathbf{W}$, but can be any other matrix, including a matrix of Word Space direct context vectors $\mathbf{C}$ or the LSA word-document matrix $\mathbf{A}$.

The experiments in this chapter showed that whilst the difference between both models boils down to a very small difference in counting between the two models, it can cause the models to either keep their difference in performance negligible (I-A-* vs. I-C-*) or diverge to a point in which the difference offsets performance considerably (all other combinations), depending on how the LSA or Word Space objects are actually employed.

Given the common ways in which LSA and Word Space are actually employed (namely, D-A-*, D-C-* and I-W-*), this offset in performance of means-based word-sense disambiguation and discrimination experiments will depend on the actual token representation configuration chosen. Finally, it was found that whilst SVD can help when LSA direct context vectors are used, the overall best results were obtained with the standard application of Word Space (D-C-UR "first-order" and I-W-UR "second order" vectors) without using SVD.

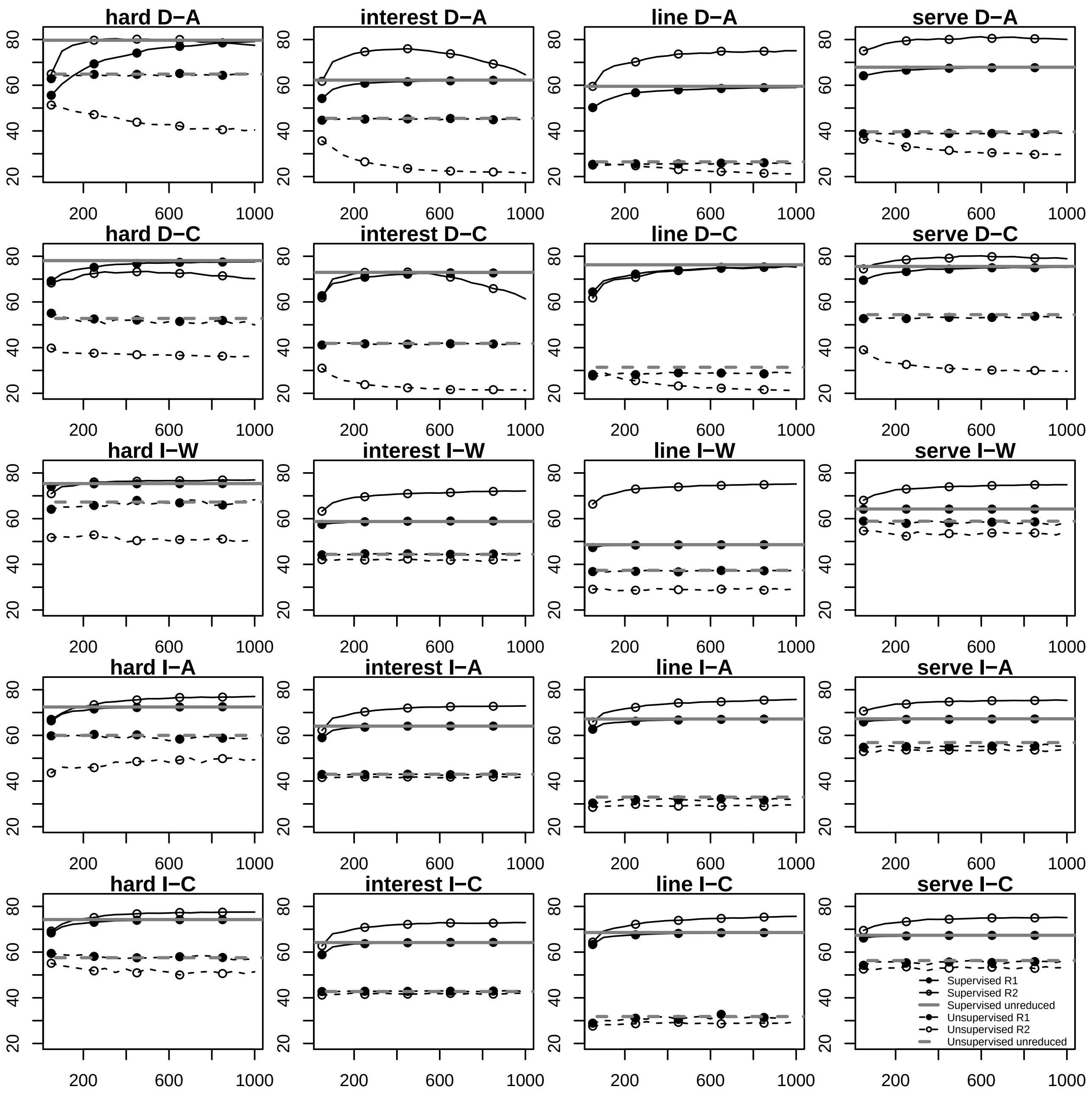


Figure 6.3.1: Accuracy plots matrix for every configuration (rows) and for each ambiguous word (columns). Each plot reports accuracy % (*y*-axis) against SVD dimensions kept (*x*-axis). Solid lines and dotted lines represent supervised and unsupervised experiments, respectively. Lines with black points represent R_1 projections and those with white points, R_2 projections. The bold, grey constant lines represent unreduced vector results for both supervised (solid) and unsupervised (dotted).

7 Word matrix consolidation as a dimensionality reduction method

Section 3.4 (p. 78) explored Singular Value Decomposition (SVD) and its use as a technique for dimensionality reduction within Latent Semantic Analysis. That section also studied how SVD is able to discover latent (hidden) meaning in the Vector Space Model while at at the same time reducing noise. Word-sense disambiguation (WSD) as well as word-sense discrimination (WSDisc) methods employ **context vectors**, which are high-dimensional objects that represent the instance of an ambiguous word. Two types of context vectors are commonly used: Word Space **direct (first-order) context vectors** ($\mathbf{c}^1$) and **indirect (second-order) context vectors** ($\mathbf{c}^2$). $\mathbf{c}^1$ vectors count direct co-occurrences of the target word with other words in the vocabulary whilst $\mathbf{c}^2$ vectors represent instances of the target word by counting the words that the words occurring in the target word's context tend to co-occur with elsewhere in the corpus (Schütze, 1998; Purandare and Pedersen, 2004).

SVD can be applied to the word matrix or to a matrix consisting of context vectors of either order. However, the usage of SVD presents the user with some empirical decisions that need to be made such as the optimal number of dimensions to keep during the dimensionality reduction process, which usually varies with the target word being discriminated. Also, the way it works is not transparent or intuitive, leaving researchers to experiment with it largely by trial and error.

Section 4.1 (p. 108) makes the case that the word matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ used in the computation of indirect context vectors could be used as an alternative method of dimensionality reduction if $n < m$. This chapter presents a dimensionality reduction method based on this insight which aims to be more intuitive than SVD. In other words, we present a method to convert a $\mathbf{c}^1(\kappa) \in \mathbb{R}^m$ vector via a $\mathbf{W} \in \mathbb{R}^{m \times n}$ matrix of lower dimensionality ($n < m$), into a $\mathbf{c}^2(\kappa) \in \mathbb{R}^n$ vector of lower dimensionality based on a more intuitive criterion for reduction.

In this chapter, a $\mathbf{W}$ matrix of lower dimensionality is obtained by consolidating together dimensions of words that are statistically associated. Roughly speaking, this method first constructs a $\mathbf{W} \in \mathbb{R}^{m \times m}$ matrix. Then it carries out hypothesis tests of word association based on the log-likelihood ratio score between each pair of words represented in the $\mathbf{W}$ matrix (Dunning, 1993). The dimensions of those words which the hypothesis test determines to be strongly associated are merged into one dimension by summing those dimensions' column vectors together. Any given dimension can be associated and therefore merged with zero, one or more dimensions. This merging procedure reduces the number of dimensions in the final

Table 7.1.1: Sample contingency table – n_{11}counts the number of times word W1 and word W2 co-occur, n_{12} the number of times W1 occurs without co-occurring with W2, etc.

	$W2$	$\neg W2$
$W1$	n_{11}	n_{12}
$\neg W1$	n_{21}	n_{22}

consolidated word matrix. The intuition behind this method is that words that are statistically associated tend to form collocations, i.e. pair of words that co-occur more often than what pure chance would allow.

7.1 Statistical word matrix

Definition 3.5.1 (p. 96) describes a word matrix **W** co-occurrence frequency between words pairs. As mentioned in Section 3.2 (p. 67) vectors (and matrices) used in vector space models often employ alternative feature functions based on different weighting schemes and to highlight the relevance of some features over the others. One such common alternative feature function is the log-likelihood ratio (Dunning, 1993), which measures how statistically significant the co-occurrence association of a word pair is. The motivation behind using log-likelihood scores comes from an interpretation of the values of each cell as the degree of association between the two words represented by the cell. Since this degree of association is based on the co-occurrence of both words, it can be taken as a measure of the strength of their syntagmatic association. Log-likelihood ratios capture better this degree of association than frequency alone. In addition, these type of features have been used in word matrices before. For example, Purandare and Pedersen (2004) constructed such word matrices with log-likelihood scores as features in their word-sense discrimination experiments.

If a co-occurrence is not statistically significant (i.e. the two words co-occur by chance), then the log-likelihood score should follow the χ^2 distribution with one degree of freedom ($\nu = 1$); if, on the contrary, the co-occurrence is significant (i.e. the two words co-occur more often than what would be expected by chance), then the log-likelihood score does not follow such a $\chi^2_{\nu=1}$ distribution. Equation (7.1.1) shows the formula for computing the log-likelihood score:

$$G^2 = 2\sum_{ij} n_{ij} \log \frac{n_{ij}}{m_{ij}} \tag{7.1.1}$$

Where n_{ij} are the observed values from the co-occurrence contingency table (see e.g. Table 7.1.1) and m_{ij} the expected values calculated from the marginals of such a contingency table (Purandare, 2004, pp. 35–48).

The raw frequency word matrix can be converted to a log-likelihood word matrix by applying the following formulas to compute the observed values for each entry w_{ij} in the matrix and then applying Equation (7.1.1):

$$
\begin{aligned}
n_{11}^{(w_{ij})} &= w_{ij} && (7.1.2)\\
n_{12}^{(w_{ij})} &= (\sum_j w_{ij}) - n_{11}^{(w_{ij})} && (7.1.3)\\
n_{21}^{(w_{ij})} &= (\sum_i w_{ij}) - n_{11}^{(w_{ij})} && (7.1.4)\\
n_{22}^{(w_{ij})} &= (\sum_{ij} w_{ij}) - n_{11}^{(w_{ij})} - n_{12}^{(w_{ij})} - n_{21}^{(w_{ij})} && (7.1.5)
\end{aligned}
$$

We call such a log-likelihood matrix **L** and each entry l_{ij}. If the original $\mathbf{W} \in \mathbb{R}^{m\times m}$, then **L** continues to be a symmetric matrix, $\mathbf{L} \in \mathbb{R}^{m\times m}$. We can use **L** exactly in the same way we use **W**, which is what Purandare and Pedersen (2004) do, but we can also go a bit further with the statistical significance tests.

