Active Learning Query Selection with Historical Information

Michael Davy

A thesis submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy

May 2009

Declaration

Michael Davy

Dated: May 18, 2009

Acknowledgements

First and foremost, I would like to thank my supervisor Saturnino Luz for his belief, support and guidance throughout my research. Without his advice and insight this thesis would not have been possible. I would like to thank all my friends, who have made the last few years a memorable and enjoyable experience. I would especially like to thank Derek Greene, Peter Byrne, Kenneth Bryan and Aisling Linehan for their help and advice in proof-reading this thesis. I would like to take this opportunity to thank Pádraig Cunningham for introducing me to the topic of machine learning and guiding me in the early stages of my research. Finally, I would like to thank my parents and family. I am exceptionally fortunate to have such a supporting and loving family and I dedicate this thesis to them.

Michael Davy

University of Dublin, Trinity College

May 2009

Abstract

This work describes novel methods and techniques to decrease the cost of employing active learning in text categorisation problems. The cost of performing active learning is a combination of labelling effort and computational overhead. Reducing the cost of active learning allows for accurate classifiers to be constructed inexpensively, increasing the number of real-world problems where machine learning solutions can be successfully applied. In this thesis we investigate strategies and techniques to reduce both computational expense and labelling effort in active learning.

Critical to the success of active learning is the query selection strategy, which is responsible for identifying informative unlabelled examples. Selecting only the most informative examples will reduce labelling effort as redundant and uninformative examples are ignored. The majority of query selection strategies select queries based on the labelling predictions of the current classifier. This thesis suggests that information from prior iterations of active learning can help select more informative queries in the current iteration. We propose *History-based* query selection strategies, which incorporate predictions from prior iterations of active learning into the selection of the current query. These strategies have been shown to increase the accuracy of classifiers produced using active learning, thereby reducing labelling effort. In addition, History-based query selection strategies are very efficient since information is reused from previous iterations of active learning.

Another contributing factor to the cost of active learning is computational expense. Query selection strategies can require considerable computation to identify the most informative examples. We investigate pre-filtering optimisation for the computationally inefficient error reduction sampling (ERS) query selection strategy. Pre-filtering restricts the number of unlabelled examples considered to a small subset of the pool, constructed using query selection strategy. Optimising ERS using pre-filtering was found to simultaneously reduce computational overhead and the labelling effort.

Contents

List of Figures

List of Tables

Associated Publications

Davy, M. & Luz, S. (2007a). Active learning with History-based query selection for text categorisation. Proceedings of the 29th European Conference on Information Retrieval Research, ECIR 2007 , 4425, 695.

Davy, M. & Luz, S. (2007b). Dimensionality reduction for active learning with nearest neighbour classifier in text categorisation problems. Proceedings of the 6th International Conference on Machine Learning and Applications (ICMLA07).

Davy, M. & Luz, S. (2008). An adaptive pre-filtering technique for error-reduction sampling in active learning. In Proceedings of the Eighth IEEE International Conference on Data Mining Workshops , IEEE Computer Society Washington, DC, USA.

Chapter 1

Introduction

This thesis investigates methods and techniques to lower the cost of applying machine learning solutions using active learning. The cost of constructing an accurate classifier in an active learning setting is a combination of the number of labels required and the computational expense of performing active learning:

$$\text{Cost} = \text{Labelling effort} + \text{Computational overhead}$$

In this thesis we investigate two fundamental ways to reduce cost of active learning. First, we investigate ways to reduce the labelling effort. The critical component of active learning is the query selection strategy, which is responsible for identifying informative unlabelled examples. Selecting only the most informative examples will reduce labelling effort as redundant or uninformative examples are ignored. We investigate ways to select more informative queries by incorporating information from prior iterations into the query selection process. Secondly, we investigate ways to reduce the computational overhead. Many of the most successful query selection strategies require considerable computational overhead to select informative unlabelled examples. Practical solutions to reduce the computational expense in applying active learning are also investigated. We examine optimisation techniques for computationally expensive query selection strategies and unsupervised dimensionality reduction techniques.

In the next section we provide some motivation for active learning and describe the research setting. Contributions made in this thesis are outlined in Section 1.2. Related work is discussed in Section 1.3. Finally, Section 1.4 describes the structure of the remainder of this thesis.

1.1 Motivation

Categorisation is an integral part of human nature. Most humans find it difficult to extract information from unstructured data. By applying structure to data, information can be retrieved easily and efficiently. Perhaps the best example of the application of structure to data is exemplified through biological taxonomy. In recent years the production and storage of textual data has increased dramatically. Efficient automatic categorisation systems are required since the scale of the data produced renders manual categorisation impractical.

A first approach (Hayes & Weinstein, 1990) was a rule-based *expert system* with hand-crafted rules constructed by a dedicated team of domain experts. This approach required considerable effort to implement; however, it produced accurate classifiers which successfully met the requirements of the problem. A serious limitation of hand-crafted rules is the *knowledge acquisition bottleneck* (Sebastiani, 2002), whereby to redeploy the system to a new domain, a new set of hand-crafted rules need to be produced, incurring considerable time and expense.

Machine learning offers a solution to the knowledge acquisition bottleneck through supervised learning. Accurate text categorisation systems can be induced from a set of training data consisting of labelled examples. The task of hand-crafting rules was replaced by the much easier task of labelling examples. Training data are constructed by labelling a large sample of unlabelled examples drawn at random from the underlying data distribution. The probably approximately correct (PAC) learning model (see Section 2.7) places a bound on the number of labelled examples required to induce an accurate classifier for a particular concept. However, in practice, training data supplied to supervised learning process typically contains many times more labelled examples than is strictly necessary in order to guarantee the production of an accurate classifier. Supervised learning lowered the cost of constructing automated categorisation systems and propelled the widespread use of machine learning solutions in many domains, allowing for considerable savings and increased productivity.

A limitation of supervised learning is the requirements it places on the availability of labelled training data. Many domains exist where there is an abundance of unlabelled data but there is a severe shortage of labelled training data due to the extremely high cost of labelling. We define this as the *label acquisition bottleneck*, where the cost of producing large amounts of labelled training data is too high and often outweighs the benefits of a machine learning solution. For example, harvesting webpages is inexpensive and many thousands can be collected in a few hours; however, acquiring labels for all those webpages would take considerable time and effort. Employing supervised learning in such domains is impractical since there are insufficient training data to ensure that an accurate classifier is produced and the cost of producing enough training data outweighs the potential benefit of the solution.

A machine learning technique known as *active learning* (Angluin, 1988; Lewis & Gale, 1994) offers a possible solution to the label acquisition bottleneck. Active learning is a technique which can significantly reduce the number of labelled examples required to induce an accurate classifier. It does so by controlling the training data and labelling only the most informative examples. The difference in the number of examples (m) required to construct an accurate classifier as specified by the PAC learning model and the number of examples (N) supplied to supervised learning in practice can be quite large, with $m \ll N$. Training data can contain many redundant and irrelevant examples which represents a considerable waste in labelling effort especially if the cost of acquiring labels is high. Active learning attempts to label only the m most informative examples. Significant reductions in the amount of labelled data required to produce an accurate classifier has been reported in the literature (Lewis & Gale, 1994; Tong & Koller, 2001).

Central to the success of active learning is the *query selection strategy* which is responsible for identifying informative examples in the pool. The primary focus of this thesis is to improve the query selection process, either by incorporating information from previous iterations to help select more informative examples or by reducing the computational overhead of applying certain query selection strategies.

Active learning is a way to significantly reduce the cost of applying a machine learning solution. Contributions made in this thesis allow for further savings to be made when producing an accurate classifier using active learning. Reduced costs of deployment could potentially expand the number of domains in which machine learning solutions can be applied.

1.2 Contributions

This thesis investigates two principled ways to reduce the cost of constructing an accurate classifier using active learning. First, we focus on the query selection strategy and develop novel methods to increase the quality of queries selected by incorporating information from prior iterations of active learning. Selecting more informative queries will lower the labelling expense and by association the cost of producing an accurate classifier. Second, active learning can be very computationally expensive. We investigate practical issues for lowering the computational overhead of performing active learning.

1.2.1 Information from Past Iterations

Active learning is an iterative process whereby in each iteration, information such as predictions on the remaining unlabelled examples are produced. This potentially valuable information is commonly discarded at the end of each iteration. We hypothesise that information from past iterations of active learning can help select more informative examples in the current iteration.

History-based Query Selection

The majority of query selection strategies in the literature base the selection of the query on the predictions from the most recent classifier. These strategies are Markovian, in that the selection of a query in the current iteration is conditional only on the current classifier. The classifier and predictions produced in each iteration of active learning are ephemeral, that is, once the query is selected, the predictions are discarded.

We hypothesise that some potentially valuable information, which could help select informative unlabelled examples, is being lost. We investigate non-Markovian query selection strategies that incorporate predictions from past iterations of active learning into the selection of queries in the current iteration.

A data structure called *History* is used to capture all predictions on the unlabelled data produced in every iteration of active learning. *History-based* query selection strategies are developed which utilise the History to select unlabelled examples based on both the current and past predictions. Empirical evaluations show that incorporating History information into the query selection process, more informative examples are selected as queries, increasing the accuracy of the classifier produced and reducing labelling effort.

Adaptive Pre-Filtering

Past information is also utilised in a proposed adaptive pre-filtering strategy, which dynamically switches among several fixed pre-filtering techniques based on the performance of each in prior iterations. The chosen pre-filtering technique is used to construct a subset from which the query will be selected. Information from past iterations is used indirectly to influence the selection of the query in the current iteration.

1.2.2 Computational Efficiency for Active Learning

Performing active learning can consume considerable computational resources. Reducing the computational overheads of active learning will subsequently reduce the cost of producing an accurate classifier, allowing machine learning solutions to be applied to many more domains.

Pre-Filtering Error Reduction Sampling

The computational overhead of employing error reduction sampling (Roy & McCallum, 2001) is directly proportional to the number of unlabelled examples it must consider as potential queries. Subset optimisation has been suggested as a way to reduce the computational overhead by restricting the number of candidate queries considered.

The most common way to construct the subset is random sub-sampling. We investigate *pre-filtering*, where a query selection strategy is used to filter informative unlabelled examples from the pool to form the subset. Pre-filtering is empirically evaluated as a way to both decrease the computational expense of error reduction sampling and simultaneously improve the quality of queries selected using error reduction sampling.

Empirically it is shown that pre-filtering can result in more accurate classifiers compared to random sub-sampling. Furthermore, pre-filtering using uncertainty sampling allows for smaller subsets to be used without decreasing the accuracy of the resulting classifier.

Dimensionality Reduction for Active Learning

Natural language text results in high-dimensional data when parsed into the vector space model. High-dimensional data is a problem since it increases both computational and storage overheads. Furthermore, some classifiers perform poorly in high-dimensional data due to the *curse of dimensionality*. Dimensionality reduction techniques are commonly applied to text data in supervised learning. However, the most frequently used technique, namely supervised feature selection, cannot be readily employed in an active learning setting since the majority of data supplied to an active learning algorithm is unlabelled. We investigate unsupervised dimensionality reduction as a possible solution to high-dimensional data in active text categorisation problems.

1.3 Related work

There has been substantial research into the problem of query selection. Chapter 3 discusses in detail some of the query selection strategies pertinent to the research in this thesis.

Chapter 5 investigates novel non-Markovian History-based query selection strategies, which incorporate information from prior iterations of active learning into the query selection process. Typically, query selection strategies have based the selection of queries on the predictions from the most recently induced classifier. Roy & McCallum (2001) discussed First order Markov active learning, where the aim is to select a query ($\mathbf{x}^*$) such that, when labelled and included in the training data, it will result in lower error than any other potential query.

It could be argued that History-based query selection strategies incorporate information about the unlabelled examples by utilising predictions from prior iterations. Query selection strategies discussed in Section 3.4.5 also incorporate information about unlabelled examples into the query selection process. However, such strategies incorporate unlabelled data directly, while History-based query selection strategies incorporate the information indirectly via predictions. In addition, query selection strategies which incorporate unlabelled information are generally computationally expensive. Error reduction sampling (Roy & McCallum, 2001) for example has a time complexity of $O(n^2)$. The use of History makes it very efficient to incorporate extra information about the unlabelled data.

Chapter 7 investigates pre-filtering as a way to reduce the computational overhead of performing error reduction sampling (Roy & McCallum, 2001). Culver *et al.* (2006) have previously used pre-filtering in error reduction sampling experiments. However, we extend the comparison both in terms of scale and techniques considered. Furthermore, the proposed adaptive pre-filtering technique is an extension of adaptive query selection strategies (Baram *et al.*, 2004; Pandey *et al.*, 2005; Donmez *et al.*, 2007). Adaptive query selection strategies dynamically choose which fixed query selection strategy to use in the current iteration of active learning based on the performance of each potential strategy in the past. Adaptive pre-filtering dynamically chooses which query selection strategy is used to form the subset for error reduction sampling. Information from past iterations is also used indirectly by the proposed adaptive pre-filtering technique. The performance of each fixed pre-filtering technique in prior iterations of active learning determines which is selected in the current iteration.

1.4 Thesis Layout

The remainder of this thesis is as follows:

Chapter 2 provides an overview of automated text categorisation. Various components of an automated text categorisation system are discussed including: parsing, dimensionality reduction, classifier induction and evaluation. The PAC model, which gives a bound on the number of labelled examples required to induce an accurate classifier, is also discussed.

Chapter 3 reviews active learning, giving details on each component of the active learning process. The critical component of an active learning algorithm is the query selection strategy, which is responsible for the selection of informative examples from the pool. Query selection techniques which have emerged in the literature are reviewed.

Chapter 4 provides details on methodology of experiments conducted including datasets, classification algorithms and evaluation metrics used in the empirical evaluation. Baseline analysis provides motivation for active learning by demonstrating the benefit in reducing labelling effort.

Chapter 5 introduces History-based query selection which incorporates predictions from classifiers in previous iterations of active learning into the selection of informative unlabelled examples from the pool. Predictions from the classifier induced in each iteration of active learning on the unlabelled examples are collected and stored in a data structure called History. Non-Markovian query selection strategies are then developed which incorporate the information from the History into the selection of queries in the current iteration.

Chapter 6 discusses dimensionality reduction for an active learning setting. The majority of data supplied to an active learner is unlabelled; therefore, the most commonly used dimensionality reduction technique used in text categorisation, namely supervised feature selection, cannot be readily employed. We suggest the use of unsupervised dimensionality reduction techniques and empirically show how they benefit a k-NN classifier.

Chapter 7 discusses pre-filtering as a way to simultaneously increase the quality of queries selected by error reduction sampling while also reducing computational overhead. An adaptive pre-filtering technique is proposed which can dynamically switch between fixed pre-filtering techniques.

Chapter 8 draws conclusions from the results presented in the thesis and suggests directions for future research.

Chapter 2

Text Categorisation

2.1 Introduction

Classification is the task of separating data into categories defined by *concepts* that represent topics of interest. A text classification system assigns natural language documents to a set of predefined categories. Manual classification of documents is feasible for small amounts of data, however, due to the rapid growth in the production and storage of natural language documents manual categorisation has become impractical.

Automated text classification systems have emerged to solve this problem. Since their introduction in the 1980s they have evolved as new advances in machine learning have been proposed. The main motivation for continued research has been the reduction in effort to construct the automated systems.

This chapter reviews the sub-systems present in a modern automated text classification system. We begin by describing the first solutions to automated text categorisation known as expert systems. Machine learning solutions which overcome the *knowledge acquisition bottleneck* are discussed including the constituent components of an automated text classification system. Details are given on how the natural language documents are transformed into a model which can be used by induction classification algorithms. Methods of reducing dimensionality are covered which help lower computational and memory overheads of categorisation systems. Finally, evaluation metrics used to measure the performance of categorisation systems are discussed. All these sub-systems combine to produce an accurate categorisation system.

2.2 Background

Classification is the task of assigning labels to data depending on whether or not they match a particular concept. Concepts are usually denoted by a label which can be used to categorise the data. For example if the input domain is e-mail and a message matches the concept of spam, it is assigned with "spam" label to signify its membership of this category.

More formally, given some input domain (X) and a set of labels ($\mathcal{Y}$), a classifier can be viewed as a Boolean-valued function which maps from input to labels according to some concept, thereby assigning membership to a particular category. The *target function* ($f : X \mapsto \mathcal{Y}$) always maps from input documents to their correct labels. The goal of automated text classification is to construct a hypothesis ($h : X \mapsto \mathcal{Y}$) which approximates the target function as closely as possible.

The manner in which the hypothesis is constructed has evolved over time, however, the objective of constructing a function, which approximates as closely as possible to the target function, remains the same.

2.2.1 Expert Systems

The first successful solution to the problem of automated text classification was delivered through *expert systems* in the 1980s. Expert systems consist of a series of hand-crafted rules constructed by dedicated domain experts. These rules form the hypothesis ($h : X \rightarrow Y$) and are used to classify previously unseen documents.

The CONSTRUE system (Hayes & Weinstein, 1990) is a well-documented example of a successful expert system. Domain experts constructed rules which classified documents based on words that occurred within them. An example rule is shown in Figure 2.1.

CONSTRUE was a major success achieving high levels of performance, practically matching precision and recall levels of human indexers. Its deployment saved millions of dollars and significantly reduced the time taken to classify documents.

While very effective, rule-based expert system are inflexible, requiring domain experts to continuously construct new rules to adapt to changes in the input. This problem, referred to as the *knowledge acquisition bottleneck* (Sebastiani, 2002), prohibits widespread use of such systems in many problem domains. Manual construction of the hypothesis is far too expensive for many domains where automated solutions could be applied but are not due to the cost of developing and maintaining the expert system being simply too high.

```
if      ("wheat" & "farm")                       or
        ("wheat" & "commodity")                  or
        ("bushels" & "export")                   or
        ("wheat" & "tonnes")                     or
        ("wheat" & "winter" and (¬ "soft"))
then WHEAT else (¬ WHEAT)
```

Figure 2.1: An if-then-else rule from CONSTRUE. For example, this rule assigns the label WHEAT to a document which contains the word "wheat" and the word "farm". Documents for which the conditions are not met are assigned the label ¬WHEAT

2.3 Automated Text Categorisation

Machine Learning offers a solution to the knowledge acquisition bottleneck through the use of classifier induction algorithms, which learn a particular concept automatically from a set of positive and negative examples of the concept. This eliminates the burden of requiring domain experts to manually construct a hypothesis using hand-crafted rules and replaces it with the far less expensive task of labelling a set of examples.

Automated Text Categorisation (ATC) is a machine learning solution to the problem of classifying natural language documents. These techniques employ induction algorithms to construct classifiers from a set of examples labelled according to the concept to be learned. Supplied with training data consisting of positive and negative examples of the concept, the induction algorithm must produce a classifier that accurately classifies unseen examples.

An ATC system is comprised of a series of sub-processes. Figure 2.2 shows how these sub-processes interact to deliver a classifier capable of accurately classifying new and previously unseen examples.

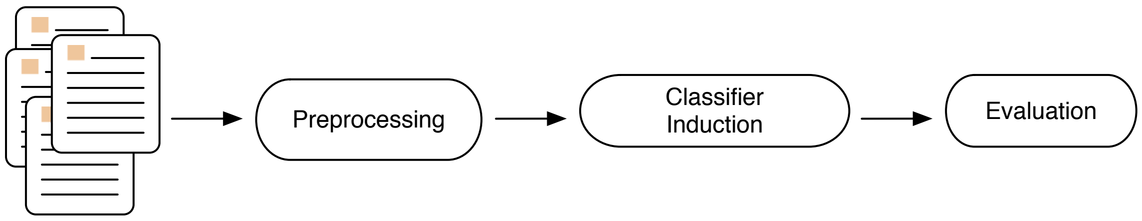


Figure 2.2: Text Categorisation Overview.

We give a brief overview of each element in an ATC system, ranging from pre-processing to evaluation metrics used to asses the quality of the classifier produced. Text categorisation is a very broad topic encompassing many disciplines. In this thesis we give details only for those aspects of text categorisation that are pertinent to our research. The interested reader is directed to (Sebastiani, 2002; Mitchell, 1997) for a full description of automated text categorisation and machine learning techniques.

2.4 Pre-processing: Parsing

Induction algorithms do not have the ability work directly with real world objects; rather they work on models, which represent real world objects. Even machine-readable document formats such as XML and PDF require transformation into an appropriate model before use with classification induction algorithms.

Parsing is the process whereby document attributes are extracted from the original raw representation and stored in a new internal representation, which can be easily interpreted by classification induction algorithms. This process is depicted in Figure 2.3

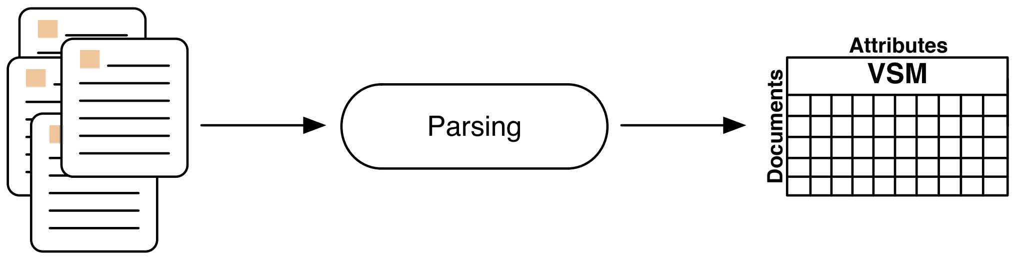


Figure 2.3: Parsing. Information from raw text documents in encoded in the Vector Space Model (VSM) which classifier induction algorithms can interpret. The VSM can be seen as a document X attribute matrix where documents are represented by the rows and attributes are represented by the columns.

2.4.1 Vector Space Model

The most commonly used model in text categorisation is the Vector Space Model (VSM) (Salton *et al.*, 1975), which represents documents as vectors of attributes. Each element of a vector stores the value of the attribute for the particular document. The VSM is often presented as a matrix where rows represent documents and columns represent *attributes*. The matrix is formed by stacking the document vectors.

Documents are described by their attributes, which can be any information about a document. A common approach for text classification is for attributes to represent the text within the document. The most straightforward approach is the *bag-of-words* (BOW) representation, where documents are considered to be an unordered set of words. Documents are parsed into their individual words (tokenised) and frequency information is recorded in a vector of length $|\tau|$ with the vocabulary (τ) containing all possible words in the corpus.

Attributes correspond to the words and their values represent the frequency of the word within the document. All position and structural information about the occurrences of the words are ignored. Despite this disposal of potentially beneficial information the BOW representation works very well and it is the *de-facto* standard in text classification.

		a	b	c	d	e	f	h
"a b a c d"	d1	2	1	1	1			
"b c d c f f"	d2		1	2	1		2	
"h f b a d"	d3	1	1		1		1	1

Figure 2.4: Vector Space Model (VSM). Each document is encoded in the VSM as a vector where each element represents the frequency of each attribute in the document. For example, the attribute "a" occurs twice in document **d1**, once in **d3** and not at all in **d2**

Increasingly sophisticated representations can be achieved by changing what constitutes an attribute. For example, n-grams consider an attribute to be a phrase rather than a word. This allows some ordering information to be retained. However, new problems arise as the vocabulary size dramatically increases. Dumais *et al.* (1998) suggest that the addition of n-gram terms actually decreases the overall performance of classification systems.

2.4.2 Weighting Schemes

As testament to the influence of Information Retrieval (IR) in machine learning, term weighing schemes (Salton & Buckley, 1988) are commonly used. Term weights were developed for use with the vector space model as a means to adjust the influence of certain attributes. There are three components to a weighting scheme:

Document Component. The document component captures term frequency information within individual documents. Binary weighting simply represents the absence or presence of a term in a document by a 0 or 1 respectively. Terms that occur more frequently in a document should be considered more important, hence given a higher weight. Term frequency (tf) measures the number of times a term (w_i) occurs in a document (d_j).

$$tf(w_i, d_j)$$

Collection Component. The collection component of the weighting scheme allows us to adjust the influence of a term. The document frequency of a term is the number of documents it appears in within the corpus.

$$DF(w_i)$$

By multiplying the term frequency by the inverse of document frequency ($tf * idf$), more influence can be given to terms which are frequent in a document but which are infrequent within all documents in the corpus ($\mathcal{D}$).

$$tf * idf(w_i, d_j) = tf(w_i, d_j) * \log \frac{|\mathcal{D}|}{DF(w_i)}$$

Normalisation Component Finally the normalisation component allows for documents of different length to be given equal weight. L2 (cosine) normalisation is commonly used as shown in (2.1).

$$x_i = \frac{tf * idf(w_i, d)}{\sqrt{\sum_{j=1}^{|\tau|} tf * idf(w_j, d)}} \tag{2.1}$$

where τ is the vocabulary. Weighting schemes range in complexity and generally have a large impact on the performance of the classification system.

2.4.3 High-dimensional Data

Text data is high-dimensional by nature. The principal causes of high-dimensional data in text stems from the richness and variety of natural language combined with the vector space model representation of documents.

The VSM dimensions will be defined by the number of documents in the corpus and the vocabulary (τ) which is the set of all unique words in the corpus. Even for relatively small corpora, the size of the vocabulary can be extremely large ($|\tau| \simeq 10^2 \ldots 10^5$ is common). Additionally, representing documents in the vector space model tends to create very sparse vectors, with the number of non-zero elements being quite low, as demonstrated in the example from Figure 2.4.

High-dimensional data increases computational expense. Comparisons between two feature vectors requires examining all features, therefore high-dimensional data requires increased computational expense. Similarly, memory and storage requirements are increased in order to handle very large vectors and matrices.

Furthermore, induction algorithms are susceptible to the *curse of dimensionality* (Mitchell, 1997), which loosely states: as the number of features increases, the performance of machine learning algorithms tends to decrease, sometimes significantly. Unlike information retrieval where matching algorithms scale well with very large numbers of features, classifier induction algorithms tend to suffer when supplied with high-dimensional data. The degradation in performance can be attributed to the presence of many irrelevant and redundant features in the data, which make the classification process all the more difficult. For example, classification algorithms such as the k-nearest neighbour (k-NN) that depend on similarity metrics are affected since examples from completely disparate topics can seem similar if they share a large number of these irrelevant features.

In an effort to reduce the overall number of terms to consider the Porter stemming algorithm (Porter, 1980) and stopword removal are sometimes used. Stemming is the collapsing of words to their common morphological root (e.g. “computers” and “computing” are stemmed to “comp”). Stemming sometimes reduces the effectiveness of classification systems hence there are times when it is not used. Stopwords are words that occur frequently in all documents. These offer little information and waste storage resources. Removal of stopwords is straightforward; a list of common stopwords is kept and while documents are being tokenised, if a token matches a stopword it is discarded.

2.5 Dimensionality Reduction

Even with stopword removal and stemming, the vocabulary can remain very large. High-dimensional data places heavy demands on computation and storage. Dimensionality Reduction (DR) is a technique that allows for significant reduction in the number of attributes used to describe the data without significant loss in overall classification performance. It can be seen as a lossy-compression algorithm which reduces the size of the data with minimal loss of information.

After parsing is complete, the original data is represented by a set of attributes. The dimensionality of this representation is typically very high. Dimensionality reduction is achieved by representing the original data using a small number of *features* as shown in Figure 2.5. A feature can be any attribute or a combination of multiple attributes and can either be selected or extracted from the original set of attributes describing the data. Here we make the distinction that the data is described by attributes after parsing while it is described using features after dimensionality reduction (if any) is performed. The number of features should be less than the number of attributes.

$$|\text{features}| \ll |\text{attributes}|$$

Of course if no dimensionality reduction is performed, then the number of features will equal the number of attributes.

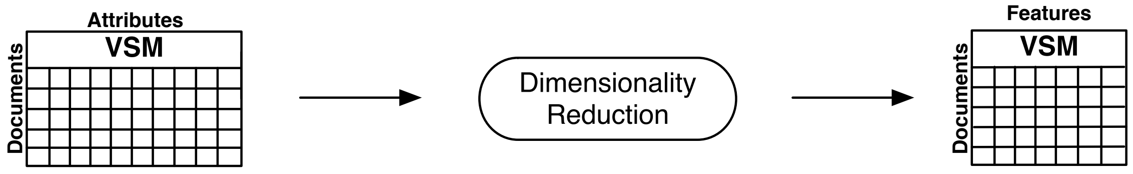


Figure 2.5: Dimensionality Reduction for the VSM. After parsing the data is represented using a large number of attributes. Dimensionality reduction represents the same data using much fewer features with minimal loss of information.

Dimensionality reduction offers a host of benefits including reducing computational overheads and sometimes increasing classifier performance (Yang & Pedersen, 1997). Given the benefits of dimensionality reduction it is commonly used as a pre-processing stage in ATC systems.

Features are found in two principled ways, namely *feature selection* and *feature extraction.* Feature selection is where a subset of the original attributes are used to represent the data. Feature extraction is where features are formed as combinations of one or more attributes.

2.5.1 Feature Selection

Feature selection seeks to find the minimum number of features that offer the best representation of the original data. The original attributes are evaluated using some performance metric and a subset of the best attributes are selected to represent the original data.

Many metrics have been proposed to evaluate the attributes. Empirical evaluation of the various metrics (Yang & Pedersen, 1997; Forman, 2003) has shown Information Gain to be one of the best performing. These studies have also shown that increased classification accuracy can be achieved when feature selection is applied. There are two principled methods to perform feature selection:

Filter-based methods seek to find the optimal subset of features irrespective of the classification algorithm used. The original attributes are ranked with an appropriate metric (e.g. Information Gain) that defines how discriminative the attribute is. A subset of the most discriminative attributes are chosen to be the features according to a user-defined threshold.

Wrapper-based feature selection is where the selection method has explicit knowledge (and use) of the classification induction algorithm. Induced classifiers are used to help select the optimal subset of attributes for the particular type of classification induction algorithm. While wrapper-based methods have the advantage of tailoring the feature selection to the classification algorithm, a major disadvantage is the computational overhead.

The simpler and computationally less expensive filter-based methods are far more prevalent in text categorisation systems.

2.5.2 Feature Extraction

Feature extraction is an alternate approach for dimensionality reduction where the original data is represented by a succinct number of new features constructed from (combinations of) the original attributes. These new features are constructed in such a way that there is minimal loss of information.

Feature extraction techniques are sometimes referred to as transformation techniques arising from the fact that the original data is transformed (projected) to a new lower dimensional representation. Perhaps the best known feature extraction technique is Principal Components Analysis (PCA) (Pearson, 1901) which transforms the data onto a set of orthogonal dimensions. It is readily performed using Eigenvalue decomposition of the covariance matrix or singular value decomposition of the data matrix. Depending on the domain, PCA is referred to as Latent Semantic Indexing (Deerwester *et al.*, 1990) in information retrieval and is also sometimes referred to as the Karhunen-Loève transform (Kirby & Sirovich, 1990).

More formally, PCA orthogonally transforms the coordinate system used to represent the data. The new coordinate system is defined by principal components which represent directions of maximal variance in the original data. Most of the information contained in the original representation can generally be represented using just a small number of principal components.

Principal components are the eigenvectors as given by the eigenvalue decomposition of the covariate matrix constructed from the centred data. Eigenvectors are ranked according to their associated eigenvalues, which signify the proportion of variance in the data accounted for by the eigenvector. Thus, the first principal component is the eigenvector with the largest associated eigenvalue, the second principal component is the eigenvector with the second largest associated eigenvalue and so on.

A small set of leading eigenvectors are selected which account for the majority of the variance and the original data is projected onto this subspace. The number of eigenvectors chosen is typically based on the proportion of variance in the data set accounted for by the chosen eigenvectors. If too few are chosen, reconstruction error will be high as the variance in the data was not well captured, similarly if too many are chosen the reduction in dimensionality can be low.

There are a number of drawbacks to PCA. The first is that features are no longer human interpretable which can cause confusion in real world scenarios. Secondly eigenvalue decomposition on very large matrices is an expensive operation.

Finally, there are assumptions made in PCA, the most notable one being that the data is linearly separable. Kernel Principal Component Analysis (Schölkopf *et al.*, 1999b) was proposed which extends the ability of PCA to find non-linear transformations of the original data. Other feature extraction techniques used in machine learning include, Independent Component Analysis (Comon, 1994), Random Projections (Bingham & Mannila, 2001) and Local Linear Embedding (Roweis & Saul, 2000).

2.5.3 Unsupervised Dimensionality Reduction

It is worth mentioning that the majority of feature selection methods and some feature extraction methods are supervised, that is, they explicitly make use of the label information. This is of particular importance to active learning problems where the majority of supplied data is unlabelled. Chapter 6 highlights this issue and proposes the use of unsupervised dimensionality reduction techniques for use in active learning problems.

2.5.4 Policy

The way in which a dimensionality reduction technique is applied is referred to as a policy. There are two distinct ways in which a dimensionality reduction technique can be applied, namely a global and local policy.

Local is where a reduced set of features are chosen for each individual category in isolation of other categories (Apté *et al.*, 1994). Essentially this means that each document will have a separate representation for every category under consideration. Local dimensionality reduction is a context-sensitive method since category information is explicitly used.

Global is where a reduced set of features are chosen irrespective of the category under consideration (Yang & Pedersen, 1997). This is a context-free method of performing dimensionality reduction.

Most dimensionality reduction techniques can be applied in both a local and global policy. However, local dimensionality reduction assumes that label information is available. This assumption does not hold in an active learning setting and we explore this issue further in Chapter 6.

2.6 Classifier Induction Algorithms

In classification we wish to build classifiers for concepts, such as "is this e-mail a spam?". Concepts correspond to a subsets of the input domain (X). For text classification, the goal is to build classifiers from a set of labelled examples that can automatically assign new documents to a set of predefined categories.

An alternative view of a concept is as a Boolean-valued function $f : X \mapsto \mathcal{Y}$ which decide membership of an input patterns to a concept. This function is referred to as the *target function*, which always correctly labels input patterns.

Induction allows the learner to learn an approximation (hypothesis) of the target function from a limited amount of examples labelled by the target function. Generally more than one hypothesis can be induced from a given training set. The set of all hypotheses consistent with the observed training data is called the version space (Mitchell, 1997).

Given that the learning algorithms induce a classifier from a set of labelled examples, this is commonly referred to as *supervised learning.* More formally, the standard supervised learning problem is stated as follows. Given a set of training examples $\mathcal{D}_\ell = \{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_m, y_m)\}$

drawn from the input space X and labelled using some unknown function $y = f(\mathbf{x})$, an induction learning algorithm will output a classifier $h(\mathbf{x})$ which is a hypothesis of the unknown target function $f(\mathbf{x})$.

There is a vast number of classifier induction algorithms available. Detailed description of all classifiers is beyond the scope of this thesis. The remainder of this section gives an overview of each of the main classification induction algorithms used throughout the research conducted. We also discuss ensemble based approaches where multiple classifiers are used.

2.6.1 k-Nearest Neighbour (k-NN)

The k-NN classifier is perhaps one of most straightforward classification algorithms. It is an instance based classifier where new documents are assigned labels based on the closest training examples as given by a similarity metric. An assumption exists that examples of the same concept should be similar to each other.

In contrast to most classification algorithms which are *eager* (process their training data immediately) the k-NN is a *lazy* algorithm, delaying processing of training data until classification time. See (Yang *et al.*, 2003) for more details on running time.