The numbers in **L** tells us how strongly associated two words are, regardless of their frequency. So, they are less biased numbers than the raw frequency counts in **W**. But in some circumstances we might just be interested in knowing whether two words are statistically associated or not. The task of determining whether the co-occurrence is statistically associated takes the shape of a χ^2 hypothesis test of independence (Manning and Schütze, 1999, Sec. 5.3.4), in which the null hypothesis is that the two words are independent (occurring by chance). The null hypothesis is accepted if the computed log-likelihood score falls on or below the $\chi^2_{\nu=1}$ critical value at a determined p-value. For example, very frequently a p-value of 0.05 is used. This p-value of 0.05 can be interpreted as guaranteeing with 95% confidence that the hypothesis is being correctly accepted or rejected. In other words, we are 95% certain that the hypothesis test gives us the correct answer. At a p-value of 0.05 the $\chi^2_{\nu=1}$ critical value is 3.841. So, if we obtain a log-likelihood score less or equal than this critical value, we accept the null hypothesis that the two words are occurring by chance, and if it is more than this value, we reject the null hypothesis that the two words are occurring by chance. We can use the outcome of this hypothesis test as binary features h_{ij} of a matrix **H**. If we accept the null hypothesis (chance occurrence) for word w_{ij} then we set the h_{ij} entry in **H** to 0. If the null hypothesis is rejected (no chance occurrence) for word w_{ij} then we set a the corresponding entry in **H** to 1. **H** is also a symmetric matrix ($\mathbf{H} \in \mathbb{R}^{m\times m}$) and again, we could easily employ **H** instead of **L** or **W** in the computation of indirect context vectors via Definition 4.1.1 (p. 109):

$$
\begin{aligned}
\mathbf{c}^{\mathrm{W}}(\kappa) &= \mathbf{W}^T\mathbf{c}^1(\kappa) && (7.1.6)\\
\mathbf{c}^{\mathrm{L}}(\kappa) &= \mathbf{L}^T\mathbf{c}^1(\kappa) && (7.1.7)\\
\mathbf{c}^{\mathrm{H}}(\kappa) &= \mathbf{H}^T\mathbf{c}^1(\kappa) && (7.1.8)
\end{aligned}
$$

Algorithm 7.1 Word Matrix dimension consolidation

```
Input: $\mathbf{W}$ of size $f_1 \times f_1$, $\mathbf{H}$ of size $f_1 \times f_1$
Output: $\widehat{\mathbf{W}}$: a version of $\mathbf{W}$ in which columns of significantly associated words (as encoded by $\mathbf{L}$) have been aggregated (summed) into single columns.
Begin
    1. $\widehat{\mathbf{W}}$ is initially empty (we will append columns to it as we go along)
    2. Initialise map to an empty set. map is a hash that maps $\mathbf{W}$ columns to $\widehat{\mathbf{W}}$ columns.
    3. $\widehat{\mathbf{w}}_1^{(c)} \leftarrow (\mathbf{w}_1^{(r)})^T$ (i.e. add transpose of first row vector $\mathbf{w}_1^{(r)}$ in $\mathbf{W}$ as first column vector of $\widehat{\mathbf{W}}$).
    4. Add mapping $1 \rightarrow 1$ to map.
    5. Initialise: $i \leftarrow 2; \hat{\imath} \leftarrow 2$
    6. For row vector $\mathbf{h}_i^{(r)}$:
        6.1. If $h_j = 0 \, \forall h_j \in \mathbf{h}_i^{(r)}$ for $j < i$, then: (i.e. if the first $i - 1$ elements of vector $\mathbf{h}_i^{(r)}$ are 0).
            6.1.1. $\widehat{\mathbf{w}}_{\hat{\imath}}^{(c)} \leftarrow (\mathbf{w}_i^{(r)})^T$
            6.1.2. Add mapping $i \rightarrow \hat{\imath}$ to map.
            6.1.3. $\hat{\imath} \leftarrow \hat{\imath} + 1$
        6.2. Otherwise:
            6.2.1. $j_f \leftarrow \min\{j | h_j \neq 0\}$ (i.e. $j_f$ is the first non-zero dimension in $\mathbf{h}_i^{(r)}$)
            6.2.2. $\hat{\jmath}_f \leftarrow \mathrm{map}(j_f)$
            6.2.3. $\widehat{\mathbf{w}}_{\hat{\jmath}_f}^{(c)} \leftarrow \widehat{\mathbf{w}}_{\hat{\jmath}_f}^{(c)} + (\mathbf{w}_i^{(r)})^T$
            6.2.4. Add mapping $i \rightarrow \hat{\jmath}_f$ to map.
        6.3. $i \leftarrow i + 1$
    7. End for
End
```

In the consolidation algorithm considered in the following section, the matrix $\mathbf{H}$ is used to identify associated words whose dimensions can be consolidated into one.

7.2 Reduction by consolidation

In this section we propose an algorithm that consolidates dimensions of word pairs that occur more often than by chance. The algorithm takes as input the unreduced $\mathbf{W}$ matrix and its corresponding $\mathbf{H}$ matrix. It starts by initialising the output object $\widehat{\mathbf{W}}$ to an empty matrix and an intermediary object map which keeps track of which $\mathbf{W}$ dimensions have been consolidated to which $\widehat{\mathbf{W}}$ dimensions. Let $\mathbf{w}_i^{(r)}$ and $\mathbf{w}_j^{(c)}$ be the i^{th} and j^{th} row vector and column vector, respectively, of $\mathbf{W}$ and, similarly, let $\widehat{\mathbf{w}}_i^{(r)}$ and $\widehat{\mathbf{w}}_j^{(c)}$ have the same analogous relationship with $\widehat{\mathbf{W}}$, as well as $\mathbf{h}_i^{(r)}$ and $\mathbf{h}_j^{(c)}$ with $\mathbf{H}$. Then, step 3 takes the transpose of the first row vector of $\mathbf{W}$, $(\mathbf{w}_1^{(r)})^T$, and makes it the first column of $\widehat{\mathbf{W}}$, $\widehat{\mathbf{w}}_1^{(c)}$. Notice that both $\mathbf{W}$ and $\mathbf{H}$ are symmetric matrices. The algorithm then proceeds to process the rest of the rows in both $\mathbf{H}$ and $\mathbf{W}$. Notice there are two counters in the algorithm: i, which keeps track of the *current row/column* in $\mathbf{W}$ and $\mathbf{H}$, and $\hat{\imath}$, used to track the *next column* to append to $\widehat{\mathbf{W}}$. Step 6.1 tests whether all the dimensions *before the diagonal* in the current row vector $\mathbf{h}_i^{(r)}$ are zero. If so, then the word represented by the i^{th} dimension in the matrix *is not statistically associated with any word represented by a lower dimension*, and therefore that dimension is copied as is to $\widehat{\mathbf{W}}$ as a new dimension (step 6.1.1). Alternatively, if there is at least a non-zero in a dimension before the diagonal, then the word represented by the the i^{th} dimension in the matrix *is indeed statistically associated with a word represented by a lower dimension*. In this case, the i^{th} dimension in $\mathbf{W}$ is consolidated (through addition) *with the lowermost statistically associated dimension* in $\widehat{\mathbf{W}}$, i.e.

the left-most non-zero dimension (steps 6.2.1 and 6.2.3).

Notice that the consolidated matrix produced by this algorithm still contains features that represent frequencies (or rather sums of frequencies). Therefore, it is still possible to transform these summed frequencies into log-likelihood scores or even hypothesis test result features as described previously. Notice as well that because the algorithm seeks to consolidate higher dimensions into lower dimensions, the result of the consolidation and the number of final dimensions will vary depending on how dimensions are ordered in **W** and **H**. The number of words that any given word is associated to can vary greatly. Placing words that are associated to many words in the lower dimensions will produce smaller consolidated matrices conflating much information into a few dimensions, possibly at the expense of losing discrimination potential. Conversely, placing those highly associated words in the upper dimensions will still compress the dimensionality, but to a lower degree. For the purposes of this chapter, we would like to propose four simple, straightforward dimension ordering techniques depending on the number of associations that a word has: 1. **Ascending order** (words with few associations occupy the lower dimensions and words with many associations occupy the higher dimensions), 2. **Descending order** (reverse of 1), 3. **Distance from the median** number of associations in ascending order and 4. **Distance from the mean** in ascending order. We also propose two additional ordering techniques: 5. **Word frequency** in descending order and 6. **Inverse Document Frequency (IDF)**[1] in ascending order. IDF could potentially be an interesting way of ordering features because words that appear across many contexts might be associated with many words but will tend to have low IDF and therefore be placed in the upper dimensions, limiting the way in which other words can be consolidated into them. These particular sorting criteria are being developed and tested in this thesis and to the best of the author's knowledge, they have not been used for this purpose before.

7.3 Experiments and conclusions

Supervised WSD experiments and unsupervised WSDisc experiments were carried out using the HILS dataset (Sec. 5.1, p. 136). Each ambiguous word subcorpus was randomly split into a training portion (60%) and a test portion (40%). To increase robustness, this splitting was done four times and experiments were carried out on each splitting and then the results averaged.

For the supervised WSD experiments, a simple Rocchio classifier based on cosine similarity was trained. Evaluation was performed via sense classification on the test portion. For the unsupervised WSDisc experiments, a K-Means clusterer based on cosine similarity was trained. We evaluate by computing accuracy scores of every cluster-to-gold standard sense mapping of test vectors and report the maximum score as the overall accuracy score. For further robustness, each K-Means exercise was run 30 times.

[1]For the purposes of this chapter, we take a word window as the interpretation of *document* in IDF.

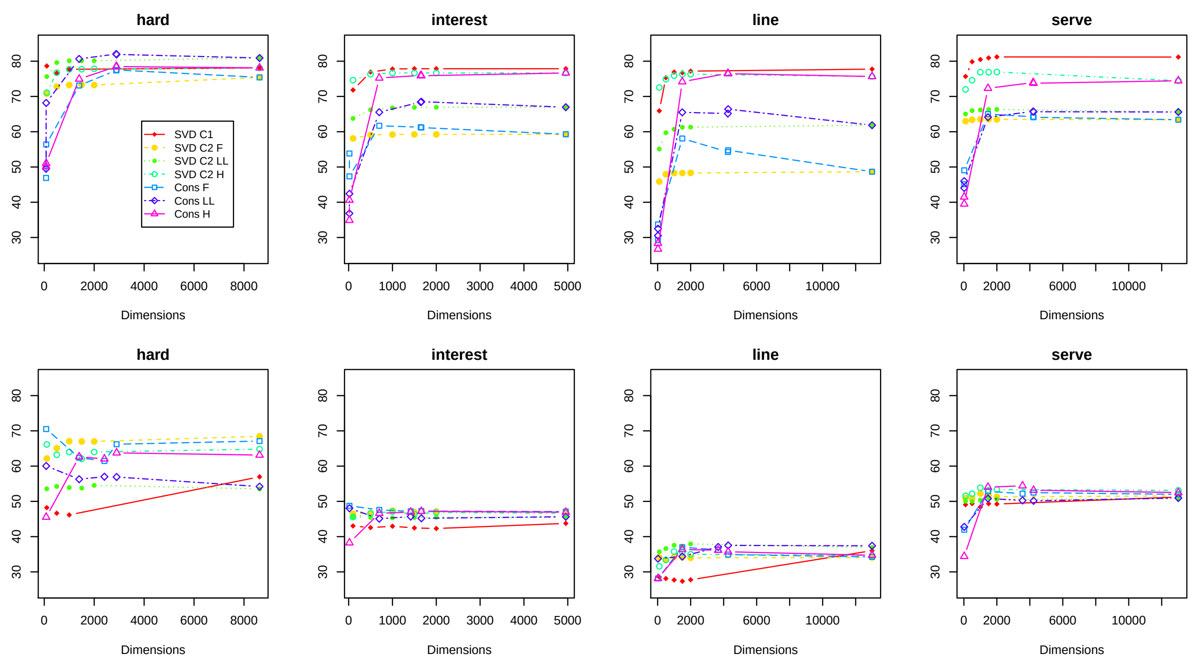


Figure 7.3.1: SVD and consolidation summary – The top of the graph shows **supervised experiment** results for ambiguous words (hard, interest, line and serve) whilst the bottom of the graph shows **unsupervised experiment** results for the same set of ambiguous words. The Y axis in all plots represents Accuracy whilst the X axis represents number of dimensions kept. The legend inside the supervised-hard plot applies to all plots.

For these experiments, SVD-reduced $\mathbf{c}^1$ vectors were created. Also, reduced $\mathbf{c}^2$ vectors were produced via SVD-reduced and consolidation-reduced word matrices. For all experiments word windows of size 20 centred at the target word (i.e. 10 words to the left and 10 words to the right) were used. All SVD reductions were performed using the R_1 projection (Sec. 3.4.2, p. 81).

Figure 7.3.1 plots the results of these consolidation experiment combinations. Since different ordering techniques yield different word matrices and different compression ratios, we decided to plot these together in the figure. For example, 'Cons F' represents results of $\mathbf{c}^2$ vectors constructed from consolidated word matrices that employ frequency features (LL stands for log-likelihood and H for binary hypothesis test results features). Each ordering technique mentioned in Section 7.2 achieves the following average degree of dimensionality reduction: frequency and descending order: 99.6%, mean: 86.7%, median: 71.97%, IDF: 67.1%, ascending order: 66.9%. For all lines depicted in the figure, the point at the far right represents the result obtained without performing any reduction.

From the figure we can see that for supervised experiments the best performer for three words is SVD for $\mathbf{c}^1$ vectors, with consolidation based on hypothesis test features closely following it. For *hard*, however, the best result was provided by consolidation with log-likelihood features using IDF/ascending order (around 3000 dimensions). In the unsupervised case, scores performed very similarly for three words. For *hard*, there was a strong preference for consolidation with frequency features using frequency/descending order (extreme left). Around the dimensions obtained by the median and IDF/ascending order, the results for the consolidation techniques are slightly better or the same as for SVD. While more extensive experimentation is needed (i.e. by using dataset with more words like those used in recent SemEval competitions as well as employing other clustering algorithms), it seems that the word matrix dimension consolidation technique can be a good alternative to SVD in unsupervised WSDisc. An advantage of the consolidation method over SVD is that an arbitrary number of dimensions does not need to be specified in advance by the user. Instead, the user specifies a dimension ordering criterion, which in turn controls the degree of dimensionality reduction. As a rule of thumb, it seems that those ordering criteria that perform compressions of round 67% are safe options, especially for supervised experiments.