For an input $\mathbf{x}_q \in X$ the k-NN will find the k training examples which are most similar as defined by the similarity metric. Let S be the set of the k most similar training examples, then, the predicted label for the input example is assigned to be the mode of the predictions of the nearest neighbours labels as shown in (2.2)

$$h(\mathbf{x}_q) \leftarrow \arg\max_{y \in \mathcal{Y}} \sum_{\mathbf{x}_i \in S} \delta(y, f(\mathbf{x}_i)) \tag{2.2}$$

where $\delta(a, b)$ equals 1 iff $a = b$ otherwise 0. The similarity of the input $\mathbf{x}_q$ and the most similar training examples can be used to give finer control on the predicted class label as shown in (2.3). Here, more similar training examples have more influence in the predicted label.

$$h(\mathbf{x}_q) \leftarrow \arg\max_{y \in \mathcal{Y}} \sum_{\mathbf{x}_i \in S} sim(\mathbf{x}_q, \mathbf{x}_i)\delta(y, f(\mathbf{x}_i)) \tag{2.3}$$

The most popular similarity metric used is Euclidean distance (2.4). Euclidean distance defines similarity as the L2 norm between two feature vectors ($\mathbf{x}_i = \{x_1, x_2 \ldots x_{|\mathcal{T}|}\}$), where the greater the distance the more dissimilar two feature vectors are.

$$d(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j|| = \sqrt{\sum_{l=1}^{|\tau|}(x_{i_l} - x_{j_l})^2} \tag{2.4}$$

While Euclidean distance works well for many types of machine learning problems, it sometimes works poorly with high-dimensional data common in text classification problems due to the curse of dimensionality. Two quite different documents might be deemed similar because they share a large number of irrelevant features. An alternative similarity metric commonly used in information retrieval tasks is cosine similarity (2.5). It defines similarity as the angle between two feature vectors. This metric can be used in place of Euclidean distance for the k-NN classifier, where neighbours are defined based on maximum similarity rather than minimum distance.

$$COS(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{||\mathbf{x}_i|| \cdot ||\mathbf{x}_j||} \tag{2.5}$$

Predictions

The output of the k-NN can be transformed into a probabilistic classifier by considering the distance to each of the neighbours. Equations (2.2) and (2.3) can be modified to return the *weight* of each predicted label. The weights are then normalised to give a pseudo-probabilistic output from the classifier predictions.

Parameters

The choice of k is an important parameter. The most straightforward approach is to assign the label of the most similar training example, referred to as a 1-NN. Varying the value of k will have an impact on the accuracy of the classifier. The optimal value for k is typically found using a validation set.

Performance

The k-NN classifier has been shown to be one of the best performing classifiers for document classification tasks (Yang & Liu, 1999). Furthermore, lazy learning algorithms are desirable for active learning given a potentially large number of iterations and a classifier being induced in each iteration.

2.6.2 Naïve Bayes

Probabilistic classifiers estimate $P(\mathcal{Y}|X)$ the probability of a label given a document. Given a new document to classify ($\mathbf{x} \in X$) a Bayesian classifier will assign a label ($y \in \mathcal{Y}$) for which $P(y|\mathbf{x})$ is highest, as shown in (2.6).

$$\hat{f}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} P(y|\mathbf{x}) \tag{2.6}$$

Induction algorithms must approximate the target function by estimating the probabilities based on observed training data. Bayes rule (Mitchell, 1997, Chapter 6) specifies how the conditional probability $P(\mathcal{Y}|X)$ can be constructed from observing training examples consisting of documents from each concept as labelled by the target function.

Using Bayes theorem (2.6) can be rewritten as:

$$h(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} P(y) \frac{P(\mathbf{x}|y)}{P(\mathbf{x})} \tag{2.7}$$

Estimates of $P(\mathcal{Y})$ are constructed from the training data by simply counting the frequency of the labels. However, estimating $P(X|\mathcal{Y})$ is more challenging since there are potentially infinite variations of X. In order to obtain reliable estimates, the training data must contain multiple copies of every possible variation of X.

The naïve Bayes classifier makes the assumption that attributes are conditionally independent. While this assumption is clearly false, significant simplifications in calculations can be made where (2.7) can be written as shown in (2.8):

$$h(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} P(y) \prod_{\mathbf{x}_i \in \mathbf{x}} P(\mathbf{x}_i|y) \tag{2.8}$$

where $P(\mathbf{x}_i|y)$ is the probability of a term for a given label and the denominator $P(\mathbf{x})$ in (2.7) is not computed since it is the same for all $y \in \mathcal{Y}$.

Multinomial Naïve Bayes

Throughout this thesis we concentrate on the *multinomial* variant of the naïve Bayes classifier as used in (McCallum & Nigam, 1998). The multinomial naïve Bayes assumes text documents are generated by a mixture model, parametrised by θ.

A classifier is induced by estimating θ from a set of labelled training data ($\mathcal{D}_\ell \subset X$). The probability of a word given a particular category is simply the fraction of times the word occurs in a document of that category.

$$\hat{\theta}_{x_i|y} = \frac{1 + \sum_{\mathbf{x} \in \mathcal{D}_\ell} tf(x_i, \mathbf{x}) P(y|\mathbf{x})}{|\tau| + \sum_{k=1}^{|\tau|} \sum_{\mathbf{x} \in \mathcal{D}_\ell} tf(x_k, \mathbf{x}) P(y|\mathbf{x})}$$

where $tf(x_i, \mathbf{x})$ is the count of the term x_i in the document $\mathbf{x}$ and $P(y|\mathbf{x})$ is the class label for the training document $\mathbf{x}$. Laplacian smoothing is used to avoid zero probabilities. Similarly, calculation of the probability of a category is simply the percentage of the training data that is of the same category.

$$\hat{\theta}_y = \frac{\sum_{\mathbf{x} \in \mathcal{D}_\ell} P(y|\mathbf{x})}{|\mathcal{D}_\ell|}$$

Using these two estimates a classifier can be constructed where the predicted category is the label ($y \in \mathcal{Y}$) with the highest probability and the probability of a label given a document ($\mathbf{x}$) is calculated using (2.9):

$$P(y|\mathbf{x}) = \frac{P(y|\hat{\theta}) \prod_{x_i \in \mathbf{x}} P(x_i|y; \hat{\theta})}{\sum_{y \in \mathcal{Y}} P(y|\hat{\theta}) \prod_{x_i \in \mathbf{x}} P(x_i|y; \hat{\theta})} \tag{2.9}$$

Performance

The naïve Bayes classifier was popular for text categorisation tasks due to its simplicity and ease of implementation. Despite the strong independence assumptions made, the classifier performed well in text classification tasks. However, it has been shown (Yang & Liu, 1999) that its performance has been eclipsed by kernel methods classifiers such as the support vector machine.

2.6.3 Support Vector Machine (SVM)

In recent years kernel methods have been extremely popular due to their excellent performance on many machine learning tasks and in particular on text categorisation tasks (Joachims, 1997). Rather than devising a different algorithm for each domain, they settle on a standard set of algorithms and transform the data to a representation that suites the set of algorithms. Kernel methods consist of two modular components. The first is a standard classification algorithm, while the second is a kernel function that is used to transform the data to allow for easier classification using the classification algorithms.

We briefly discuss kernels and how they can be used to transform the data. The set of classification algorithms used in kernel methods are standard and have only one prerequisite: they must be described in terms of dot-products only.

Kernel

A classifier can easily be induced to correctly classify data from a domain which is linearly separable. However, for many real world problems the classes in the input space are unlikely to be linearly separable. One solution is to use a classifier specifically for each domain, e.g. using the non-linear classifier such as the k-NN. Kernel methods use an alternative strategy whereby the data is re-coded so that it becomes much easier to classify. A standard classification algorithm can therefore be used on all domains.

The function $\Phi(\mathbf{x})$ transforms (re-codes) the original input into a new feature space. Typically the dimensionality of the transformed feature space is much higher than the original feature space. Figure 2.6 shows an example where the data is non-separable in 2 dimensions but is separable in 3 dimensions. Explicitly transforming each input into the new feature space is computationally expensive (and potentially infeasible when infinite dimensions are used).

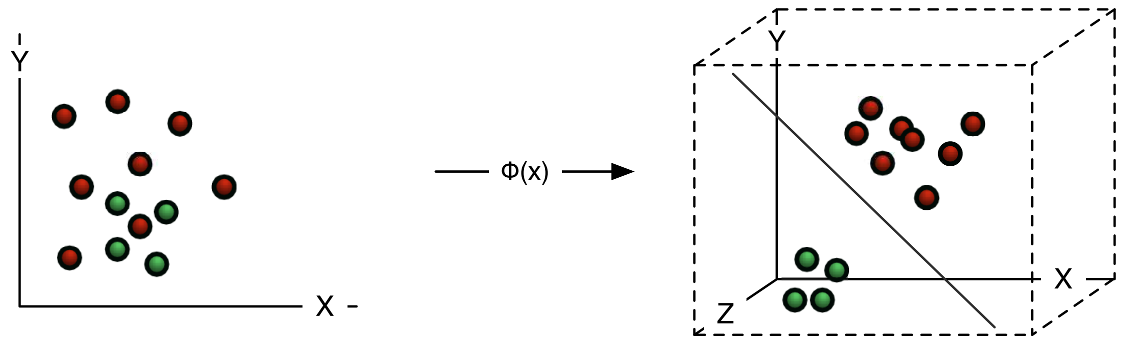


Figure 2.6: Kernel mapping. A trivial example showing that linear classification becomes easier once the original two-dimensional data is mapped to another three-dimensional feature space.

A “kernel” is a similarity measure that can be thought of as a dot product in some, user defined, feature space, as shown in (2.10). The *kernel trick* refers to the fact that the dot product of two projected examples $\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$ can be calculated without ever explicitly transforming the original data.

For example, consider the transformation function that maps from two dimensional feature space ($\mathbf{x} = \{x_1, x_2\}$) to three dimensional feature space ($\Phi(\mathbf{x}) = \{x_1^2, x_2^2, \sqrt{2x_1x_2}\}$)

Table 2.1: Commonly used Kernels.

Kernel	Formula
Linear	$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$
Polynomial	$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^d$
Gaussian	$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})}$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \tag{2.10}$$

$$\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle = \langle \{x_{i_1}^2, x_{i_2}^2, \sqrt{2x_{i_1}x_{i_2}}\}, \{x_{j_1}^2, x_{j_2}^2, \sqrt{2x_{j_1}x_{j_2}}\} \rangle \tag{2.11}$$

$$= x_{i_1}^2 x_{j_1}^2 + x_{i_2}^2 x_{j_2}^2 + 2x_{i_1}x_{i_2}x_{j_1}x_{j_2} \tag{2.12}$$

$$= \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2 \tag{2.13}$$

In this way we can compute the dot product of two instances in the transformed feature space using (2.13) meaning that we never have to explicitly evaluate the coordinates in the new feature space (as is done in (2.11)). The power of the kernel trick becomes evident when the transformation function (Φ) projects the original data into possibly infinite dimensions. A number of kernel functions that are commonly used in the literature are given in Table 2.1.

Gram Matrix

The Gram matrix (sometimes called the kernel matrix) is formed by evaluating the kernel function on the training data and storing the results in a matrix $\mathbf{K}$ where its element are:

$$\mathbf{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$$

The dimensions of the matrix are $m \times m$ where m is the number of documents in the training data. The Gram matrix offers an interface between the kernel component and the classifier component in a kernel machine.

Classifier Component

The second component of a kernel machine is a classification learning algorithm. The support vector machine (Vapnik, 1995; Schölkopf *et al.*, 1999a) is a linear learner that separates the training data using a decision hyperplane ($\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$). The classification of an example ($\mathbf{x}$) is given by its projection along the hyperplane, plus some offset b and is shown in (2.14).

$$\hat{f}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b \tag{2.14}$$

Linear classifiers are binary therefore the predicted label is determined based on which side of the hyperplane the example lies. Figure 2.7 shows a linear classifier which separates the data. In the case of $\mathcal{Y} = \{\pm 1\}$ the classification is simply the sign of predicted values as shown in (2.15)

$$h(\mathbf{x}) = sign(\hat{f}(\mathbf{x})) \tag{2.15}$$

A special characteristic of the hyperplane produced by an SVM is that the margin is maximised. The margin is the distance to the separating hyperplane from the point closest to it. It is desirable to have a large margin since it will be better able to tolerate noisy examples. In general, for a linear classifier to generalise well, a large margin should be sought. Finding the hyperplane which maximises the margin can be formulated as the following optimisation problem.

$$\begin{aligned} Minimise &: ||w|| && (2.16) \\ subject\,to &: \forall_{(\mathbf{x},y)\in\mathcal{D}_\ell} : y\langle \mathbf{w}\cdot\mathbf{x}\rangle + b \geq 1 && (2.17) \end{aligned}$$

Lagrange multipliers are used to translate the difficult *primal* optimisation problem (2.16) into an equivalent *dual* quadratic maximisation optimisation problem (2.18) for which there exists efficient algorithms.

$$\begin{aligned} Maximise &: -\sum_{i=1}^{m}\alpha_i + \frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j\langle \mathbf{x}_i\cdot\mathbf{x}_j\rangle && (2.18) \\ subject\,to &: \sum_{1=1}^{m}\alpha_i y_i = 0 \; with \quad \alpha_i \geq 0 && (2.19) \end{aligned}$$

The result of the optimisation process is a set of coefficients α. Only those training examples which have $\alpha_i > 0$ contribute to the construction of the hyperplane. These examples are called the *support vectors* and define the margin. Figure 2.7 shows a linear classifier constructed from three support vectors (highlighted) that lie on the margin.

Classification can now be evaluated in terms of dot products between the input instance and the training data set examples ((2.20) and (2.21)). Additionally only the support vectors need to be evaluated allowing for computational speed-ups to be made.

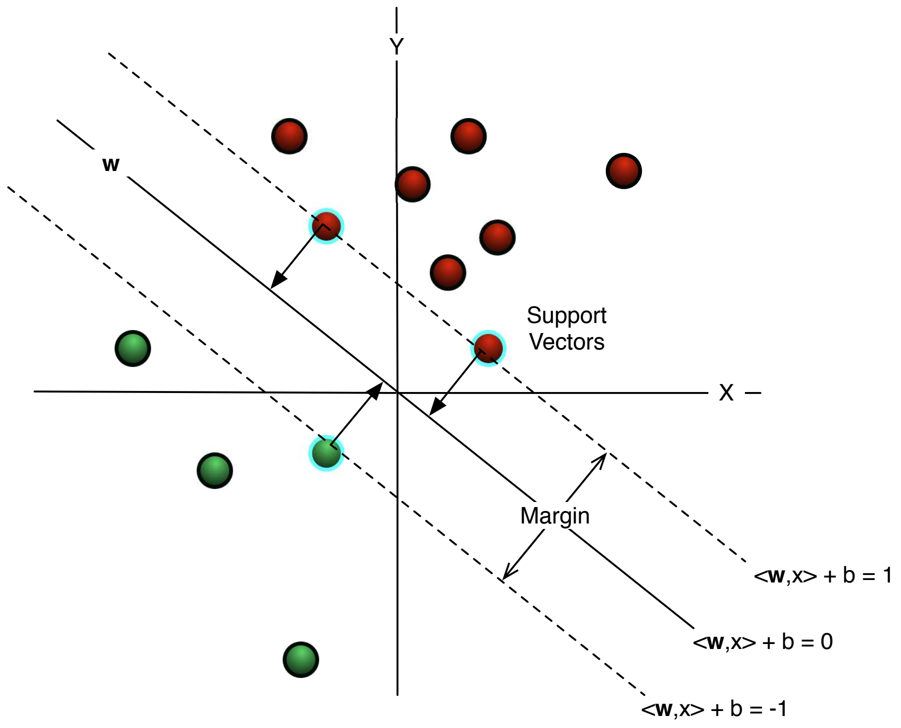


Figure 2.7: SVM Classifier. The SVM is a linear classifier that constructs a hyperplane **w** separating the data. The hyperplane is constructed in such a way to create maximal (margin) distance between the two classes. Computational efficiency is achieved since only the three Support Vectors are required to classify further instances. In this particular example the offset parameter b is zero since **w** passes through the origin.

$$\hat{f}(\mathbf{x}) = \sum_{i}^{m} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b \tag{2.20}$$

$$h(\mathbf{x}) = sign(\hat{f}(\mathbf{x})) \tag{2.21}$$

Both the optimisation problem of finding the hyperplane and the classification function are expressed in terms of dot products. These can easily be "kernelised" by substituting the dot product for an appropriate kernel function. Eqn. 2.18 and (2.20) can be substituted with (2.22) and (2.23) respectively. This allows classification algorithms to be applied in the new feature space while only ever operating on the original feature space by using the kernel function.

$$Maximise: \quad -\sum_{i=1}^{m} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{2.22}$$

$$\hat{f}(\mathbf{x}) = \sum_{i}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{2.23}$$

Performance

Support vector machines have been shown to be one of the best performing classifiers for text classification (Yang & Liu, 1999; Joachims, 1997). Since the classification learning algorithm is fixed, kernel machines place the importance on choosing the correct kernel rather than choosing the correct classifier. Sometimes choosing the appropriate kernel is almost as difficult as choosing the correct classifier.

2.6.4 Ensembles

The "No Free Lunch" theorem (Wolpert, 1995) states that there is no single learning algorithm that always induces the most accurate classifier for every domain. Each learning algorithms will perform differently depending on the domain. Yang & Liu (1999) considers the performance of many different learning algorithms for the task of text classification. In general, for supervised learning tasks, a number of different learning algorithms will be considered and the one which outputs the best performing classifier, as evaluated on a validation set, is chosen.

An alternative solution is to combine multiple classifiers together. An ensemble (or committee) (Breiman, 1996) is a collection of classifiers whose individual decisions are combined in order to classify new examples. The members of the ensemble as referred to as the *base-learners*. Ensembles have been shown to increase accuracy when used with reasonably accurate base-learners. Ensembles perform best when the constituent base-learners are diverse. A number of methods for constructing ensembles are given below:

- *Stacking.* The easiest way to ensure diversity in the ensemble is to use different learning algorithms for each base-learner.
- *Parameters.* Most learning algorithms are characterised by a set of user defined parameters. An ensemble can be constructed by altering the parameters supplied to each base-learner. For example, different k values for the k-NN algorithm.

- *Representations.* Diversity can be established by training each base-learner using a separate representation of the data. This is similar to the idea of Co-Training (Blum & Mitchell, 1998) where multiple representations of the same data exist. Similarly, one could use different kernels for each base-learner to transform the original data into different representations.

- *Training Data.* The most common way to construct an ensemble is to train each base-learner with different training data. Two bootstrapping approaches have been successfully used in the literature, namely *boosting* and *bagging.*

Bagging

Bagging (Breiman, 1996) is where an ensemble of N members is constructed by sampling N bootstrap samples from the labelled training data, with replacement. Each base-learner is induced from one of the bootstrap samples.

Boosting

Boosting, in particular AdaBoost (Schapire, 1999) is an iterative process to construct a set of complementary base-learners for an ensemble. It iteratively adds new base-learners such that the training data of base-learners added in successive iterations are tailored by the misclassifications of the base-learner in previous iterations.

A weak base-learner is induced from a subset constructed by sub-sampling from the training data according to the weights. Initially all weights are uniform but are updated based on the misclassifications of the weak base-learner. Training examples which are incorrectly classified have their weights adjusted so that they are more likely to be chosen in subsequent iterations of boosting. In this way sampling is focused on these hard to classify examples. The output of AdaBoost is an ensemble formed using all the induced weak base-learners.

Ensemble Predictions

Predictions from an ensemble are obtained by combining the predictions of the members. Majority voting is the most commonly used method and simply finds the mode of the predictions in the ensemble (see (2.24)).

$$\hat{f}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{N} \hat{f}_i(\mathbf{x}) = y \tag{2.24}$$

where $\hat{f}_i$ is the prediction given by the i^{th} member of the ensemble.

The use of ensembles (committees) in machine learning has grown due to the increased performance achieved. However, considerable computational overheads are encountered since numerous classifier must be induced and considered for each prediction.

2.6.5 Multi-Class Classification

We have discussed mainly binary classification problems, where there are only two possible labels for an instance and the task is to construct a function which maps input examples to one of the two possible labels. However, we often wish to solve classification problem that have more than two labels. Multi-class classification is where the number of labels is greater than two.

Certain classifiers such as the k-NN are capable of easily extending to the multi-class setting while others such as Support Vector Machines do not. To overcome this, multi-class classification problems can be decomposed into a series of binary classification problems. Two principled techniques can be employed to achieve this, namely One-v-Rest and One-v-One. Below we give some details on both techniques. For a comprehensive evaluation of various multi-class techniques please see (Hsu & Lin, 2002).

One-v-Rest

One-v-Rest (1vR) decomposes a n-class problem into n separate binary classification problems where one concept is compared with all other concepts. For example, a multi-class classification problem to identify digits would construct 10 binary classifiers, one for each digit. One such classifier would identify the digit 7 and considers all other digits to be negative examples of this concept (Vapnik, 1995).

One-v-One

An alternative approach is One-v-One (1v1) where a series of binary classifiers are constructed for each possible pairing of concepts. Reusing the digits classification example, a classifier is constructed to for each pair of digits. In this way, $\frac{n(n-1)}{2}$ classifiers are constructed.

2.7 PAC Learning Model

An obvious question regarding induction algorithms is how many examples are required to produce an accurate classifier. Ideally we would like the induction algorithm to construct a classifier consistent with the target function for any possible input. In other words we wish the classifier to have zero error. Unfortunately this is infeasible for two reasons.

First, to achieve zero error requires training data consisting of labelled examples for every possible input pattern. This is infeasible for all but the most trivial of problems. Second, the training data is a random sample of examples. There exists a small probability that the sample contains noisy examples that mislead the induction algorithm and result in the production of a poor classifier. Hence, the construction of a classifier consistent with the target function may not always be possible.

The Probably Approximately Correct (PAC) (Valiant, 1984; Mitchell, 1997) learning model studies the number of examples required to achieve a classifier with a bounded error rate. It does so by weakening the demands on the induction algorithm. The induction algorithm is not required to succeed in constructing a classifier for every possible sequence of randomly drawn training examples. Failure is bound by some arbitrarily small parameter δ. Furthermore, the algorithm is not required to produce a zero-error classifier. Rather it is required to produce a classifier that has error bounded by ϵ.

These bounds presented allow PAC to state that, a learner will output a classifier (hypothesis) with probability $(1-\delta)$ that has error less than ϵ after observing a sufficient number of training examples and after performing a reasonable amount of computation.

The number of examples required is intrinsically linked to the efficiency of the learning algorithm in the PAC learning model. Training data size is limited by the time required to process it since PAC requires the output of a hypothesis after a reasonable amount of computation.

In document classification tasks, explicit calculation of PAC bounds is generally not conducted. A heuristic of supplying as much training data as possible is typically employed. There are many domains where the collection of labelled training data is impractical due to the high cost of acquiring labels. Active learning has been proposed as a technique to overcome this problem and is capable of construction of an accurate classifier from the small number of labelled training data. Chapter 3 describes active learning in greater detail.

2.8 Evaluation

A classifier induced from a set of labelled training examples will be deployed to classify new and previously unseen examples from the input space. Evaluation is concerned with estimating the performance of induced classifiers on future examples.

Induction of classifiers with zero error requires a labelled training example for every possible input pattern. Since this is impractical for all but the most trivial of classification problem, induced classifiers will misclassify a certain percentage of data. The optimal performance metric to use is *true error* which is the probability that the classifier will misclassify a randomly drawn instance from the input space.

$$Error(h) = P(h(\mathbf{x}) \neq f(\mathbf{x}))$$

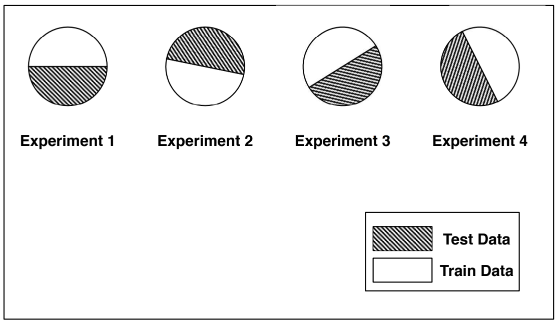
The true error of a classifier is unknown, however, an estimate can be calculated using a number of machine learning evaluation techniques.

2.8.1 Data Partitioning

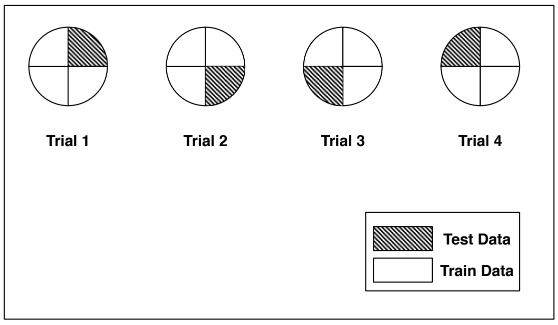
Estimates are constructed from a *test set*, which is a set of examples labelled by the target function and is used to evaluate the classifier performance. The test set can be constructed in a number of ways but, crucially, test data is completely unknown to the induction algorithm. In other words, test data is never used in training a classifier. Two of the most popular methods for constructing a test set are:

- *Holdout Testing.* This is where the training data is split into two mutually exclusive sets. One set becomes the training data from which the classifier is induced. The second is used as the test and is used to estimate the performance of the induced classifier. Figure 2.8(a) shows this process where half of the data is used as training data while half is held back for testing data. This is called 50/50 train/test split. However, it is considered more beneficial to have a larger test set as used in the RCV1 dataset (Lewis *et al.*, 2004).

- *k-fold-cross-validation.* When the number of labelled training data is small the holdout method becomes infeasible. k-fold-cross-validation overcomes this by problem by segmenting the data in to k folds (or sets). k trials of the experiment are then conducted where in each trial one fold is held back as testing data while all other folds are used as training data. The fold used as testing data changes in each trial. Results are averaged over all trials of the experiment. Figure 2.8(b) shows this for 4-fold cross validation.



(a) 50/50 Split



(b) k-Fold

Figure 2.8: Data partitioning methods. The entire circle represents all the data in the corpus. (a) randomly divides the dataset into two sets, where one is used for training while the other is held back for testing. (b) shows 4-fold cross validation where in each trial one fold is held for training while all others are used for testing. The results are averaged over all trials.

The misclassifications made by the classifier on the test set are stored in a *confusion matrix* from which a number of performance metrics can be calculated. In the next section we discuss the confusion matrix next as well as a number of performance metrics commonly used in the literature. Evaluation of classifiers is an important aspect of machine learning and there are many strategies and metrics in the literature. In this review we concentrate on the most popular evaluation metrics used for text categorisation tasks.

2.8.2 Confusion Matrix

Central to all of the following metrics is the *confusion matrix* (sometimes referred to as the contingency matrix) which encodes the errors made by a classifier when evaluated on a test set. The classifier is used to predict the labels for the members of the test set and the predictions are compared to their true labels as given by the target function.

Table 2.2 shows the confusion matrix for a binary classification problem ($\mathcal{Y} = \{-1, 1\}$). Predictions given by the classifier (h) are compared to the labels given by the target function (f). The table consists of four entries:

- True Positives (TP) counts the number of examples where $h(\mathbf{x}) = 1$ and $f(\mathbf{x}) = 1$

- False Positives (FP) counts the number of examples where $h(\mathbf{x}) = 1$ and $f(\mathbf{x}) = -1$
- False Negative (FN) counts the number of examples where $h(\mathbf{x}) = -1$ and $f(\mathbf{x}) = 1$
- True Negative (TN) counts the number of examples where $h(\mathbf{x}) = -1$ and $f(\mathbf{x}) = -1$

These four counts can be combined to evaluate a classifiers performance. Generally, high-performing classifiers will have relatively few false positives and false negatives.

	$f(\mathbf{x}) = 1$	$f(\mathbf{x}) = -1$
$h(\mathbf{x}) = 1$	TP	FP
$h(\mathbf{x}) = -1$	FN	TN

Table 2.2: Confusion Matrix for a binary classifier. True labels are given by the columns while the predicted labels are given by the rows.

2.8.3 Accuracy

The accuracy of a classifier is perhaps the most often used metric. It examines effectiveness of the induced classifier to correctly classify examples from the test set and is calculated as shown in (2.25).

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.25}$$

Accuracy gives the percentage of predictions which were consistent with the true labels as given by the target function. Analogously, *error* is percentage of predictions inconsistent with the true labels and is simply $1 - Acc$.

Accuracy is best used when there is little to no bias in the data. If the data consists of a large percentage of a single class, then a trivial acceptor classifier (assigns the label of the majority class found in the training data) will achieve high accuracy. For this reason accuracy is rarely used as an evaluation metric for imbalanced datasets (Yang & Liu, 1999; ?).

2.8.4 Precision and Recall

Precision (π) and Recall (ρ) are commonly used performance metrics in information retrieval systems. They are less sensitive to imbalanced dataset and can also be used in a classification setting. Precision is defined to be the probability that examples predicted by the classifier as being positive are indeed positive. More formally, when the prediction is $h(\mathbf{x}) = 1$ the true label is $f(\mathbf{x}) = 1$. It can be calculated from the confusion matrix as shown in (2.26):

$$\pi = \frac{TP}{TP + FP} \tag{2.26}$$

Similarly, recall is the probability that positively labelled examples are given positive predictions by the classifier. More formally, given $f(\mathbf{x}) = 1$ the probability that $h(\mathbf{x}) = 1$ also. It is calculated from the confusion matrix as shown in (2.27):

$$\rho = \frac{TP}{TP + FN} \tag{2.27}$$

When dealing with multi-class classification, precision and recall must be averaged over the whole category set ($\mathcal{Y}$). There are two ways to do this:

Macro-averaging is where precision and recall are computed per-category then the mean precision and recall values are computed. This is a local policy where each *category* is given equal weight.

Micro-averaging is where precision and recall are computed from the combined confusion matrix over all categories. This is effectively summing over all the individual decisions made by the classifier. Micro-averaging is a global policy, where each *decision* is given equal weight therefore more frequent topics will receive a higher weight in the average. When micro-averaging is performed, it is denoted by the super-script ^{M} in the remainder of this thesis.

2.8.5 F_1 Measure

Neither precision or recall make sense in isolation. The F_1 is a way to combine the separate precision and recall performance metrics into a single value. This allows for easier comparisons since only one value needs to be compared. It is defined to be the harmonic mean of precision and recall, where attributes equal importance to both and is calculated as shown in (2.28):

$$F_1 = \frac{2\pi\rho}{\pi + \rho} \tag{2.28}$$

In this thesis we compute F_1 using both macro- and micro- averaged precision and recall. F_1 calculated using micro-averaged precision and recall is denoted at F_1^M.

2.9 Summary

In this chapter, we have reviewed the many sub-systems involved in the construction of a automatic text classification system. Figure 2.9 shows how the various sub-systems are combined to produce an automated text categorisation system.

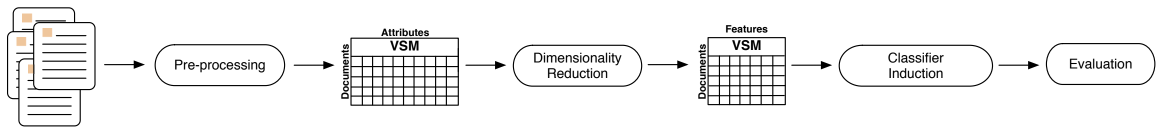


Figure 2.9: Automated Text Categorisation System

We reviewed the vector space model which is central to traditional machine learning and described feature extraction methods for constructing the VSM from raw data. The causes of high-dimensional data in text categorisation was discussed. Automatic dimensionality reduction techniques were reviewed and the benefits to text categorisation were given. Dimensionality reduction is a particular problem for active learning settings which is discussed further in Chapter 3 along with proposed solutions. We reviewed a number of classification induction algorithms, which learn classifiers from a set of examples. The number of examples required to learn an accurate classifier was described by the PAC learning model.

While labelling examples is an easier task than constructing rules by hand, domains exist where even the task of labelling is prohibitively expensive. Active learning is a solution to reduce the number of labels required to induce an accurate classifier and is discussed in detail in Chapter 3.

Chapter 3

Active Learning

3.1 Introduction

Supervised learning techniques assume the availability of a large number of labelled examples. Training data supplied to a supervised learning algorithm are constructed by labelling a large random sample of unlabelled examples. However, in many real-world domains the cost associated with constructing a large corpus of training material, suitable for use in supervised learning, is prohibitively high. Unlabelled examples, on the other hand, are plentiful and cheap to collect.

High labelling costs restrict the application of machine learning solutions to domains where the availability of labelled data is plentiful. In this chapter, we discuss *active learning* which is a technique in machine learning used to produce high quality (read accurate) classification systems from a small set of labelled examples.

Active learning expedites the learning process by allowing the learner control over the training data. Training data is constructed by the learner sequentially selecting the most informative examples to label from a pool of unlabelled examples. Significant reductions in labelling effort have been demonstrated when active learning was used to construct the training data (Lewis & Gale, 1994; Tong & Koller, 2001).

Rather than randomly select training examples, an active learner will carefully select and label only the most informative examples. Active learning is directly beneficial to domains where the cost of labelling is high but also has potential benefits for practically all domains where it can be used to expedite the learning process.

This chapter provides an extensive review of the active learning literature. As a motivation for active learning, the problem of limited training data in discussed in Section 3.2. Section 3.3 discusses active learning, giving details on the components that form the active learning process as well as describing a canonical active learner. Query selection is a critical component to active learning and we describe this in detail in Section 3.4. As with traditional supervised learning, evaluating performance is an important aspect to active learning research. Methods and techniques used to measure the performance of active learning are given in Section 3.5. Finally, we summarise and detail research conducted in latter chapters of this thesis in Section 3.6.

3.2 Limited Training Data

In many real-world problems unlabelled data is abundant and cheap to collect while labelling an example incurs a cost. In some domains the costs are prohibitively expensive resulting in severely restricted amounts of labelled training data. Limited amounts of training data pose a problem for supervised learning techniques which rely on the availability of large amounts of labelled training data to produce high-quality classifiers. The net result is that supervised learning techniques cannot be applied to a variety of domains.

In this section we discuss the cost of labelling examples in text categorisation problems. Furthermore the PAC learning model is revisited to discover how many examples are required to construct a classifier with a bounded error (ϵ).

3.2.1 Cost of Labelling

Even in domains where acquisition of large amounts of labelled data is common, the cost associated with obtaining the labelled data is often overlooked. As an example we look at the recently released RCV1 corpus (Lewis *et al.*, 2004), which comprises of over $800,000$ manually categorised Reuters news articles collected in the 12 months between August 1996 and August 1997. Lewis *et al.* (2004) report that Reuters, at one stage, employed 90 staff to categorise 5.5 Million English news articles produced in one year and estimate the labelling effort for the RCV1 corpus to be 12-person years. This represents a considerable undertaking in terms of both time and resources. Furthermore, despite strict quality checks the resulting corpus contained many labelling errors.

Since labelling is an expensive task, a classification system should be induced from as few examples as is strictly necessary. Furthermore reducing the labelling effort also diminishes the possibility for errors since even manual classification is not infallible (precision and recall for humans are estimated to be approximately 90% (Hayes & Weinstein, 1990))

3.2.2 PAC Bounds

It is desirable to know *a priori* how many labelled examples are required in order to construct a classifier with a bounded error. The PAC learning model (see Section 2.7) allows for an upper bound to be placed on the number of labelled examples required (m) to produce a classifier of given accuracy (ϵ) with high probability ($1 - \delta$).

Sample complexity (Mitchell, 1997, Chapter 7) measures the growth in the number of required training examples in connection with the problem. In most practical settings this is the factor that limits the success of the learner due to the limited availability of training data.

$$m \geq \frac{1}{\epsilon}\left(\ln(|H|) + \ln(\frac{1}{1-\delta})\right) \tag{3.1}$$

The sample size (m) is described in terms of the PAC learning model in (3.1). It places a bound on the number of examples required to successfully learn a target concept. Sample size is dependent on the size of the hypothesis space $|H|$ and grows linearly with $1/\epsilon$ and logarithmically with δ.

The size of the hypothesis space is difficult to quantify for many learning algorithms and can even be infinite. This means that calculation of the sample complexity can be difficult, if not impossible, when the hypothesis space is infinite. Vapnik-Chervonenkis (VC) Dimensions (Mitchell, 1997, Chapter 7) is an alternate measure of the complexity of the hypothesis space. VC measures the size of the subset of S ($S \subset X$) which can be shattered. The concept of shattering is where, for every dichotomy of the set S, there exists some consistent hypothesis $h \in H$. If S cannot be shattered by H then there must exist some dichotomy for which there does not exist a consistent hypothesis in H.

The upper bound on the number of training examples required to probably approximately learn a target concept can then be formulated using the VC of the hypothesis space rather than the potentially infinite value $|H|$. Sample complexity calculated using VC is shown in (3.2) which is analogous to (3.1). The VC-Dimensionality of linear decision surfaces in an r dimensional space is $r + 1$.

$$m \geq \frac{1}{\epsilon}\left(4\log_2(\frac{2}{\delta}) + 8 * VC(H)\log_2(\frac{13}{\epsilon})\right) \tag{3.2}$$

For example, if a consistent learner is to learn the target concept in a 10 dimensional space 95% of the time and achieve an error of less than 10% then it requires at most m training examples, where m is calculated as follows:

$$207.3147 \geq \frac{1}{0.1}\left(4\log_2(\frac{2}{0.05}) + 8 * 11\log_2(\frac{13}{0.1})\right)$$

The highest cost of applying a machine learning solution is the cost associated with collecting sufficient labelled training data. Ideally only the minimum number of examples required to achieve an accurate classifier should be labelled. However, in practice a large amounts of labelled data are supplied to the supervised learner in order to guarantee the PAC bounds are met and an accurate classifier is output. In some domains the cost of labelling is extremely high and the cost of collecting sufficient labelled training data restricts the application of machine learning solutions. We refer to this as the labelling acquisition bottleneck and in the next section we discuss techniques to overcome it.

3.3 Active Learning

The PAC learning model places an upper bound on the number of training examples required (m) in traditional supervised learning problem. Supervised learning algorithms are typically supplied with a large set of labelled training data ($|\mathcal{D}_\ell| \gg m$) to ensure an accurate classifier is produced. Figure 3.1 depicts this batch process whereby the training data are supplied to the learner. The learner outputs a classifier induced from the training data.

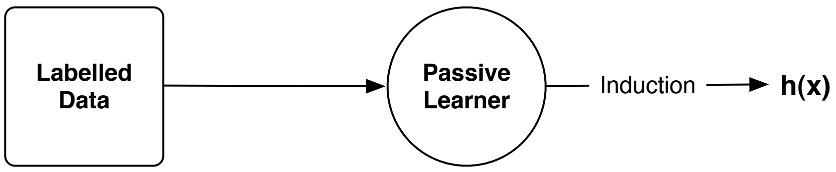


Figure 3.1: Passive supervised learning has no control over the training data, it is simply supplied to the classifier learning algorithm. Induction is used to construct a classifier (h) from all the training data.

Training data for supervised learning are formed by labelling an independent and identically distributed ($i.i.d$) sample of the underlying data distribution $P(\mathbf{x})$. In this way, the learner has no control over the construction of the training data. However, there are many domains where constructing large amounts of labelled training data is impractical due to the expense of labelling (labelling acquisition bottleneck). Ideally one would want to construct an effective training set composed of a small number of highly informative examples.

Active learning is a machine learning technique that gives the learner control over the training data. Humans interact with the active learning system to label examples chosen by the learner from a source of unlabelled examples. Figure 3.2 depicts an active learner, where the batch process of passive learning is replaced with an interactive process.

The set of training data selected by active learning should not contain redundant or uninformative examples allowing for a reduction in the labelling effort. Ideally, only the minimum number of examples required to induce an accurate classifier should be labelled ($|\mathcal{D}_\ell| = m$). Once the active learning process is finished, the output is the same as passive learning, namely a classifier induced from all the labelled training data.

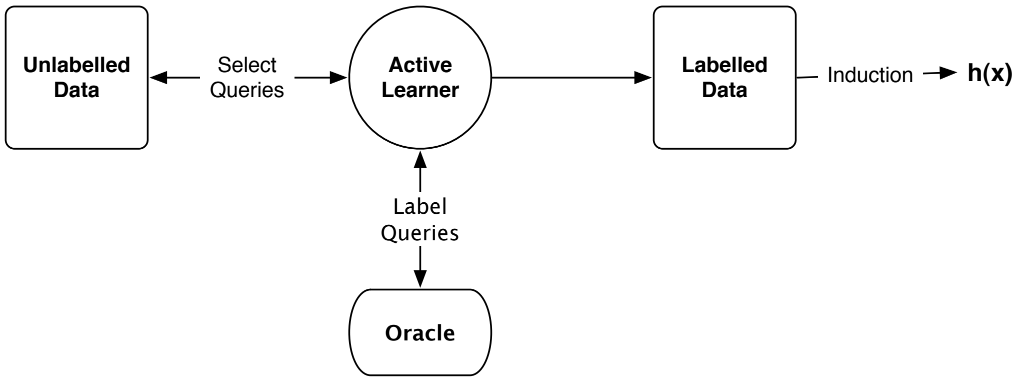


Figure 3.2: Active learning is given control over the training data. It constructs the training data by interacting with an external entity (the oracle) before applying the traditional supervised learning algorithm to construct the classifier.

The process of active learning proceeds as follows. An active learner is given just a small amount of labelled training data but has access to a large amount of unlabelled data. Training data are accumulated iteratively whereby in each iteration, the active learner carefully selects unlabelled examples and acquires their true labels via interaction with an external entity (the oracle). The process is iterated until a stopping criterion is met. Training data produced via active learning are populated with only the informative examples selected by the learner, thereby expediting the learning process and reducing the overall labelling burden.

It is the role of the query selection strategy to identify informative examples from the unlabelled data. Informative examples are often those which are difficult to classify. Challenging examples often contain more information about the target function than easy to classify examples. This is similar to the idea of boosting (Schapire, 1999), which focuses the training data on difficult to classify examples. However, unlike boosting, the label information is not known *a priori* in the active learning setting.

As motivation for active learning, the toy example shown in Figure 3.3 is commonly used (Cohn *et al.*, 1994; Dasgupta, 2005). Given some data that lies on the real line, consider the concept of learning a thresholding function h_w from $H = \{h_w : w \in \Re\}$:

$$h_w(\mathbf{x}) = \begin{cases} 1 & x > w \\ 0 & otherwise \end{cases} \tag{3.3}$$

VC theory states that if the underlying distribution can be separated perfectly by some hypothesis in H (the realisable case) then the number of examples required to construct a classifier with error ϵ is $m = O(1/\epsilon)$. One may simply supply m randomly chosen labelled training examples and return any consistent classifier.

If m unlabelled examples were drawn from P and laid down as shown in Figure 3.3, their hidden labels form a series of 0s followed by 1s. The goal is now to find the point w at which the transition occurs. Active learning, in this case a binary search, can be employed and requires just $m = O(\log 1/\epsilon)$ examples to produce a $\epsilon-$accurate threshold function.

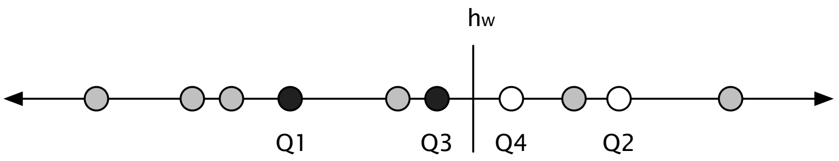


Figure 3.3: Active Learning Toy Example. Starting with all unlabelled examples (grey circles) active learning selects queries using a binary search (Q1-Q4). The true labels are revealed (black circle = 0 and white = 1) and the location of the transition is established using very few labelled examples.

This gives an exponential reduction in the number of labels required. Applying the active learning approach to more challenging concept classes is regrettably not so straightforward. The benefit achieved by employing active learning is dependent on the specific hypothesis class and the domain.

In the ideal case active learning can exponentially reduce the number of labelled training examples required. However, there may be domains, where in order to induce the perfect classifier, given some training sample of size m, all m labels are required.

In its worst case, active learning cannot be worse than randomly labelling examples. In other words, the passive learning setting. Empirical results have found that significant reductions (orders of magnitude) in the number of labelled examples required were achieved when active learning was employed in text categorisation tasks (Lewis & Gale, 1994; McCallum & Nigam, 1998; Tong & Koller, 2001).

3.3.1 Canonical Active Learning Algorithm

The pseudo-code for a canonical active learner is given in Algorithm 1. Active learning is seeded with a very small amount of labelled training data ($\mathcal{D}_\ell$) and a large amount of unlabelled data ($\mathcal{D}_U$).

In each iteration a classifier is induced from all the known labelled training data. While this step is not strictly necessary it is often done to evaluate performance and the classifier is commonly used to help select the query from the unlabelled examples. Queries are chosen according to a query selection strategy and presented to the oracle whereby its true label is obtained. The newly labelled query is then added to the labelled training data.

The amount of labelled data grows as active learning progresses. In this way, queries from previous iterations help select more informative examples in subsequent iterations. These steps are iterated until a stopping criterion is met. Once stopped, a final classifier is induced from all known labelled training data.

Algorithm 1: Canonical Active Learner

```
Data: $\mathcal{D}_\ell$: the labelled data.
Data: $\mathcal{D}_U$: the unlabelled data source.
Result: $\hat{f}$: classifier induced from all training data
repeat
    induce classifier from $\mathcal{D}_\ell$ ;
    select query ($\mathbf{x}$) from $\mathcal{D}_U$ and obtain label ($y$);
    update labelled data $\mathcal{D}_\ell \leftarrow \mathcal{D}_\ell \bigcup \{\mathbf{x}, y\}$ ;
until stop criterion reached ;
```

This canonical active learner forms a basis for the majority of active learning strategies found in the literature. Active learning has been applied in a number of domains giving rise to a wide variety of settings. The most critical component of the active learning algorithm is the query selection strategy which we describe in detail in Section 3.4. In the remainder of this section we focus on the other components and parameters of the active learner.

3.3.2 Source of Unlabelled Examples

One of the main distinguishing factors in the active learning literature is the source of unlabelled examples. Membership queries (Angluin, 1988) *construct* unlabelled examples based on the most uncertain aspects of a learners knowledge.

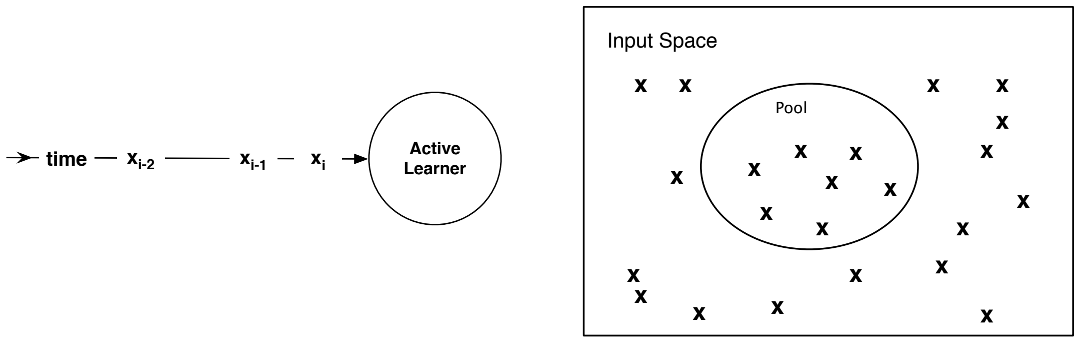
This form of active learning has certain advantages in that informative unlabelled examples arbitrarily close to the decision boundary can be constructed without the learner ever needing to observe such an example in the real world. However, membership queries are less popular in text categorisation tasks since constructed unlabelled examples tend to be difficult (if not impossible) for the oracle to label. This is due to the fact that the oracle is assumed to be human and the constructed examples are often incomprehensible random collections of terms making manual classification impossible (Baum & Lang, 1992).

Rather than constructing artificial examples, an alternative approach is to filter queries from a source of unlabelled examples. Within the active learning literature there are two sources:

Stream-based active learning is where queries are selected from an *infinite* stream of unlabelled examples supplied to the active learner (see Figure 3.4(a)). In each iteration the learner must decide whether to select the unlabelled example as a query or reject it and move on to the next unlabelled example in the stream. This is an on-line decision where the learner only has access to the current unlabelled example, it cannot consider alternatives.

Pool-based active learning is where the learner is given the set of *all* unlabelled examples from which it must select queries (see Figure 3.4(b)). This differs from stream-based active learning since every possible unlabelled example can be evaluated before the query is selected. Pool-based active learning is seen as an off-line problem.

Stream-based active learning has been applied to some problems (Saunier *et al.*, 2004; Dagan & Engelson, 1995). However, pool-based active learning is by far the more popular (Hoi *et al.*, 2006; Lewis & Gale, 1994; Schohn & Cohn, 2000; Tong & Koller, 2001) in active learning settings. Unless stated otherwise, when we refer to active learning in this thesis, we are referring to pool-based active learning.



(a) Stream-Based Active Learning (b) Pool-Based Active Learning

Figure 3.4: Graphical representation of Stream-based and Pool-based active learning. Stream-based active learning is where the learner is presented with an infinite stream of unlabelled examples. Pool-based active learning is where the learner is given a set of all unlabelled examples

3.3.3 Granularity

The canonical active learning algorithm is an iterative process where in each iteration a query is selected. The *granularity* refers to the number of queries chosen in each iteration. A query selection strategy ranks the unlabelled examples such that highly ranked examples are selected as queries. Various levels of granularity can be achieved by selecting more or fewer queries per iteration of active learning. The finest granularity is where just one query is selected - the highest ranked unlabelled example. Lower granularity is achieved by selecting a batch of queries per iteration by simply selecting the top N unlabelled examples as queries.

Finer granularity means that fewer queries will be selected per iteration resulting in maximal learning. However, finer granularity increases computational costs per query selected. To offset the computational expense batches of queries can be selected in each iteration. While the cost per query will decrease, learning may not be maximal since if the batch size is too large there is a possibility for sub-optimal queries to be selected.

Cohn *et al.* (1994) analyse the effect of various batch sizes in active learning. It was found that smaller batch sizes increased the effectiveness of active learning. However, the authors point out that the advantages obtained must be weighed against the increased computational cost incurred with finer granularity. Similarly, Schohn & Cohn (2000) also analysed batch size. Again it was found that smaller batch sizes led to greater improvement in the performance of active learning for small training sets.

Batch Selection

As already mentioned batches can be selected as the top N ranked unlabelled examples as given by some query selection function. A more principled way of constructing the batch of queries is given by Hoi *et al.* (2006), which selects queries such that the redundancy between the unlabelled examples is minimised. The set of queries is chosen such that the Fisher information is maximised. Alternatively, clustering can be utilised to select batches of queries (Xu *et al.*, 2003). Queries correspond to the cluster centres (metoids) obtained by clustering the top ranked examples as given by the query selection strategy.

3.3.4 Stopping Criteria

The canonical active learning algorithm is supplied with a parameter to determine when to stop the iterative process. Deciding when to stop the active learning process is a surprisingly difficult task and remains an open problem. A stopping criterion is dependent on the constraints of the problem being solved. We broadly define two types of stopping criteria, namely *adaptive* and *fixed*. While this is not an exhaustive list the majority of criteria encountered in the literature can be defined by one of the two types.

Fixed

Fixed stopping criteria terminate the active learning process once a resource has been exhausted. By far the most straightforward and popular stopping criterion for active learning is a limit on the number of labels the oracle will supply. Once the oracle resource is exhausted, it will refuse to label any more examples. The active learner must achieve the highest possible accuracy in the limited number of iterations. Similarly exhaustion of alternative resources can also be used to stop active learning. One such resource is the pool of unlabelled examples. Once all unlabelled examples have been queried then active learning stops due to the fact there are no more unlabelled examples.

Adaptive

Adaptive stopping criteria terminate the active learning process as soon as a desired level of performance is achieved. This requires an effective monitoring mechanism for the performance of the active learner. Performance in supervised learning is typically evaluated using a hold-out test set. Given the scarcity of labelled data in an active learning setting, the use of a test set is not applicable.

The performance of an active learner can be calculated by conducting cross-validation on the labelled data. However this is impractical for two reasons. First there is considerable computational overhead associated with this approach. Second, cross-validation is performed on data assumed to be sampled *i.i.d.* according to the underlying distribution. Since the active learner selects the training data, this assumption is violated.

Schohn & Cohn (2000) proposed a self-contained stopping criterion for active learning using a SVM classifier. An assumption is made that the data is linearly separable. In this case active learning is terminated once all unlabelled examples lying within the margin are exhausted. This is determined when the distance from each unlabelled example to the hyperplane is greater than the distance of all the support vectors. Empirical evaluation found this margin-based stopping criterion worked well; stopping active learning slightly after the maximum performance was achieved.

Vlachos (2007) suggest an alternative stopping criterion for active learning using uncertainty sampling. After each iteration of active learning the confidence of the current classifier is calculated using a separate large unannotated dataset, which has undergone the same preprocessing as the training and pool data. Intuitively, as active learning progresses and more training data is available, confidence should increase. Uncertainty sampling (Lewis & Gale, 1994) selects those examples that are difficult to classify and are also dissimilar to those already in the training data. A point will be reached where all the remaining unlabelled examples will have confident predictions. Queries added after this point will not contribute novel information, as they are similar to those already in the training data. Additionally if the labels contradict the predictions this will invalidate generalisations induced from knowledge obtained in previous iterations of active learning leading to confidence remaining unchanged or falling. While the authors show that confidence tracks the performance of the classifier, they do not offer a definitive stopping criterion. Stopping after 3 successive declines in confidence was shown empirically to work well. However, it was left to future work to determine the optimal number of iterations of declining confidence to stop active learning.

3.3.5 Oracle

The canonical active learning algorithm must acquire labels for the selected queries. The oracle is an external entity whose role is to return the true label for queries presented to it. In passive learning the training data is labelled using an oracle prior to use. Conversely, in active learning, the training data is accumulated by iteratively interacting with the oracle.

It is generally assumed that the oracle is human, however this need not be the case. The oracle could be an expensive test, such as a fMRI scan or drilling a test hole in the sea bed. In the setting of text categorisation, however, the oracle is assumed to be human. As mentioned previously, human oracles pose a problem for membership queries as examples presented may not be interpretable (Baum & Lang, 1992).

A further assumption is that the oracle is infallible, that is, no noise is introduced to the learning process from the oracle. This is a strong assumption which is not always valid. The A^2 algorithm (Balcan *et al.*, 2006) is an agnostic active learner which works in the presence of arbitrary forms of noise. The strong assumption of infallibility can therefore be relaxed in this case. In our experiments we do not make the assumption that the oracle is human but we do assume it is infallible.

3.3.6 Summary

By far the most important component in active learning is the query selection strategy, which is responsible for the identification of informative unlabelled examples from the pool. This section summarised the remaining components and parameters of a canonical active learning algorithm. Details were given on the oracle, source of unlabelled examples, granularity and stopping criteria. In the next section we describe in detail the query selection strategy.

3.4 Query Selection

Critical to the success of active learning is the ability to select informative queries from a source of unlabelled examples. Within the literature a number of heuristics have emerged that attempt to identify and select the most informative examples. Classifiers induced from training data constructed using these heuristics have been shown to outperform classifiers induced using standard passive supervised learning.

3.4.1 Random Sampling

As previously noted, training data for supervised learning is constructed in a batch manner by randomly drawing a set of examples independent and identically distributed (*i.i.d.*) from the input distribution $P(\mathbf{x})$ and labelling them. This process is emulated in an active learning setting by using the *random sampling* query selection strategy. Random sampling selects queries at random from the source of unlabelled examples.

In the worst case, active learning should perform at the same level as random sampling. Random sampling is therefore typically used as a baseline from which other query selection strategies are assessed. Queries selected using alternative selection strategies should result in more accurate classifiers.

3.4.2 Furthest-First

Hochbaum & Shmoys (1985) propose the Furthest First (FF) strategy, which selects queries that are most dissimilar to all other known labelled examples. Similarity is defined using a metric such as Euclidean distance. Similar to random sampling, the FF selection strategy can be applied without the construction of a classifier. The input space is maximally explored since in each iteration the furthest unlabelled example is selected as the query. While FF will rapidly explore the input space, convergence to the supervised learning performance will be slow. The algorithm was used in a kernel machine context in (Baram *et al.*, 2004) where distance was defined using squared Euclidean distance.

3.4.3 Uncertainty Reduction

Uncertainty reduction based strategies select queries which are difficult to classify given the current training data. Such examples often reside in regions of the input space for which there is little information known.

Regions of Uncertainty

In (Cohn *et al.*, 1994) a *region of uncertainty* is defined to be the area of the input space that is currently not well defined by the known labelled data. More formally, the region of uncertainty $R(\mathcal{D}_\ell)$ is defined as given in (3.4)

$$R(\mathcal{D}_\ell) = \{\mathbf{x} : \exists_{h_i, h_j \in H} \cdot h_i(\mathbf{x}) \neq h_j(\mathbf{x})\} \tag{3.4}$$

where h_i and h_j are two hypotheses both consistent with the training data. Selecting unlabelled examples from outside the region as queries will not benefit learning since that area is already well defined by the known labelled examples. Queries selected from within the region of uncertainty will increase the knowledge of the domain and further restrict the region.

Explicit calculation of the region of uncertainty is an expensive and difficult task for all but the most trivial problems. One way to approximate sampling from the region of uncertainty is to sample unlabelled examples close to the decision boundary. Uncertainty is defined as the ambiguity surrounding the predicted class for an unlabelled example. Examples close to the decision boundary are by definition difficult to classify therefore have high uncertainty.

Uncertainty Sampling

Lewis & Gale (1994) proposed uncertainty sampling (US) which was the first selective sampling technique for a single classifier active learning. US is a straightforward query selection strategy which can be applied to any classification algorithm. The only prerequisite is that the classifier provide some measure of confidence about its predictions. It was demonstrated using a probabilistic classifier for a binary classification problem, where the probability of class membership was used as a metric of confidence for the prediction. Predictions close to 0 or 1 were considered confident while predictions close to 0.5 were considered uncertain. Queries are selected as those most uncertain unlabelled examples, that is, those with predictions closest to 0.5 as shown in (3.5)

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{D}_U} |\hat{P}(y|\mathbf{x}) - 0.5| \tag{3.5}$$

where $h(\mathbf{x}) = \hat{P}(y|\mathbf{x})$ is a probabilistic classifier trained using all known examples of the target function $P(y|\mathbf{x})$. Predictions close to 0.5 represent those unlabelled examples close to the decision boundary. These examples correspond to sampling from the region of uncertainty.

The idea of uncertainty sampling was later extended for use with Support Vector Machines (Schohn & Cohn, 2000; Campbell *et al.*, 2000) where these query selection strategies attempt to maximally narrow the existing margin. Uncertainty was defined as proximity to the decision hyperplane (see Section 2.6.3 for more information on SVM hyperplanes) and the region of uncertainty corresponds to the margin of the SVM. Selecting queries from outside the margin will have little effect on the classifier while selecting queries from within the margin will cause maximal change in the hyperplane. Evaluating distance to the hyperplane is an inexpensive procedure requiring just one dot product calculation. Uncertainty based on distance to the hyperplane was also used in (Tong & Koller, 2001) which was referred to as SIMPLE. Empirical evaluation on real world text corpora showed that improvements were consistently made over random sampling.

Despite its simplicity, uncertainty sampling has demonstrated strong performance in text categorisation tasks (Lewis & Gale, 1994; Tong & Koller, 2001). Throughout this thesis uncertainty sampling is used as a baseline from which other query selection strategies are measured.

3.4.4 Version Space Reduction

A theoretically motivated way to select examples is based upon maximally reducing the size of the version space (H) (Mitchell, 1997). The version space is comprised of all hypotheses consistent with the known labelled training data. Assuming the target hypothesis is present in the version space, the goal of supervised learning is to identify the target hypothesis from H.

Active learning reduces the *sample complexity* for the supervised learning task. Selecting queries which maximally reduce the size of the version space is advantageous since each query will restrict the possible space in which the target hypothesis can lie. Ultimately the target hypothesis can be found using exponentially fewer labelled examples (Seung *et al.*, 1992; Freund *et al.*, 1997) by selecting queries that reduce maximally the size of the version space. Although similar to the idea of uncertainty reduction, version space reduction is theoretically proven while uncertainty reduction is mostly empirically driven.

Motivation

A greedy active learning strategy was proposed in (Dasgupta, 2005) and is shown to be as good at minimising the sample complexity as *any* other strategy. More formally the greedy strategy will select queries that cause maximal disagreement within the version space. Given an unlabelled example $\mathbf{x}$, let C^+ be the number of hypotheses in H that predict a positive label and C^- be the number of hypotheses in H that predict a negative label. Queries are selected such that $\min(C^+, C^-)$ is largest.

For most, non-trivial problems the geometry of the version space is complex, making the area of $C^\pm$ impractical to explicitly calculate. A number of strategies have emerged in the literature which attempt to select those queries which will maximally reduce the version space using various approximations.

QBC

The Query-By-Committee (QBC) selection strategy (Seung *et al.*, 1992; Freund *et al.*, 1997) attempts to find queries which reduce the version space by utilising the principle of maximal disagreement. A committee of classifiers is constructed using the (zero temperature) Gibbs algorithm, which selects hypotheses at random from the version space. Predictions are obtained from both hypotheses for all unlabelled examples. If the predictions from the committee members disagree for a particular unlabelled example then it is selected as a query, otherwise it is rejected.

The QBC query selection strategy has been applied to a variety of inductive learning algorithms including perceptrons (Freund *et al.*, 1997) and naïve Bayes (McCallum & Nigam, 1998). QBC is commonly used in stream-based active learning.

SG-Net

SG-Net (Cohn *et al.*, 1994) reduces the version space by selecting queries from the region of uncertainty, which is defined by two subsets of the version space. Within the version space (H) there exists a partial ordering in the generality of the hypotheses. Two subsets of the version space are maintained, namely $S \subset H$ which contains the *most specific* consistent hypotheses and $G \subset H$ which contains the *most general* hypotheses.

Queries are selected from the region defined by the intersection of S and G. If the query's true label is positive then hypotheses in S will become inconsistent and be removed from S, conversely if the true label is negative then some hypotheses in G will become inconsistent and be removed from the version space. In either case, the version space is reduced with every query labelled.

Unlike QBC, SG-Net is only applicable to relatively simple learning problems, since as the complexity increases so does the difficulty in computing and maintaining approximations of the version space.

Single Classifier Solution

Strategies discussed so far have utilised committees in order to find queries that minimise the version space. A single classifier solution was developed (Tong & Koller, 2001), which utilises the duality between the feature space (X) and the parameter space ($\mathcal{W}$) of a Support Vector Machine. By definition, points in parameter space correspond to hyperplanes in feature space. Similarly, points in feature space correspond to hyperplanes in parameter space, where observing a labelled example $\mathbf{x}$ in feature space restricts the set of hyperplanes to those which correctly classify $\mathbf{x}$.

Queries should be selected so that the version space is reduced as much as possible. This corresponds to queries that bisect the version space, that is, half of the hypotheses predict one label while the remainder predict the opposite label. Tong & Koller (2001) propose three strategies:

SIMPLE which is based on (Campbell *et al.*, 2000; Schohn & Cohn, 2000) selects the unlabelled example closest to the hyperplane. The motivation for this strategy is that a SVM corresponds to the largest radius hypersphere that can fit inside the current version space, where the unit vector $\mathbf{w}$ is the centre of the hypersphere in parameter space. Selecting examples in X close to the hyperplane corresponds to selecting the hyperplane in $\mathcal{W}$ that is closest to $\mathbf{w}$, thereby bisecting the version space and maximally reducing its size. The SIMPLE strategy assumes the version space is symmetrical. However, this is often not the case.

MinMax was proposed to deal with the relatively strong assumption of SIMPLE and allows for non-symmetrical versions spaces. Similar to ERS, MinMax is an exhaustive process where each unlabelled example $\mathbf{x}$ is tentatively labelled with every possible label and a SVM is induced from $\mathcal{D}_\ell \bigcup \{\mathbf{x}, y\}$.

The size margin of the resulting SVM is proportional to the radius of the hypersphere in the version space. Dealing with a binary classification problem, two SVMs will be constructed per unlabelled example ($\mathbf{x}$), with $margin^+$ being the size of the margin produced by labelling $\{\mathbf{x}, +1\}$. Similarly, $margin^-$ being the size of the margin produced by labelling $\{\mathbf{x}, -1\}$.

Queries are selected as those examples which most equally divide the version space, that is, result in similarly sized hyperspheres. MinMax selects the unlabelled example for which $\min(margin^+, margin^-)$ is largest.

RatioMargin is a further adaption of MinMax where the possibility of an elongated version space is taken into consideration. It examines the relative sizes of the margins and chooses the query as the unlabelled example for which $\min(\frac{margin^-}{margin^+}, \frac{margin^+}{margin^-})$ is largest.

Both MinMax and RatioMargin offer improvement over SIMPLE during the early iterations of active learning but are computationally far more expensive to run.

3.4.5 Incorporating Unlabelled Data

Query selection strategies described thus far have based the selection of queries only on the information contained in the labelled training data. Potentially a lot of information is contained in the large number of unlabelled examples at the disposal of the active learning. In this section we discuss query selection strategies which utilise unlabelled data information to select informative unlabelled examples.

Stream-based active learning implicitly incorporates density as unlabelled examples are drawn according to the underlying distribution $P(\mathbf{x})$. However, query selection decisions must be made on-line, not allowing for the consideration of alternative unlabelled examples. Conversely pool-based active learning allows all unlabelled examples to be considered but the implicit modelling of $P(\mathbf{x})$ is lost.

Clustering

Clustering the unlabelled data provides information about the underlying distribution of the data $P(\mathbf{x})$. This information can be incorporated into query selection strategies so that queries are chosen as those unlabelled examples for which there is high uncertainty and also are representative of many other unlabelled examples. Selecting queries from dense regions of the input space mitigates against the possibility of selecting *outliers* (uncertain examples are found in sparse and uninteresting regions of the input space). Outliers offer little to no information about the target concept and waste valuable labelling effort. Furthermore, the true decision boundary should reside in low density regions (Chapelle *et al.*, 2003).

Representative sampling (Xu *et al.*, 2003) incorporates clustering with uncertainty sampling to select the *most important uncertain* queries. In each iteration of active learning, the k-means algorithm (Forgy, 1965) is used to cluster those unlabelled examples that lie within the margin of the SVM. Queries are chosen to be the unlabelled examples closest to the centroids. Empirical evaluations demonstrate improvements during the early iterations of active learning while uncertainty sampling outperforms in later iterations. As active learning progresses, the quality and quantity of training data increases resulting higher accuracy classifiers induced in each iteration. A high accuracy SVM can better separate the data, meaning fewer unlabelled examples within the margin. Clustering with few examples will not be capable of providing a good estimate of the underlying data distribution.

McCallum & Nigam (1998) extend the QBC query selection strategy to incorporate unlabelled data. First it is used to provide an approximation of density $P(\mathbf{x})$ by measuring the average distance from an example to all other examples. Second, a committee of naïve Bayes classifiers are formed by sampling according to the classifier parameter distribution specified by the training data. Since active learning is seeded with a very small amount of training data, the parameter estimates can be poor leading to poor classification accuracy. Expectation-Maximisation (EM) is applied to the unlabelled examples to produce better estimates for the naïve Bayes classifier.

Pool-leveraged active learning runs EM to convergence on each committee member before QBC is used to select the query. Density weighted pool-based sampling selects queries as the unlabelled example with the largest product of density and disagreement among the committee. In this way queries are those unlabelled examples which are most uncertain and also representative of many other unlabelled examples.

Nguyen & Smeulders (2004) propose a probabilistic framework that incorporates knowledge of the density of a region $P(\mathbf{x})$ into the query selection strategy. The motivation is that examples in the centre of clusters are more informative and should be selected first. Furthermore, samples from the sample cluster are likely to have the same label hence the number of samples taken from the same cluster should be reduced.

Clustering information is used to approximate $P(\mathbf{x})$ and is incorporated directly into a formal model. The model is trained using logistic regression trained on the set of cluster representatives. A Gaussian noise model is then used to infer the class label for non-representative examples. The model allows for representative queries to be selected from the pool while also avoiding repeated sampling from the same cluster. Queries selected are those unlabelled examples which are in proximity to the decision boundary but also reside in regions of high density.

A drawback of employing clustering algorithms such as k-means is that extra parameters need to be set. The success of k-means depends directly on the correct value of k being supplied. In an active learning setting where labelled data is scarce it is unclear how good values for parameters can be found given that validation sets are not available.

Expected-Error Reduction

An alternative method to utilise unlabelled data to select queries is to examine the effect on the remainder of the unlabelled data. Chapter 7 examines techniques in which to increase the quality of examples selected by Error-Reduction Sampling (ERS) (Roy & McCallum, 2001).

Cohn *et al.* (1996) proposes an ideal query selection strategy, namely one which selects queries that, once labelled and incorporated into the training data, will result in lower future error on text examples. Using an appropriate loss function (L) the expected error of the learner is calculated as shown in (3.6).

$$E_{\hat{P}_{\mathcal{D}_\ell}} = \int_{\mathbf{x}} L(P(y|\mathbf{x}), \hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x}))P(\mathbf{x}) \tag{3.6}$$

where $P(y|\mathbf{x})$ is the output of the target function while $\hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x})$ is the output of the hypothesis trained from all known labelled examples.

Unfortunately, closed form calculation of expected future error is intractable for most problems. ERS constructs an estimate of the expected future error using the unlabelled data. ERS selects queries as those unlabelled examples that generates the lowest expected error on the remainder of the unlabelled pool. The expected error on the remainder of the unlabelled examples is calculated as shown in (3.7) using the entropy of the posterior class distribution on a sample of the unlabelled examples and a log-loss function.

$$\tilde{E}_{\hat{P}_{\mathcal{D}^*_\ell}} = \frac{1}{|\mathcal{D}_u|} \sum_{x \in \mathcal{D}_u} \sum_{y \in \mathcal{Y}} \hat{P}_{\mathcal{D}^*_\ell}(y|x) \log(\hat{P}_{\mathcal{D}^*_\ell}(y|x)) \tag{3.7}$$

The true label (y) for unlabelled examples is not available therefore the expected error must be estimated using every possible label. The prediction from the current classifier is used to give a weighted average over all the possible labels as shown in (3.8):

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{D}_u} \sum_{y \in \mathcal{Y}} \hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x})\ \tilde{E}_{P_{\mathcal{D}_\ell \bigcup \{\mathbf{x}, y\}}} \tag{3.8}$$

In contrast to alternative selection strategies which consider unlabelled examples in isolation, ERS selects queries based on the effect their labelling will have on the remainder of the unlabelled examples. Queries selected by ERS are those examples which maximally reinforce the existing belief of the learner.

A related technique is that of look-ahead query selection (Lindenbaum *et al.*, 1999) which estimates the *utility* of knowing the true label for an unlabelled example for the k-Nearest

Neighbour algorithm. Queries are selected as the unlabelled example which has the highest utility. This strategy is similar to ERS where the log-loss function in (3.7) is replaced with a 0/1 loss function.

Guo & Greiner (2007) proposed MCMI[min] which is similar to ERS except rather than take the weighted average as done in (3.8) the y value which cause the lowest entropy in the posterior predictions is used.