This chapter has shown that the computation of $\mathbf{c}^2$ vectors can be seen as a matrix transformation involving a multiplication of $\mathbf{c}^1$ vectors by a word matrix, with the potential to reduce the dimensionality of the resulting $\mathbf{c}^2$ vectors. As a secondary contribution, this chapter also introduces a new type of word matrix features: binary hypothesis test results.

This work can be extended in many different ways. For example, by using weighting schemes during consolidation. But since the matrix by which $\mathbf{c}^1$ vectors are multiplied can be any arbitrary matrix, it could be possible to select a suitable matrix that uses features other than word co-occurrence statistics.

8 Measuring MWE compositionality via Word Space

Sinclair (1991, ch. 8) advances two principles in which meaning arises from language: an **open-choice principle** and an **idiom principle** (see Section 2.2.5, p. 52). The open-choice principle states that after a single word is completed in a stream of text or speech, there is a vast amount of possible words that can follow that word, and the only condition restricting exactly which words are admissible is grammaticalness. The idiom principle, on the other hand, states that a language user has at his or her disposal a vast set of pre-constructed phrases that can be used as whole units, even if these phrases could be separated and analysed into smaller constituent units or even individual words. Sinclair argues that in normal language use, it seems that speakers tend to strike a balance between the two principles, with some bias towards recycling pre-constructed phrases[1]. Possible motivations for this phrase re-use could lie in economy of effort or the quick nature of real-time conversation, or perhaps simply because similar situations tend to recur in life. As it was studied in Section 2.1.2 (p. 33) the notion of recurring multi-word phrases is termed **collocation** in the linguistics literature, which in the computational linguistics / natural language processing literature, the term **multi-word expression** is often used instead.

As noted in Section 2.1.2, it seems that there are no hard rules to determine what constitutes a collocation other than statistical significance and near co-occurrence and of course, human intuition. In fact, most computational studies on collocations rely on probabilistic criteria to automatically identify collocations in corpora (see e.g. Church and Hanks (1990); Church et al. (1991) and Manning and Schütze (1999, ch. 5)). However, a naïve frequentist analysis could lead to labelling frequent, uninteresting word combinations such as *in the* as collocations. And even if stop words are filtered out, frequent and purely compositional word combinations could be captured as a collocation. As mentioned in Section 2.1.2 (p. 33), Benson (1989) puts forward a lexicographical approach for determining whether a frequent word combination is a collocation by listing as lexical units in a dictionary only those collocations with peculiar semantics and avoiding any free, trivial and unnecessary word combinations. For example, the following adjectives should not be considered collocates of *evidence*: *absolute, additional, certain, considerable, conspicuous, gratifying, hopeful*, even if they frequently modify this noun. This is because they form normal and literal "adjective plus noun" combin-

[1]He notes however that in certain traditions, such as in poetry, there will be a bias towards originality and therefore to the open-choice principle.

ations that any learner or speaker of English could determine by just applying this grammar rule without recurring to any external knowledge. Benson mentions that collocations are not only frequent word combinations but **arbitrary** and frequent word combinations.

From a computational perspective, it is important to detect and handle this arbitrariness in a number of applications. For example, a machine translation engine should be able to translate the French *prendre une décision* as *make a decision* and not literally as **take a decision*.

Because of the importance of handling non-literal collocations correctly, the non-compositionality feature is usually regarded as the most important characteristic of a collocation. And so the early work on statistical word association (Church and Hanks, 1990; Church et al., 1991) has been extended to detect non-compositional collocations (Lin, 1999; Bu and Zhu, 2010). But some efforts have been made in applying the Vector Space Model in the detection of compositionality (Schone and Jurafsky, 2001; Baldwin et al., 2003; Bannard et al., 2003). It was soon realised that it was difficult to categorically distinguish every collocation as either compositional or non-compositional, and that instead shades or degrees of compositionality should be considered for a given collocation (Bannard et al., 2003; Katz and Giesbrecht, 2006). For example, while it is clear that *red herring* is completely non-compositional and *orange juice* is very much compositional, *heavy smoker* lies somewhere in between: it does describe a type of smoker, one who smokes too much but not an overweight one. So, we could say that *heavy smoker* has some sort of medium-to-low compositionality but it is by no means fully non-compositional. In Section 2.1.2 *red light* was given as a collocation example that, depending on meaning or use, will have a different compositionality degree. It is clear that in Example (2.1.2) (p. 36) *red light* is clearly compositional since it literally describes light that is red in colour. Whilst (2.1.3) describes a scenario in which a light that is red in colour is involved, it can be argued that *red light* has acquired the more abstract meaning of a traffic control device signalling the car to stop. This usage involves some literal usage (the signal is in the form of a light that is red in colour) but it also carries other meanings associated to traffic conventions as well as electronic devices designed to enforce these conventions in actual streets. Since (2.1.4) involves decoding a metaphor by exploiting additional knowledge (i.e. a notion of traffic conventions mapped into a business situation), this usage of *red light* exhibits very low compositionality.

The remainder of this chapter describes our research in measuring the degree of compositionality of collocations (Maldonado-Guerra and Emms, 2011) and our participation in the shared task at the 2011 Workshop on Distributional Semantics and Compositionality (Biemann and Giesbrecht, 2011).

8.1 DISCO 2011 Shared Task

This section describes the shared task and evaluation methods whilst Section 8.2 describes the system and methods implemented and Section 8.4 gives details about the performance of the system.

The shared task at the 2011 Workshop on Distributional Semantics and Compositionality (DISCO) (Biemann and Giesbrecht, 2011) was the first attempt at creating a common publicly available dataset and evaluation framework for the measurement of compositionality of multi-word expressions[2].

8.1.1 The multi-word expressions

The organisers of the shared task provided a list of multi-word expressions (MWEs), all of them bigrams, in German (303 MWEs) and in English (349 MWEs) and a copy of the German and English WaCky corpora (Baroni et al., 2009) each having a size between 1 and 2 billion tokens. Two versions of each corpus were provided: a raw, plain-text version of the corpus and a tokenised, part-of-speech-tagged (POS-tagged) and lemmatised version.

In order to obtain gold-standard compositionality scores for these MWEs, each MWE was judged for compositionality in a scale between 0 and 10 by a human annotator, with 0 being fully non-compositional and 10 being fully compositional. Human annotators were able to see the MWE used in a sample sentence. There are at most 5 sample sentences per MWE, and each sample sentence-MWE combination is judged for compositionality by at most 4 human annotators[3]. This gives a total of at most 20 compositionality scores per MWE. The average of these 20 judgements for an MWE was multiplied by 100 and used as the overall compositionality judgement for that MWE.

These overall compositionality judgements are termed the *numerical human judgements* or the *numerical scores* and submitted systems were evaluated based on the average difference between their numerical compositionality outputs and these numerical human judgements. However, since some applications could benefit more from a less fine-grained classification of MWEs into a *low compositionality* category, *medium compositionality* category and a *high compositionality* category, these numerical scores were converted into what the shared task organisers call *coarse-grained compositionality labels* or *bins*. Organisers categorised MWEs based on their numerical score into these three categories by applying the following rules: MWEs with numerical judgements in the range 0-25 were deemed to be in the *low compositionality* bin, MWEs with judgements between 38 and 62 fell into the *medium compositionality* bin, whilst MWEs in the 75-100 range were assigned to the *high compositionality* category. Any MWEs that fell in the 26-37 and 62-74 ranges were not categorised and were excluded from the coarse-grained evaluation of systems. Systems were expected to classify each MWE into a *high*, *medium* or *low* category and they were evaluated using a precision score against the organiser's categorisation.

The $\sim$300 MWEs per language were randomly split into a development set and a test set

[2]Details on the shared task, results, datasets and evaluation scripts are available at http://disco2011.fzi.de/
and
http://aclweb.org/aclwiki/index.php?title=DISCo_2011_shared_task_data:_Compositionality_judgments_%28Repository%29

[3]A small number of sample sentences were removed because annotators expressed they did not provide useful context and a few judgements given by some annotators were deemed invalid by the shared task organisers for various reasons.

Table 8.1.1: Distribution of MWEs between the development (train and validation) set and the test set. The right-hand side of the table shows the distribution of the MWEs selected for the coarse-grained evaluation (MWEs with numeric scores near the borders around the low-medium and medium-high borders were removed)

All MWEs					Selected for coarse-grained eval			
Language	EN		DE		EN		DE	
Train	140	40.11%	119	39.27%	107	34.85%	98	36.30%
Validation	35	10.03%	35	11.55%	26	8.47%	23	8.52%
Test	174	49.86%	149	49.17%	174	56.68%	149	55.19%
TOTAL	349		303		307		270	

in an approximately 50%-50% manner. The development set was further divided into a 80% training and 20% validation subsets. The development set was provided to participants early in the competition in order to develop and fine tune their systems or to perform any training that was deemed to be necessary. The test set was provided to participants only during the last week of the competition and the official evaluation scores was based on the system's output (both numerical and coarse-grained) of this test set. The numerical and coarse-grained human judgements on the test set were withheld by the organisers during the competition and were not made available until some time after the end of the competition. Table 8.1.1 details how MWEs are actually distributed across the the development and the test sets, highlighting the slight difference in distribution in the MWEs selected for coarse-grained evaluation.

There were three part of speech combination types for the MWEs provided in the shared task: adjective + noun (e.g. *big fish*), subject (noun) + verb (e.g. *client wants*) and verb + object (noun) (e.g. *improve access*).

8.1.2 Evaluation of systems' output

As previously mentioned, a system's output is evaluated only on its ability of measuring the compositionality of the test set's MWEs. Performance scores are computed for the numerical task and for the coarse task as follows.

The performance score of a system in the numerical task is measured as the average difference between the numerical gold-standard human judgement and the system's numerical compositionality output scores. If G is the set of numerical gold-standard human judgements for all MWEs and if S is the system's numerical compositionality output scores for all MWEs, then the system's score in the numerical task is *NUMSCORE* and is computed as:

$$NUMSCORE(S, G) = \frac{1}{N}\sum_{i=1}^{N} |g_i - s_i| \qquad (8.1.1)$$

Where g_i is the gold-standard score for the i-th MWE, s_i is the system score for the i-th

MWE and N is the size of G. If the system does not provide a value for one or more s_i, the scoring script automatically assumes a value of 50 for that s_i. Since *NUMSCORE* is an average difference score, the lower the value, the better the performance. A score of 0 would be a perfect score.

For the coarse-grained task a precision score is used. If S and G are, respectively, the system's and gold-standard coarse labels for all MWEs, then the system's score in the coarse-grained task is *COARSE* and is computed as:

$$COARSE(S,G) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases}1 & g_i = s_i\\ 0 & otherwise\end{cases} \tag{8.1.2}$$

Where g_i is the gold-standard label category (*low*, *medium* or *high*) for the i-th MWE, s_i is the system label category for the i-th MWE and N is the size of G. If a system does not report a label for an MWE, a value of *medium* is assumed for that MWE. Since this is a precision score, the higher the value, the better. A score of 1 would be a perfect score.

As previously mentioned, evaluations were performed on all MWEs in the test set and overall numerical task and coarse-grained task evaluation scores were provided on them. In addition, MWEs were scored by part-of-speech combination type (ADJ+N, V+SUBJ, V+OBJ).

In addition to these scores, the organisers also computed the Spearman's rho and Kendall's tau correlation values on the numerical compositionality scores.

8.1.3 Issues regarding the human judgements

An issue with this method of averaging compositionality judgements for an MWE used in different sentences is that it conflates different senses or usages of the MWE that potentially have different degrees of compositionality. This concern can be verified from the shared task data itself. Sentences (8.1.4), (8.1.3) and (8.1.5) are actual samples from the English WaCky corpus used as context in the manual compositionality annotation of *heavy metal*:

(8.1.3) Current projects Emissions of mercury (Hg) and other *heavy metals* from volcanoes.

(8.1.4) On the north shore is a large fort, which has a formidable battery of *heavy metal*, over which a strong guard is mounted every day.[4]

(8.1.5) *Heavy metal* supergroup Velvet Revolver continue to struggle in the UK with their new single 'Fall To Pieces'.