3.4.6 Multiple Strategies

Query selection strategies are generally fixed and do not change over the life-cycle of the active learner. A strategy will generally either *explore* or *exploit*. Exploration-based strategies, such as random sampling or furthest-first, discover new information by traversing previously unexplored regions of the input space. Conversely, exploitation-based strategies, such as uncertainty sampling, leverage the learners internal knowledge to focus sampling from confined regions of the input space. Neither exploration- nor exploitation- based strategies dominate in active learning.

Empirical studies (Donmez *et al.*, 2007; Xu *et al.*, 2003) found that exploration is desirable in the early iterations of active learning while exploitation is optimal in the later iterations. This motivates the need for an adaptive query selection strategy that can balance exploration and exploitation in order to increase the performance and stability of active learning.

Recently online active learning algorithms (Baram *et al.*, 2004; Donmez *et al.*, 2007; Osugi *et al.*, 2005; Pandey *et al.*, 2005) have emerged to dynamically switch between a number of different query selection strategies in each iteration. These online strategies adaptively choose which query selection strategy to use based on their relative performance. In this way, a particular query selection strategy is applied during the iterations where it is most useful. Combining diverse strategies allows for greater flexibility in the selection of queries and potentially a more robust selection strategy across multiple datasets. Managing multiple strategies then becomes the problem of balancing exploration and exploitation in order to achieve the highest performance possible.

Combining Strategies

In this section we discuss ways that have emerged to combine strategies. Typically two alternative query selection strategies are combined. Methods for combining many query selection strategies are discussed later.

A straightforward method of combining two query selection strategies is presented in (Osugi *et al.*, 2005). Here the choice between using the explorative KFF (Baram *et al.*, 2004, Section 4.3) or using the exploitative SIMPLE (Tong & Koller, 2001) query selection strategy is decided at random. A biased coin is flipped in each iteration of active learning and exploration-based query selection is performed if the result is heads otherwise exploitation-based query selection is performed.

The probability associated with performing an exploration-based query selection is based upon how successful the previous exploration queries were. Success is defined to be a change in the classifier induced after the query has been incorporated. This is measured using the predicted labels (V) of the unlabelled data prior to the query and the predicted labels (V') obtained once the query was incorporated. It is calculated as the cosine of the angle between the two vectors as shown in (3.9):

$$\frac{\langle V \cdot V' \rangle}{||V||\ ||V'||} \tag{3.9}$$

Intuitively exploration should cause a significant change in the classifier as previously unseen regions of the domain are encountered. If exploration does not lead to a change in the classifier, then it is deemed unsuccessful and the chances of exploration on the next iteration of active learning is reduced. Empirical evaluation of the above algorithm was conducted on both artificial and real world data. Results show that the adaptive algorithm either outperforms or keeps track with pure query selection strategies. The algorithm favours exploration in early iterations before tending towards exploitation in later iterations.

Guo & Greiner (2007) proposes Mm+M which combines two query selection strategies, namely MCMI[min] and uncertainty sampling. During the MCMI[min] strategy an optimistic guess is made on the label of y for the query example. The true label is revealed by the oracle in the next iteration. If the optimistic guess is incorrect, uncertainty sampling us used to select the query otherwise MCMI[min] continues to select the query. Mm+M was reported to be statically significantly better on 12 of 17 UCI dataset it was evaluated on.

Donmez *et al.* (2007) proposed dual strategy active learning (DUAL) which combines two query selection strategies, namely density-weighted uncertainty sampling (DWUS) as proposed in (Nguyen & Smeulders, 2004) with uncertainty sampling. Xu *et al.* (2003) previously found that density-based query selection strategies perform well in the early iterations of active learning while uncertainty-based strategies performed better in the later iterations.

DWUS favours those unlabelled examples with lower uncertainty but with higher information (due to density). This is advantageous in the early iterations where many unlabelled examples are highly uncertain. However, in later iterations informative queries reside in low-density regions. Since DWUS favours queries from dense regions unlabelled examples with high uncertainty are not selected, decreasing performance.

The primary objective of DUAL is to achieve better performance in the later iterations of active learning by finding the optimal point at which to switch from the density-based to uncertainty-based query selection. The *cross-over* point is found by predicting the low derivative of the expected error as shown in (3.10):

$$\frac{\partial_\epsilon(DWUS)}{\partial x_t} \leq \delta \tag{3.10}$$

where the expected error is calculated as the average expected error of DWUS on the unlabelled data. Monitoring the average expected error allows for an estimate of when performance saturates. In this case it is when it falls below δ which is a small fixed, user defined, value. When the cross-over point is reached DUAL uses both DWUS and US to select queries. Unlike the hybrid method in (Xu *et al.*, 2003) DUAL determines dynamically when to switch over to uncertainty sampling. Empirically DUAL was shown to be better than each of the sampling strategies used in isolation.

Multi-Armed Bandit

In the multi-armed bandit (MAB) problem a gambler is presented with k non-identical slot machines[1] and must choose one machine to play. This decision is repeated over a number of trials T and the objective of the gambler is to maximise their total reward.

At the heart of the MAB problem is a trade-off between exploration and exploitation: choosing to always play a single machine can prevent discovery of a better machine and choosing to play all machines can prevent maximising reward. Combining several query selection strategies can be formulated as a MAB problem. A number of existing reinforcement learning algorithms that solve the MAB problems can then be applied.

One way to combine strategies is to formulate the task as an opaque Multi-Armed Bandit (MAB) (Vermorel & Mohri, 2005) problem, where the learner must choose among several *machines* to play, with the goal of maximising the overall *reward*. A number of trials are conducted which give a finite horizon.

[1]Sometimes this is stated as one slot machine with several arms but it is conceptually easier to consider several one-armed bandit machines

In an active learning setting the learner must choose to play one of the k machines (query selection strategies) in each iteration. Playing a machine represented selecting a query using the chosen query selection strategy. After playing the machine the reward is calculated. Ideally the reward should be the true accuracy of the classifier induced from all labelled data. However, this information is not available so estimates must be used.

Pandey *et al.* (2005) proposes a technique of combining two query selection strategies using an opaque MAB. The *follow the perturbed leader* (FPL) algorithm was used to choose amongst the k slot machines (query selection strategies). FPL chooses the slot machine to play based on the performance of each machine when it was played in the past. The machine which has performed best in the past is chosen. A perturbation is added to allow for exploration.

In each iteration, the learner must choose between an explorative strategy (Random Sampling) and an exploitative strategy (Uncertainty Sampling). In this so called *hard* choice the learner must play only one query selection strategy. Soft choice allowed the learner to use both strategies but limits the number of query examples each strategy can select. Reward is the expected error, calculated using cross validation on the training data [2].

Empirical results demonstrated an increase in the learning rate of active learning when the combined strategy was used. In addition, combining the two query selection strategies using both hard and soft choice techniques increased performance of active learning. The lift in performance was attributed to the learner favouring the more optimal query selection strategy in different stages of active learning (i.e. random exploration in the early iterations while exploitation in the later iterations). While this strategy combines just two strategies, the use of MAB allows for more strategies to be combined.

Baram *et al.* (2004) propose COMB, which combines three different query selection strategies. However, active learning is formulated as a problem of prediction with expert advice (PEA). In this setting the goal is to utilise and combine k *experts* in order to maximise reward. Each expert will give *advice* on how to play the n slot machines.

In active learning the k experts correspond to the query selection strategies while the n slot machines are the unlabelled pool of examples. In each iteration the advice from the experts is combined and the slot machine to play is chosen. COMB utilises the EXP4 (Auer *et al.*, 2003; Vermorel & Mohri, 2005) algorithm which combines the advice of the experts to select the next slot machine to play.

[2]this strategy is not ideal given the small amounts of training data

The query selection strategies combined are: SIMPLE which is an uncertainty sampling strategy as described in (Tong & Koller, 2001), ERS which is the strategy described in (Roy & McCallum, 2001) but adapted to work with a SVM and KFF which is a furthest first strategy that is designed to work in the kernel-defined feature space. Both SIMPLE and ERS are exploitation-based strategies since they select the next query example based on the current hypothesis. KFF is an exploration-based strategy, which seeks to find the most dissimilar examples to those already labelled.

Each of the three query selection strategies provides advice on which slot machine to play next, essentially ranking the machines. The three 'ranked' advice vectors are combined to construct an advice probability vector from which the slot machine to play is chosen. The slot machine with the highest advice probability is greedily selected in each iteration to be played. As previously stated each slot machine corresponds to an unlabelled example and the query corresponds to the chosen slot machine. Classification entropy maximisation was used as the reward.

Empirically evaluations of COMB on a number of dataset show that its performance is consistently strong. The algorithm generally always either tracked the performance of the best strategy in hindsight or outperformed it.

The performance of both approaches using MAB algorithms to incorporate more than one query selection function to active learning (Baram *et al.*, 2004; Pandey *et al.*, 2005) demonstrates that performance increase can be achieved if exploration and exploitation are carefully balanced.

3.4.7 Summary

Query selection is critical to the success of active learning since it must identify the informative unlabelled examples. In the latter chapters of this thesis we investigate novel query selection strategies which can identify more informative examples, thereby reducing the labelling effort and ultimately the cost of applying machine learning solutions. Furthermore, we investigate techniques to reduce the computational expense of existing query selection strategies, thereby reducing the computational overhead of active learning and lowering the cost of deploying machine learning solutions.

3.5 Evaluation

The goal of active learning is to induce an accurate classifier using the minimal number of labelled training examples. It is therefore desirable to measure the performance of active learners allowing for comparisons to be made between various competing strategies.

The performance of an induced classifier is dependent on the training data supplied to it. Alternative query selection strategies will result in the construction of different training sets. Various query selection strategies can be compared by examining the performance of the classifiers induced from the training data selected by each of the strategies.

Performance in supervised learning tasks is typically measured by means of a hold-out test set (see Section 2.8). Most active learning research is conducted using a pre-labelled corpus, hence a hold-out test set can be used to track the performance of the various active learning strategies. We discuss the techniques and performance metrics commonly reported in the literature.

3.5.1 Fixed Intervals

The performance of alternative active learning strategies can be reported by comparing performance values obtained from the various query selection strategies after a predetermined number of iterations. The classifier induced in the i^{th} iteration is evaluated on a hold-out test set using an appropriate performance metric. In the most extreme case, performance is only evaluated once. This form of evaluation is computationally inexpensive since it is only performed after a number of iterations of active learning. This style of evaluation only records a snap-shot of the performance of the active learning.

3.5.2 Learning Curves

A straightforward way in which to display the performance of active learning throughout its life cycle is to evaluate the performance of the classifier induced in each iteration. A classifier is induced from all the known labelled data and evaluated against a hold-out test set in each iteration. A series of performance values are obtained, one for each iteration of active learning. Plotting these values gives the learning curve for the particular query selection strategy (as shown in Figure 3.5). Alternative query selection strategies will result in different queries being selected and thereby producing a different learning curve.

The top-line performance of active learning is to achieve the same performance as su-

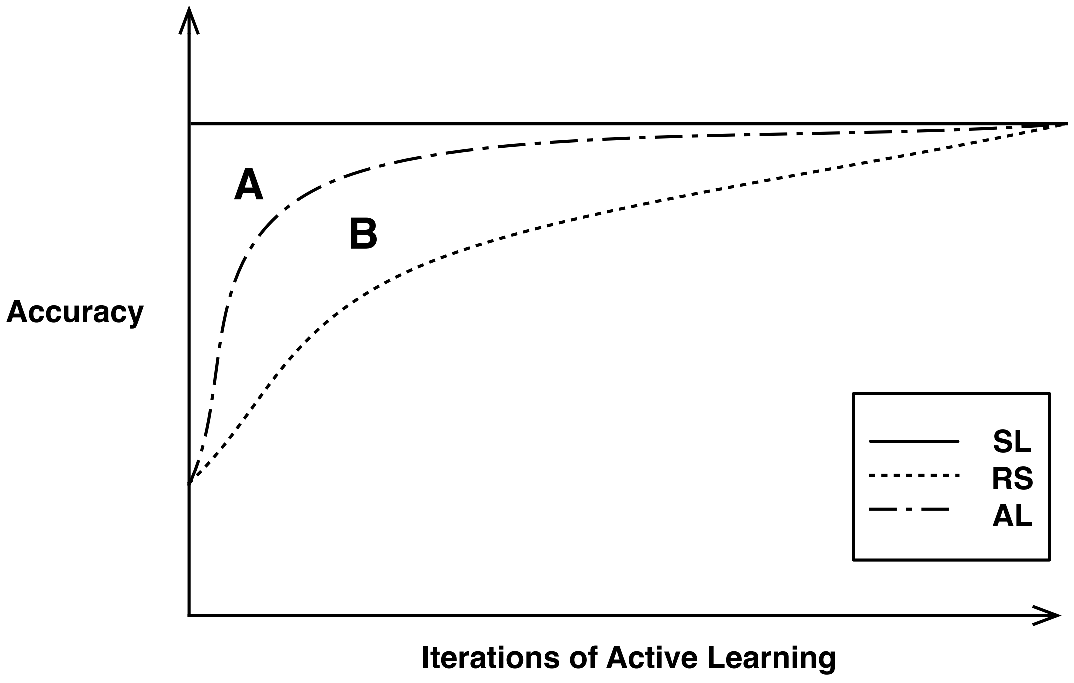


Figure 3.5: Learning Curves for active learning. The plot shows the top-line performance obtained by supervised learning (SL) as well as the baseline performance of random sampling (RS). The active learner should outperform the baseline and approaching the top-line performance using the minimal number of queries

pervised learning when all unlabelled examples are labelled. The baseline performance is random sampling. Active learning should approach the top-line performance using the minimum number of queries. Furthermore the active learner should also outperform the baseline.

Learning curves are an easy way to report the performance of various query selection strategies. The success of strategies can be shown visually by plotting their learning curves. *Ceteris paribus*, the optimal strategy corresponds to the highest performing learning curve.

3.5.3 Deficiency

Deficiency (Baram *et al.*, 2004) is a global performance metric for active learning and measures the ratio between area A and B of Figure 3.5. The area A corresponds to the deficiency between the top-line supervised learning accuracy (trained on all examples in the pool) and the active learner. Area B corresponds to the deficiency between the top-line supervised learner (trained on all examples in the pool) and random sampling. The deficiency value

($D_A(AL)$) is calculated as shown in (3.11)

$$D_A(AL) = \frac{\sum_{i=1}^{T} A(SL) - A_i(AL)}{\sum_{i=1}^{T} A(SL) - A_i(RS)} \tag{3.11}$$

where T is the number of iterations of active learning, $A(SL)$ is the accuracy achieved by passive learning using the entire pool and $A_i(AL)$ is the accuracy achieved by the active learner after the i^{th} iteration. Similarly $A_i(RS)$ is the accuracy achieved by random sampling after the i^{th} iteration. While accuracy is used in the calculation of deficiency, alternative performance metrics can be substituted in (e.g. F_1) when appropriate.

3.5.4 *p*-value Plot

Proposed by Becker (2008) the plots are formed using the p-value obtained from performing a paired t-test at every iteration of active learning. The plot shows at which iterations each strategy is statistically significantly better than random sampling.

3.5.5 Confidence Intervals

Confidence Intervals (Culver *et al.*, 2006) are a method of combining the p-values obtained from performing one tail paired t-tests in every iteration of active learning into a single value. This allows for easier comparison between alternative strategies to be made.

Statistical significance is established between two alternative strategies by taking the median confidence level at which they differ across all iterations of active learning. At each iteration of active learning the maximum confidence level at which the difference between two alternative strategies is calculated. A number of confidence level thresholds $\{0.6, 0.7, 0.8, 0.9, 0.95, 0.975, 0.99, 0.995\}$ are considered. For example, if the null hypothesis is rejected at $\alpha = 0.05$ then the confidence level for that iteration is 0.95. One tailed paired t-test have a null hypothesis $H_0 : A > B$. Positive values correspond to iterations where strategy A is significantly better than B, similarly negative values correspond to iterations where strategy B is significantly better.

A series of confidence level values (t_i), one for each iteration, is returned where the values correspond to the confidence threshold (e.g. $\{t_1, t_2, \ldots, t_m\}$). The median confidence level across all iterations of active learning is used to establish significance. Using the median rather than the mean offers greater protection from outliers and also requires at least half of the iterations to be significantly better in order to be considered significantly better overall.

3.5.6 Semi-supervised performance metrics

The scarcity of labelled data in active learning problems means that supervised learning performance measurement techniques, in particular hold-out test sets, cannot be readily applied. The performance metrics discussed above can be used in conjunction with cross-validation on the labelled data. However as pointed out by (Schohn & Cohn, 2000; Baram *et al.*, 2004) unless the labelled data is sufficiently large, cross-validation will provide substantially biased estimates. This is due to the fact that the training data selected by active learning is not *i.i.d.* since the query selection strategy will favour difficult to classify examples.

Classification Entropy Maximisation (CEM) (Baram *et al.*, 2004, Section 6) is a *semi-supervised* performance metric. It is defined as the binary entropy of the classification predictions given by the most current classifier over the set of unlabelled examples and is calculated as shown in (3.12)

$$CEM(\hat{P}_{\mathcal{D}_\ell}) = \mathbf{H}(\frac{|\hat{P}_{\mathcal{D}_\ell}(\mathcal{D}_u) = +|}{|\mathcal{D}_u|}) \tag{3.12}$$

where $|\hat{P}_{\mathcal{D}_\ell}(\mathcal{D}_u) = +|$ is the number of positive predictions given by the classifier when run on the unlabelled data and $\mathbf{H}$ is the binary entropy[3]. CEM gives higher values to a more even distribution of labels and was seen to closely resemble the true error estimate.

3.5.7 Summary

Results reported in the empirical evaluations in this thesis use one or more of the metrics described here. Where possible we use multiple metrics to ensure the performance of active learning is accurately and comprehensively reported.

3.6 Summary

This chapter reviewed the active learning process. Motivated by scarcity of labelled training data, active learning has been shown to construct accurate classifiers from very small amounts of training data. The active learning process was described in detail with a canonical process. A large number of parameters are used to control the active learning process, each of which were discussed.

We reviewed the query selection strategy arguably the critical component of an active learning system. It is responsible for selecting informative examples from the unlabelled

[3]The binary entropy of a random variable with bias p is: $\mathbf{H}(p) = \mathbf{H}(1-p) = -p\log(p) - (1-p)\log(1-p)$

pool and ultimately determines the success of active learning. Query selection is an active research topic and as such a wide variety of strategies have emerged. We highlighted a number of heuristics used to select queries.

The goal of the research in this thesis is to reduce the labelling effort. The dual problem is increasing the learning rate of active learning. In the latter chapters of this thesis we empirically evaluate novel query selection strategies which aim to select more informative examples in an efficient and practical manner.

Chapter 4

Methodology and Baseline Analysis

4.1 Introduction

In this chapter we describe the experimental methodology used in the empirical evaluations with the goal of providing sufficient information to aid reproducibility of the results reported in this thesis. We begin by describing the real world text corpora and partitions of the data used in the empirical evaluation. Details on pre-processing, classification induction algorithms and evaluation metrics (reviewed in Chapter 2) are given.

Performing active learning for text categorisation problems in real world scenarios presents a bewildering number of settings and parameters. Chapter 3 reviewed the components of active learning and the parameters which are used to control the process. Active learning parameters used in the empirical evaluation in latter chapters are given.

Baseline analysis is performed to demonstrate not only the benefit of active learning over passive learning but also to highlight the active learning process used in the empirical evaluations in the latter chapters of this thesis. Despite its simplicity uncertainty sampling (US) has demonstrated strong performance in text categorisation tasks (Lewis & Gale, 1994; Tong & Koller, 2001). Throughout this thesis uncertainty sampling is used as a baseline from which other query selection strategies are measured.

4.2 Datasets

Experiments were conducted on two well-established, publicly available benchmark text categorisation corpora. Where possible we used standardised *splits* and techniques to partition the corpora into training and testing sets. Below we describe each corpus, giving details on the pre-processing performed.

4.2.1 R10

The Reuters-21578 is a collection of 21,578 news articles collected from the Reuters newswire in 1987. There are 135 categories into which the articles are classified. Multiple labels are possible whereby one article can be assigned to many categories. Additionally, some articles are not assigned to any category. Some 8,676 articles had not been manually indexed and are usually removed from the corpus, leaving 12,902 articles.

Debole & Sebastiani (2005) examine the R10, a subset of the Reuters-21578 that considers only the top ten categories ranked by the number of positive training examples. Table 4.1 shows the number of each category found in the training and testing sets. The distribution of categories is imbalanced making the task of automated text categorisation challenging.

Table 4.1: R10 category distribution

Category	Train	Test	Rank
Acq	1650	719	2
Corn	181	56	10
Crude	389	189	5
Earn	2877	1087	1
Grain	433	149	4
Interest	347	131	7
Money-Fx	538	179	3
Ship	197	89	9
Trade	369	117	6
Wheat	212	71	8

Pre-processing

Documents were parsed into the vector space model (see Section 2.4). Individual words formed the features used, where the total number of features was the lexicon of all unique words in the training set. Stopwords were removed and stemming was applied in order to help reduce the dimensionality of the resulting vector space model. Term weighting (see Section 2.4.2) ($tf * idf$) was applied, where the idf component was computed using only the training data for empirical soundness. Feature selection was not applied.

Partition

Standardised splits of the data have been used in the text categorisation literature in order to help reproducibility and allowing for direct comparisons to be made. The most common is the ModApté split (Apté *et al.*, 1994) which divides the corpus into a set of 9,603 training examples and a set of 3,299 testing examples. The division is done based on chronological order, with test examples being newer than training examples. This mirrors real world deployment where the classifier would have to deal with new previously unseen documents.

Pool

To reduce computational cost, a subset of the training examples are used as the unlabelled pool for active learning experiments. Figure 4.1 shows the methodology used in the experiments which closely resembles that of (Tong & Koller, 2001). In each trial of an experiment, a pool of 1,000 examples was formed by randomly selecting examples from the 9,603 training data. Seed examples are drawn at random from the unlabelled pool and labelled. The pool size at the start of active learning is therefore 1,000 minus the number of seed examples supplied.

In contrast to passive supervised learning, pool-based active learning considers the training data to consist of a pool of unlabelled examples. A separate set of training data exists which initially contains just the labelled seed examples - which are chosen at random from the pool. Queries selected from the pool are placed in the training data once labelled by the oracle.

4.2.2 20NG-4

The 20 Newsgroups corpus[1] consists of approximately 20,000 newsgroup articles collected from 20 Usenet topics by Ken Lang. The dataset is a popular benchmark corpus used in a variety machine learning tasks including text categorisation and clustering. Topics range from sports to atheism.

The 20NG-4 is a subset of the 20 newsgroups dataset, consisting of four binary classification problems of varying difficulty were used in the experiments. Table 4.2 describes the four sub-problems which were previously used in active learning experiments (Schohn & Cohn, 2000). This dataset was used extensively for empirical evaluation in this thesis due to its size

[1]http://people.csail.mit.edu/jrennie/20Newsgroups/

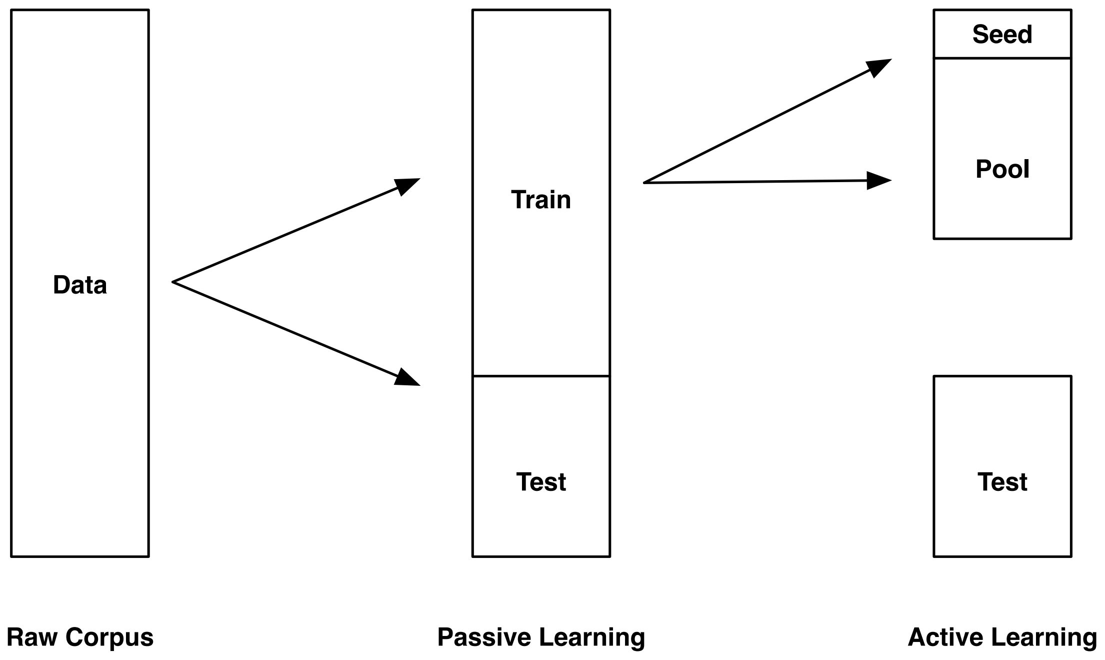


Figure 4.1: Data split for the R10 dataset. The original raw corpus (left) is divided into training and testing sets using the ModApté split (middle). The training set is further divided (right) into a small training set consisting of just the seed examples and a pool of 1,000 unlabelled examples drawn at random from the training data. The test set remains unchanged.

and range of problems.

Pre-processing

Documents in the 20NG-4 are parsed into the vector space model. Individual words formed the features used where the total number of features was the lexicon of all unique words in the training set. Stopwords were removed and stemming was applied in order to help reduce the dimensionality of the resulting vector space model. Feature selection was not performed and documents were normalised for length.

Partition

The 20NG-4 can be partitioned in a number of ways (see Section 2.8.1). Two methods of splitting this dataset are used in the empirical evaluations in this thesis:

50/50 Split This is where the data is randomly divided into two subsets, where one set is

Table 4.2: 20NG-4 Sub-problems

Label	Data	Difficulty
(AR)	alt.atheism vs. talk.religion.misc	Difficult
(GX)	comp.graphics vs. comp.windows.x	Moderately Difficult
(WH)	comp.os.ms-windows.misc vs. comp.sys.ibm.pc.hardware	Moderately Easy
(BC)	rec.sports.baseball vs. sci.cryptography	Easy

used for training while the other is held back for testing. This is referred to as 50/50 train/test split (Schohn & Cohn, 2000). Several trials of the experiments are conducted, each with a different random 50/50 split. Results are then averaged across all the trials of the experiment. Each problem had approximately 2,000 examples giving a training set size of roughly 1,000 and a test set size of roughly 1,000. Unless otherwise stated, ten trials of the experiment were conducted.

k-Fold Cross Validation An alternative approach to split this dataset is k-Fold cross validation, where the data is partitioned into k folds (or sets) and k trials of the experiment are conducted - one for each fold. In each trial of the experiment a different fold is held back as test data while all other folds are used as training data. The results of the k trials are averaged to produce the final results. 10-fold cross validation was used in our experiments, where each problem of the 20NG-4 had approximately 2,000 examples giving a fold size of roughly 200. The test set consisted of just one fold while the training set consisted of 9 folds (approximately 1,800 examples).

Pool

Unlike the R10 dataset, the number of documents in each of the 20NG-4 problems is sufficiently small that the unlabelled pool was formed using all the original training data. Seed training examples are extracted at random from the unlabelled pool and labelled. The pool in the 50/50 partition therefore consists of approximately 1,000 minus the s seed examples while in the k-Fold validation consists of 1,800 unlabelled examples minus the s seed examples.

4.3 Software

Where possible we have attempted to use open-source machine learning software to implement our experiments. When this was not feasible we constructed bespoke software to meet our requirements. Below we will discuss the software used in the various components of the experiments conducted.

4.3.1 Pre-processing

We built a custom parsing engine in order to allow for greater flexibility in processing text data. The Berkeley parsing engine (BPE) is built using modNLP[2] and is responsible for the conversion of textual documents into document vectors within a vector space model. The model can be stored and used later with classification algorithms in both passive and active learning settings.

Documents are tokenised based on white-space characters and the frequency of each token in the corpus is maintained. The Matrix Toolkit for Java (MTJ)[3] provides matrix functionality, used in the vector space model. Due to the sparse nature of text data, computational savings where made by using the sparse versions of both matrices and vectors.

Dimensionality reduction is commonly combined into the pre-processing stages in an effort to reduce memory and storage overheads. Feature selection (Section 2.5.1) was incorporated into BPE in order to allow for reductions in dimensionality to be made prior to storing the model. Several feature selection techniques are implemented including document frequency and information gain.

A common interface is required between the parsing engine and the classification system. The use of the vector space model suggested the use of a matrix-based representation. The matrix market[4] was developed by NIST to allow for the convenient access to repositories of test data for use in comparative studies of algorithms. A number of file formats are used to efficiently store large matrices. We used the coordinate file format to store the output of the processing engine. The choice of this format was motivated by the sparse nature of text data.

[2]http://modnlp.berlios.de
[3]http://ressim.berlios.de/
[4]http://math.nist.gov/MatrixMarket

4.3.2 Classification

A number of different machine learning software libraries are available each offering different benefits and drawbacks. While no one software library met all our requirements we combined several in order to perform the experiments. We strived to utilise open-source machine learning libraries for two reasons. First, since the software is open-source we could easily adapt and change the existing software to meet our requirements. Secondly, the use of existing, freely available software increases reproducibility.

WEKA

Waikato Environment for Knowledge Analysis (Witten & Frank, 2005) is a machine learning library written in Java. It consists of a vast collection of classification induction algorithms ranging from decision trees to support vector machines. WEKA was chosen since it is open-source and is commonly used in machine learning experiments. The primary use for WEKA was to supply implementations of naïve Bayes and k-NN.

Spider

The Spider is a Matlab toolbox for performing supervised, unsupervised and semi-supervised machine learning. It is an open source toolbox available for download online[5]. The spider implements a number of classification induction algorithms and encapsulates many existing machine learning libraries and offers a unified interface to them using the *data object*. In our research we have used the Spider toolbox for performing principal component analysis on text categorisation data as well as providing an implementation of a support vector machine.

4.3.3 Active Learning

Existing machine learning software is primarily used for passive supervised learning. Our requirements were for an experimental platform on which we could quickly develop experiments that could be run in conjunction with existing open-source machine learning libraries. We decided to implement bespoke software to meet the active learning requirements of our research. Matlab was chosen as the development platform given its ability to handle matrices as well as easy integration to Java. We developed a framework of utility functions and scripts which we could assemble in a script to perform an experiment.

[5]http://www.kyb.tuebingen.mpg.de/bs/people/spider/

4.4 Experiments in Active Learning

In this section we detail the experimental settings used in the empirical evaluations in the latter chapters of this thesis. Chapter 3 reviewed the various components of an active learning system and detailed the parameters used to control the process. Query selection was identified as the critical component of the active learning system and a wide variety of query selection strategies were reviewed.

In order to fully compare alternative query selection strategies, every possible combination of active learning parameters must be evaluated for each strategy. The number of experiments quickly becomes unwieldy. For this reason it is common to fix all parameters except those which are directly under investigation. In this way, fair comparisons can be made between various strategies using fewer empirical evaluations.

We begin by discussing implementation details for the various components of the canonical active learning algorithm (see Section 3.3.1). A common set of parameters used to control the active learning process are given along with values. These parameters can be fixed in order to reduce the number of experiments conducted to evaluate a query selection strategy. Finally we detail the evaluation criteria used for the empirical evaluations.

4.4.1 Active Learning Components

Active learning combines several components in order to perform its task. These components are supplied to the learner and are outside its control. We give details on these components below.

Unlabelled Data

Unlabelled data can be obtained in a number of ways (see Section 3.3.2). In this thesis we focus on pool-based active learning. Since annotated corpora are used, the unlabelled pool was formed by taking the appropriate training data and masking the labels.

Oracle

Labelling text documents is a tedious and time consuming job. If a human oracle was used only a limited number of active learning experiments could be conducted. Rather than having a human-in-the-loop for active learning experiments the role of the oracle is commonly emulated by performing experiments on annotated corpora. Queries are selected from the pool using the query selection strategy and the true label is obtained by revealing the masked label supplied in annotated corpus. In this way the human oracle is emulated allowing for more active learning experiments to be conducted.

Classifier

A small set of classifier induction algorithms (see Section 2.6) were used in the empirical evaluation in the latter chapters. Throughout this thesis we perform *homogenous* active learning, where the classifier used to provide predictions on the unlabelled data is of the same type as the final classifier produced when active learning is stopped. Heterogeneous active learning (Lewis & Catlett, 1994; Iyengar *et al.*, 2000) is where a different type of classifier is used to provide predictions than the one used as the final classifier.

Seed Data

Seed examples form the initial training set supplied to the active learner and are crucial for the success of active learning in the early iterations. In order to allow for fair comparisons to be made between alternative active learning strategies, seed data is fixed across each trial of an experiment. This ensures that any changes to the training data are strictly due to the strategy in use. Seeds are selected by randomly sampling a number of unlabelled examples from the pool and labelling them prior to the start of active learning. In most experiments an equal number of positive and negative examples are selected as seed data. This removes initial bias from the first classifier and also ensures that examples of minority classes are represented in the training data.

4.4.2 Parameters

Section 3.3 described the parameters used to control the active learning process. Below we give details on a common set of parameters used in all active learning experiments in this thesis.

Stopping Criteria

In most experiments a *fixed* stopping criterion was employed. We stopped active learning when a resource was exhausted, where the resource chosen was either the unlabelled pool or the oracle.

Where possible active learning was allowed to continue until all unlabelled examples from the pool were exhausted. This approach gives a more detailed representation of the performance of active learning over its entire life-cycle. However, the size of the pool is typically extremely large since it is inexpensive to collect unlabelled data. In order to reduce computational overhead we generally halted active learning when the oracle was exhausted. The oracle is willing to label a predefined number of unlabelled examples. Typically, the number is relatively low since it is believed that the oracle (human) will quickly tire of labelling examples.

In the empirical evaluation we choose to stop active learning after 100 or 250 iterations as we consider this to be a reasonable number of queries to label. It is unrealistic for an oracle to label thousands of examples especially if we assume the oracle is human. In addition, if labelling requires considerable time or is expensive to perform (e.g. biological tests) then only a small number of queries are likely to be labelled.

Granularity

Using a fixed stopping criterion means the objective of the active learner is to achieve maximal performance within the limited number of queries allowed. In this setting more emphasis is given to selecting informative examples rather than computational efficiency. We wished to reduce the labelling burden therefore the finest granularity was used in the empirical evaluations. In this case, we trade computational expense for reduction of labelling effort.

4.4.3 Evaluation

In order to compare alternative active learning strategies, a way to evaluate the relative performance is required. In Section 3.5 a number of the most popular evaluation metrics were discussed. All are based on evaluating the current classifier using a holdout test set. This is possible when performing active learning in laboratory conditions where annotated corpora are used.

Experiments conducted in latter chapters of this thesis evaluated the current classifier against a holdout test set for every iteration of active learning. A series of performance values are obtained for each strategy. The confusion matrix obtained was stored and from this confusion matrix a number of common performance metrics can be calculated and plotted to form a learning curve. Additionally, metrics such as CEM can be calculated from the stored confusion matrices.

The correctness of a performance metric depends on the data and parameters of the experiment. In order to achieve a more complete and comprehensive analysis of the performance of various active learning strategies we conducted our evaluation using the performance metric (accuracy, F_1, ...) most suitable to the setting. Where possible we also display results using multiple evaluation metrics.

4.5 Baseline Analysis

Baseline analysis was performed to highlight the difference between active and passive learning. In order for active learning to be considered beneficial it should demonstrate a significant reduction in the labelling effort required to produce an accurate classifier.