In the 0-100 scale, the average human compositionality judgement for *heavy metal* in (8.1.3) was 87.5. A possible explanation for this high compositionality assessment might be due to the categorisation of mercury as a heavy metal in Chemistry and/or to its popular perception as a metal that is heavy in mass/weight, i.e. literally "a heavy metal". In (8.1.4), *heavy*

[4]This sentence is part of a description of the fort in a (now closed) prison in Liverpool. The full text of the article can be retrieved from http://www.genuki.org.uk/big/eng/LAN/Gazetteer/L.htm

metal received a medium compositionality evaluation of 67.5. In this sentence, the meaning of *battery* is "the platform or fortified work, on or within which artillery is mounted (sometimes including the guns or mortars there mounted)"[5]. Despite the sentence describing a platform made literally of a metal that is heavy, it only achieved a medium compositionality score, possibly due to the rather obscure or specialised military sense of *battery*. Perhaps if the annotators had been more familiar with the sense of the word or if more context had been provided, a higher compositionality score might have been assigned for *heavy metal* in this sentence. By contrast, it was clear for annotators that the music-related sense of *heavy metal* as used in (8.1.5) is of low compositionality since they assigned it an average judgement of 20.

The overall average of these three distinct senses or usages of *heavy metal* would be 58.3, a medium compositionality judgement[6]. It is easy to see that considering this average as a single gold-standard score for every occurrence of the MWE is a very strong weakness in the shared task evaluation method. If usages of an MWE are both highly compositional and highly non-compositional and if sample sentences are equally distributed across these usages, the compositionality score would tend to be around 50. Since it has been observed that polysemous words usually have skewed sense distributions (McCarthy et al., 2004)[7], it is reasonable to assume that a polysemous MWE will also have a skewed sense or usage distribution. Therefore, an overall average of human judgements based on sentences drawn from a sufficiently big random sample of a corpus could be seen as a weighted human judgement giving more weight to more frequent senses or usages of the MWE than rarer senses or usages. Unfortunately, as it has been mentioned previously, the sample taken from the corpus was an extremely small 5 sentences per MWE, making the obtained average an unrealistic representation of the overall sense/usage distribution for an MWE.

Nevertheless, configuration 3 of the system submitted to the competition (p. 169) assumes that the golden-standard score is indeed a weighted average of compositionality scores, with a bias towards the most frequent senses, and thus attempts to detect whether an MWE has different senses and attempts to construct a weighted compositionality score from the senses it detected.

8.2 Methodology

This section describes the compositionality measurement methods we developed for the DISCO shared task. One source of inspiration for the methods used by our system for measuring compositionality is the work of McCarthy et al. (2003), who measured the similarity between a

[5]Oxford English Dictionary Online. Entry for *battery*, sense 5. http://www.oed.com

[6]The overall average for the five sentences annotated (that include the three here presented) was 52.5, a medium compositionality score.

[7]Some senses of a word are much more frequent than other senses. As an example, contrast the very frequent electrical meaning of *battery* (as in 'this radio takes AA batteries') vs. its much rarer military sense illustrated by (8.1.4).

phrasal verb (a main verb and a preposition like *blow up*) and its main verb (*blow*) by comparing the words that are closely semantically related to each, and use this similarity as an indicator of compositionality. However, our method is more similar to that of Katz and Giesbrecht (2006), who exploited the same intuition but in a different manner. First, they built "true" type vectors for all MWEs and a corresponding estimated "composed" type vector for each candidate MWE. This candidate composed type vector is the sum of the MWE's constituent words, essentially a vector compositionality method (Guevara, 2011). They argue that these composed vectors estimate the type vector of the MWE in compositional usages. A cosine value between this composed vector and the true MWE vector is interpreted as an estimate of the MWE compositionality: if the value is high (i.e. if there is little difference between the composed and true vectors), they conclude that the MWE is compositional, otherwise they conclude it is non-compositional since the contexts of the composed and true vectors are different. They obtained relatively good results in their test corpus and a set of German MWEs.

The method presented here also uses the cosine measure as a proxy for compositionality grading. However, it does not compute any vector estimates. Instead, it computes cosine similarities directly between the MWE vector and each of the constituent's word vectors in three different configurations, all of which are defined in Section 8.3. Our system can be regarded as fully unsupervised as it does not employ any parsers in its processing or any external data other than the corpus and a candidate collocation list. Within the setting of the DISCO shared task, this candidate collocation list was compiled by the organisers and provided to participants. In an application scenario, this list could be automatically extracted from a corpus.

8.3 Preliminary definitions

All sliding windows for the word vectors and context vectors used are of length $l = 20$, centred at the target word or target bigram. The feature vocabulary $\mathcal{V}_m$ (vector dimensions) for all vectors is set to the top $m = 2,000$ most frequent content words in the corpus (i.e. stop words are excluded). Although definitions in Section 3.5 (p. 95) are for types and tokens of individual words, they can be extended to multi-word expressions in various ways. Since the DISCO shared task only considers bigrams, we have extended the distributional vectors for bigrams only as follows: for any **bigram type** $\tau\ \upsilon$, its **tokens** are taken to be occurrences of the sequence $\kappa\gamma\lambda$, where κ is the token realising τ (i.e. $\tau = \text{type}(\kappa)$), γ can be any sequence of intervening words of length t, $0 \leq t \leq 3$, and λ is the token realising υ (i.e. $\upsilon = \text{type}(\lambda)$). In sum, $\kappa\gamma\lambda$ is a token of $\tau\ \upsilon$. In this way, the definitions of direct (first-order) and indirect (second-order) context vectors, as well as word vectors, can be carried over to multi-word expressions.

We assume that each collocation (bigram) provided has a **node** (headword) and a **collocate** (modifier). Given a bigram we assign a node and a collocate role to each of its constituent

Table 8.3.1: Node/collocate assignment by part of speech combination in collocation

POS Combination	**Node**	**Collocate**
Adjective + Noun	Noun	Adjective
Subject (noun) +Verb	Verb	Subject
Verb + Object (noun)	Verb	Object

words based on the part of speech of these words. As previously mentioned, MWEs were provided in three different part of speech combinations: adjective + noun, subject (noun) + verb and verb + object (noun). Table 8.3.1 describes how the node/collocate assignment was performed based on this.

The system implements three methods. These three methods are implemented via three configurations detailed as follows:

Configuration 1 Given a collocation candidate $\tau\ \upsilon$, assuming that τ is the collocate and υ is the node, the following objects are built from a corpus:

1. A word vector for the collocation candidate, following the extension described above: $\mathbf{w}(\tau\ \upsilon)$
2. A word vector for the collocate: $\mathbf{w}(\tau)$
3. A word vector for the node: $\mathbf{w}(\upsilon)$

Then, configuration 1 measures the compositionality of the collocation candidate as the average of two cosine similarity measures: the measure between the collocation candidate word vector $\mathbf{w}(\tau\ \upsilon)$ and the collocate word vector $\mathbf{w}(\tau)$, and the measure between the collocation word vector and the node vector $\mathbf{w}(\upsilon)$.

$$c_1 = \frac{1}{2}\begin{bmatrix} \cos\left(\mathbf{w}(\tau\ \upsilon), \mathbf{w}(\tau)\right) \\ + \cos\left(\mathbf{w}(\tau\ \upsilon), \mathbf{w}(\upsilon)\right) \end{bmatrix} \tag{8.3.1}$$

Configuration 2 The second configuration of our system compares the occurrences of the node υ when accompanied by the collocate τ forming the collocation candidate $\tau\ \upsilon$ separately, with the occurrences of the τ appearing outside the vicinity of υ (i.e. not forming $\tau\ \upsilon$). If $\text{coll}(\kappa, \tau)$ is a binary function that returns 1 if a specific token κ forms a collocation with τ (i.e. whether in the vicinity of κ there is a token instance of τ) and 0 otherwise, let

$$\mathbf{w}^{\tau}(\upsilon) = \sum_{\forall\kappa:\text{type}(\kappa)=\upsilon} \mathbf{c}^1(\kappa)\,\text{coll}(\kappa, \tau) \tag{8.3.2}$$

be the word vector computed from all the occurrences of the node υ that form a collocation with τ and conversely, let

$$\mathbf{w}^{\overline{\tau}}(\upsilon) = \sum_{\forall\kappa:\text{type}(\kappa)=\upsilon} \mathbf{c}^1(\kappa)\,(1 - \text{coll}(\kappa,\tau)) \tag{8.3.3}$$

be the word vector representing the occurrences of υ not occurring in the vicinity of τ. In this configuration, the compositionality score is then computed by

$$c_2 = \cos\left(\mathbf{w}^{\tau}(\upsilon), \mathbf{w}^{\overline{\tau}}(\upsilon)\right) \tag{8.3.4}$$

The intuition behind this configuration is that if the node tends to co-occur with more or less the same words in both cases (producing a high cosine score), then the meaning of the node is similar regardless of whether the collocation's collocate is present or not, implying a high degree of compositionality. If on the other hand, the node co-occurs with somewhat differing words in the two cases (a low cosine score), then we assume that the presence of the collocation's collocate is markedly changing the meaning of the node, implying a low degree of compositionality.

Configuration 3

This configuration attempts to address the issue of polysemy in collocation candidates as exemplified with the case of *red light*. A clustering technique as used in word-sense discrimination is employed in order to exploit semantic differences that may naturally emerge from each context in which the collocation candidate and its constituents are used.

In word-sense discrimination and induction, clustering is used to group occurrences of a target word according to its sense or usage in context as it is expected that each cluster will represent a different sense or usage of the target word. However, since within the setting of the DISCO shared task, the contexts that human annotators referred to when judging the compositionality of the collocations were not provided, our system employs a workaround that uses a weighted average when measuring compositionality. This workaround is explained in what follows.

In this configuration, the system first builds word vectors for the 20,000 most frequent words in the corpus (see Eq. 3.5.6 on page 101), and then uses these to compute the second-order context vectors for each occurrence of the collocation and its constituents in the corpus (3.5.7). After context vectors for all occurrences have been computed, they are clustered using CLUTO's repeated bisections algorithm[8]. The vectors are clustered across a small number K of clusters (we employed $K = 4$). The expectation was that each cluster will represent a different contextual usage of the collocation, its node and its collocate. Figure 8.3.1 depicts how a context vector space could be partitioned with $K = 4$.

The system then for each cluster k builds the word vectors $\mathbf{w}_k(\tau\ \upsilon)$, $\mathbf{w}_k(\tau)$, and $\mathbf{w}_k(\upsilon)$ for the collocation candidate, its node and its collocate, from the contexts grouped within the cluster k. The compositionality measure for the third configuration is then basically a

[8]http://glaros.dtc.umn.edu/gkhome/views/cluto/

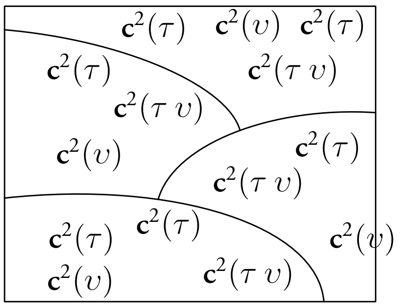

$\mathbf{c}^2(\tau)$ $\mathbf{c}^2(\upsilon)$ $\mathbf{c}^2(\tau)$
$\mathbf{c}^2(\tau)$
$\mathbf{c}^2(\tau\ \upsilon)$
$\mathbf{c}^2(\tau\ \upsilon)$
$\mathbf{c}^2(\upsilon)$
$\mathbf{c}^2(\tau)$
$\mathbf{c}^2(\tau\ \upsilon)$
$\mathbf{c}^2(\tau)$ $\mathbf{c}^2(\tau)$
$\mathbf{c}^2(\upsilon)$
$\mathbf{c}^2(\upsilon)$ $\mathbf{c}^2(\tau\ \upsilon)$

Figure 8.3.1: Conceptual depiction of a clustered second-order context vector space.

weighted average over the clusters of the c_1 score using each cluster, that is:

$$c_3 = \sum_{k=1}^{K} \frac{\|\mathcal{C}_k\|}{2N} \begin{bmatrix} \cos\left(\mathbf{w}_k(\tau\ \upsilon), \mathbf{w}_k(\tau)\right) \\ + \cos\left(\mathbf{w}_k(\tau\ \upsilon), \mathbf{w}_k(\upsilon)\right) \end{bmatrix} \tag{8.3.5}$$

where $\|\mathcal{C}_k\|$ is the number of contexts in cluster $\mathcal{C}_k$ and N is the total number of contexts across all clusters.