The supervised learning accuracy is the accuracy attained by a classifier induced via supervised learning when the entire pool is labelled. We consider how many labelled examples are required by both passive and active learning to achieve the supervised learning accuracy. Active learning should converge to the supervised learning accuracy at a faster rate than passive learning.

4.5.1 Experiment Settings

Experiments were performed on the R10 dataset as described in Section 4.2.1. Ten trials of the experiments were conducted where a new pool 1,000 unlabelled was constructed in every trial.

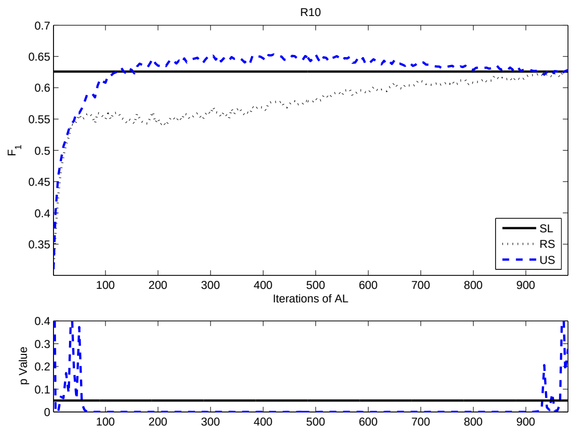
Labelled training data consists of seed data plus labelled queries from the pool. Seed data was constructed by selecting ten positive and ten negative examples from the pool at random. In order for a fair comparison to be made active learning using both query selection strategies were supplied with the same seed data.

Active learning was performed using uncertainty sampling (Lewis & Gale, 1994) where the query is selected as the most uncertain example as determined by the prediction confidence of the most recently induced classifier. The classifier used was a committee consisting of 10 members and was constructed using bagging (see Section 2.6.4). The WEKA bagging classifier returns a class probability estimate from the committee for any unlabelled examples presented to it. Predictions for all examples in the pool are obtained from the committee and queries are selected based on the uncertainty of these predictions. The unlabelled pool is never replenished and active learning was continued until the pool was exhausted. Granularity was set to 1, meaning that just a single query was selected in each iteration.

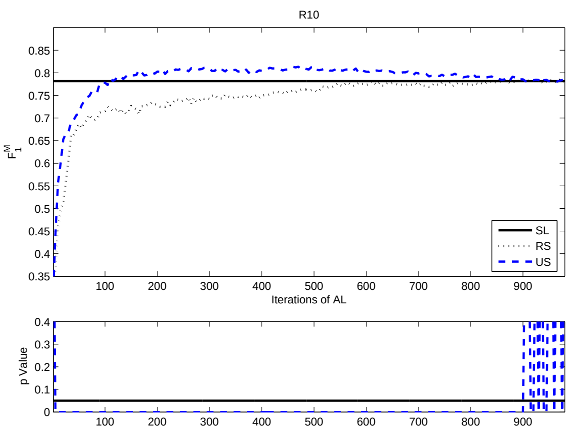
Passive learning is emulated by random sampling whereby the selection of the query from the pool is done at random. Results are reported using macro and micro averaged F_1. Additional learning curves for individual categories are given in Appendix A.

4.5.2 Results

Figure 4.2 shows the results obtained by performing active learning until the pool of unlabelled examples was fully exhausted. As the number of labelled examples increases, the accuracy achieved by both of the active learning strategies converge to the passive supervised learning accuracy. The supervised learning accuracy is denoted using the solid horizontal line (SL) in Figure 4.2. Active learning using random sampling (RS) is denoted as the dotted line and active learning using uncertainty sampling (US) is denoted as the dashed line.



(a) Macro F_1



(b) Micro F_1

Figure 4.2: Results obtained using a bagged multinomial naïve Bayes classifier with Active Learning run until pool exhausted, seeded with 10 examples from each class and selecting 1 query per iteration. F_1 is used as the evaluation metric and evaluation on the test set was done after every 5 iterations.

Both active and passive learning will converge to the supervised learning accuracy as the pool of examples are labelled and added to the training data. The power of active learning is in the rate at which the convergence happens. Random sampling can be seen to slowly converge to the supervised learning accuracy while uncertainty sampling is seen to rapidly converge to the supervised learning accuracy. The results reported here are consistent with previous research (Lewis & Gale, 1994).

Table 4.3 shows the number of queries required by both random sampling and uncertainty sampling to achieve the supervised learning accuracy.

Table 4.3: Queries required to achieve supervised learning accuracy on R10 dataset.

	RS	US
Macro F_1	972	125
Micro F_1	840	103

The benefit of performing active learning becomes evident when the number of queries required to achieve the supervised learning accuracy is examined. It was found that random sampling required 972 queries before reaching the supervised learning macro F_1 value while 840 queries were required to reach the micro F_1 value.

Conversely, active learning requires significantly fewer queries. At just 125 queries the supervised learning macro F_1 value is achieved, while just 103 queries were required to achieve the supervised learning macro F_1 value.

By carefully selecting examples using uncertainty sampling a reduction of 847 queries was observed for the macro F_1 when compared to random sampling. Similarly, a reduction of 737 queries was observed for micro F_1 when compared to random sampling. This demonstrates a significant reduction in labelling effort.

Statistical Significance

Statistical significance is demonstrated using a paired t-test. Directly under the learning curves in Figure 4.2 a plot of the p-value associated with performing a paired t-test (as proposed in (Becker, 2008)) is given. The paired t-test compares uncertainty sampling with random sampling for every iteration of active learning. Statistical significance at $\alpha = 0.05$ is shown in the plot as a solid line. Results obtained demonstrate that uncertainty sampling is statistically significantly better than random sampling over the majority of the iteration of active learning.

Random sampling is competitive only in the very early iterations and again in the later iterations. Initially, with small amounts of training material, the classifiers induced tend to be poor, hence providing poor predictions on the unlabelled data. In these situations uncertainty sampling is no better than random sampling. However, as the size of the training data increases, classifiers induced are more accurate leading to better predictions on the unlabelled data. Uncertainty sampling can now leverage the information given by the predictions to select more informative examples than random sampling.

As active learning approaches the later iterations, random sampling is seen to become competitive once again. This is due to the fact that the pool is almost depleted and both strategies converge to the supervised learning rate.

4.5.3 Discussion

The results obtained are consistent with previous research that has shown active learning to significantly reduce the labelling effort compared to supervised learning. It was observed that an 87 percent reduction in the number of queries was achieved when uncertainty sampling was employed.

Uncertainty sampling demonstrates significant improvements over random sampling. However, during the early and latter iterations of active learning random sampling was competitive with uncertainty sampling. This result supports the idea that there is not a globally optimal query selection strategy, rather different strategies are optimal during distinct phases of the active learning life cycle. Donmez *et al.* (2007) found that exploration based query selection strategies, such as random sampling, offer greater benefit during the initial iterations of active learning. In latter iterations, once a good model of the problem is established, exploitation-based query selection strategies such as uncertainty sampling are optimal.

An interesting phenomenon which has been previously reported in the literature (Schohn & Cohn, 2000) is that active learning can achieve higher accuracy than that of a passive learner trained on all $1,000$ labelled examples. This is also observed in the reported baseline analysis.

4.6 Summary

This chapter detailed the methodologies used for empirical evaluations throughout this thesis. Details of datasets and software used were given to aid reproduction of results presented in latter chapters. Active learning is controlled by a large number of parameters. In order to curtail the number of experiments required when comparing two query selection strategies certain parameters are fixed. We gave details on the parameters used in the active learning experiments conducted and supply default values.

Baseline analysis on the R10 dataset demonstrated the ability of active learning to significantly reduce the labelling effort. Despite its relative simplicity, uncertainty sampling is one of the best performing query selection strategies in active learning. Throughout this thesis we compare alternative query selection strategies to both uncertainty sampling and random sampling.

In the following chapters we propose methods and techniques which improve the performance of active learning. Novel query selection strategies are developed that are capable of selecting more informative examples and techniques are investigated to decrease the computational overhead of active learning.

Chapter 5

History-Based Query Selection

5.1 Introduction

In this chapter we propose *History-based* query selection strategies, which incorporate information from prior iterations of active learning into the selection of informative unlabelled examples as queries in the current iteration.

Query selection strategies typically exhibit the Markov property (Sutton & Barto, 1998, Chapter 3) that implies that selection of future queries are based only on the current classifier's predictions. In addition, predictions obtained in the current iteration are ephemeral and are discarded after the query is selected. We hypothesise that valuable information is being lost and propose to capture and incorporate information from prior iterations to help select more informative queries in the current iteration.

Predictions from every iteration of active learning are collected and stored in a data structure we call *History*. The History allows for the prediction for a particular unlabelled example to be retrieved for every iteration of active learning thus far. Novel History-based query selection strategies are proposed which utilise the History to incorporate predictions given in prior iterations of active learning into the query selection process. History-based query selection strategies are non-Markovian since they select queries based on information from both the current and past iterations of active learning. Furthermore, the use of the History makes History-based query selection strategy very efficient since they reuse predictions given in prior iterations of active learning.

Empirical evaluations on a real-world text corpora demonstrate that increased effectiveness is achieved from classifiers produced by active learning when historical predictions are incorporated into the query selection process. Increasing the quality of queries selected will reduce the labelling effort and ultimately reduce the cost of applying machine learning solutions to problems.

Section 5.2 discusses the background and motivation for History-based query selection. In Section 5.3 we describe the data structure History that is used to store the predictions from each iteration of active learning. History-based query selection strategies are developed in Section 5.4 which utilise information from the History into the selection of informative unlabelled examples. Section 5.5 discusses filtered History-based query selection, which place more emphasis on the most recent predictions. Empirical evaluations and results are reported in Section 5.6 and discussion are given in Section 5.8.

5.2 Background

The canonical query selection strategy (see Chapter 3) involves construction of a classifier from the known labelled data, obtaining predictions on the unlabelled data and using the predictions to select the next query. The query selection strategy exhibits the Markov property (Sutton & Barto, 1998, Chapter 3) which implies that selection of the query is based only on the predictions of the current classifier. For example, uncertainty sampling (Lewis & Gale, 1994) selects queries from an unlabelled pool of examples based on uncertainty in the prediction from the current classifier. Roy & McCallum (2001) discuss first order Markov active learning which states that the aim is to select a query ($\mathbf{x}^*$) such that, when its true label (y) is acquired from the oracle and added to the training data, the classifier trained on the resulting training data has lower error than any other unlabelled example ($\mathbf{x}$).

Predictions obtained in the current iteration of active learning are ephemeral: once used to select the query, they are discarded. Potentially valuable information in the prediction confidence of each remaining unlabelled example received in every iteration of active learning is being lost.

We hypothesise that capturing the predictions about unlabelled examples from prior iterations of active learning can be useful in the selection of queries in the current iteration. Potentially, classifiers produced in prior iterations are more accurate than the current classifier. This may be particularly evident in the early iterations of active learning, where the predictive power of a classifier can be poor (due to the very small training data) resulting in a sub-optimal query being selected and misdirecting learning. In this case it may be advantageous to use predictions from a classifier in a previous iteration of active learning.

Informative queries are generally those examples which have high uncertainty and reside close to the decision boundary Lewis & Gale (1994); Tong & Koller (2001). We hypothesise that these informative unlabelled examples should remain uncertain over many iterations of active learning. Examining the predictions over the last d iterations of active learning, consistently uncertain unlabelled examples can be identified.

We propose History-based query selection strategies that incorporate information from prior iterations of active learning. The information in the History supplies the query selection process with some additional information about the unlabelled examples. These History-based query selection strategies are non-Markovian since the selection of the query in no longer based only on the predictions given by classifiers in the current iteration.

5.3 History

In this section we describe the data structure History that is used to store the predictions from each iteration of active learning. Active learning is an iterative process whereby in every iteration one or more queries are selected to be labelled by the oracle and incorporated into the training data.

Predictions made in each iteration of active learning are ephemeral. We propose to store and utilise the historical predictions in a data structure we refer to as History. In essence, the History is a matrix, where each column represents the predictions for all unlabelled examples (rows) in an iteration of active learning. For any particular unlabelled example the predictions given by a classifier in any iteration of active learning, thus far, can be retrieved from the History. This information can then be incorporated into a query selection strategy to help select more informative examples.

Figure 5.1 shows how the History is integrated into the active learning process. In each iteration a classifier (h) is induced from all known labelled data and is used to provide predictions ($h(\mathbf{x})$) on the remaining unlabelled examples. In contrast to traditional active learning, the History is now used to store the predictions of each iteration of active learning. Queries are selected using the History rather than just the current classifiers predictions. Once the query is selected and labelled by the oracle $\{\mathbf{x}^*, y\}$ it is added to the training data.

The History is an efficient way to store information from prior iterations of active learning. The space complexity of the History is bounded by the number of unlabelled examples since

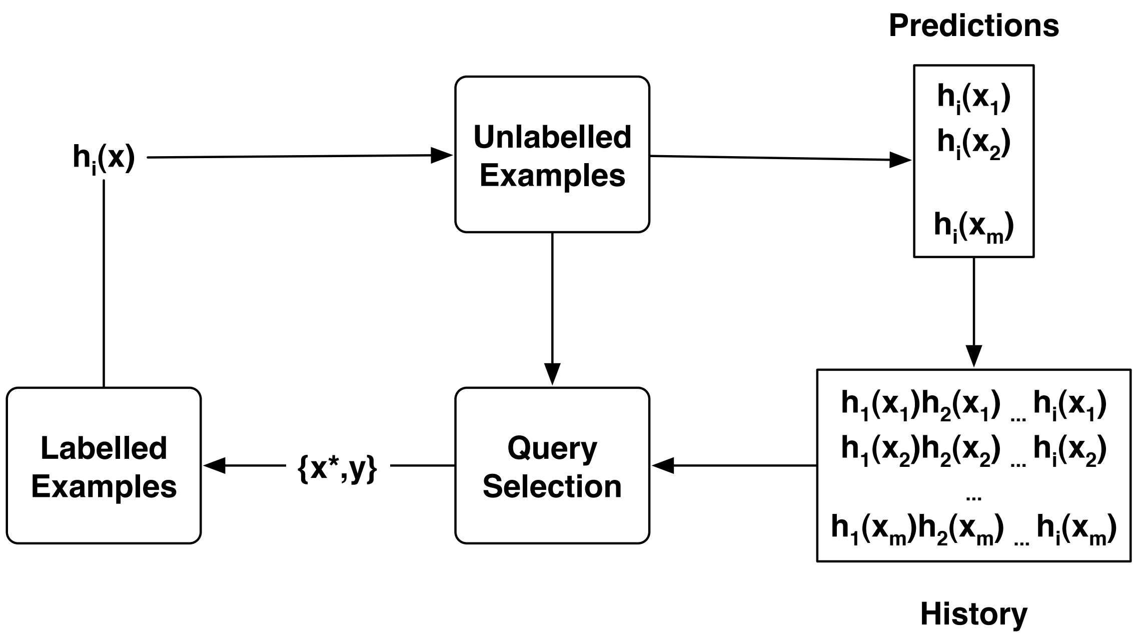


Figure 5.1: History process. In each iteration of active learning a classifier is induced and provides predictions on the unlabelled examples. Predictions are stored in the History, which is then used by the query selection process to select the query. The labelled data is updated with the newly labelled query and the process continues.

only the predicted values for the unlabelled data are stored. This gives a space complexity of $O(nT)$, where n is the number of unlabelled examples ($|\mathcal{D}_U|$) and T is the number of iterations of active learning. The time complexity is linear with the number of predictions since values are just stored.

5.4 History-Based Query Selection

In this section we describe ways in which information from the History can be incorporated into a query selection strategy. The History is an extra resource that a query selection strategies can utilise in order to help select more informative unlabelled examples. Predicted values for a given unlabelled example are retrieved from the history using the $Hist(\mathbf{x}, j)$ function, where $\mathbf{x}$ is the unlabelled example and j is a parameter which controls from how far back in the History the prediction should be retrieved ($j = 0$ is the current iteration, $j = 1$ is from the last iteration, and so on).

Two History-based query selection strategies are proposed, each of which utilise History in a different way. These History-based query selection strategies are controlled using two parameters, namely *History depth* and *History delay.*

History Depth

Exactly how far back in the History predictions are sought is controlled by the History depth parameter d. A depth of $d = 0$ will use just the most current predictions, which is the same as performing uncertainty sampling.

The most recent classifier is also likely to be the most accurate since it is induced from the largest amount of available labelled data. For this reason more recent predictions should be favoured over older predictions. History depth allows us to set a limit on how far back in the History a particular History-based query selection strategy will seek prediction from.

History Delay

Before a History-based query selection strategy can start, it requires the History it to contain a sufficient numbers of predictions. History delay is a parameter used to delay the use of History-based query selection strategies until such time when the History is populated with enough predictions. It does this by only applying History-based query selection strategies after a given number of iterations have occurred. Non History-based query selection strategies are employed until the number of iterations satisfy the History delay criterion.

5.4.1 History Uncertainty Sampling (HUS)

In Section 3.4.3 we discussed uncertainty sampling (US) which is a straightforward query selection strategy which can be applied to any probabilistic classification algorithms. Queries are selected as those examples which have the highest uncertainty. Uncertainty is defined as the lack of confidence in the prediction given by the current classifier whereby a prediction close to 0.5 gives high uncertainty.

We propose History Uncertainty Sampling (HUS) as an extension to uncertainty sampling. HUS attempts to identify those unlabelled examples which have been consistently uncertain over a number of iterations. Queries are selected based on the sum of the uncertainty in the prediction obtained over the last d iterations of active learning, as shown in (5.1). On the i^{th} iteration the query is selected as the unlabelled example which has the highest cumulative uncertainty over the past d iterations of active learning.

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{D}_u} \sum_{j=0}^{d} |Hist(\mathbf{x}, j) - 0.5| \tag{5.1}$$

where $Hist(\mathbf{x}, j)$ retrieves the prediction for the given unlabelled example from the History, The second parameter j controls how far back in the History predictions are retrieved, where $j = 0$ retrieves the predictions from the current classifier, $j = 1$ retrieves the prediction given by the classifier in the previous iteration of active learning and so on.

Table 5.1 shows an example History for three documents. In the 5^{th} iteration of active learning, uncertainty sampling would select document $\mathbf{x}_1$ as the query since this has a prediction of 0.49 which is the most uncertain. Considering a history depth of $d = 3$, the query selected using HUS will be document $\mathbf{x}_2$. HUS examines the historical predictions where it can be seen that document $\mathbf{x}_2$ has also received highly uncertain predictions over the last two iterations.

Table 5.1: History Uncertainty Sampling Example. Predictions for the three document in all five iterations are stored in the History. HUS with a history depth of three ($d = 3$) uses the predictions highlighted by bold text to select the query

	1	2	3	4	5	US Rank	HUS Rank
$\mathbf{x}_1$	0.61	0.74	**0.80**	**0.81**	**0.49**	1*	2
$\mathbf{x}_2$	0.59	0.56	**0.45**	**0.49**	**0.70**	2	1*
$\mathbf{x}_3$	0.32	0.21	**0.22**	**0.81**	**0.11**	3	3

5.4.2 History Kullback-Leibler Divergence (HKLD)

An alternative view of the History is as a collection of classifiers, one for every iteration of active learning. We propose History Kullback-Leibler Divergence (HKLD) which constructs a committee from classifiers induced in prior iterations of active learning. Queries are then selected as those unlabelled examples which cause the most disagreement in the committee.

A committee of size d can be constructed by simply combining the predictions of the past d iterations. While the committee cannot be used to label new examples it can provide predictions on the remaining unlabelled examples in the pool.

In the Query By Committee (QBC) query selection strategy (see Section 3.4.4) queries are selected as those unlabelled examples which cause maximal disagreement among the committee members. We used Kullback-Leibler (KL) divergence to the mean (McCallum &

Nigam, 1998) to measure disagreement as it incorporates the uncertainty of each committee member's predictions.

KL divergence (5.2) measures the difference between two probability distributions P and Q. It is sometimes referred to as *relative entropy* when used in information theory.

$$D(P||Q) = \sum_{\mathbf{x} \in X} P(\mathbf{x}) \log \frac{P(\mathbf{x})}{Q(\mathbf{x})} \tag{5.2}$$

KL divergence to the mean, measures the difference between the individual committee member class probability distribution and the mean class probability distribution for the committee. The mean class probability distribution (5.3) for a committee of size d is simply the average prediction for all d members.

$$\bar{P} = \frac{1}{d} \sum_{m} P_m(\mathcal{Y}|\mathbf{x}) \tag{5.3}$$

The KL divergence to the mean for a particular example ($\mathbf{x}$) with a committee of d members is calculated as in (5.4):

$$KLD(\mathbf{x}) = \frac{1}{d} \sum_{i=1}^{d} D(\, P_i \,||\, \bar{P}\,) \tag{5.4}$$

where the class distribution of an individual committee member is P_i. The query is selected as the unlabelled example which results in the most disagreement among the committee as shown in (5.5). Queries selected by HKLD tend to be unlabelled examples which have erratic and confident predictions.

$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathcal{D}_u} KLD(\mathbf{x}) \tag{5.5}$$

While committee based query selection strategies such as QBC typically have high computational overheads associated with them, HKLD has practically no additional computational expense. Since the History contains predictions, no extra computational effort is required to induce or employ a classifier to obtain predictions for the unlabelled pool.

5.5 Filtered History-Based Query Selection

The current classifier is likely to be the most accurate since it is induced from all known labelled information. It makes intuitive sense to lend more weight to the predictions it gives rather than predictions from further back in the History. A straightforward way to accomplish this is by applying the History-based query selection strategies only to a subset of the pool populated with the most uncertain unlabelled examples as defined by the most recently induced classifier.

Figure 5.2 shows the updated process whereby the predictions given by the current classifier induced from all known labelled training data is used to filter examples from the pool and populate the subset. History-based query selection strategies are then applied only to the subset rather than the entire pool of unlabelled examples.

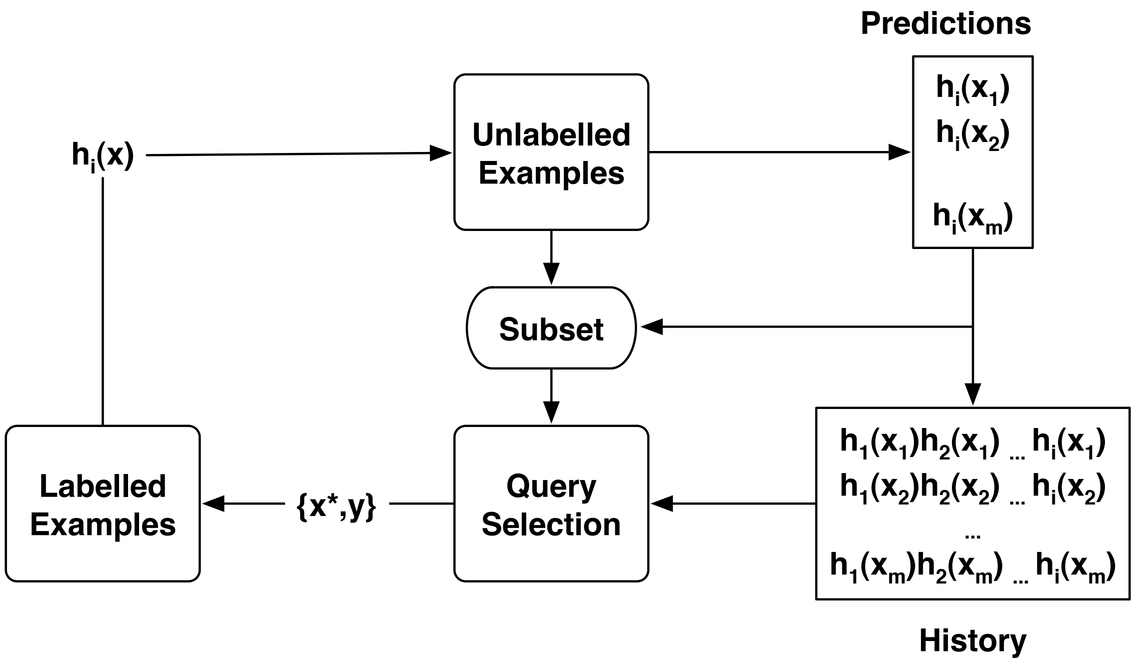


Figure 5.2: Filtered History process. In each iteration of active learning a classifier is induced and provides predictions on the unlabelled examples. Predictions are stored in the History, which is then used by the query selection process to select the query. The labelled data is updated with the newly labelled query and the process continues.

5.5.1 Subset Construction

In our experiments uncertainty sampling was used to form the subset. The subset is constructed by selecting the t most uncertain examples as given by the current classifier predictions. Subsets of various sizes ($t = \{10, 25, 50, 100\}$) were compared in the empirical evaluation.

5.5.2 History Uncertainty Sampling Filtered (HUSF)

In each iteration of active learning a subset is constructed by filtering the t most uncertain unlabelled examples from the pool as given by uncertainty sampling. HUS is then applied to the subset and a query is selected. The use of the subset ensures that the query is uncertain in the current iteration and also uncertain in the past d iterations of active learning.

5.5.3 History Kullback-Leibler Divergence Filtered (HKLDF)

Similar to HUSF, a subset is formed in each iteration of active learning by filtering the t most uncertain unlabelled examples from the pool and queries are selected by applying HKLD only to the subset. The use of the subset in HKLD ensures the query is uncertain in the current iteration and also has caused disagreement in the committee formed using classifiers in the past d iterations of active learning.

5.6 Empirical Evaluation

Through empirical evaluation we establish if incorporating information from the History is beneficial to active learning. Lewis & Gale (1994) have shown uncertainty sampling to be a top performing query selection strategy, reducing labelling effort by orders of magnitude. Uncertainty sampling was taken as a baseline and the various History-based query selection strategies were compared against it.

Experiments were evaluated on the 20NG-4 and R10 datasets, as described in Section 4.2. The 20NG-4 dataset consists of four binary text classification problems (AR, BC, GX and WH) of varying difficulty, while the R10 dataset consists of ten binary classification tasks. All documents were tokenised on non-word characters, stopwords removed and stemming applied using the Porter algorithm (Porter, 1980). Feature selection was not performed. 10-fold cross validation was performed.

Active learning was seeded with equal number of positive and negative examples. In the case of the 20NG-4 dataset, 3 positive and 3 negative seeds were randomly chosen from the unlabelled pool, while for the R10 dataset, 10 positive and 10 negative examples were randomly chosen from the unlabelled pool. In each iteration just one query is selected and active learning is stopped after 100 iterations. In order for fair comparison to be made, the same seeds were supplied to each of the strategies under investigation.

History-based query selection strategies can only be applied when the History is populated with some information. The History delay parameter was used to postpone the use of History-based query selection strategies, whereby uncertainty sampling is performed until such time as the History is populated with sufficient information. Through empirical evaluation we settled on a History delay value of 10. This means that uncertainty sampling was used for the first ten iterations of active learning, while the respective History-based query selection strategy is utilised for the 11th and subsequent iterations.

5.6.1 Preliminary Experiments

Preliminary experiments were conduced using a k-NN classifier, where cosine similarity (2.5) was used in place of Euclidean distance. The History depth (d) was tested for $d = \{3, 5, 7\}$.

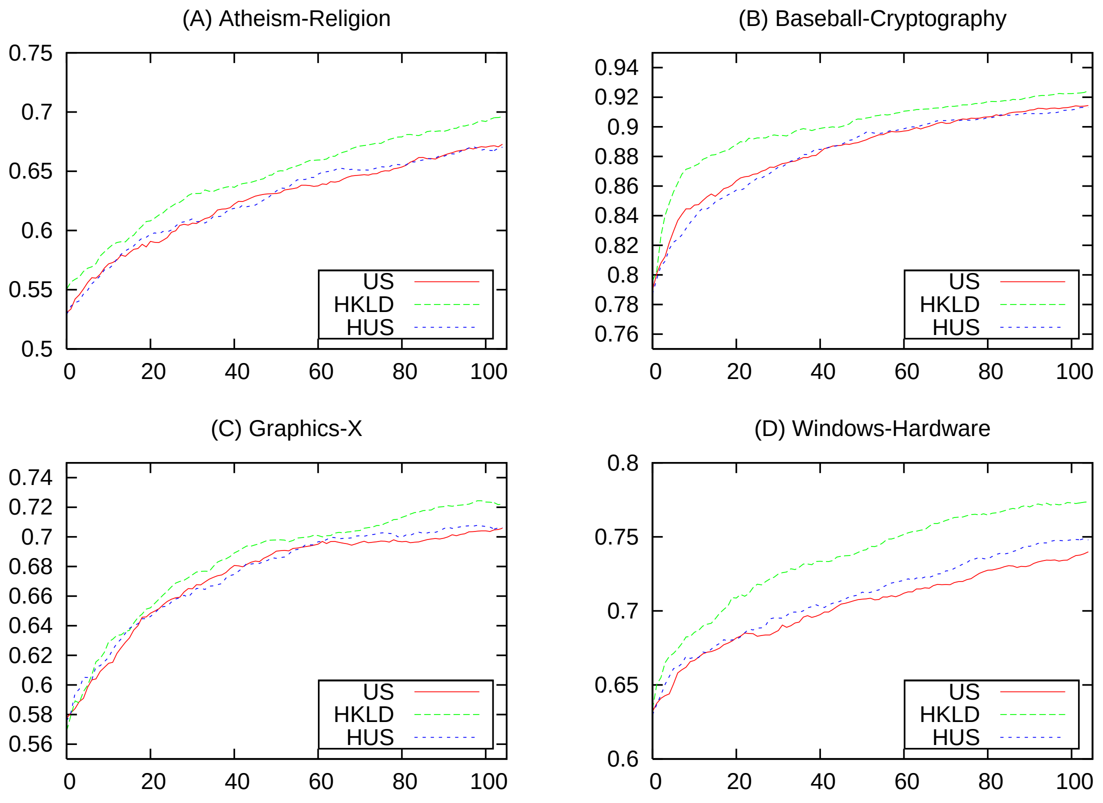


Figure 5.3: 20NG-4 results with history depth ($d = 3$). Accuracy is given on the Y axis and the iterations of active learning are given on the X axis. Active learning was stopped after 100 iterations.

Figure 5.3 displays the learning curves obtained. Accuracy was averaged over the ten trials of 10-Fold cross validation, where active learning was halted after 100 iterations. A History depth of $d = 3$ was found to be optimal in these experiments. In all four problems accuracy at the 100^{th} iteration was increased by (AR) 2.178%, (BC) 0.946%, (GX) 2.027% and (WH) 3.887%, when HKLD was used. All increases are statistically significant using a paired t-test ($\alpha = 0.05$). The results for HUS, however, are inconclusive. We believe this was due to the failure of HUS to capture the variation in class prediction.

While the experiments demonstrated a positive effect when incorporating History information using the k-NN, the classifier was changed for the full evaluation as several problems emerged in implementation. Specifically, the high dimensional data used in these text categorisation exasperated the curse of dimensionality for the k-NN resulting in poor accuracy, extra computational expense and long running times. In Chapter 6 we examine ways to make the k-NN classifier more competitive using dimensionality reduction. In the following experiments we used a naïve Bayes classifier. This classifier has the considerable advantage of being parameterless[1].

5.6.2 HUS

History Uncertainty Sampling selects as the query the unlabelled example that has been most uncertain over the past d iterations of active learning. In the following experiments, a number of history depth values $d = \{3, 5, 7\}$ are compared.

Figure 5.4 displays the aggregate learning curves (see Section 3.5.2) for HUS performed on all four problems in the 20NG-4. HUS is seen to outperform random sampling, however, while competitive, none of the variants of HUS are seen to outperform the baseline of uncertainty sampling. Examining the p-value plot it is seen that HUSd_3 offers some advantage over US between iteration 25 and 40. In the later iteration of active learning, uncertainty sampling is seen to outperform all other strategies.

Figure 5.5 plots the learning curves for HUS conducted on the R10 dataset using both macro- and micro-averaged F_1. HUS was observed to outperform the baseline of uncertainty sampling in the early iterations of active learning when it was applied. The History-based query selection strategies were delayed until the tenth iteration. A lift in effectiveness is observed for HUS from iteration ten until approximately iteration 25. This lift is also shown in the p-value plot. Uncertainty sampling was seen to be optimal in the later iterations of active learning outperforming HUS with all possible History depths.

[1]A threshold can be tuned via a validation set but we used the default threshold of 0.5

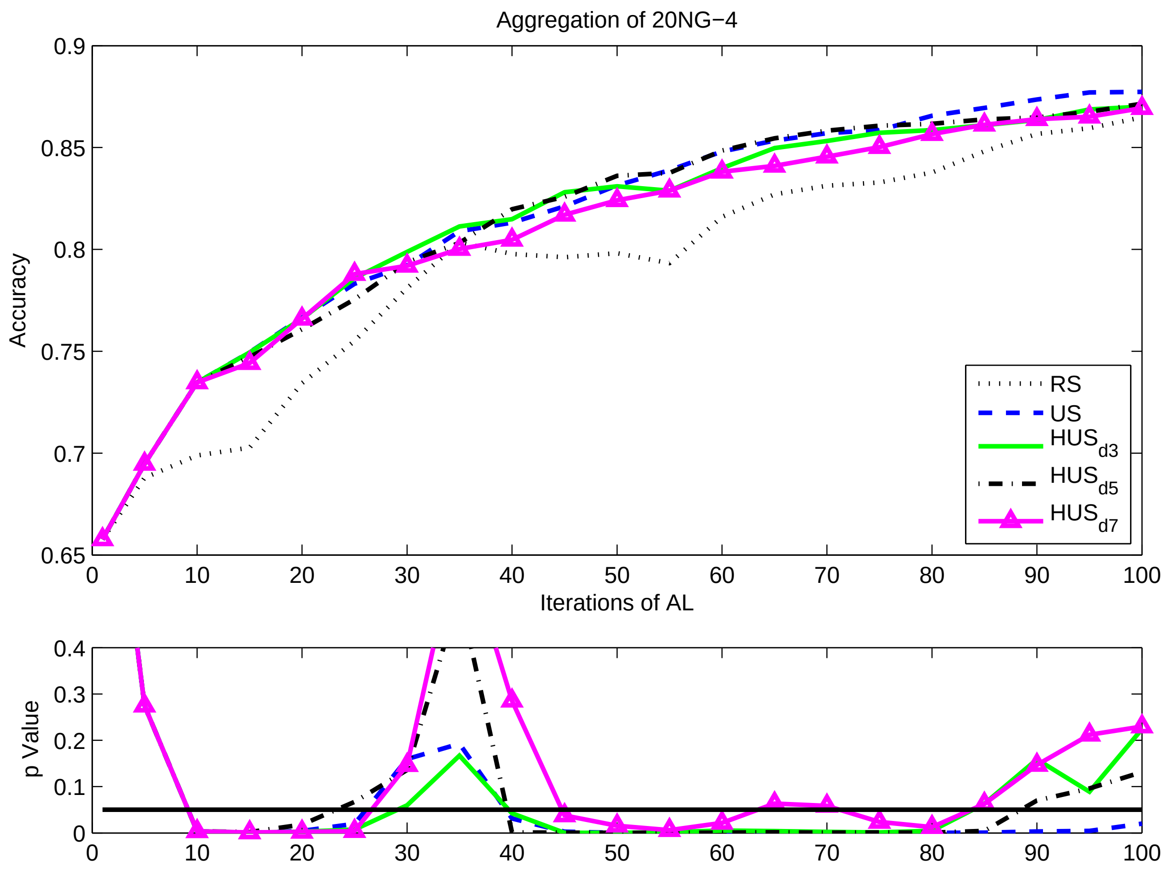
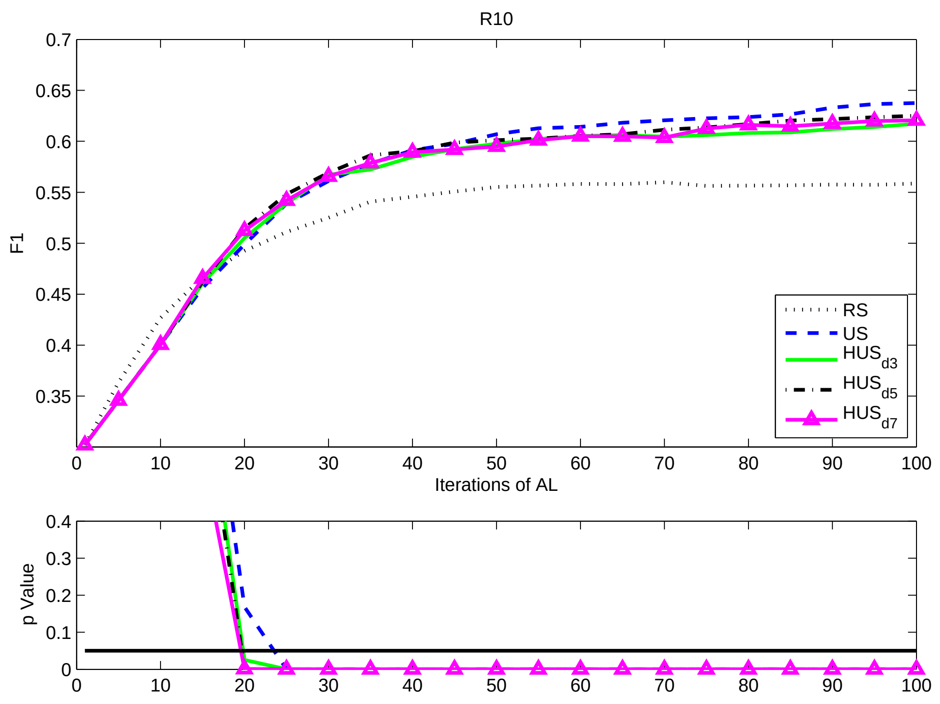
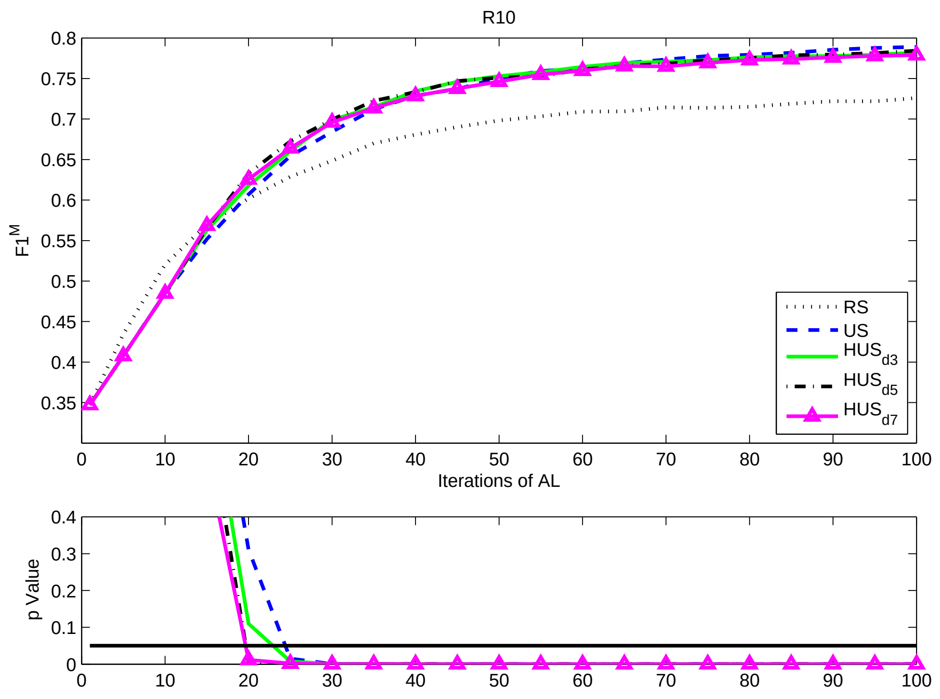


Figure 5.4: Aggregate HUS results averaged over all four 20NG-4 problems. Obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



(a) F_1



(b) F_1^M

Figure 5.5: HUS results obtained from R10 experiments using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 10 examples from each class and selecting 1 query per iteration. F_1 (computing using both macro-avering and micro-averaging) was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)

Deficiency

Deficiency values (see Section 3.5.3) for HUS on the 20NG-4 are reported in Table 5.2. A reduction in deficiency was observed in a number of the individual problems (e.g. the AR ($d = 5$), WH ($d = 3$) and BC ($d = 3$) problems). However, on average HUS did not decrease deficiency.

Table 5.2: Deficiency Values for HUS on 20NG-4. Lowest deficiency is indicated by bold text. Averages across the four problems as well as across the different History depths are also given

	AR	BC	GX	WH	Avg
US	0.8241	0.5656	**0.7521**	1.0488	0.7976
HUS$_{d3}$	0.8286	**0.5437**	0.7910	**1.0348**	0.7995
HUS$_{d5}$	**0.7981**	0.6223	0.7602	1.0939	0.8186
HUS$_{d7}$	0.8342	0.6121	0.7895	1.1390	0.8437
Avg	0.8203	0.5927	0.7802	1.0892	0.8206

Table 5.3 reports the deficiency values obtained when performing HUS on the R10 dataset. Given the imbalanced nature of the data F_1 is commonly used to report effectiveness. We calculated Deficiency as standard with Accuracy as well as using both macro- and micro-averaged F_1. It can be seen that lower values obtained when HUS is employed when considering deficiency calculated using accuracy and micro-averaged F_1, however this is not observed when deficiency is calculated using macro-averaged F_1. We believe this is an artefact of HUS performing better on a per-document basis than a per-category basis and corresponds to the learning curves reported in Figure 5.5.

Table 5.3: Deficiency Values for HUS on R10. Lowest deficiency is indicated by bold text. Average deficiency over all three History depths is also given

	Acc	F_1	F_1^M
US	0.8793	**0.6229**	0.7115
HUS$_{d3}$	0.8569	0.6946	0.6980
HUS$_{d5}$	**0.8505**	0.6464	**0.6924**
HUS$_{d7}$	0.8672	0.6723	0.7143
Avg	0.8582	0.6711	0.7016

Summary

History uncertainty sampling was observed to give benefit to active learning only in the early iterations. In later iterations uncertainty sampling was seen to be optimal. We hypothesise that HUS would benefit by placing more emphases on predictions given by the most recent classifier, giving less emphasis to predictions in previous iterations. Filtered HUS (HUSF) is proposed to address this problem.

5.6.3 HKLD

The following experiments examined the performance of HKLD query selection. HKLD considers those unlabelled examples which consistently change their label assignment as being the most interesting queries. Various values for history depth $d = \{3, 5, 7\}$ are compared in these experiments.

Figure 5.6 displays the learning curves obtained for HKLD applied to the 20NG-4 dataset. The results demonstrate that incorporating History information using HKLD did not improve the effectiveness of active learning. In fact, it reduced effectiveness, in some cases performing worse than random sampling.

Figure 5.7 reports the learning curves obtained by applying HKLD to the R10 dataset. Effectiveness is measured using both macro- and micro- averaged F_1. It was observed that HKLD performed poorly on this dataset consistently under-performing the baseline of uncertainty sampling. The micro-averaged F_1 showed that HKLD showed some improvement over uncertainty sampling in the early iterations (between iteration 10 and approximately iterations 25), however, uncertainty sampling quickly recovered and was optimal for the remaining iterations.

Deficiency

Deficiency values for HKLD conducted on the 20NG-4 dataset are given in Table 5.4. The reported values confirm that HKLD did not outperform the baseline of uncertainty sampling. Furthermore, the performance of HKLD was worse than random sampling, as indicated with deficiency values greater than one.

Table 5.5 reports the deficiency values obtained by performing HKLD on the R10 dataset. Due to the imbalanced nature of the dataset F_1 is generally used to measure effectiveness. We calculated deficiency as standard using accuracy but also calculate using both macro- and

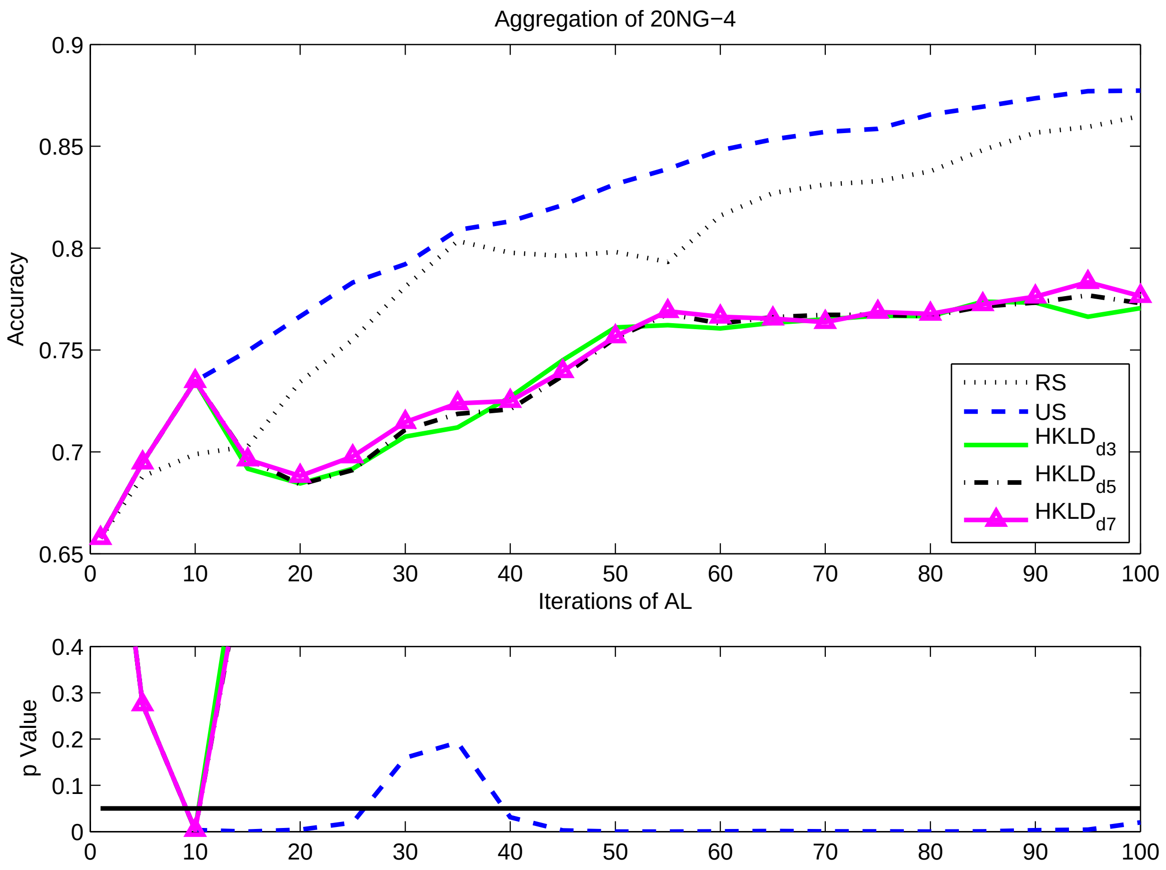
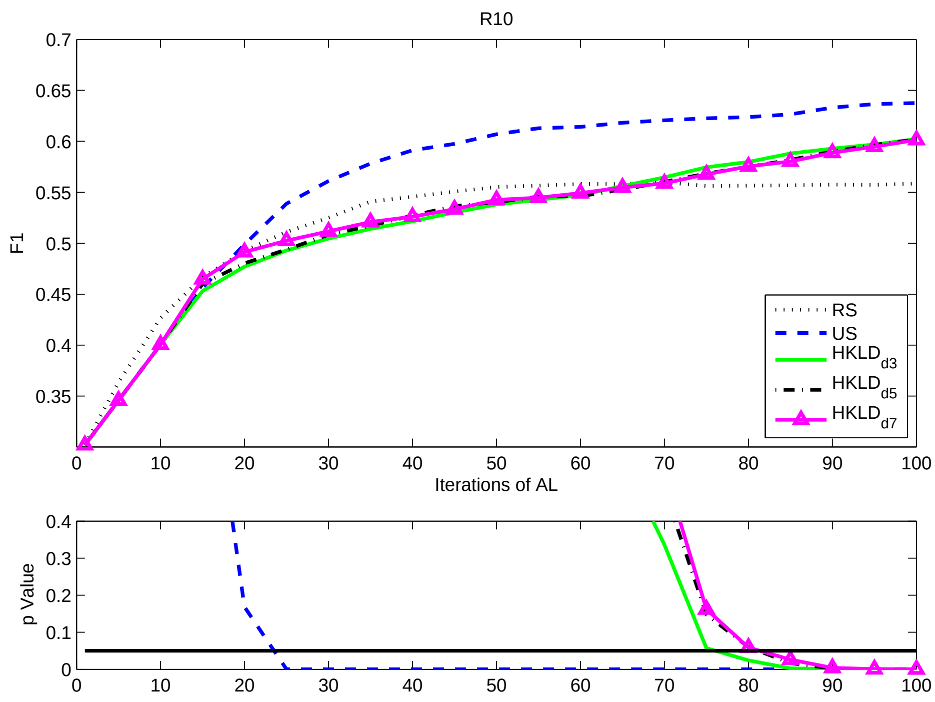


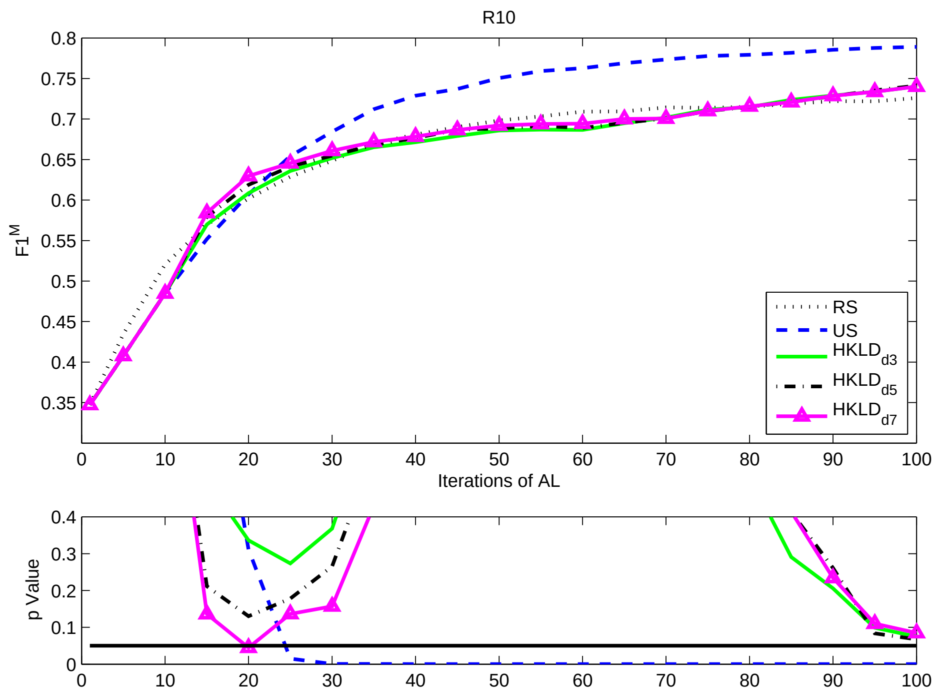
Figure 5.6: Aggregate HKLD results averaged over all four 20NG-4 problems. Obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)

Table 5.4: Deficiency values for HKLD on 20NG-4. Lowest deficiency is indicated by bold text. Averages across the four problems as well as across the different History depths are also given

	AR	BC	GX	WH	Avg
US	**0.8241**	**0.5656**	**0.7521**	**1.0488**	0.7976
HKLD$_{d3}$	1.1141	2.9535	1.0680	1.6113	1.6867
HKLD$_{d5}$	1.0890	2.9437	1.0611	1.6364	1.6825
HKLD$_{d7}$	1.0883	2.9439	1.0431	1.6003	1.6689
Avg	1.0971	2.9470	1.0574	1.6160	1.6794



(a) F_1



(b) F_1^M

Figure 5.7: HKLD results obtained from R10 experiments using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 10 examples from each class and selecting 1 query per iteration. F_1 (computing using both macro-avering and micro-averaging) was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)

micro- averaged F_1. The reported values all demonstrate that HKLD did not outperform the baseline of uncertainty sampling. Furthermore, it can be seen that HKLD did not outperform random sampling as indicated by deficiency values greater than one.

Table 5.5: Deficiency Values for HKLD on R10. Lowest deficiency is indicated by bold text. Average deficiency over all three History depths is also given

	Acc	F_1	F_1^M
US	**0.8793**	**0.6229**	**0.7115**
HKLD$_{d3}$	1.2735	1.0198	1.0435
HKLD$_{d5}$	1.2399	1.0134	1.0198
HKLD$_{d7}$	1.2212	1.0047	1.0062
Avg	1.2449	1.0127	1.0232

Summary

We hypothesise the poor performance of HKLD is due to the fact that the query selection strategy gives equal emphasis to each committee member. The current classifier is the most accurate since it is induced from all known labelled training data. Therefore its predictions should be considered more informative than historical predictions. Filtered variants of HKLD are proposed in Section 5.5 that address this problem.

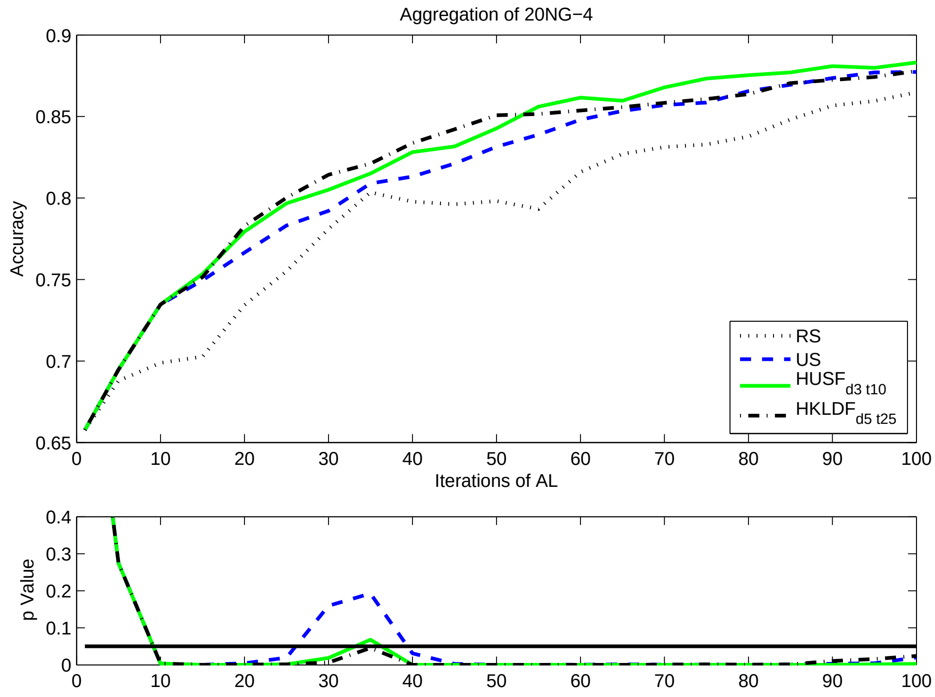
5.6.4 HUSF

HUSF places more importance on the predictions given by the most recent classifier induced. Active learning was conducted in the same manner as described previously in Section 5.5.2. Various subset sizes $t = \{10, 25, 50, 100\}$ and History depths $d = \{3, 5, 7\}$ were compared.

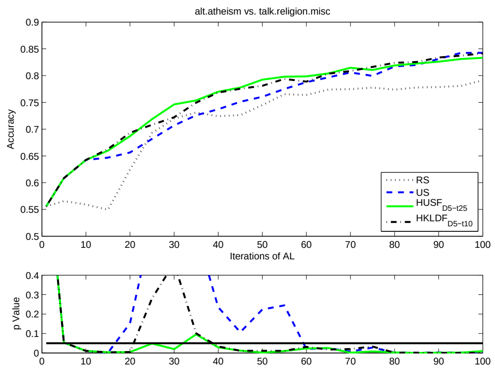
Given the large number of possible combinations of parameters we only present the plot for HUSF using optimal parameter found during empirical evaluation. A sensitivity analysis of the History depth and subset size parameters is given in Section 5.6.6 and plots for all permutations of parameters is given in Appendix B.

Figure 5.8 plots the learning curves for HUSF applied to the 20NG-4 dataset. Accuracy is calculated as the average accuracy over all four sub-problems. From the plot it can be seen that HUSF outperforms uncertainty sampling over all iterations of active learning.

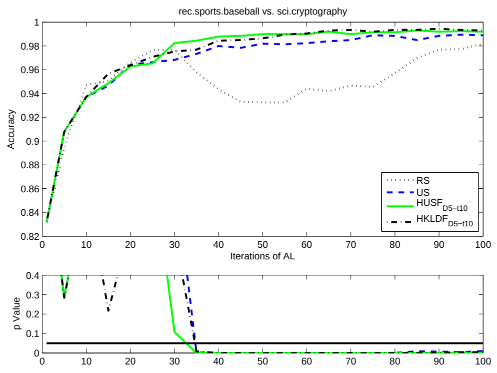
Figure 5.9 displays the learning curves obtained by applying HUSF to the R10 dataset. F_1 is commonly used as an effectiveness metric due to the imbalanced nature of the R10



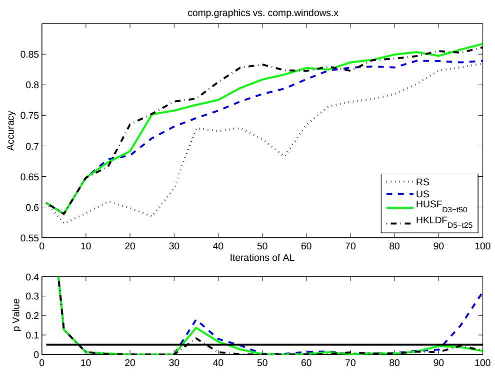
(a) Optimal



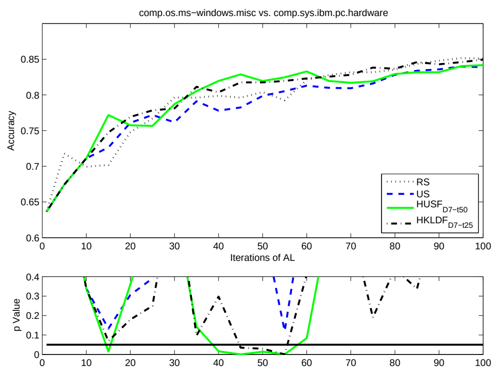
(b) AR



(c) BC

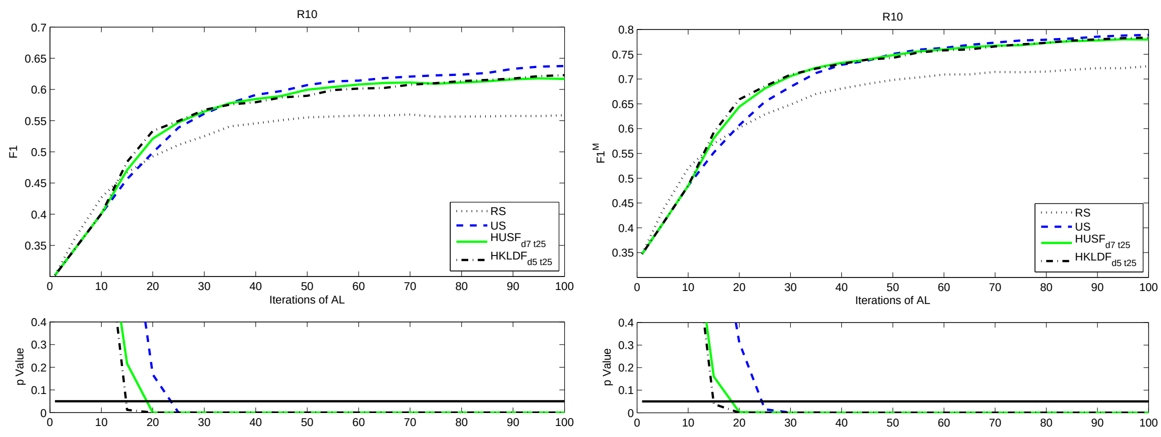


(d) GX



(e) WH

Figure 5.8: HUSF and HKLDF learning curves for optimal History depth and subset size on the 20NG-4 dataset. (a) displays an aggregate plot of all four 20NG-4 problems. The plots for each of the four 20NG-4 problems is given underneath.



(a) F_1 (b) F_1^M

Figure 5.9: HUSF and HKLDF learning curves for optimal History depth and subset size on the R10 dataset

dataset. We computed both macro- and micro- averaged F_1. It can be seen that HUSF is of greatest benefit in the early iterations of active learning. HUSF outperforms the baseline of uncertainty sampling between iteration 10 and approximately iteration 25. However, uncertainty sampling is optimal in the subsequent iterations of active learning.

Deficiency

Table 5.6 reports the deficiency values obtained for HUSF on the 20NG-4 dataset. HUSF is seen to outperform uncertainty sampling in all combinations of history depth and subset size. This trend is also evident in the average across all four sub-problems (rightmost column in Table 5.6).

Table 5.7 reports the deficiency values for HUSF on the R10 dataset. Deficiency is calculated using accuracy as well as both macro- and mirco- averaged F_1. Considering deficiency calculated using accuracy HUSF is seen to outperform the baseline of uncertainty sampling over all combinations of History depth and subset size. However, using F_1 only certain combinations of the two parameters resulted in lower deficiency values than the baseline. We discuss the sensitivity of HUSF to these parameters further in Section 5.6.6.

Summary

HUSF achieved greater effectiveness in the early iterations of active learning in both datasets considered. In the 20NG-4 dataset the increase in effectiveness was observed over all itera-

Table 5.6: Deficiency values for HUSF on the 20NG-4. Bold text signifies minimum deficiency achieved in each subset size (t)

	AR	BC	GX	WH	Avg
US	0.8241	0.5656	0.7521	1.0488	0.7977
HUSF$_{d3\,t10}$	**0.7397**	0.5129	**0.6779**	**1.0132**	0.7359
HUSF$_{d5\,t10}$	0.8043	**0.4615**	0.6961	1.0781	0.7600
HUSF$_{d7\,t10}$	0.8022	0.4878	0.6981	1.0542	0.7605
HUSF$_{d3\,t25}$	0.7685	0.5471	0.6786	1.1006	0.7737
HUSF$_{d5\,t25}$	**0.7481**	0.5558	**0.6781**	1.0108	0.7482
HUSF$_{d7\,t25}$	0.7625	**0.4731**	0.6811	**1.0031**	0.7299
HUSF$_{d3\,t50}$	0.8156	0.5375	**0.6752**	1.0382	0.7666
HUSF$_{d5\,t50}$	0.7901	**0.4751**	0.7206	0.9974	0.7458
HUSF$_{d7\,t50}$	**0.7565**	0.4984	0.7004	**0.9694**	0.7312
HUSF$_{d3\,t100}$	**0.7507**	0.5511	0.7228	**1.0411**	0.7664
HUSF$_{d5\,t100}$	0.7617	0.5856	0.6984	1.0467	0.7731
HUSF$_{d7\,t100}$	0.8031	**0.5440**	**0.6781**	1.0658	0.7728

Table 5.7: Deficiency values for HUSF on the R10. Bold text signifies minimum deficiency achieved in each subset size (t)

	Acc	F_1	F_1^M
US	0.8793	0.6229	0.7115
HUSF$_{d3\,t10}$	**0.8301**	0.6325	0.6739
HUSF$_{d5\,t10}$	0.8313	**0.5980**	**0.6711**
HUSF$_{d7\,t10}$	0.8323	0.6131	0.6766
HUSF$_{d3\,t25}$	0.8262	**0.6346**	0.6866
HUSF$_{d5\,t25}$	0.8273	0.6381	0.6890
HUSF$_{d7\,t25}$	**0.8242**	0.6629	**0.6840**
HUSF$_{d3\,t50}$	0.8682	0.6243	0.7139
HUSF$_{d5\,t50}$	**0.8415**	**0.6147**	**0.6891**
HUSF$_{d7\,t50}$	0.8438	0.6410	0.6951
HUSF$_{d3\,t100}$	0.8720	**0.6474**	**0.7149**
HUSF$_{d5\,t100}$	0.8752	0.6734	0.7307
HUSF$_{d7\,t100}$	**0.8609**	0.6894	0.7223

tions, while in the R10 it was only observed only in the initial iterations of active learning.

Increases in effectiveness were reported irrespective of the parameters used, namely History depth and subset size. However, the magnitude of the increase is directly related to the parameters used. Sensitivity analysis of the parameters used in filtered History-based query selection strategies is discussed in Section 5.6.6.

5.6.5 HKLDF

In this section we present results obtained by selecting queries using HKLDF. This History-based query selection strategy considers examples informative if they are uncertain in the current iteration and if there is a large disagreement among the committee formed by the classifiers in the last d iterations of active learning. A variety of subset sizes $t = \{10, 25, 50, 100\}$ and History depths $d = \{3, 5, 7\}$ are evaluated.

As with the HUSF experiments, given the large number of possible combinations of parameters we only present the plot for HKLDF using optimal parameter found during empirical evaluation. A sensitivity analysis of the History depth and subset size parameters is given in Section 5.6.6 and plots for all permutations of parameters is given in Appendix B.

Figure 5.8 plots the aggregate learning curves for HKLDF applied to the 20NG-4 dataset. From the plot it can be seen that HKLDF only outperforms uncertainty sampling until approximately iteration 70 where it then converges to the baseline. It was also found that HKLDF does not outperform the alternative filtered HBQS strategy in this dataset.

Learning curves obtained by applying HKLDF to the R10 dataset are given in Figure 5.9. It can be seen that HKLDF is of greatest benefit in the early iterations of active learning, where it outperforms uncertainty sampling between iteration 10 and approximately iteration 25 (iterations 10 to approximately 35 for F_1^M). Additionally, HKLDF is seen to marginally outperform HUSF in the iterations where both outperform the baseline. Uncertainty sampling becomes optimal in the later iterations of active learning.

Deficiency

Table 5.8 reports the deficiency values obtained from the 20NG-4 dataset. HKLDF is seen to outperform uncertainty sampling in the majority of combinations of history depth and subset size.

Table 5.9 reports the deficiency values for HKLDF on the R10 dataset. HKLDF is seen to outperform the baseline of uncertainty sampling over all combinations of History depth

Table 5.8: Deficiency values for HKLDF on the 20NG-4. Bold text signifies minimum deficiency achieved in each subset size (t)

	AR	BC	GX	WH	Avg
US	0.8241	0.5656	0.7521	1.0488	0.7977
HKLDF$_{d3\,t10}$	0.7906	0.5311	0.7106	**1.0269**	0.7648
HKLDF$_{d5\,t10}$	**0.7621**	**0.4539**	0.7181	1.0437	0.7444
HKLDF$_{d7\,t10}$	0.7918	0.5313	**0.6973**	1.0495	0.7675
HKLDF$_{d3\,t25}$	0.8091	**0.4826**	0.6915	1.0214	0.7511
HKLDF$_{d5\,t25}$	**0.7741**	0.5325	**0.6531**	1.0665	0.7565
HKLDF$_{d7\,t25}$	0.7790	0.4942	0.7574	**0.9436**	0.7435
HKLDF$_{d3\,t50}$	**0.7632**	0.5279	**0.6955**	1.0551	0.7604
HKLDF$_{d5\,t50}$	0.7739	**0.5248**	0.7149	1.1025	0.7794
HKLDF$_{d7\,t50}$	0.8461	0.4901	0.7086	**1.0388**	0.7709
HKLDF$_{d3\,t100}$	0.8371	0.4946	0.7101	1.0774	0.7798
HKLDF$_{d5\,t100}$	0.8314	0.5248	**0.6816**	1.1871	0.8062
HKLDF$_{d7\,t100}$	**0.7740**	**0.4930**	0.6990	**1.0454**	0.7529

Table 5.9: Deficiency values for HKLDF on the R10. Bold text signifies minimum deficiency achieved in each subset size (t)

	Acc	F_1	F_1^M
US	0.8793	0.6229	0.7115
HKLDF$_{d3\,t10}$	0.8130	0.6434	0.6737
HKLDF$_{d5\,t10}$	0.8156	0.6749	0.6852
HKLDF$_{d7\,t10}$	**0.8091**	**0.6332**	**0.6729**
HKLDF$_{d3\,t25}$	0.8043	0.6622	0.6814
HKLDF$_{d5\,t25}$	**0.7947**	**0.6606**	**0.6715**
HKLDF$_{d7\,t25}$	0.8109	0.6817	0.6764
HKLDF$_{d3\,t50}$	**0.7944**	0.6751	**0.6650**
HKLDF$_{d5\,t50}$	0.8030	**0.6500**	0.6725
HKLDF$_{d7\,t50}$	0.8142	0.6993	0.6906
HKLDF$_{d3\,t100}$	0.8403	0.7278	0.7158
HKLDF$_{d5\,t100}$	**0.7981**	**0.6794**	**0.6738**
HKLDF$_{d7\,t100}$	0.8195	0.6871	0.6875

and subset size. Averages across all History depths and subset sizes are also reported. On average HKLDF is seen to outperform the baseline.

Summary

Filtered History KL Divergence showed improved effectiveness compared to the baseline of uncertainty sampling. In both datasets HKLDF gives increased effectiveness in the early iterations of active learning, however, this increase can be short lived. In the 20NG-4 dataset the improvement was noted for all iterations of while in the R10 the improvement was only observed in the initial iterations and uncertainty sampling was optimal in later iterations of active learning.

As with HUSF, the parameters used for HKLDF directly related to the magnitude of the increase in effectiveness. Sensitivity analysis of the parameters used in filtered History-based query selection strategies is discussed in next section.

5.6.6 History Parameters

In this section we report the results obtained for various combinations of the History parameters, namely History depth (d) and subset size (t). Both of the filtered History-based query selection strategies are sensitive to these parameters. In order to help identify a possible trend we consider the average deficiency across each parameter.

HUSF

Table 5.10 compares the different deficiency values obtained using the various permutations of History depth and subset size in the HUSF query selection strategy. In this case we use the average deficiency value obtained across all four sub-problems of the 20NG-4 dataset. From these results it can be seen that on average, lower deficiency is obtained for larger History depths and medium sized subsets.

Results obtained for HUSF conducted on the R10 dataset are given in Table 5.11. Deficiency was calculated using accuracy in this case. The results show that on average, lower deficiency values were recorded when HUSF was employed with larger History depths and smaller subsets (25).

Table 5.10: HUSF Deficiency Values for the 20NG-4. Various combinations of History depth and subset size are compared

	t10	t25	t50	t100	Avg
US	0.7976	0.7976	0.7976	0.7976	0.7976
HUSF$_{d3}$	0.7359	0.7737	0.7666	0.7664	0.7607
HUSF$_{d5}$	0.7600	0.7482	0.7458	0.7731	0.7568
HUSF$_{d7}$	0.7605	0.7299	0.7312	0.7728	0.7486
	0.7521	0.7506	0.7479	0.7708	

Table 5.11: HUSF Deficiency Values for the R10. Various combinations of History depth and subset size are compared. Deficiency is calculated using accuracy

	t10	t25	t50	t100	Avg
US	0.8793	0.8793	0.8793	0.8793	0.8793
HUSF$_{d3}$	0.8301	0.8262	0.8682	0.8720	0.8491
HUSF$_{d5}$	0.8313	0.8273	0.8415	0.8752	0.8438
HUSF$_{d7}$	0.8323	0.8242	0.8438	0.8609	0.8403
	0.8312	0.8259	0.8512	0.8694	

HKLDF

Table 5.12 reports the deficiency values obtained by performing HKLDF on the 20NG-4 dataset. Values in the table represent the average deficiency obtained over all four sub-problems. It can be seen that for HKLDF, lower deficiency values are obtained for larger History depth and smaller subsets (25).