For all three configurations, the value reported as the numeric compositionality score was the corresponding value obtained from (8.3.1), (8.3.4) or (8.3.5), multiplied by 100. Each configuration's numeric scores c_i were binned into the three coarse compositionality classes by comparing them with the configuration's maximum value through (8.3.6).

$$\text{coarse}(c_i) = \begin{cases} \textit{high} & \text{if } \frac{2}{3}max \leq c_i \\ \textit{medium} & \text{if } \frac{1}{3}max < c_i < \frac{2}{3}max \\ \textit{low} & \text{if } c_i \leq \frac{1}{3}max \end{cases} \tag{8.3.6}$$

8.4 Results and conclusion

Table 8.4.1 shows the evaluation results for the three system configurations and two baselines. The left-hand side of the table shows the average difference between the gold-standard numeric score and each configuration's numeric score. The right-hand side reports the precision on binning the numeric scores into the coarse classes. Evaluation scores are reported on all collocations and on the collocation subtypes separately. Row **R** is the baseline suggested by the workshop organisers, assigning random numeric scores, in turn binned into the coarse categories. Row **A** shows the performance of a constant output baseline, assigning all collocations the *mean* gold-standard numeric score from the training set: 66.45, and then applying the binning strategy (8.3.6) to this – which always assigns the coarse category *high*.

The first thing to note from this table is that configurations 1 and 2 generally outperform configuration 3, both on the mean difference and coarse scores. Configuration 1 slightly outperforms configuration 2 on the mean numeric difference scores, whilst configuration 2

Table 8.4.1: Evaluation results of the three system configurations and two baselines on the test dataset. Best system scores on each grammatical subtype highlighted in bold.

C	Average differences (numeric)				Precision (coarse)			
	ALL	A-N	S-V	V-O	ALL	A-N	S-V	V-O
1	**17.95**	**18.56**	20.80	**15.58**	53.4	**63.5**	19.2	62.5
2	18.35	19.62	**20.20**	15.73	**54.2**	**63.5**	19.2	**65.0**
3	25.59	24.16	32.04	23.73	44.9	40.4	**42.3**	52.5
R	32.82	34.57	29.83	32.34	29.7	28.8	30.0	30.8
A	16.86	17.73	15.54	16.52	58.5	65.4	34.6	65.0

Table 8.4.2: Some corpus statistics: the number of matched collocations per subtype (**Instances**) and the average number of intervening words per subtype (**Avg intervening**).

	A-N	S-V	V-O
Instances	177254	11092	121317
Avg intervening	0.0684	0.3867	0.4612

is very close to and slightly better than configuration 1 on the coarse precision scores. The exception is that configuration 3 was the best performer on the coarse precision scoring for the **S-V** subtype.

The R baseline is outperformed by configurations 1, 2 and 3; roughly speaking where 1 and 2 outperform R by d, configuration 3 outperforms R by around $d/2$. The A baseline generally outperforms all our system configurations. It seems to be also a quite competitive baseline for other systems participating in the shared task (see Tables 8.4.3 and 8.4.4).

The other trend apparent from the table is that performance on the **V-O** and **A-N** subtypes tends to exceed that on the the **S-V** subtype.

An examination of the gold standard test files shows that the distribution over the *low, medium* and *high* categories is similar for both **V-O** and **A-N**, in both cases close to 0.08/0.27/0.65, with *high* covering nearly two-thirds of cases, whilst for **S-V** the distribution is quite different: 0.0/0.654/0.346, with *medium* covering nearly two-thirds of cases. This is reflected in the A baseline precision scores, as for each subtype these will necessarily be the proportion of gold-standard *high* cases. This explains for example why the A baseline is much poorer on the **S-V** cases (34.6) than on the other cases (65.0, 65.4).

Looking further into the differences between the three subtypes, Figure 8.4.1 shows the gold standard numeric score distribution across the three collocation subtypes (**Test GS**), and the corresponding distributions for scores from the system's first configuration (**Conf 1**). This shows in more detail the nature of the poorer performance on **S-V**, with the gold standard having a peak around 50-60, and the system having a peak around 70-80. For the other subtypes the contrast in the distributions seems broadly consistent with the mean numeric difference scores of Table 8.4.1.

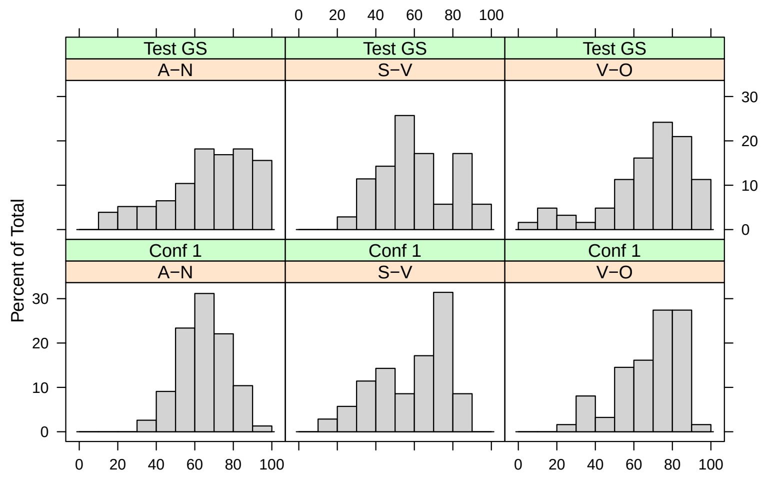


Figure 8.4.1: The distribution of the gold standard numeric score vs. the distribution of the system's first configuration numeric scores.

One can speculate on the reasons for the system's poorer performance on the **S-V** subtype. The system treats intervening words in a collocation in a particular way, namely by ignoring them. This is one option, and another would be to include them as features counted in the vectors. Table 8.4.2 shows the average intervening words in the occurrences of the collocations. **S-V** and **V-O** are alike in this respect, both being much more likely to present intervening words than collocations of the **A-N** subtype. So the explanation of the poorer performance on **S-V** cannot lie there. Also because the average number of intervening words is low, we believe it is unlikely that including them as features will impact performance significantly.

Table 8.4.2 also gives the number of matched collocations per subtype. The number for the **S-V** collocations is an order of magnitude smaller than for the other subtypes. Although the collocations supplied by the organisers are in their base form, the system attempts to match them 'as is' in the unlemmatised version of the corpus. Whilst for **A-N** and **V-O** the base-form sequences relatively frequently do double service as inflected forms, this is far less frequently the case for the **S-V** sequences (e.g. *user see* (**S-V**) is far less common than *make money* (**V-O**)). This much smaller number of occurrences for **S-V** cases, or the fact that they are drawn from syntactically special contexts, may be a factor in the relatively poorer performance. This perhaps is also a factor in the earlier noted fact that although configuration 3 was generally outperformed, on the **S-V** subtype the reverse occurs.

The unlemmatised version of the corpus was used because initial experimentation with the validation set produced slightly better results when employing raw words as features rather than lemmas. A possibility for future work would be to to refer to lemmas for matching collocations in the corpus, but to continue to use unlemmatised words as features.

In sum, the two simplest configurations of a totally unsupervised system yielded surprisingly good results at measuring compositionality of collocations in raw corpora. In comparison with

Table 8.4.3: Numerical evaluation results for all systems in the DISCO 2011 Shared Task for English (Biemann and Giesbrecht, 2011)

Numerical	Responses	ALL	A-N	S-V	V-O
Number of phrases		174	77	35	62
0-response baseline	0	23.42	24.67	17.03	25.47
Random baseline	174	32.82	34.57	29.83	32.34
UCPH-simple.en	174	**16.19**	14.93	21.64	**14.66**
UoY: Exm-Best	169	16.51	15.19	**15.72**	18.6
UoY: Pro-Best	169	16.79	**14.62**	18.89	18.31
UoY: Exm	169	17.28	15.82	18.18	18.60
SCSS-TCD: conf1	174	17.95	18.56	20.80	15.58
SCSS-TCD: conf2	174	18.35	19.62	20.20	15.73
Duluth-1	174	21.22	19.35	26.71	20.45
JUCSE-1	174	22.67	25.32	17.71	22.16
JUCSE-2	174	22.94	25.69	17.51	22.60
SCSS-TCD: conf3	174	25.59	24.16	32.04	23.73
JUCSE-3	174	25.75	30.03	26.91	19.77
Duluth-2	174	27.93	37.45	17.74	21.85
Duluth-3	174	33.04	44.04	17.60	28.09
submission-ws	173	44.27	37.24	50.06	49.72
submission-pmi	96	-	-	52.13	50.46
UNED-1: NN	77	-	17.02	-	-
UNED-2: NN	77	-	17.18	-	-
UNED-3: NN	77	-	17.29	-	-

the other systems submitted to the competition, configuration 1 and configuration 2 ranked in fifth and sixth places (out of 18 places), respectively, in the overall numeric evaluation. In the overall coarse score evaluation, configuration 2 reached fifth place and configuration 1 sixth place. Our system performed best in the **V-O** subtype: configuration 1 and configuration 2 ranked second and third places, respectively, in the numeric evaluation, whilst in the coarse evaluation, configuration 2 ranked first place with configuration 1 reaching a joint second-place with another system. Configuration 3 reached 10th place in the numeric evaluation and 9th place in the coarse evaluation. Tables 8.4.3 and 8.4.4 (adapted from Biemann and Giesbrecht, 2011) show the numeric evaluation results and the coarse evaluation results, respectively, for all systems in the competition. The SCSS-TCD systems correspond to the three configurations described here.

Table 8.4.4: Coarse evaluation results for all systems in the DISCO 2011 Shared Task for English (Biemann and Giesbrecht, 2011)

Coarse	Responses	ALL	A-N	S-V	V-O
Number of phrases		118	52	26	40
0-response baseline	0	0.356	0.288	0.654	0.250
Random baseline	118	0.297	0.288	0.308	0.300
Duluth-1	118	**0.585**	0.654	0.385	0.625
UoY: Exm-Best	114	0.576	0.692	0.500	0.475
UoY: Pro-Best	114	0.567	**0.731**	0.346	0.500
UoY: Exm	114	0.542	0.692	0.346	0.475
SCSS-TCD: conf2	118	0.542	0.635	0.192	**0.650**
SCSS-TCD: conf1	118	0.534	0.640	0.192	0.625
JUCSE-3	118	0.475	0.442	0.346	0.600
JUCSE-2	118	0.458	0.481	0.462	0.425
SCSS-TCD: conf3	118	0.449	0.404	0.423	0.525
JUCSE-1	118	0.441	0.442	0.462	0.425
submission-ws	117	0.373	0.346	0.269	0.475
UCPH-simple.en	118	0.356	0.346	0.500	0.275
Duluth-2	118	0.322	0.173	0.346	0.500
Duluth-3	118	0.322	0.135	**0.577**	0.400
submission-pmi	-	-	-	0.346	0.550
UNED-1-NN	52	-	0.289	-	-
UNED-2-NN	52	-	0.404	-	-
UNED-3-NN	52	-	0.327	-	-

9 Conclusions and future work

Perhaps, the most fundamental contribution presented in this thesis is the observation that the Word Space word matrix can be seen as a linear map that converts direct (first-order) context vectors in $\mathbb{R}^m$ into indirect (second-order) context vectors in $\mathbb{R}^n$ (Def. 4.1.1, p. 109). Although the operation described by this linear map is really equivalent to the sum or averaging operations traditionally described in the literature (Def. 3.5.4, p. 101), it has the advantage of making more transparent the application of SVD, and dimensionality reduction methods in general, to Word Space (Def. 4.2.1, p. 112; Def. 4.2.5, p. 117). It is from this contribution that all of the other theoretical contributions of the thesis emerge in one way or another. For example, this linear transformation formulation makes the relationships between LSA and Word Space presented in Chapter 4 more transparent and obvious. It also makes easier the viability of the transformation itself as a method of dimensionality reduction in its own right (Chapter 7), and helps understand how the different components of Word Space fit together (Chapter 3).

Chapters 5, 6, 7 and 8 test some of these theoretical insights in actual empirical settings. These chapters confirmed some of the expectations of the earlier chapters but sometimes provided surprising results, like the difference in performance between the two types of SVD projections for supervised and unsupervised experiments or the slightly bigger than expected difference in performance between LSA and Word Space context vectors.

This thesis lifts the veil of mysticism covering SVD in many computational lexical semantics works by showing that LSA and Word Space have some clear relationships. Researchers have intuitively acknowledged a link between the two models, with some making simplistic assumptions regarding this link. This is the first time these assumptions are tested and we show that whilst there is indeed a clear link between the two models, their relationship is not as straightforward as one might initially think. In addition, the experimental results reflect this non-trivial relationship between the models, indicating that success, or lack thereof, in one model is not necessarily indicative of the other model's performance. However, given that the relationship between the two models is clearly known, it is possible to use a system implementing one model in order to make representations in the other model by modifying the values appropriately in the system's matrices. These findings have wider implications in areas where semantic vectors are used: there are potentially many works that repeat experiments already published simply because there is a lack of transparency in the literature regarding the methods of semantic vector construction used. Works such as the present one and that of Dinu et al. (2012) show the potential for simplifying and clarifying the methods used in the field.

This chapter presents in a summarised form each of the key contributions made by this thesis, and provides future avenues to expand on these contributions.