Table 5.12: HKLDF Deficiency Values for the 20NG-4. Various combinations of History depth and subset size are compared

	t10	t25	t50	t100	Avg
US	0.7976	0.7976	0.7976	0.7976	0.7976
HKLDF$_{d3}$	0.7648	0.7511	0.7604	0.7798	0.7640
HKLDF$_{d5}$	0.7444	0.7565	0.7794	0.8062	0.7716
HKLDF$_{d7}$	0.7675	0.7435	0.7709	0.7529	0.7587
	0.7589	0.7504	0.7702	0.7708	

Table 5.13 gives the deficiency values when HKLDF was employed in the R10 dataset. Deficiency, in this case, is calculated using accuracy. It can be seen that lower deficiency values for HKLDF are recorded when the History depth is medium with a smaller subset size

Table 5.13: HKLDF Deficiency Values for the 20NG-4. Various combinations of History depth and subset size are compared

	t10	t25	t50	t100	Avg
US	0.8793	0.8793	0.8793	0.8793	0.8793
HKLDF$_{d3}$	0.8130	0.8043	0.7944	0.8403	0.8130
HKLDF$_{d5}$	0.8156	0.7947	0.8030	0.7981	0.8028
HKLDF$_{d7}$	0.8091	0.8109	0.8142	0.8195	0.8134
	0.8126	0.8033	0.8039	0.8193	

(25).

Summary

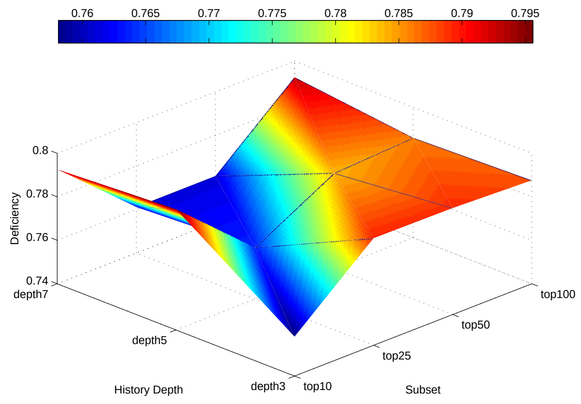
Sensitivity analysis showed that on average small to medium sized subsets ($t = \{25, 50\}$) were optimal in the two datasets considered, while large subsets ($t = 100$) did not produce the lowest deficiency. A larger History depth was advantageous in both HUS and HKLD. Figure 5.10 gives a surface plot of the two parameters for both HUSF and HKLDF in both datasets considered.

5.6.7 Running Time

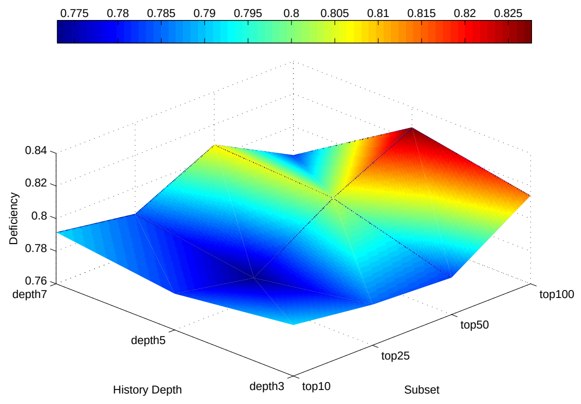
In this section we describe the overall computation overhead as recorded on the experimental apparatus. Experiments were performed on a 2.2GHz Intel Core 2 Duo processor with 2GB of RAM running at 667MHz. The maximum running time of uncertainty sampling was 0.0037 seconds. Averaging across all iterations of active learning US had a mean running time of less than 0.0001 seconds.

History-based query selection strategies will incur extra running time since they perform extra calculations. At a History depth of 3, HUS was found to have an average running time of 0.0011 seconds (averaged over all runs of active learning). HKLD was the most computationally expensive method due to the KL calculations. At a History depth of 3, HKLD had an average time of 0.0500 seconds. Larger History depths added to the computational expense. At depth 7, the average running time for HUS was 0.0012 seconds and for HKLD was 0.0710 seconds.

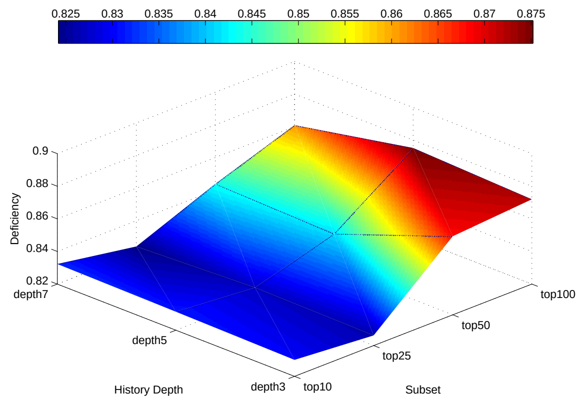
Filtering the pool also increases the computational expense, however, the number of candidates considered decreased substantially, thereby reducing the calculations performed



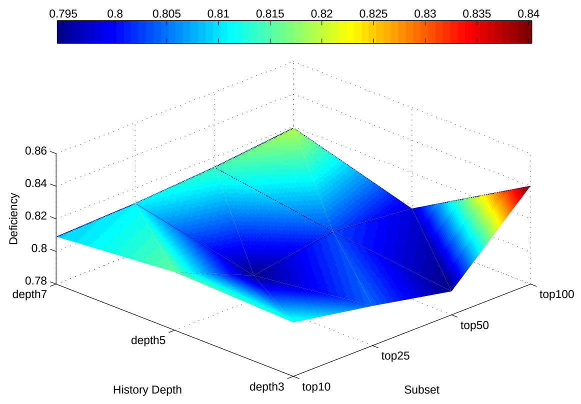
(a) 20NG HUSF



(b) 20NG HKLDF



(c) R10 HUSF



(d) R10 HKLDF

Figure 5.10: Surface plot of the filtered History-based query selection parameters

to select the next query. The average running time of HUSF and HKLDF at History depth of 7 were 0.0010 seconds and 0.0019 seconds respectively. Some computational savings are made at lower History depths with HUSF and HKLDF reporting an average running time at depth 3 of 0.0010 seconds and 0.0017 seconds respectively.

The increase in computational running time of History-based query selection strategies compared to uncertainty sampling is significant. However, HBQS strategies are still entirely practical with running times less than 0.1 seconds. We are confident that the potential benefit of increased effectiveness of active learning, where the overall aim is to reduce labelling effort, mitigates the increased computational expense.

5.7 Related Work

Boosting (Schapire, 1999) is a machine learning technique that focuses the training on the hard to classify examples in the supplied training data. This is somewhat similar to the idea of the History-Based query selection strategy HUS, which selects queries as those unlabelled examples which have constantly been difficult to classify. In particular, selecting queries based on their predictions over successive iterations of active learning is somewhat similar to the re-weighting in Boosting. However, boosting is applied to labelled training data while active learning operates on unlabelled data.

The QBC (Freund *et al.*, 1997) query selection strategy selects queries based on disagreement among committee members. The proposed History Kullback-Leibler Divergence (HKLD) similarly selects queries as those unlabelled examples with maximal disagreement in a committee. However, the committee is constructed in a principally different manner. Furthermore, we utilise Kullback-Leibler Divergence to the mean (Roy & McCallum, 2001) as the metric to measure disagreement.

Query selection strategies discussed in Section 3.4.5 also incorporate information about unlabelled examples into the query selection process. However, such strategies incorporate unlabelled data directly while History-based query selection strategies incorporate the information indirectly via predictions. In addition, query selection strategies which incorporate unlabelled information are generally computationally expensive. Error reduction sampling for example has a computational complexity of $O(|\mathcal{D}_U|^2)$. The use of History makes it very efficient to incorporate extra information about the unlabelled data.

5.8 Discussion

Active learning strategies typically select the query based on the predictions of the most recently induced classifier. This is first order Markov active learning whereby the assumption is that the current training data contains all the relevant information required to select the best query. This chapter introduces History-based query selection strategies (HBQS), which challenge this idea by incorporating predictions from prior iteration of active learning into the query selection process. These non-Markovian strategies attempt to leverage potentially informative classifiers in past iterations to identify those unlabelled examples which are consistently uncertain. We hypothesise that examples which are difficult to classify over many iterations of active learning correspond to those examples near the decision boundary and therefore are the most informative queries to select.

Through empirical evaluation, we wished to show that incorporating information from previous iterations of active learning via the History can help select more informative queries. Two History-based query selection strategies were proposed, each of which used the History in a different manner. One is a single classifier solution, while the other emulates a query by committee technique. History Uncertainty Sampling (HUS) selects queries based on the cumulative uncertainty over the past d iterations. This is an extension of the idea of uncertainty sampling technique which considers uncertainty not only in the current iteration but also in past iterations. History KL Divergence (HKLD) selects queries based on uncertainty in a committee formed using classifiers produced the last d iterations of active learning.

Uncertainty sampling (US) was used as a baseline to compare the performance of the proposed History-based query selection strategies since it is a very effective query selection strategy that works well across many domains. The more familiar baseline of random sampling is also given to show the benefit of the active learning strategies over passive learning.

In both the R10 and the aggregated 20NG-4 results HUS demonstrated marginal increases in effectiveness over the baseline of US in the early iterations of active learning. However, US recovered in the later iterations, where it became optimal. HKLD performed poorly in experiments conducted on both datasets. While there was some evidence of improvement over the baseline in the early iterations on the R10 dataset, this was not evident on the 20NG-4 where it failed to outperform even random sampling.

We attribute the poor performance of these HBQS strategies to the equal emphasis given to all predictions. The current classifier is generally considered to be the most accurate, since it is trained on all known labelled training data. Predictions from the more recent iterations of active learning should be given more emphasis than predictions taken from deep in the History. Filtered HBQS strategies give more emphasis to predictions from the current classifier by selecting queries from a subset of the most uncertain examples. The subset is formed by filtering from the pool the t most uncertain examples as given by uncertainty sampling.

HUSF and HKLDF are filtered variants of the HBQS strategies. HUSF selects queries from the subset based on the cumulative uncertainty of the unlabelled examples. Similarly HKLDF selects queries from the subset based on the disagreement among committee members formed using the classifiers from previous predictions. Results obtained for the filtered HBQS strategies show considerable increases in the effectiveness of active learning compared with both the non-filtered HBQS strategies and the baseline. On average it was found that HUSF was more effective on the 20NG-4 dataset while HKLDF was most effective on the R10 data.

The impact of filtered HBQS strategies tends to be in the early iterations of active learning. This was observed in the R10 and the 20NG-4 datasets whereby the filtered HBQS strategies outperformed US in the iterations directly after the History delay while converging in later iterations of active learning.

History depth and subset size are important parameters for the HBQS strategies. We showed that filtered HBQS was capable of outperforming the baseline of US using any combination of the History depth and subset size parameters. Analysis of the results obtained for the various values used showed that on average larger History depths led to lower deficiency values reported. A medium sized subset was also found, on average, to result in lower deficiency values.

An important characteristic of an active learning strategy is that the number of parameters that require tuning be minimal, since there are typically no validation sets. This was one of the motivations for the use of the multinomial naïve Bayes classifier. While we report optimal values of t and d for the filtered HBQS we did not use a validation set to select these. We believe that a more consistent approach would be an adaptive solution, which would dynamically alter the parameters based on the current performance of active learning. In the early iterations of active learning a large History depth could be used while in later iterations, a small History depth could be employed, emulating uncertainty sampling. This is proposed as future work since, at the time of writing, there does not currently exist an effective metric to measure active learning performance.

The increased performance of History-based approaches are achieved with remarkably little extra computational expense since the strategies reuse predictions made in previous iterations. Some increased memory is required to store the History and there is a significant increase in computation compared to US but overall the proposed HBQS strategies are entirely practical making their use in active learning settings advantageous.

5.9 Summary

This chapter introduced History-based query selection for active learning, an inexpensive method to increase the effectiveness of active learning by incorporating classifier predictions from previous iterations of active learning.

Most query selection strategies are Markovian, whereby they select queries based only on the current classifier. Moreover, the predictions used in every iteration of active learning are ephemeral and are duly discarded after the query has been selected. We investigated non-Markovian History-based query selection strategies which selected queries based on predictions given in both current and past iterations of active learning.

A data structure known as the History is used to store all predictions given on the remaining unlabelled examples in every iteration of active learning. History-based query selection strategies select unlabelled examples incorporating information from the History. Empirical evaluation on benchmark text corpus demonstrated that incorporating historical information using a filtered scheme led to increased effectiveness of active learning compared to uncertainty sampling.

Difficulties were encountered when attempting to use a k-NN classifier in text categorisation active learning problems. In Chapter 6 we investigate this problem further and suggest methods to overcome this problem. The filtered HBQS schemes discussed in this chapter also motivated work described in Chapter 7, in particular pre-filtering.

Chapter 6

Dimensionality Reduction for Active Learning

6.1 Introduction

In this chapter we highlight the difficulties arising from performing dimensionality reduction in an active learning setting and investigate possible solutions. This research was motivated by a desire to use the k-NN classifier in active learning experiments for text categorisation. The k-NN was shown to be a high-performance classifier for text categorisation when suitable dimensionality reduction is employed (Yang & Liu, 1999). However, if dimensionality reduction is not performed, the k-NN can perform poorly due to the *curse of dimensionality* (Mitchell, 1997).

Automated text categorisation utilises the vector space model representation, which results in very high-dimensional data for natural language text. The number of dimensions corresponds to the number of unique terms in the corpora. Sparsity is a consequence of the high dimensionality where the number of non-zero elements of a feature vector for a document is likely to be very small ($\leq 5\%$). Synonymity, polysemy and homonymy all exacerbate the problem further.

High-dimensional data has a dramatic impact on learning systems, increasing the computational overhead and in some cases limiting the choice of classifier to those which are capable of operating successfully in high-dimensional data.

Dimensionality reduction techniques are used in machine learning to reduce the number of features used to represent the data. The most successful dimensionality reduction techniques for text categorisation are supervised feature selection methods (Yang & Pedersen, 1997). However, performing supervised feature selection is a significant problem in an active learning setting since the majority of supplied training data are unlabelled.

We investigate the application of unsupervised dimensionality reduction to active learning on text categorisation problems. The application of unsupervised dimensionality reduction will bring all the benefits of reduced computational and storage overheads while also allowing for greater flexibility in the choice of classifier used in active learning. We first highlight the issues surrounding performing dimensionality reduction in an active learning setting, before examining which established techniques are applicable.

Section 6.2 discusses the background of performing dimensionality reduction in an active learning setting. Suggested unsupervised dimensionality reduction techniques for use in active learning settings are discussed in Section 6.3. Empirical evaluation on benchmark corpora is given in Section 6.4 and discussion on the result obtained are given in Section 6.5.

6.2 Background

In text categorisation tasks supervised feature selection techniques (see Section 2.5) are commonly used to reduce dimensionality. Yang & Pedersen (1997) have shown such techniques to yield significant reductions in dimensionality with no loss in accuracy of the resulting classifier and in some cases to show a small increase in accuracy.

Supervised feature selection techniques explicitly utilise label information from the supplied training data. Supervised learning is supplied with a large amount of labelled training data hence supervised feature selection can be successfully employed. However, a problem exists for active learning settings, where the majority of the training data are unlabelled, therefore supervised techniques cannot be readily employed.

Within the literature two solutions to the problem of dimensionality reduction in active learning settings have emerged. The first approach is to circumvent the problem by using pre-labelled benchmark corpora. Supervised feature selection can be applied prior to experimentation of active learning techniques (Hoi *et al.*, 2006). However, in real world applications the label information will not available *a priori* thereby limiting the applicability of this approach.

In the second approach dimensionality reduction is not performed. Experiments are conducted on very high-dimensional text data ($10^2 \ldots 10^5$ features is not uncommon). Operating with such high-dimensional data has several implications including significantly increased computational overhead. Furthermore, the choice of classifier used is restricted to those which can successfully operate in high-dimensional data. The potential benefits offered by dimensionality reduction, including reduced storage and computational overheads, are never realised. In the next section we explore an alternative solution for reducing the dimensionality of text categorisation data in active learning problems.

6.3 Unsupervised Dimensionality Reduction

Unsupervised dimensionality reduction techniques offer the ability to significantly reduce dimensionality without the need for labelled training data. Unlike supervised dimensionality reduction, label information is not explicitly required by unsupervised techniques. Two well established unsupervised dimensionality reduction techniques are considered for use in active learning settings below.

6.3.1 Document Frequency Global (DFG)

Document frequency (Apté *et al.*, 1994; Sebastiani, 2002) is a straightforward and efficient feature selection technique where features are chosen based on the number of documents within the corpus in which they occur. Preference is given to those terms which have a high frequency within the corpus.

The preference for high frequency features is a somewhat surprising criteria for a dimension reduction technique since this is in contrast to beliefs from Information Retrieval, where low frequency features are assumed to be informative while high frequency features are assumed to be irrelevant. However, many of the low frequency terms appear only once in the corpus and offer little discriminative power. As such their removal will not degrade the performance of a resulting classifier. Additionally high frequency topic neutral terms (stopwords) are removed before document frequency is performed. What remains are terms which appear with medium frequency in the corpus.

Yang & Pedersen (1997) have shown that dimensionality can be reduced by a factor of ten (removal of 90 percent of terms) using document frequency with no loss in the accuracy of the resulting classifier. Despite its simplicity, the performance of document frequency is comparable to the best performing feature selection methods when dimensionality is reduced by up to a factor of ten. Advanced techniques such as Information Gain are superior when more than 90 percent of terms are removed.

Document Frequency performed Globally

As discussed in Section 2.5.4 dimensionality reduction can be performed using either a *local* or *global* policy. Document frequency behaves differently depending on which policy is used.

Local document frequency is where a reduced set of features are selected based on the document frequency of features for each individual category. In practice this can mean that each document will have a distinct representation for every category. This is referred to as context-sensitive since the representation used is dependent on the category under consideration.

Global document frequency is where a reduced set of features are selected using document frequency irrespective of category. Only one reduced set of features will be used for all categories. This is referred to as context-free since the representation is not dependant on the category.

Performing document frequency using a local policy requires explicit knowledge of label information and is therefore not suitable for an active learning setting where the majority of data are unlabelled. Conversely, performing document frequency using a global policy is an unsupervised dimensionality reduction technique and therefore is applicable to an active learning setting.

6.3.2 Principal Components Analysis (PCA)

Principal Components Analysis, as discussed in Section 2.5.2 is a feature extraction method which projects high-dimensional data on to a new low-dimensional space with minimum loss of information. It is an unsupervised feature extraction technique, which makes it suitable for use in an active learning setting.

PCA has been used extensively for image processing while a related technique called Latent Semantic Indexing (Deerwester *et al.*, 1990) has been used in the Information Retrieval domain. LSI employs singular value decomposition on the Term $\times$ Document matrix. However, the result is the same as performing PCA on a covariance matrix.

A criticism levelled at PCA (and all feature extraction techniques) is that the new features are not human readable. This is in contrast to feature selection, which uses a subset of the original attributes to express the data therefore the features are still readable after dimensionality reduction has been applied. Feature extraction constructs new features from the original attributes often combining several into a single new feature. The result is that while the new feature is very expressive, it is illegible to humans.

PCA for Text Categorisation.

Given a set of m examples, principal component analysis will first centre the data by constructing the mean of the data μ (as given in (6.1)) and subtracting this from each example. Centering the data is not essential but can remove irrelevant variance as it reduces the overall sum of the eigenvalues.

$$\mu = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i \tag{6.1}$$

The covariance matrix (C) is constructed as the dot product of the centered examples as given in (6.2) (centering is incorporated into the construction of the covariance matrix).

$$C = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T \tag{6.2}$$

The eigenvalue problem (6.3) is solved by performing eigenvalue decomposition on C. The solution is a set of eigenvectors (v) and their associated eigenvalues (λ).

$$Cv = \lambda v \tag{6.3}$$

The d largest eigenvalues are sorted in descending order ($\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_d$) and their associated eigenvectors stacked to form the transformation matrix $W = [v_1, v_2, v_3, \ldots, v_d]$. For a given example $\mathbf{x}$ it can be transformed into the PCA reduced space by (6.4).

$$y = W^T\mathbf{x} \tag{6.4}$$

The original set of examples and any subsequent data can be transformed to the d-dimensional space using (6.4).

Variance

The value of d is an important factor in the success of PCA. Since the eigenvalues correspond to the amount of variance accounted for by their associated eigenvector, the proportion of variance accounted for by the first d eigenvectors can be calculated as given in (6.5):

$$\frac{\lambda_1 + \lambda_2 + \ldots + \lambda_d}{\lambda_1 + \lambda_2 + \ldots + \lambda_d + \ldots + \lambda_{|m|}} \tag{6.5}$$

.

6.4 Empirical Evaluation

Experiments were conducted to examine the effect of the proposed unsupervised dimensionality reduction techniques on the performance of active learning.

6.4.1 Experiment Settings

In order to assess the impact of dimensionality reduction, three representations of the data were used in the experiments:

FULL This representation is where no dimensionality reduction is performed. All features obtained from parsing the corpora were used. Stopwords and punctuation were removed and stemming was performed using the Porter stemming algorithm (Porter, 1980).

DFG This representation was constructed by applying document frequency with a global policy to the FULL representation. The data is expressed by a subset of the original features. Only 10 percent of the most frequent features were retained.

PCA This representation was constructed by transforming the FULL representation using principal component analysis. In our experiments we choose the leading d components which account for 90 percent of the variance in the data. Eigenvalue decomposition results in m eigenvalues eigenvector pairs, where m is the number of documents. The value of d is found by iteratively increasing d until the output of (6.5)

In the case of DFG and PCA, the reduced (or transformed) feature sets were found using the seed data and the unlabelled data only. The test data was subsequently re-expressed using the reduced set of features. This ensures that no information is leaked from the testing data.

Active Learning

Active learning performed on the 20NG-4 was seeded with three positive and three negative examples. Similarly, for the R10 dataset active learning was seeded with ten positive and ten negative examples of the class. In each iteration just a single query is selected, which corresponds to the finest granularity. Active learning was not terminated until the pool of unlabelled examples was exhausted.

Datasets

Two standard benchmark corpora previously described in Section 4.2, namely the R10 and the 20NG-4 corpora were used. Similar to (Schohn & Cohn, 2000) a 50/50 train/test split was used in the 20NG-4 experiments. The data was randomly divided into two sets where one was used as training while the other was held back for testing. The R10 used the ModApté split where ten binary classification problems were constructed in a one-v-rest manner (see Section 2.6.5).

Accuracy (see Section 2.8) is used as the performance metric for the 20NG-4 dataset. However, due to the unbalanced class distribution in the R10 dataset the F_1 value of precision (π) and recall (ρ) was chosen as the performance metric, (where $F_1 = \frac{2\pi\rho}{\pi+\rho}$). F_1 was calculated using both macro-averaged and micro-averaged variants of precision and recall.

Classifier Details

Yang & Liu (1999) demonstrated the k-NN to be a high-performance classifier for text categorisation. However, it is sensitive to curse of dimensionality. While it is not commonly used for active learning text categorisation tasks, we chose the k-NN since it is expected to benefit greatly from dimensionality reduction. The k value was fixed at 3 in our experiments. The optimal value for k is typically found using validation data, which is not available in active learning. A low value for k is also important for the early iterations of active learning since the number of training examples can be very low.

The Spider toolbox for Matlab (see Section 4.3.2) was used to perform the experiments due to the fact it can easily perform principal components analysis and has a k-NN classifier. However, the k-NN in Spider did not fully meet our requirements. We extended the functionality of the k-NN by transforming the output of the classifier into a class membership probability estimate $P(y|\mathbf{x})$. This was done based on the inverse distance of the neighbours to the query example in a manner that is similar to that of the WEKA implementation.

6.4.2 Dimensionality

Table 6.1 shows the dimensionality of each representation used in the experiments. DFG was configured to retain only the top 10 percent of the most frequently occurring features while PCA was configured to adaptively select the number of features which account for 90 percent of the variance in the data.

Table 6.1: Dimensionality of each representation

	FULL	DFG	PCA
R10	19868	1986	309
AR	1427	142	194
GX	1953	195	23
WH	1968	196	232
BC	1985	198	210

The number of features retained is an important parameter for document frequency. We performed an experiment to assess the sensitivity of this parameter on the accuracy obtained using passive supervised learning. The number of retained features was varied from 1 percent to 100 percent and the accuracy obtained from a k-NN operating in the resulting representation was observed.

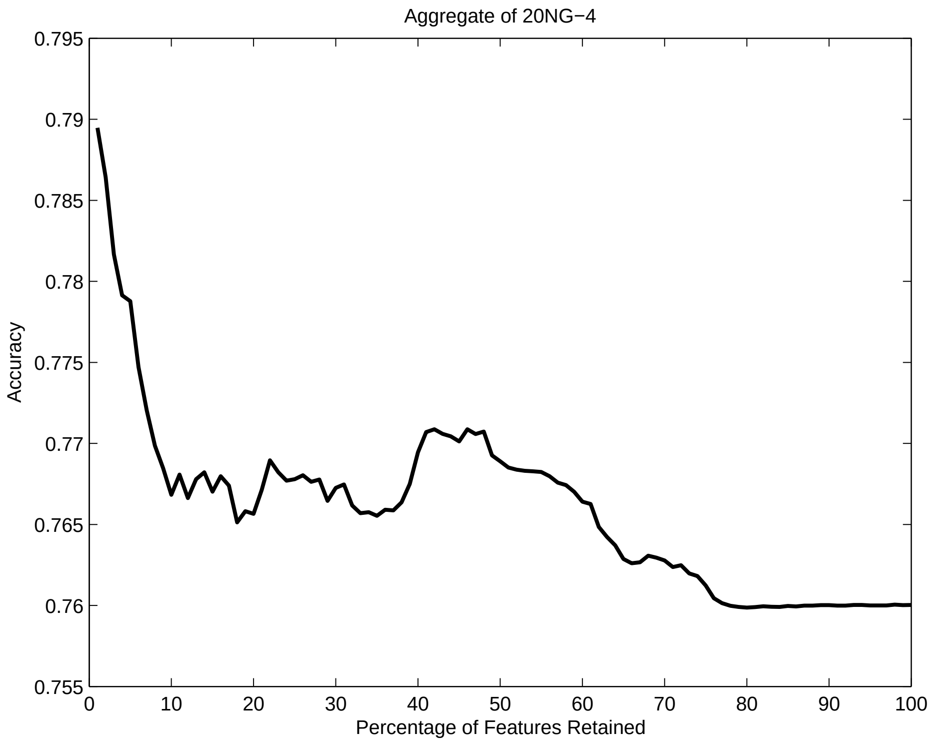
Figure 6.1 shows the results obtained from the 20NG-4 dataset. It is observed that in the individual sub-problems the optimal number of features retained can vary. However, considering the aggregate of the 20NG-4, it can be seen that accuracy is optimal when a small percentage of the most discriminating features are retained (the maximum point resides in the interval of 1 to 10 percent).

Figure 6.2 shows the results obtained from the R10 dataset. Similarly, it can be seen that accuracy is optimal when a small percentage of the most discrimination features are retained.

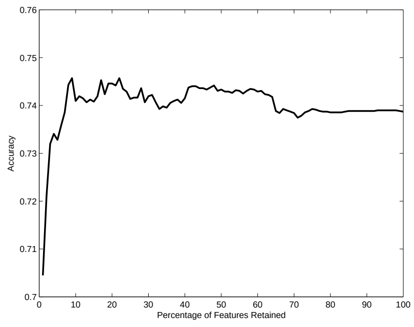
6.4.3 Learning Curves

Experiments were conducted to establish if unsupervised dimensionality reduction can benefit a k-NN in the active learning process. To show this we made comparisons between a k-NN operating on the full feature set (FULL) and a k-NN operating on either of the dimensionality reduced data (DFG or PCA).

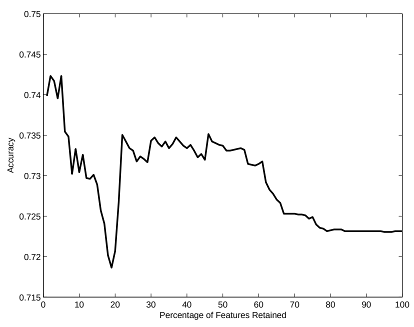
Figure 6.3 shows the learning curves for active learning using each of the representations for the R10 dataset. The FULL representation is the worst performing achieving the lowest



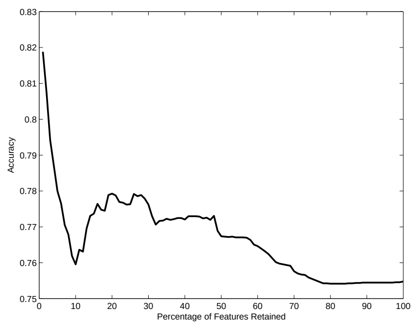
(a) Aggregate 20NG-4



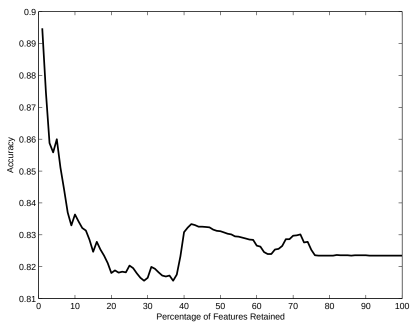
(b) AR



(c) GX

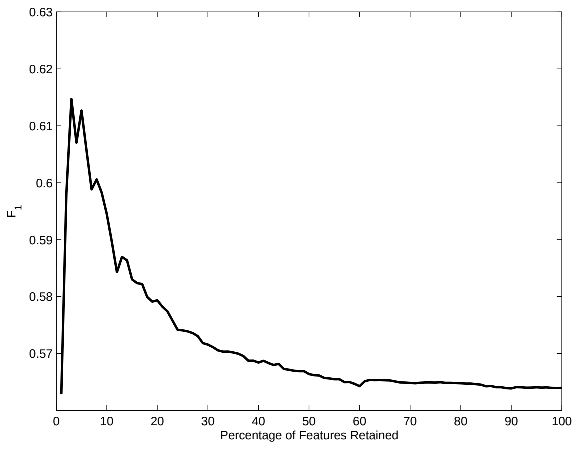


(d) WH

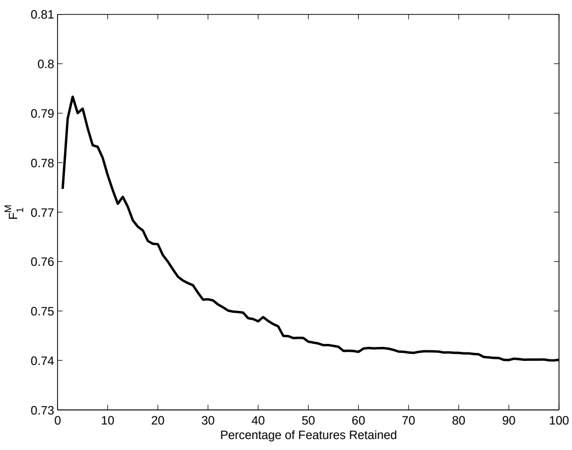


(e) BC

Figure 6.1: Document frequency aggressiveness for 20NG-4 obtained using a 3-NN classifier.



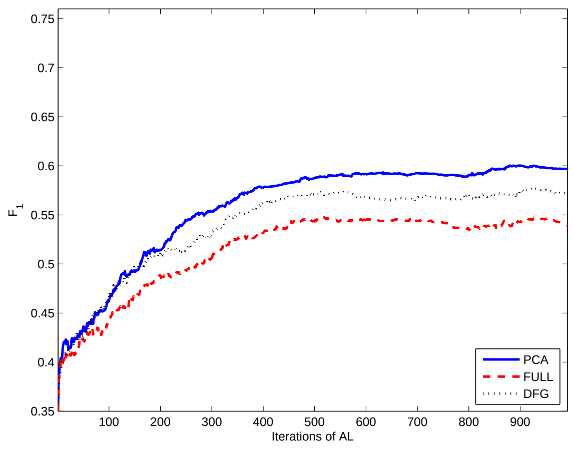
(a) Macro F_1



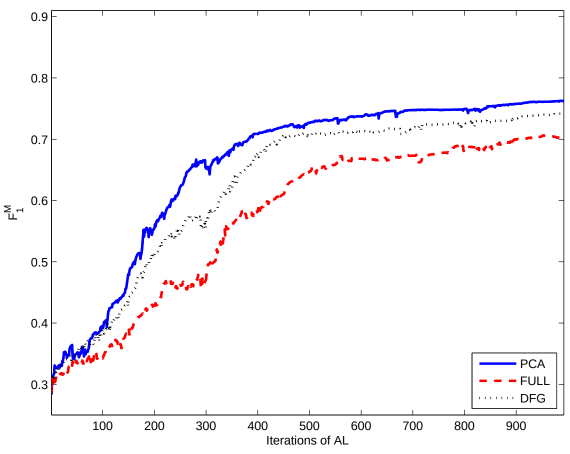
(b) Micro F_1

Figure 6.2: Document frequency aggressiveness for R10 obtained using a 3-NN classifier.

overall F_1 values in practically all iterations of active learning. This can be attributed to the poor performance of the k-NN in high-dimensional data.



(a) Macro F_1



(b) Micro F_1

Figure 6.3: Learning curves for R10 obtained using a 3-NN classifier in three representations of the data: FULL contains all possible features, PCA is the d- dimensional representation given by principal components analysis and DFG is the reduced representation given by the top 10 percent of features as given by document frequency performed with a global policy

Increases in F_1 are achieved when dimensionality is reduced using both unsupervised techniques. Document frequency with a global policy (DFG) achieves a higher overall F_1 value than FULL, while the best performance is obtained when dimensionality is reduced using principal components analysis (PCA).

Similar results are obtained from the 20NG-4 dataset. Figure 6.4 shows the learning curves for the 20NG-4. The performance of the k-NN is weakest when the FULL representation of the data is used. The k-NN operating in the DFG representation performed better than the k-NN operating in the FULL representation while the best performance was achieved by the k-NN operating in the PCA representation.

6.4.4 Labelling Effort

An alternative way in which to assess the impact of dimensionality reduction is to establish how long it takes an active learner in each representation to achieve the accuracy obtained from passive supervised learning using the entire pool in the FULL representation. Table 6.2 shows the number of queries required in order to achieve the passive supervised learning level for the R10 dataset.