9.1 Summary of contributions

The R_1 vs. R_2 projection contention in LSA The case was made that there are (at least) two contending formulations of SVD projections in the LSA literature which we called R_1 and R_2 (Secs. 3.4.2, p. 81 and 3.4.3, p. 86). It was argued that both projections do not preserve cosine measures between vectors and that therefore must perform differently in actual WSX experiments. And indeed, the experiments in Chapter 6 show very different performance results for both projection types. In the analysis of both projections in Section 3.4.2 it was mentioned that R_1 projections were better representations than R_2 projections, intuitively. The experiments in Chapter 6 show that, depending on the actual kind of context/segment vector used, the R_2 projections can perform much worse than R_1 projections or at least very similarly, but never better, something congruent with our analytical predictions. However, for several kind of vectors (notably LSA's segment vectors, D-A-R2), R_2 projections can perform surprisingly better than R_1 projections. Unfortunately the vast majority of works in the literature that employ LSA or SVD on Word Space do not specify exactly whether they employ R_1, R_2 or even some other way, and in consequence some of their conclusions might actually depend on the type of projection used.

Expression of indirect context vectors as a linear transformation Chapter 4 introduced an alternative computation of indirect (second-order) context vectors in $\mathbb{R}^n$ as a linear transformation of direct (first-order) context vectors in $\mathbb{R}^m$ via a word matrix in $\mathbb{R}^{m \times n}$, i.e. $\mathbf{c}^2(\kappa) = \mathbf{c}^{\mathbf{W}}(\kappa) = \mathbf{W}^T\mathbf{c}^1(\kappa)$. Effectively, such a word matrix maps vectors in $\mathbb{R}^m$ to $\mathbb{R}^n$. This alternative but equivalent formulation makes the relationship of indirect context vector construction more akin to a method of dimensionality reduction (that is if $n < m$). It makes the application of SVD to Word Space more transparent and natural: $\mathbf{c}^{\mathrm{R}_1(\mathbf{W})}(\kappa) = \mathbf{\Sigma}_\mathbf{k}\mathbf{U}_\mathbf{k}{}^T\mathbf{c}^1(\kappa)$ and $\mathbf{c}^{\mathrm{R}_2(\mathbf{W})}(\kappa) = \mathbf{U}_\mathbf{k}{}^T\mathbf{c}^1(\kappa)$. Finally, in all of these equations, the matrix $\mathbf{W}$ can be substituted by either a matrix of direct (first-order) context vectors $\mathbf{C}$, LSA's word-segment matrix $\mathbf{A}$ or any other suitable matrix. This is effectively a simple, transparent and general way of producing indirect context vectors. A summary of the context vector configurations explored in this thesis is given in Table 4.2.1 (p. 111).

The Word Space word matrix and the LSA word-segment matrix are approximations of each other Despite the inherent differences between the unreduced Word Space word matrix and the unreduced LSA word-segment matrix (i.e. the matrix in Salton's original vector space model), in their respective reduced forms both matrices are very similar. This is an indirect consequence of a similarity between unreduced LSA segment vectors and unreduced Word Space direct context vectors, i.e. $\mathbf{d_j} = \mathbf{c}^1(\kappa) + \mathbf{g}(\kappa)$ (Def. 3.5.2, p. 124). The consequence of this is that $\mathbf{A} \approx \mathbf{C}$, i.e. the matrices of segment vectors and direct context vectors are approximate to each other. So, the unreduced and the SVD-reduced token vectors produced by either matrix should be similar: D-A-* $\approx$ D-C-* and I-A-* $\approx$ I-C-*.

But the relationship given by 3.5.2 translates into the more complicated relationship between the unreduced Word Space word matrix **W** and the unreduced LSA word-segment matrix **A**: $\mathbf{A}\mathbf{A}^T = \mathbf{W} + \mathbf{F}$ (Conj. 4.3.2, p. 125). But if **W** is symmetric, then the SVD of **A** is similar to the SVD of **W**, a consequence of Theorem 4.3.1 (p. 131). This implies that the SVD-reduced versions of indirect token vectors derived from **A**, **C** and **W** should also be similar: I-A-R1/2 $\approx$ I-C-R1/2 $\approx$ I-W-R1/2.

The performance of all of these token vector representations were tested in WSX experiments. These experiments showed that whilst the difference between both models boils down to a very small difference in counting between the two models, it can cause the models to either keep their difference in performance negligible (I-A-* vs. I-C-*) or diverge to a point in which the difference offsets performance considerably (all other combinations), depending on how the LSA or Word Space objects are actually employed.

Given the common ways in which LSA and Word Space are actually employed (namely, D-A-*, D-C-* and I-W-*), this offset in performance of means-based word-sense disambiguation and discrimination experiments will depend on the actual token representation configuration chosen. Finally, it was found that whilst SVD can help when LSA direct context vectors are used, the overall best results were obtained with the standard application of Word Space (D-C-UR "first-order" and I-W-UR "second order" vectors) without using SVD.

Systematic comparisons of Word Space direct and indirect token representations
The geometric properties of parallelism and angular spread were compared for sense vectors derived from direct and indirect Word Space context vectors.

By studying the parallelism property of actual direct and indirect sense vectors using the same features, it was found that direct and indirect sense vectors are not approximations of each other. The implication of this is that it is no possible to predict the performance of one vector type based on the performance of the other in actual WSX experiments.

The angular spread property showed that direct sense vectors are more spread than indirect sense vectors, implying that direct context vectors will potentially perform better than indirect context vectors in means-based WSX experiments. On actual supervised word-sense disambiguation experiments, it was shown that direct context vectors performed considerably better than indirect context vectors, largely following the predictions from the angular spread analysis. The interpretation of this is that the better angular spread of direct context vectors outweighs the semantic benefits (such as finding similarities between related contexts that do not use the same words) that indirect context vectors bring in. However, in the unsupervised word-sense discrimination experiments, the performance between direct and indirect context vectors is quite comparable and well below the results observed in the supervised setting. The interpretation in this case is that the semantic benefits provided by indirect context vectors are perhaps playing a part in matching and sometimes beating the performance of direct vectors. The fact that unsupervised scores fall well below supervised scores is not a surprise and is a trend well documented in the literature.

Word matrix consolidation as a viable dimensionality reduction technique A method for reducing the dimensionality of context vectors is proposed. This method is based on a consolidated word matrix in which the dimensions representing statistically associated word types are merged together. An algorithm that merges less important dimensions into more important dimensions is proposed and several definitions of "dimension importance" are explored. The proposed method is comparable in WSX performance to SVD.

Semantic similarity between word vectors can be used as a proxy to grade the compositionality of multi-word expressions Word vectors from Word Space are tested empirically in a task designed to grade (measure) the compositionality (or degree of "literalness") of multi-word expressions (MWEs). Cosine similarity measures are taken between a word vector representing the full MWE, and word vectors representing each of its individual member words in order to measure the deviation in co-occurrence distribution between the MWE and its individual members. It was found that this deviation in co-occurrence distributions does correlate with human compositionality judgements of MWEs. The experiments were performed on the 2011 DISCO Shared Task. Some issues with the way the gold standard human judgements were averaged were highlighted in this thesis. Namely, that these averages do not take into account the proportions in which compositional and non-compositional usages of the shared task MWEs are used in the corpus, producing a somewhat flawed gold standard. An attempt to work around this issue was provided in the form of Configuration 3, a variation of our method that clusters occurrences of MWEs and attempts to come up with a weighted averaged compositionality score. Unfortunately, this attempt was largely unsuccessful and the simpler Configurations 1 and 2 performed better.

9.2 Future work

Model comparisons in other tasks This thesis concentrated mostly on word-sense disambiguation and discrimination tasks with some limited attention to the automatic grading of compositionality in multi-word expressions. Whilst these are all important tasks, there other areas of lexical semantics where the vector space models studied in this thesis can be applied and their performance compared. This will enable to draw more general, task-independent conclusions for these models. Examples of tasks to consider are word clustering, TOEFL synonym tests, word association prediction or semantic priming, terminology extraction and terminology consistency checking, text classification and clustering, etc.

Geometric comparisons between LSA and Word Space Geometric experiments (parallelism and angular spread, Chapter 5) that directly compared sense vectors computed from direct or indirect context vectors were conducted for Word Space. These same experiments can be computed in the across the SVD-reduced Word Space and LSA token representations (both direct and indirect).

Deeper performance analysis between the R_1 and R_2 projections Section 3.4.2 (p. 81) argued that R_1 projections should be more appropriate than R_2 projections for the type of comparisons that we wanted to perform in WSX experiments in LSA and Word Space. Empirically, it was found that for certain configurations, such as plain vanilla LSA (D-A-R1/2), R_1 projections performed better than R_2 projections in unsupervised word-sense discrimination experiments, but on supervised word-sense disambiguation experiment the exact opposite was the case (R_2 projections performed better than R_1 projections). In Section 6.3.1 (p. 149), it was hypothesised that perhaps R_2 projections have a noise reduction effect in the supervised case that aids in classification, but perhaps hinders the generalisations needed to successfully cluster in an unsupervised manner. More analysis is warranted in this area. For example, additional angular spread experiments (like the ones already proposed), as well as experimentation in other natural language processing tasks could shed some light to this question.

Performance of direct vs. indirect Word Space token spaces based on corpus size In Section 5.5 (p. 143) a comparison was made between the direct vs. indirect WSX benchmarks performed by Purandare and Pedersen (2004) and Maldonado-Guerra and Emms (2012) (contained in Chapter 5) where it was argued that Purandare and Pedersen did not perform their benchmark based on minimal feature pairs and so their conclusions regarding the effectiveness of direct and indirect vectors as a function of corpus size (namely that direct vectors perform better in large corpora whilst indirect vectors perform better in small corpora) could instead be due to other factors (such as the actual features they used). It would be interesting to repeat the experiments reported in Chapter 5 by varying the size of the corpora used. One way could be by taking random samples of different sizes of the HILS and NYT datasets. In addition, so far these experiments have focused on news texts. For future work, experiments will also be run in corpora from several other domains.

Study alternative vector space models There are other popular vector space models that were not studied in this thesis. For one, any vector space model that uses syntactic or deeper linguistic features were not considered. But there are other "cousins" of Word Space and LSA that deserve to be compared against in some of the frameworks presented in this thesis. These include the Word Space version introduced in Schütze and Pedersen (1993) that considers both syntagmatic and paradigmatic relations in a single word matrix, as well as the closely related Hyperspace Analogue to Language (HAL) (Burgess and Lund, 1997, 2000), as well as newer tensor-based representations (Turney and Pantel, 2010). It would be interesting, for example, to use Schütze and Pedersen's model to grade the compositionality of multi-word expressions, and whether their sensitivity to syntagmatic and paradigmatic relations helps in the task. Also, it will be interesting to substitute SVD with other methods of dimensionality reduction such as Random Indexing (Moen et al., 2013) and Non-Negative Matrix Factorisation (Van De Cruys, 2008) in LSA and Word Space.

More fine-grained context definitions The introduction in Section 4.2.1 (p. 111) listed three possible options for creating matrices of direct context vectors. In experimentation, only one such option was explored, the one that was deemed more similar to LSA. However, it is still interesting to investigate whether the other options, especially option 1 which provides more redundant co-occurrence information, are able to produce better- or worse-performing SVD spaces. For example, perhaps SVD is able to generalise better when it is given that much information. Also, each token represented in option 1 is seen as a discrete, separate feature from each other. But in reality, they could be grouped, somehow, into types but without turning them into word vectors. It would be interesting if that information could be exploited by SVD or other method of dimensionality reduction.

Hybrid direct and indirect vector clustering in a single experiment Indirect (second-order) context vectors have not consistently produced significantly better results than direct context vectors. The angular spread experiments in Chapter 5 showed that the differences between different sense vectors constructed from indirect context vectors were quite small, presumably making it difficult to tell different senses apart. At the same time, the generalisation performed by its second-order coverage does seem to help somewhat in clustering, even if that same second-order characteristic is perhaps the cause for the low difference in angular spread in sense vectors. A proposal would be to use some sort of hybrid clustering approach based on K-Means in which second-order context vectors are used as the initial randomly selected seeds, but similarity computation is performed using first-order context vectors. Also, subsequent cluster centroids (sense vectors) are computed from first-order context vectors. Means-based sense vectors generalise senses. So, in a way they themselves are second-order vectors, hence the idea of using second-order context vectors as the initial randomly-selected centroids. But as the algorithm progresses, perhaps it is better to concentrate on the first-order context vectors. First-order context vectors are never compared between each other, they are only compared against the "second-order" sense vectors. So, their perceived weaknesses of sparsity or lack of semantic sensitivity to semantically related contexts that use different words is perhaps not completely warranted.

A related experiment to try out is to use clustering algorithms based on k-neighbours techniques. Since they involve pairwise comparisons between individual context vectors, it is likely that second-order context vectors fare better than first-order context vectors in those clustering techniques that do rely on pairwise comparisons.

Linear transformations as methods of vector composition The word vectors produced by LSA and Word Space represent singleton words. There is a growing interest in methods that allow the combination of such individual word vectors to represent larger textual units such as phrases and perhaps even full sentences. These methods are known as compositional distributional semantics models (Mitchell and Lapata, 2008). The two most basic methods of vector composition are vector addition, i.e. summing two word vectors $\mathbf{u} + \mathbf{v}$ in a manner not unlike that used to compute indirect (second-order) context vectors, and pairwise vector multiplication, i.e. $\mathbf{u} \odot \mathbf{v} = [u_1 v_1, \ldots, u_n v_n]$. Variations on these two basic methods exist, some of which are documented by Guevara (2011). These methods could be easily expressed in terms of the linear transformations presented in Chapter 4. For example, the pairwise vector multiplication can be more conventionally expressed by the matrix multiplication $\mathbf{u} \times \text{diag}(\mathbf{v})$ where $\text{diag}(\mathbf{v})$ is a function returning a diagonal matrix with the elements of vector $\mathbf{v}$ along its diagonal and zeroes everywhere else. Some of the other variations of vector composition could find expression as a linear transformation, too. It is hoped that this alternative expression reveals additional useful properties, just like it did for Word Space in this thesis. Besides, there are works that are starting to find a connection between models of vector composition and methods of dimensionality reduction, such as Non-Negative Matrix Factorisation (Van de Cruys et al., 2013).

An analysis of relation types captured by SVD spaces in LSA and Word Space We reviewed some of the theory regarding syntagmatic and paradigmatic relations and how these are captured by the unreduced versions of $\mathbf{A}$ and $\mathbf{W}$, respectively. However, the degree to which each relation is represented by the SVD-reduced spaces in these models (especially in Word Space) remains largely unexplored. This could be researched by performing experiments in tasks that depend more on one type of relation than the other, like some of the experiments performed by Sahlgren (2006) and Utsumi (2010).

Further alternative methods of dimensionality reduction based on a word matrix Chapter 7 offered a way of dimensionality reduction based on the word matrix. Rather than using the obvious method of selecting n word dimensions as column features, it attempted to merge words together that were syntagmatically associated. Whilst the results were promising, this particular method of reducing dimensions is not necessarily the best. It could perhaps be more promising to merge paradigmatically related dimensions, i.e. by clustering them with each cluster representing a dimension in the new reduced space. Each word would then be mapped to one or two dimensions, i.e. the dimension(s) of the cluster whose word vector belong to. Different clustering techniques could be used, from graph methods, soft clustering techniques like Expectation-Maximisation to more traditional hard-clustering methods like K-Means. And the degree of association of a word vector to a cluster could be used as the dimension's weight.

Bibliography

Agirre, E. and Edmonds, P., editors (2005). *Word Sense Disambiguation: Algorithms and Applications*. Springer.

Agirre, E. and Edmonds, P. (2007). *Word Sense Disambiguation: Algorithms and Applications*. Springer.

Agirre, E. and Stevenson, M. (2007). Knowledge Sources for WSD. In Agirre, E. and Edmonds, P., editors, *Word Sense Disambiguation: Algorithms and Applications*, pages 217–251. Springer.

Anton, H. and Rorres, C. (2000). *Elementary Linear Algebra: applications version*. John Wiley & Sons, 8th edition.

Austin, J. L. (1962). *How to do things with words*. Clarendon Press, Oxford.

Baldinger, K. (1980). *Semantic Theory*. Blackwell, Oxford.

Baldwin, T., Bannard, C., Tanaka, T., and Widdows, D. (2003). An Empirical Model of Multiword Expression Decomposability. In *Proceedingsof the ACL-2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 89–96, Sapporo.

Banerjee, S. and Pedersen, T. (2003). The Design, Implementation, and Use of the Ngram Statistics Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, Mexico City.

Bannard, C., Baldwin, T., and Lascarides, A. (2003). A Statistical Approach to the Semantics of Verb-Particles. In *Proceedings of the ACL-SIGLEX Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, number 1, pages 65–72, Sapporo.

Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Bartell, B. T., Cottrell, G. W., and K., B. R. (1992). Latent Semantic Indexing is an Optimal Scaling Special Case of Multidimensional Scaling. In *Proceedings of the 15th annual international ACM SIGIR conference on research and development in information retrieval*, pages 161–167, Copenhagen.

Benson, M. (1989). The Structure of the Collocational Dictionary. *International Journal of Lexicography*, 2(1):1–14.

Berry, M. W. (1992). Large sparse singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49.

Berry, M. W., Dumais, S. T., and O'Brien, G. W. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, 37(4):573–595.

Biemann, C. and Giesbrecht, E. (2011). Distributional Semantics and Compositionality 2011: Shared Task Description and Results. In *Proceedings of the Distributional Semantics and Compositionality workshop (DISCo 2011) in conjunction with ACL 2011*, Portland, OR.

Boleda, G., Padó, S., and Utt, J. (2012). Regular polysemy: A distributional model. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 151–160, Montreal.

Bond, F. (2005). *Translating the Untranslatable: A Solution to the Problem of Generating English Determiners*. CSLI Publications, Stanford, CA.

Bruce, R. and Wiebe, J. (1994). Word-Sense Disambiguation using Decomposable Models. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 139–146, Las Cruces, NM.

Bu, F. and Zhu, X. (2010). Measuring the Non-compositionality of Multiword Expressions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, number August, pages 116–124, Beijing.

Budanitsky, A. and Hirst, G. (2001). Semantic distance in WordNet : An experimental , application-oriented evaluation of five measures. In *Proceedings of the Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, Pittsburgh, PA.

Buitelaar, P., Alexandersson, J., Jaeger, T., Lesch, S., Pfleger, N., Raileanu, D., Von der Berg, T., Klöckner, K., Neis, H., and Schlarb, H. (2001). An Unsupervised Semantic Tagger Applied to German. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, pages 52–57, Tzigov Chark.

Buitelaar, P., Magnini, B., Strapparava, C., and Vossen, P. (2007). Domain-Specific WSD. In *Word Sense Disambiguation: Algorithms and Applications*, pages 275–298. Springer.

Bullinaria, J. A. and Levy, J. P. (2007). Extracting semantic representations from word co-occurrence statistics: a computational study. *Behaviour research methods*, 39(3):510–26.

Burgess, C. and Lund, K. (1997). Modelling Parsing Constraints with High-dimensional Context Space. *Language and Cognitive Processes*, 12(2):177–210.

Burgess, C. and Lund, K. (2000). The Dynamics of Meaning in Memory. In Dietrich, E. and Markman, A., editors, *Cognitive Dynamics: Conceptual Representational Change in Humans and Machines*, pages 117–156. Lawrence Erlbaum.

Carpuat, M. and Wu, D. (2007). Improving statistical machine translation using word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, number June, pages 61–72, Prague.

Carreras, X. and Màrquez, L. (2004). Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In *Proceedings of the Eighth Conference on Computational Natural Language learning*, pages 89–97, Boston, MA.

Charoenporn, T., Kruengkrai, C., Theeramunkong, T., and Sornlertlamvanich, V. (2007). An EM-Based Approach for Mining Word Senses from Corpora. *IEICE TRANSACTIONS on Information and Systems*, 90(4):775–782.

Chomsky, N. (1972). *Language and mind*. Harcourt Brace Jovanovich, New York, NY.

Church, K., Gale, W., Hanks, P., and Hindle, D. (1991). Using statistics in lexical analysis. In Zernik, U., editor, *Lexical acquisition: exploiting on-line resources to build a lexicon*, pages 115–164. Lawrence Erlbaum, Hillsdale, NJ.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Cohen, T. and Widdows, D. (2009). Empirical distributional semantics: methods and biomedical applications. *Journal of Biomedical Informatics*, 42(2):390–405.

Cowie, A. P. (1988). Stable and creative aspects of vocabulary use. In Carter, R. and McCarthy, M., editors, *Vocabulary and language teaching*. Longman, London.

Davidson, D. (1967). Truth and Meaning. *Synthese*, 17(3):304–323.

de Marneffe, M.-C., Archambeau, C., Dupont, P., and Verleysen, M. (2005). Local Vector-based Models for Sense Discrimination. In *Proceedings of the 6th International Workshop on Computational Semantics*.

de Marneffe, M.-C. and Dupont, P. (2004). Comparative study of statistical word sense discrimination techniques. In *Actes des 7es Journées internationales d'Analyse statistique des Données Textuelles (JADT 2004)*.

de Saussure, F. (1916). *Cours de linguistique générale. Édition critique préparée par Tullio de Mauro (1967, 1995)*. Payot, Paris.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Dinu, G., Thater, S., and Laue, S. (2012). A comparison of models of word meaning in context. In *2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 611–615, Montréal.

Dumais, S. T., Letsche, T. A., Littman, M. L., and Landauer, T. K. (1997). Automatic Cross-Language Retrieval Using Latent Semantic Indexing. In *Proceedings of the AAAI Spring Symposium on Cross-Language Text and Speech Retrieval*, pages 115–132, Stanford, CA.

Dunning, T. (1993). Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19(1):61–74.

Emms, M. and Maldonado-Guerra, A. (2013). Latent Ambiguity in Latent Semantic Analysis? In *Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods*, pages 115–120, Barcelona.

Erk, K. (2007). A Simple, Similarity-based Model for Selectional Preferences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 216–223, Prague.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Fillmore, C. J. (1982). Frame semantics. In The Linguistic Society of Korea, editor, *Linguistics in the Morning Calm: Selected Papers from SICOL-1981*, pages 111–137. Hanshin, Seoul.

Firth, J. R. (1951). Modes of meaning. In *Papers in Linguistics 1934-1951*, pages 190–215. Oxford University Press, Oxford.

Firth, J. R. (1957). A synopsis of linguistic theory 1930-1955. In *Studies in Linguistic Analysis, pp. 1-32. Oxford: Philological Society. Reprinted in F.R. Palmer (ed.), Selected Papers of J.R. Firth 1952-1959*, pages 1–32. Longman, London.

Foltz, P. W., Laham, D., and Landauer, T. K. (1999). The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 1(2).

Fowler, H. N. (1921). *Plato in Twelve Volumes, Vol. 12*. Hardvard University Press, Cambridge, MA.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., and Lochbaum, K. E. (1988). Information Retrieval using a Singular Value Decomposition Model of Latent Semantic Structure. In *Proceedings of the 11th Anual International*

ACM Conference on Research and Development in Information Retrieval (SIGIR), pages 465–480, Grenoble.

Gaustad, T. (2001). Statistical corpus-based word sense disambiguation: Pseudowords vs. real ambiguous words. In *Companion Volume to the Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL/EACL 2001) - Proceedings of the Student Research Workshop*, pages 61–66, Toulouse.

Geeraerts, D. (2010). *Theories of lexical semantics*. Oxford University Press, Oxford.

Goddard, C. and Wierzbicka, A., editors (2002). *Meaning and Universal Grammar: Theory and Empirical Findings*. Benjamins, Amsterdam.

Golub, G. H. and Van Loan, C. F. (1989). *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, 2nd edition.

Gong, Y. and Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pages 19–25, New Orleans, LA.

Guevara, E. (2011). Computing Semantic Compositionality in Distributional Semantics. In Bos, J. and Pulman, S., editors, *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, pages 135–144, Oxford. Association for Computational Linguistics.

Guthrie, J. A., Guthrie, L., Wilks, Y., and Aidinejad, H. (1991). Subject-dependent co-occurrence and word sense disambiguation. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 146–152, Berkeley, CA.

Halliday, M. A. K. (1961). Categories of the theory of grammar. *Word*, 17(3):241–292.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3):146–162.

Hovy, E., Navigli, R., and Ponzetto, S. P. (2013). Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

Ide, N. and Véronis, J. (1998). Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1):1–40.

Katz, G. and Giesbrecht, E. (2006). Automatic Identification of Non-Compositional Multi-Word Expressions using Latent Semantic Analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19, Sydney.

Katz, J. J. and Fodor, J. A. (1963). The Structure of a Semantic Theory. *Language*, 39(2):170–210.

Kilgarriff, A. (1997). I don't believe in word senses. *Computers and the Humanities*, (31):91–113.

Kontostathis, A. and Pottenger, W. M. (2006). A Framework for Understanding Latent Semantic Indexing (LSI) Performance. *Information Processing and Management*, 42(1):56–73.

Landauer, T. K. (2007). LSA as a Theory of Meaning. In Landauer, T. K., McNamara, D. S., Dennis, S., and Kintsch, W., editors, *Handbook of Latent Semantic Analysis*, pages 3–34. Routledge, New York, NY.

Landauer, T. K. and Dumais, S. T. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104(2):211–240.

Lawson, C. L. and Hanson, R. J. (1974). *Solving least squares problems*. Prentice-Hall, Englewood Cliffs, N.J.

Leacock, C., Chodorow, M., and Miller, G. A. (1998). Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics*, 24(1):147–165.

Leacock, C., Towell, G., and Voorhees, E. (1993). Corpus-Based Statistical Sense Resolution. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 260–265, San Francisco, CA.

Lehrer, A. (1974). *Semantic Fields and Lexical Structure*. North-Holland, Amsterdam.

Lesk, M. (1986). Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of the 1986 ACM SIGDOC Conference*, pages 24–26, Toronto.

Levin, B. (1993). *English Verb Classes and Alterations*. University of Chicago Press, Chicago, IL.

Levin, E., Sharifi, M., and Ball, J. (2006). Evaluation of Utility of LSA for Word Sense Discrimination. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, Companion Volume: Short Papers*, number 1998, pages 77–80, New York, NY.

Lin, D. (1999). Automatic identification of non-compositional phrases. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 317–324, Morristown, NJ, USA. Association for Computational Linguistics.

Lin, D. and Pantel, P. (2001). DIRT – Discovery of Inference Rules from Text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 323–328, San Francisco, CA.

Lyons, J. (1963). *Structural Semantics*. Blackwell, Oxford.

Lyons, J. (1966). Firth's theory of meaning. In Bazell, C., Catford, J., Halliday, M., and Robins, R., editors, *In Memory of J. R. Firth*, pages 288–302. Longman, London.

Lyons, J. (1977). *Semantics*. Cambridge University Press, Cambridge.

Lyons, J. (1981). *Language and linguistics*. Cambridge University Press, Cambridge.

Maldonado-Guerra, A. and Emms, M. (2011). Measuring the compositionality of collocations via word co-occurrence vectors: Shared task system description. In *Proceedings of the Distributional Semantics and Compositionality workshop (DISCo 2011)*, Portland, OR.

Maldonado-Guerra, A. and Emms, M. (2012). First-order and second-order context representations: geometrical considerations and performance in word-sense disambiguation and discrimination. In *Actes des 11es Journées internationales d'Analyse statistique des Données Textuelles (JADT 2012)*, pages 675–686, Liège.

Malinowski, B. (1935). *Coral Gardens and their Magic: A study of the Methods of Tilling the Soil and of Agricultural Rites in the Trobriand Islands. Volume II: The Language of Magic and Gardening*. Allen & Unwin, London.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Martin, D. I. and Berry, M. W. (2007). Mathematical foundations behind latent semantic analysis. In Landauer, T. K., McNamara, D. S., Dennis, S., and Kintsch, W., editors, *Handbook of Latent Semantic Analysis*, pages 35–55. Routledge, New York, NY.

Martinez, D. and Baldwin, T. (2011). Word sense disambiguation for event trigger word detection in biomedicine. *BMC bioinformatics*, 12((Suppl 2):S4).

McCarthy, D., Keller, B., and Carroll, J. (2003). Detecting a continuum of compositionality in phrasal verbs. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment, Volume 18*, pages 73–80, Sapporo. Association for Computational Linguistics.

McCarthy, D., Koeling, R., Weeds, J., and Carroll, J. (2004). Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 577–583, Barcelona. Association for Computational Linguistics.

McIntosh, A. (1966). *Patterns and ranges: Papers in general, descriptive and applied linguistics*. Longman, London.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Miller, G. A. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, number June, pages 236–244, Columbus, Ohio. Citeseer.

Moen, H., Marsi, E., and Gambäck, B. (2013). Towards Dynamic Word Sense Discrimination with Random Indexing. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, number 2009, pages 83–90, Sofia.

Montague, R. (1974). *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven, CT.

Moro, A., Raganato, A., and Navigli, R. (2014). Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38.

Navigli, R. (2009). Word sense disambiguation. *ACM Computing Surveys*, 41(2):1–69.

Navigli, R. and Crisafulli, G. (2010). Inducing Word Senses to Improve Web Search Result Clustering. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 116–126, Cambridge, MA.

Navigli, R. and Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Ng, H. T. and Beng Lee, H. (1996). Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 40–47, Santa Cruz, Ca.

Och, F. J. (2002). *Statistical Machine Translation : From Single-Word Models to Alignment Templates*. PhD thesis, Rheinisch-Westfälischen Technischen Hochschule Aachen.

Ogden, C. K. and Richards, I. A. (1936). *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Routledge and Kegan Paul, London, fourth edition.

Oh, J.-H. and Choi, K.-S. (2002). Word sense disambiguation using static and dynamic sense vectors. In *Proceedings of the 19th international conference on Computational linguistics*, volume 1, Taipei.

Palmer, M., Ng, H. T., and Dang, H. T. (2007). Evaluation of WSD Systems. In Agirre, E. and Edmonds, P., editors, *Word Sense Disambiguation: Algorithms and Applications*, pages 75–106. Springer.

Papadimitriou, C. H., Tamaki, H., Raghavan, P., and Vempala, S. (1998). Latent Semantic Indexing: A Probabilistic Analysis. In *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems*, pages 159–168, Seattle, WA.

Paul, H. (1920). *Prinzipien der Sprachgeschichte*. Halle, 5th edition.

Pecina, P. (2005). An extensive empirical study of collocation extraction methods. In *Proceedings of the ACL Student Research Workshop*, number June, pages 13–18, Ann Arbour, Michigan. Association for Computational Linguistics.

Pedersen, T. (2007). Unsupervised Corpus-Based Methods for WSD. In Agirre, E. and Edmonds, P., editors, *Word Sense Disambiguation: Algorithms and Applications*, chapter 6, pages 133–166. Springer.

Pedersen, T. (2010). The effect of different context representations on word sense discrimination in biomedical texts. In *Proceedings of the First ACM International Health Informatics Symposium*, pages 56–65, Arlington, VA. ACM Press.

Pedersen, T. and Bruce, R. (1997). Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, volume 2, pages 197–207, Providence, RI.

Peirsman, Y., Heylen, K., and Geeraerts, D. (2008a). Size matters: tight and loose context definitions in English word space models. In *Proceedings of the ESSLLI Workshop on Distributional Lexical Semantics*, pages 34–41, Hamburg.

Peirsman, Y., Heylen, K., and Speelman, D. (2008b). Putting things in order. First and second order context models for the calculation of semantic similarity. In *Actes des 9es Journées internationales d'Analyse statistique des Données Textuelles (JADT 2008)*, pages 907–916, Lyon.

Pino, J. and Eskenazi, M. (2009). An Application of Latent Semantic Analysis to Word Sense Discrimination for Words with Related and Unrelated Meanings. In *Proceedings of the NAACL HLT Workshop on Innovative Use of NLP for Building Educational Applications*, number June, pages 43–46, Boulder, CO.

Purandare, A. (2004). *Unsupervised Word Sense Discrimination by Clustering Similar Contexts*. Msc thesis, University of Minnesota.

Purandare, A. and Pedersen, T. (2004). Word sense discrimination by clustering contexts in vector and similarity spaces. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 41–48, Boston, MA.

Rapp, R. (2003). Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the Ninth Machine Translation Summit*, pages 315–322, New Orleans, LA.

Resnik, P. (2007). WSD in NLP Applications. In *Word Sense Disambiguation: Algorithms and Applications*, pages 299–337. Springer.

Rosario, B. (2000). Latent Semantic Indexing: An overview. Technical report, University of California at Berkeley.

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. (2002). Multiword Expressions: A Pain in the Neck for NLP. *Third International Conference on Computational Linguistics and Intelligent Text Processing (Lecture Notes in computer Science)*, 2276:1–15.

Sagi, E., Kaufmann, S., and Clark, B. (2011). Tracing semantic change with Latent Semantic Analysis. In Allan, K. and Robinson, J. A., editors, *Current Methods in Historical Semantics*, pages 161–183. Mouton de Gruyter, Berlin.

Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words*. Ph.d. thesis, Stockholm University.

Salton, G. (1971). *The SMART Retrieval System–Experiments in Automatic Document Processing*. Prentice-Hall, Upper Saddle River, NJ.

Salton, G., Wong, A., and Yang, C.-S. (1975). A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620.

Schone, P. and Jurafsky, D. (2001). Is knowledge-free induction of multiword unit dictionary headwords a solved problem. In *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, pages 100–108, Pittsburgh, PA.

Schütze, H. (1992). Dimensions of meaning. *Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, pages 787–796.

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Schütze, H. and Pedersen, J. (1993). A Vector Model for Syntagmatic and Paradigmatic Relatedness. In *Making Sense of Words: Proceedings of the Conference*, pages 104–113, Oxford.

Sebastiani, F. (2001). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47.

Sinclair, J. (1966). Beginning the study of lexis. In Bazell, C., Catford, J., Halliday, M., and Robins, R., editors, *In Memory of J. R. Firth*, pages 410–431. Longman, London.

Sinclair, J. (1991). *Corpus, Concordance, Collocation*. Oxford University Press, Oxford.

Sinclair, J. (2004). *Trust the Text: Language, Corpus and Discourse*. Routledge, London.

Singleton, D. (2000). *Language and the Lexicon: An Introduction*. Arnold, London.

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Stubbs, M. (2002). *Words and Phrases: Corpus Studies of Lexical Semantics*. Blackwell, Oxford.

Sugiyama, K. and Okumura, M. (2009). Semi-supervised Clustering for Word Instances and Its Effect on Word Sense Disambiguation. In Gelbukh, A., editor, *Proceedings of the 10th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2009)*, pages 266–279, Mexico City. Springer.

Trier, J. (1934). Das sprachliche Feld. Eine Auseinandersetzung. *Neve Jahrbücher für Wissenschaft und Jugendbildung*, (10):428–49.

Turney, P. D. (2001). Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*, pages 491–502, Freiburg.

Turney, P. D. (2006). Similarity of Semantic Relations. *Computational Linguistics*, 32(3):379–416.

Turney, P. D. and Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Utsumi, A. (2010). Exploring the Relationship between Semantic Spaces and Semantic Relations. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*, pages 257–262, Valletta.

Utsumi, A. and Suzuki, D. (2006). Word Vectors and Two Kinds of Similarity. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL 2006) Main Conference Poster Sessions*, pages 858–865, Sydney.

Van De Cruys, T. (2008). Using Three Way Data for Word Sense Discrimination. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 929–936, Manchester.

Van de Cruys, T., Poibeau, T., and Korhonen, A. (2013). A Tensor-based Factorization Model of Semantic Compositionality. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1142–1151, Atlanta, GA.

Van Gompel, M., Hendrickx, I., Van den Bosch, A., Lefever, E., and Hoste, V. (2014). SemEval-2014 Task 5 : L2 Writing Assistant. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 36–44, Dublin.

Véronis, J. (2004). HyperLex: lexical cartography for information retrieval. *Computer Speech and Language*, 18(3):223–252.

Wang, T. and Hirst, G. (2010). Near-synonym Lexical Choice in Latent Semantic Space. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, number August, pages 1182–1190, Beijing.

Weaver, W. (1955). Translation. In Locke, W. N. and Booth, A. D., editors, *Machine translation of languages: fourteen essays*, pages 15–23. MIT Press, Cambridge, MA.

Weisgerber, L. (1954). Die Sprachfelder in der geistigen Erschließung der Welt. pages 34–49. Westkulturverlag Anton Hain, Meisenheim/Glan.

Widdows, D. (2004). *Geometry and Meaning*. CSLI Publications, Stanford, CA.

Wittgenstein, L. (1968). *Philosophical Investigations (Philosophische Untersuchungen)*. Blackwell, Oxford, 3rd edition.

Wolfe, M. B. W., Schreiner, M. E., Rehder, B., Laham, D., Foltz, P. W., Kintsch, W., and Landauer, T. K. (1998). Learning from text: Matching readers and texts by Latent Semantic Analysis. *Discourse Processes*, 25:309–336.

Yarowsky, D. (1992). Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 454–460, Nantes. Association for Computational Linguistics.

Yarowsky, D. (1993). One sense per collocation. In *Proceedings of the workshop on Human Language Technology*, pages 266–271, Plainsboro, NJ. Morgan Kaufmann Publishers.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivalling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.

Zelikovitz, S. and Hirsh, H. (2001). Using LSI for text classification in the presence of background text. In *Proceedings of the tenth international conference on Information and knowledge management (CIKM'01)*, pages 113–118, New York, NY. ACM Press.

Index