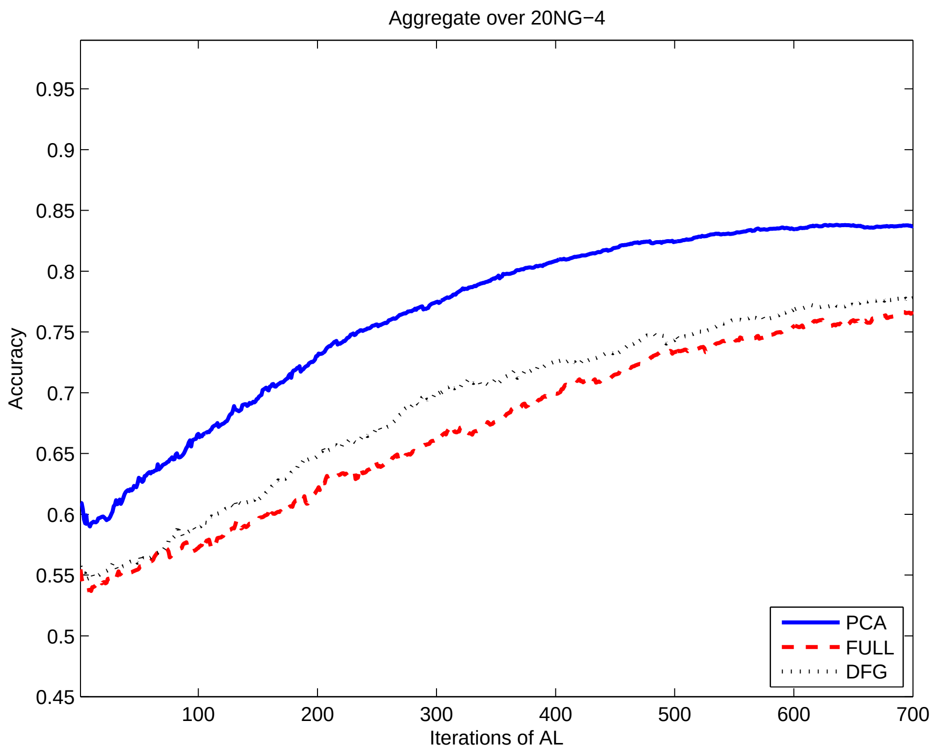
Table 6.2: Iterations of active learning required to achieve supervised passive learning performance on R10. Percentage of pool labelled is given in brackets

	Full	DFG	PCA
MacroF_1	454 (46%)	324 (33%)	243 (25%)
MicroF_1	923 (93%)	444 (45%)	385 (39%)

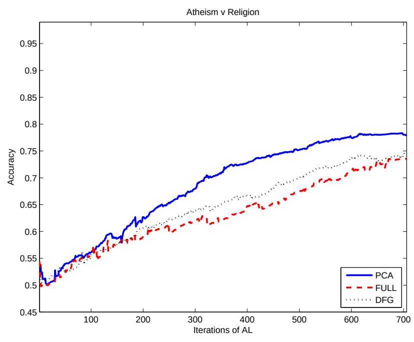
The application of unsupervised dimensionality reduction can be seen to reduce the number of queries required. PCA offers the greatest reduction in labelling effort, requiring only 25 percent of the pool to be labelled in the case of macro F_1 and 39 percent in the case of micro F_1. Reductions are also achieved when dimensionality is reduced using DFG, however the reductions are not as pronounced as those given by PCA.

Results from the 20NG-4 dataset exhibit similar characteristics to those found in the R10 dataset. Table 6.3 shows the number of queries made before the supervised passive learning rate was achieved for the 20NG-4 dataset. The application of dimensionality reduction using DFG led to a decrease in the number of queries required in three of the four problems.

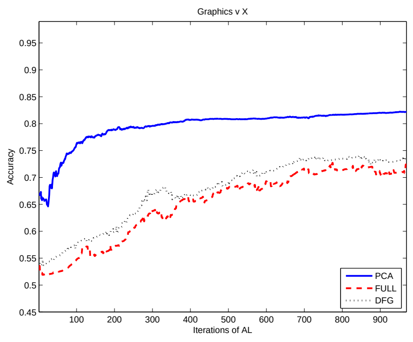
The k-NN operating in the PCA representation was the top performer. In three of the four problems the number of queries was approximately halved while in the GX problem only 58 queries (6 percent of the pool) required labels. In contrast, 773 queries (80 percent of the pool) were required to achieve the supervised passive learning rate when the k-NN was operating in the FULL representation.



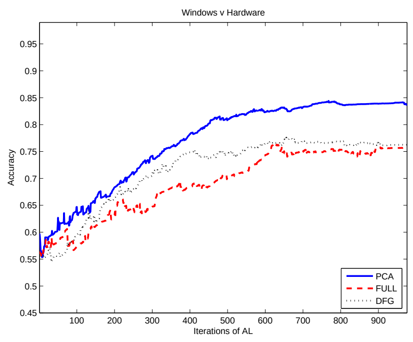
(a) Aggregate 20NG-4



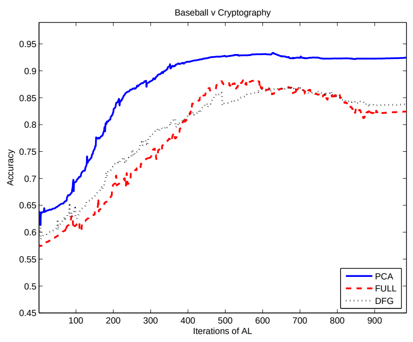
(b) AR



(c) GX



(d) WH



(e) BC

Figure 6.4: Learning curves for 20NG-4 obtained using a 3-NN classifier in three representations of the data: FULL contains all possible features, PCA is the *d*- dimensional representation given by principal components analysis and DFG is the reduced representation given by the top 10 percent of features as given by document frequency performed with a global policy

Table 6.3: Iterations of active learning required to achieve performance of supervised learning for 20NG-4. Percentage of pool labelled is given in brackets

	Full	DFG	PCA
AR	672 (95%)	597 (85%)	416 (59%)
GX	773 (80%)	661 (68%)	58 (6%)
WH	616 (64%)	553 (57%)	342 (35%)
BC	408 (42%)	428 (44%)	201 (21%)

6.4.5 Fixed Stopping Criteria

Active learning is used in domains where the cost of acquiring label information is high. For this reason it is common to use a fixed stopping criterion (see Section 3.3.4). If active learning is stopped after 250 iterations the increase in F_1 on the R10 dataset using the PCA representation (compared to FULL) is (Macro) **0.0496** (Micro) **0.1561** while the increase in F_1 of the DFG representation compared to FULL is (Macro) **0.0219** (Micro) **0.0802**. Bold text indicates statistical significance ($\alpha = 0.05$).

Similarly for the 20NG-4 dataset the increase in accuracy when active learning is performed in the PCA representation is: (AR) **0.0574** (GX) **0.1883** (WH) **0.0643** (BC) **0.1517** while the increase in accuracy of DFG compared to FULL is: (AR) 0.0255 (GX) **0.028** (WH) 0.0316 (BC) 0.0386. Bold text, again, indicates statistical significance ($\alpha = 0.05$).

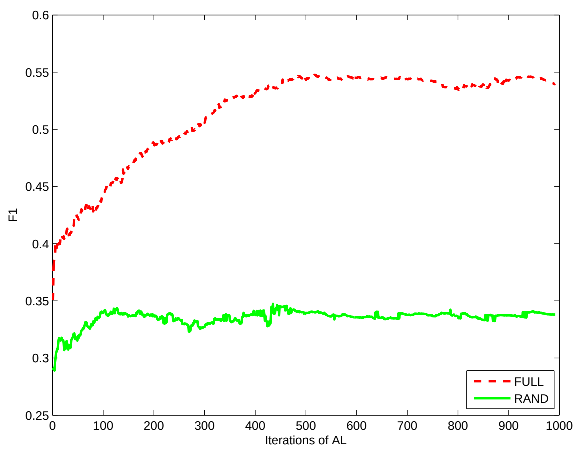
6.4.6 Random Feature Selection

It could be the case that the observed improvements in performance is due simply to the positive effect of reducing the number of features has on the classifier, irrespective of the quality of the reduced set. In order to test that possibility we compared performance of the baseline to random feature selection (Forman, 2003).

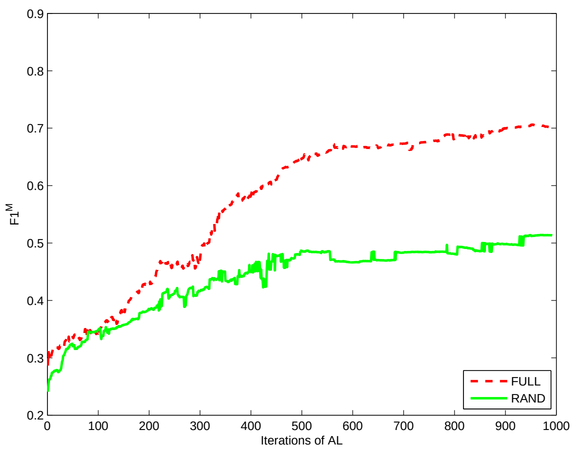
Figure 6.5 plots the performance of random feature selection (RAND) with respect to the original feature set (FULL) on the R10 dataset. The performance of Rand is significantly worse which shows features selected by the unsupervised techniques are discriminative.

6.4.7 Supervised Dimensionality Reduction

To establish how competitive the unsupervised methods are with a top performing supervised dimensionality reduction technique, we conducted active learning using a k-NN operating in a representation described using just the top 500 features as given by Information Gain (IG500)



(a) Macro F_1



(b) Micro F_1

Figure 6.5: Performance of RAND compared to FULL. Similar to the other dimensionality reduction techniques in this chapter, random feature selection was performed in each trial of the experiment and ten trials were conducted. Results shown are the average of the ten trials. Iterations of active learning is given on the X-axis and the F_1 is given on the Y-axis.

(Hoi *et al.*, 2006).

Figure 6.6 shows the results obtained from experiments conducted on the 20NG-4 dataset. DFG offers some improvements over the FULL representation. However, PCA is very competitive with the supervised technique and in some cases (e.g. Figure 6.6(c)) outperforms it.

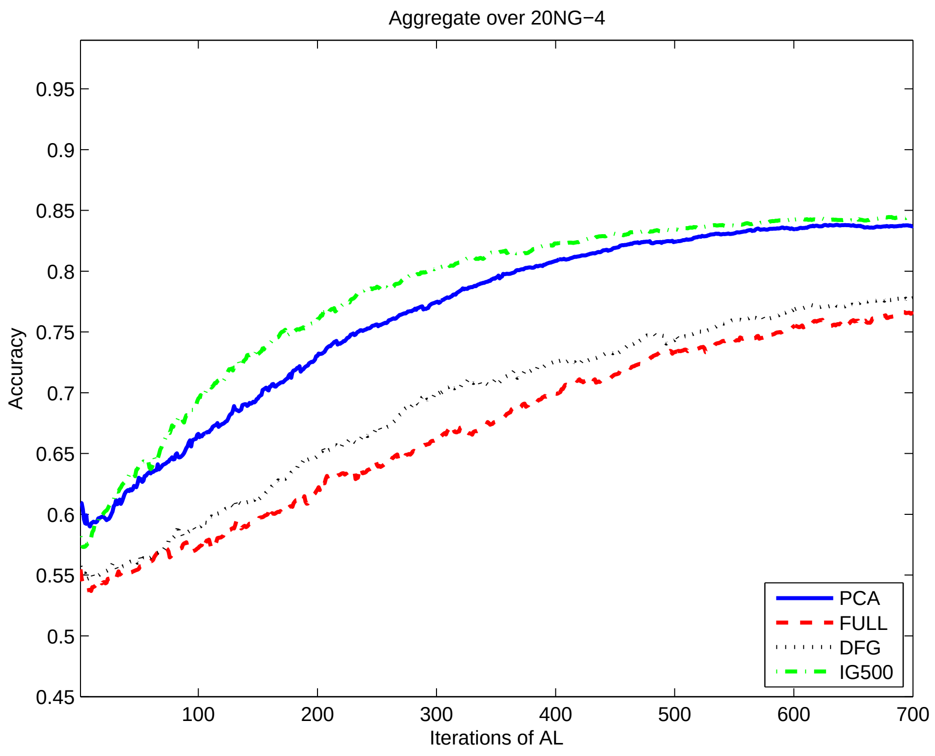
The strong performance of PCA is encouraging and we believe this is due to the adaptive way in which the number of features retained are selected. However, the computational overhead of performing PCA is not to be underestimated and must be considered when choosing the dimensionality reduction technique.

6.4.8 Alternative Classifiers

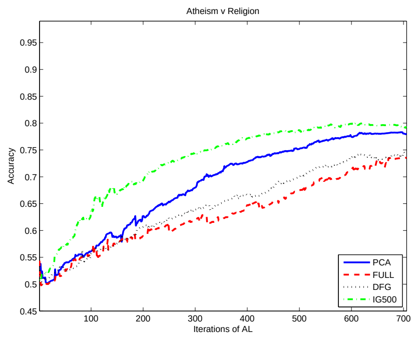
To assess the improvement achieved by the application of unsupervised dimensionality reduction, the k-NN is compared against a number of alternative classifiers, namely a support vector machine and a multinomial naïve Bayes classifier. The support vector machine is implemented using the Spider implementation[1] while the multinomial naïve Bayes is from WEKA.

Figure 6.7 shows that while the SVM still maintains a clear advantage the combination

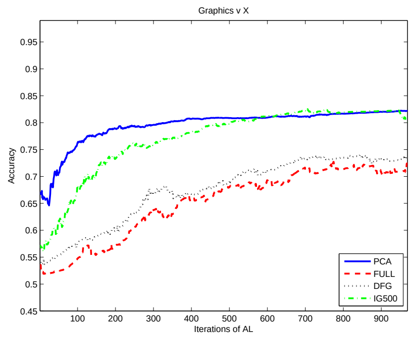
[1]the andre optimiser was used



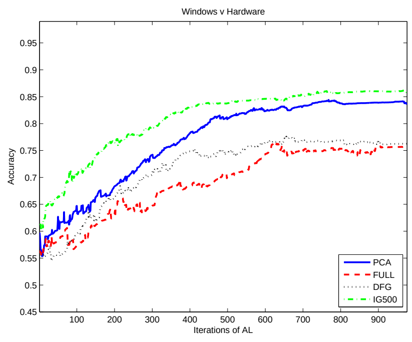
(a) Aggregate 20NG-4



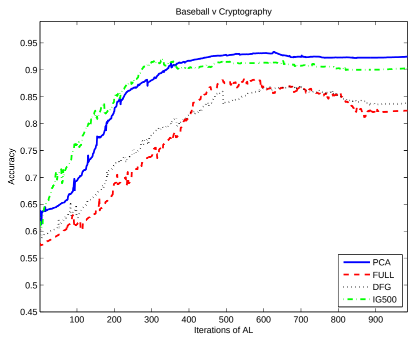
(b) AR



(c) GX



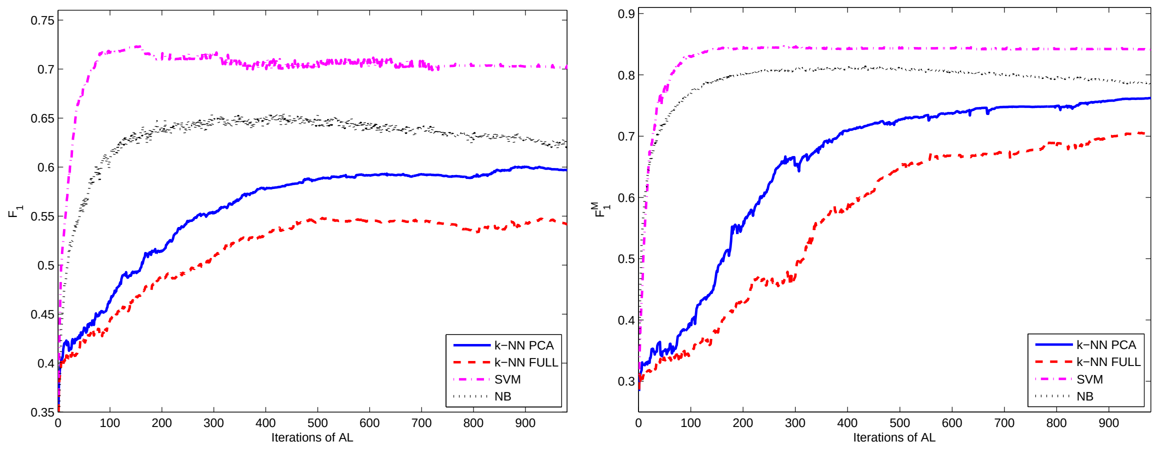
(d) WH



(e) BC

Figure 6.6: Comparison of supervised feature selection using information gain retaining the top 500 features (IG500) to unsupervised dimensionality reduction using DFG and PCA. Results obtained using a 3-NN classifier

of k-NN with unsupervised dimensionality reduction does offer more competition than a k-NN operating within the FULL representation. While there is improvement, the lift in performance offered by alternative classifiers should not be ignored.



(a) Macro F_1 (b) Micro F_1

Figure 6.7: Performance of SVM and naïve Bayes classifiers compared to k-NN on the R10 Dataset. Iterations of active learning is given on the X-axis and the the F_1 is given on the Y-axis.

6.5 Discussion

High-dimensional data is common in text categorisation problems. Supervised feature selection techniques, which are typically used to reduce dimensionality in text categorisation problems cannot be employed since the majority of data supplied to the learner are unlabelled. We investigated unsupervised techniques to reduce dimensionality of text categorisation data in active learning settings without adversely effecting the accuracy of the induced classifier. Reducing dimensionality provides benefits by reducing computational and storage overheads but also allows greater flexibility in the selection of classifier used in active learning.

Two unsupervised dimensionality reduction techniques were investigated for use in active learning settings, namely document frequency performed globally and principal components analysis. Empirical evaluation in Section 6.4.3 shows that the suggested unsupervised dimensionality reduction increases the performance of active learning using a k-NN. The k-NN was chosen for use in the experiments as it suffers from the curse of dimensionality and can perform poorly in high-dimensional data. However, it exhibits state-of-the-art performance when operating in a reduced feature set given by an appropriate supervised feature selection (Yang & Liu, 1999)

Comparisons were made between active learning operating in the original FULL feature set and active learning operating in the reduced feature sets as given by DFG and PCA respectively. Reducing dimensionality using DFG offered some performance increase compared with the FULL representation. However, the k-NN performed best in the representation as given by performing dimensionality reduction using principal component analysis.

PCA was found to outperform DFG in all experiments conducted. We believe this was due to PCA adaptively choosing the number of features to retain (the d Eigenvectors which account for 90 percent of the variance in the data). Conversely, DFG statically reduced the dimensionally of the data. In the 20NG-4 experiments, for instance, PCA reduced dimensionality to just 23 features in the GX problem. Subsequently classification in the reduced feature set was considerably easier leading to higher performance of active learning and a large reduction in the labelling effort (58 compared to the baseline of 773).

It may be argued that the increased performance of active learning is simply due to the reduction in features, irrespective of the method used to reduce dimensionality. Section 6.4.6 demonstrated the performance increase was as a result of the selection of discriminative features since random feature selection failed to achieve similar results to DFG.

Section 6.4.7 compares the relative performance of the two unsupervised techniques to a top performing supervised technique on the 20NG-4 dataset. Yang & Pedersen (1997) demonstrated Information Gain as being a top performing supervised dimensionality reduction technique for text categorisation problems. Reducing dimensionality using PCA was seen to be competitive with Information Gain retaining only the top 500 ranked features (Hoi *et al.*, 2006). This demonstrates that the unsupervised techniques can deliver significant reductions in dimensionality comparable with the best performing supervised techniques.

While PCA was shown to be best performing unsupervised dimensionality reduction technique for the k-NN, the associated computational overhead is high, which limits its applicability to very large datasets. Conversely, document frequency offers some increased performance at much lower computational expense. Active learning prefers to incur computational expense in favour of manual labelling, therefore the additional cost of principal component analysis may be justified. If the cost of PCA is too high, alternative classifiers such as those compared in Section 6.4.8 could be used, since the SVM and naïve Bayes classifiers operate well in high-dimensional data and do not require dimensionality reduction.

6.6 Summary

Motivated by the desire to use the k-NN classifier for active learning in text categorisation problems, we highlight the specific problems of performing dimensionality in an active learning setting. High-dimensional data is common in text categorisation problems due to the size of the vocabulary in natural text documents. Dimensionality reduction is commonly employed to shrink the size of the data while maintaining all the salient information contained in the original representation.

Supervised dimensionality reduction is typically used for text categorisation problems which assumes the presence of label information. This is a significant problem in an active learning setting where the majority of the data is unlabelled.

This chapter investigated unsupervised dimensionality reduction for use in active learning settings. It was found that unsupervised dimensionality reduction clearly benefits the k-NN classifier making it more competitive. Reducing dimensionality using principal components analysis was found to be competitive with a top-performing supervised feature selection technique (commonly used in text categorisation). The use of unsupervised dimensionality reduction techniques allows for greater flexibility in the choice of classifier used in active learning.

Chapter 7

Adaptive Pre-Filtering Error Reduction Sampling

7.1 Introduction

In this chapter we investigate the pre-filtering optimisation technique for error reduction sampling (ERS) (Roy & McCallum, 2001; Lindenbaum *et al.*, 1999). The computational overhead of error reduction sampling is extremely high, requiring $O(n^2)$ classifications to select each query, where $n = |\mathcal{D}_U|$. We hypothesise that pre-filtering optimisation will increase the quality of queries selected by error reduction sampling as well as reduce computational overhead by allowing for smaller subsets to be used. Furthermore, we propose a novel adaptive pre-filtering technique which dynamically switches among a number of fixed pre-filtering techniques based on performance in past iterations.

Roy & McCallum (2001) have proposed subset optimisation as a technique to reduce the computational expense of ERS. Subset optimisation reduced computational overhead by restricting the number of unlabelled examples considered by error reduction sampling. The most popular technique for constructing the subset is random sub-sampling (ERS-RS), which randomly selects unlabelled examples from the pool to form the subset. We investigate an alternative technique called pre-filtering, which constructs the subset in a more principled manner. Pre-filtering populates a subset with unlabelled examples filtered from the pool using a query selection strategy. The pre-filtered subset will contain a large amount of potentially informative unlabelled examples compared to sub-sampling. We hypothesise that pre-filtered subset optimisation will increase the quality of queries selected by error reduction sampling as well as reduce computational overhead by allowing for smaller subsets to be used.

A novel adaptive pre-filtering technique is proposed (ERS-FPL) that constructs the subset using a number of alternative query selection strategies. Adaptive query selection (Baram *et al.*, 2004; Pandey *et al.*, 2005; Osugi *et al.*, 2005; Pandey *et al.*, 2005) dynamically switches among a number of fixed query selection strategies, where the strategy that has performed best in the past iterations of active learning is used to select the query in the current iteration. We extend this idea to the task of pre-filtering, where a single fixed query selection strategy is chosen to construct the subset based on performance in prior iterations. ERS-FPL should perform as well as the best performing fixed pre-filtering technique.

Section 7.2 gives some motivation for error reduction sampling, pre-filtering and the proposed adaptive pre-filtering technique. Error reduction sampling is described in detail in Section 7.3. Subset optimisation, including random sub-sampling and pre-filtering, is discussed in detail in Section 7.4, while Section 7.5 describes the proposed adaptive pre-filtering technique. Details of the experimental settings and results from the empirical evaluation conducted are given in Section 7.6. Finally, discussions on the findings obtained are given in Section 7.7.

7.2 Background

The success of active learning depends directly on the ability to identify and label only the most informative examples. In pool-based active learning (see Section 3.3.2) it is responsible for the selection of informative queries from a large pool of unlabelled examples. Informative examples are those which maximise learning of the *target concept* and result in the lowest future error on test examples. An ideal query selection strategy is to select queries based on reduction of future error, since this is the metric on which active learning performance is ultimately evaluated.

Section 3.4.5 discusses two query selection strategies proposed in the literature which select unlabelled examples based on the criterion of directly reducing future error. In this chapter we explore error-Reduction Sampling (ERS) (Roy & McCallum, 2001), which selects queries that directly minimise *expected* future error. ERS is not frequently used in active learning due to its large computational overhead - selecting a query requires $O(|\mathcal{D}_U|)$ classifications. Roy & McCallum (2001) have proposed *subset optimisation* as a way to reduce the computation expense of ERS. Subset optimisation applies ERS to only a small fraction of examples from the pool.

The most popular technique for constructing the subset is random sub-sampling (ERS-RS), which randomly select unlabelled examples from the pool to form the subset. This has two main drawbacks. First, the subset must be large enough to provide a representative sample of the pool, restricting the computational reduction possible. Second, there is a chance that the randomly selected example contains uninformative examples wasting effort as ERS is applied to them.

Pre-filtering, also proposed by (Roy & McCallum, 2001), constructs the subset with unlabelled examples filtered from the pool using a query selection strategy. In this way the subset will contain potentially more informative unlabelled examples compared to sub-sampling. Pre-filtering will not lead to a decrease in computational overhead with respect to random sub-sampling. However, it can increase the quality of the examples in the subset allowing for smaller subset to be used and reductions in labelling effort by improving active learning learning rates.

Active learning typically employs a single fixed query selection strategy throughout its entire life-cycle. Empirically it has been found (Donmez *et al.*, 2007; Xu *et al.*, 2003) that different query selection strategies are desirable in different iterations of active learning. This motivates the need for an adaptive query selection strategy that can balance exploration and exploitation in order to increase the accuracy of active learning. Online adaptive query selection strategies (Baram *et al.*, 2004; Pandey *et al.*, 2005; Osugi *et al.*, 2005; Pandey *et al.*, 2005) dynamically switch among a number of fixed query selection strategies, where at each iteration of active learning the best performing strategy is used to select the query. We develop this idea to propose a novel adaptive pre-filtering technique which adaptively chooses the query selection strategy used to filter examples from the pool. The choice is based on the performance of each strategy when it was chosen in past iterations of active learning.

7.2.1 Related Work

ERS has been shown to be a very strong performing query selection strategy (Culver, 2006; Roy & McCallum, 2001). Due to the computational overhead, random sub-sampling has been typically employed when performing ERS (Roy & McCallum, 2001; Baram *et al.*, 2004; Culver *et al.*, 2006). Results obtained for ERS-RS in the empirical evaluation conducted are consistent with these findings.

Recently, uncertainty sampling was used as pre-filtering for ERS in (Culver *et al.*, 2006;

Culver, 2006). The subset was formed by selecting only the N most uncertain examples. It was found that pre-filtering using uncertainty sampling increased accuracy when the base classifier for active learning was an SVM. However, no improvement was found when a naïve Bayes classifier was used. The poor performance was attributed to the inaccurate probability estimates generated.

In contrast to (Culver *et al.*, 2006), empirical evaluations in this thesis were conducted on text categorisation problems using a multinomial naïve Bayes classifier, where Bagging was used to increase the quality of the probability estimates obtained. This setting is similar to that of (Roy & McCallum, 2001) where ERS was shown to achieve higher accuracy than uncertainty sampling. In addition, we investigate an alternative exploration based pre-filtering strategy (ERS-FF) and also examine the effect of subset size for both sub-sampling and pre-filtering.

Furthermore, we propose a novel adaptive pre-filtering technique, which constructs the subset by dynamically switching among a number of alternative query selection strategies throughout the active learning process. While online adaptive query selection strategies have emerged from the literature (Baram *et al.*, 2004; Donmez *et al.*, 2007; Osugi *et al.*, 2005; Pandey *et al.*, 2005), these strategies are concerned with the selection of queries and not with the construction of a subset for use with ERS. We extend this idea to the problem of selecting a pre-filtering technique in an adaptive fashion.

7.3 Error-Reduction Sampling (ERS)

As discussed in Section 3.4.5 error reduction sampling (Roy & McCallum, 2001), in contrast to alternative selection strategies that optimise different criteria, selects queries that directly minimise expected future error. Closed form calculation of expected error is impractical for all but the most trivial of problems. However, estimates can be calculated using only the unlabelled data. For a probabilistic classifier trained on all known labelled data $\hat{P}_{\mathcal{D}_\ell}$ an estimate of future error is calculated using (7.1):

$$E_{\hat{P}_{\mathcal{D}_\ell}} = \int_{\mathbf{x}} L(P(y|\mathbf{x})\ ,\ \hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x}))P(\mathbf{x}) \tag{7.1}$$

where L is an appropriate loss function. The loss function L measures the disagreement between the predicted class and the true class. In the case of error reduction sampling, log-loss (7.2) is used:

$$L = \sum_{y \in \mathcal{Y}} P(y|\mathbf{x}) \log(\hat{P}(y|\mathbf{x})) \tag{7.2}$$

Rather than estimating the error over the full distribution of data $P(\mathbf{x})$, sample estimation is used over the unlabelled pool $\mathcal{D}_U$ to calculate the estimated future error, as shown in Eqn. 7.3. The true output distribution $P(y|\mathbf{x})$ is unknown therefore the current classifier ($\hat{P}_{\mathcal{D}_\ell}$) predictions are used.

$$\tilde{E}_{\hat{P}_{\mathcal{D}_\ell}} = \frac{1}{|\mathcal{D}_U|} \sum_{x \in \mathcal{D}_U} \sum_{y \in \mathcal{Y}} \hat{P}_{\mathcal{D}_\ell}(y|x) \log(\hat{P}_{\mathcal{D}_\ell}(y|x)) \tag{7.3}$$

The true label for unlabelled examples is not available therefore the expected error must be estimated using every possible labelling. The prediction from the current classifier is used to give a weighted average over all the possible labels as shown in (7.4):

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{D}_U} \sum_{y \in \mathcal{Y}} \hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x}) \ \tilde{E}_{P_{\mathcal{D}_\ell \bigcup \{\mathbf{x}, y\}}} \tag{7.4}$$

The use of the log-loss means that queries selected are those examples which maximise the sharpness of the learner's posterior belief of the unlabelled examples. In other words, ERS will select queries which maximally reinforce the current learner's existing belief over uncertain unlabelled examples.

```
Algorithm 2: Error Reduction Sampling : Active Learner
  Data: $\mathcal{D}_\ell$: the labelled data.
  Data: $\mathcal{D}_U$: the unlabelled data source.
  Result: $\hat{P}_{\mathcal{D}_\ell}$: classifier induced from all training data
  repeat
      induce classifier from $\mathcal{D}_\ell$ ;
      forall $\mathbf{x} \in \mathcal{D}_U$ do
          foreach $y \in \mathcal{Y}$ do
              Estimate expected error for $\mathcal{D}_\ell \bigcup \{\mathbf{x}, y\}$ ;
              weight each expected error by $\hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x})$ ;
      select query $(\mathbf{x}^*)$ with lowest expected future error;
      update labelled data $\mathcal{D}_\ell \leftarrow \mathcal{D}_\ell \bigcup \{\mathbf{x}^*, y\}$ ;
  until stop criterion reached ;
```

Algorithm 2 shows the pseudo-code for an active learner using error reducing sampling. Each unlabelled example in the pool ($\mathbf{x} \in \mathcal{D}_U$) is considered a candidate query. The unlabelled examples are assigned a possible label and tentatively added to the training data ($\mathcal{D}_\ell^* = \mathcal{D}_\ell \bigcup \{x, y\}$). A new classifier is induced ($\hat{P}_{\mathcal{D}_\ell^*}$) using the updated training data and the expected future error is estimated using (7.3). This is repeated for all possible labels and the estimated expected error for $\mathbf{x}$ is calculated as an average, weighted by the current classifiers predictions $\hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x})$. The query is selected as the unlabelled example that resulted in the lowest estimated expected error

A related technique to ERS is that of look-ahead query selection (Lindenbaum *et al.*, 1999) which estimates the *utility* of knowing the true label for an unlabelled example. Queries are selected as the unlabelled example that has the highest utility. This strategy is similar to ERS where the log-loss function in (7.3) is replaced with a 0/1 loss function.

Computational Efficiency

Implemented naïvely, error-reduction sampling is hopelessly inefficient. For a binary classification problem, selecting a query would require the induction of $|\mathcal{D}_U|^2$ classifiers for every iteration of active learning. Typically the size of the unlabelled pool is large in active learning rendering ERS impractical for most real world problems. Several optimisations that reduce the associated computational overhead have been suggested (Roy & McCallum, 2001):

- Many classification induction algorithms support incremental training, whereby adding and incorporating additional training data is far less expensive than inducing a new classifier from all available training data. Computational expense can be reduced if an incremental training implementation is used for ERS due to the large number of classifiers which are induced in every iteration of active learning.

- Estimates can be expensive to calculate especially if the pool is very large. A good estimate may be obtained from a subset of the pool. Sub-sampling and pre-filtering are ways in which a subset of the large pool is constructed. Estimates of future error are then found for just the examples in the subset.

Sub-sampling is frequently used to make ERS more computationally efficient. While pre-filtering is less common we believe it has the potential to be more beneficial to ERS than sub-sampling. Section 7.4 describe each technique in detail while Section 7.5 proposes a novel adaptive pre-filtering technique.

7.4 Subset Optimisation

The computational expense of ERS is proportional to the size of the unlabelled pool. Restricting the number of unlabelled examples in the pool can significantly reduce computational overhead. To achieve this, a subset of the pool ($\mathcal{D}'_U$) can be constructed and ERS applied to only those examples within the subset.

The manner in which the subset is constructed is the topic of investigation in this chapter. Two ways in which to construct the subset are random sub-sampling and pre-filtering. Random sub-sampling has been used frequently when ERS is applied. However, we believe the less commonly used pre-filtering technique has some significant advantages. We now describe both random sub-sampling and pre-filtering in greater detail.

7.4.1 Random Sub-sampling

Sub-sampled error-reduction sampling (ERS-RS) is the most frequently used optimisation (Roy & McCallum, 2001; Baram *et al.*, 2004). The subset is constructed by randomly selecting unlabelled examples from the (large) pool.

$$\mathcal{D}'_U \subset \mathcal{D}_U \quad \textbf{s.t.} \quad |\mathcal{D}'_U| \ll |\mathcal{D}_U|$$

Random sub-sampling is an efficient method of populating the subset. Furthermore, selecting examples at random ensures there is a variety of examples in the subset, provided the samples are large enough. However, with large subsets there is a possibility that it contains a significant number of uninformative examples.

7.4.2 Pre-filtering

Pre-filtering constructs the subset in a more principled way than random sampling. Query selection strategies are used to select the examples which form the subset. Intuitively, the subset should contain more informative examples since the uninformative examples are filtered out by the query selection strategies.

Practically all query selection strategies can be used as a pre-filter. Care should be taken however that the application of the pre-filtering does not increase computational overhead significantly. Two query selection strategies are proposed for use as pre-filters to construct the subset:

ERS-US is where Uncertainty Sampling is used as a pre-filtering technique (Culver *et al.*, 2006) to construct the subset. The N most uncertain examples from the pool of unlabelled examples are used to populate the subset. Uncertainty is defined by the prediction confidence of the classifier induced in the current iteration of active learning. Uncertainty sampling is a pure exploitation-based query selection strategy that selects examples close to the current decision boundary.

ERS-FF is where a Furthest First query selection strategy (Hochbaum & Shmoys, 1985) is used to populate the subset. Furthest First selects queries which are most dissimilar to all known labelled examples. Similarity is defined using a standard metric such as Euclidean distance. Furthest first is a pure exploration-based selection strategy that selects queries from regions of the input space where there is currently little or no knowledge.

Constructing the subset by filtering examples from the pool using a query selection strategy is expected to benefit ERS. This is because the subset should contain more informative examples than a subset constructed using random sub-sampling, resulting in an increase in the accuracy of active learning.

Random sub-sampling will be more computationally efficient to employ than pre-filtering, since the query selection strategy used to populate the subset will incur extra computational costs. However, we hypothesise the increased accuracy obtained from the pre-filtered subset will justify the extra computational expense.

Furthermore, we expect pre-filtering will allow for smaller subsets to be used without effecting the accuracy of the classifier produced using active learning. In this way, pre-filtering will not only reduce computational overhead but can also increase the accuracy of the classifier produced using active learning, offering significant advantages over random sub-sampling.

7.4.3 Subset Size

The running time of ERS is directly related to the size of the subset. The subset should be small enough to obtain a significant computational advantage but cannot be too small to ensure the sample contains some informative examples. This matter is discussed further in Section 7.6.7 where sensitivity analysis is conducted to determine the impact of subset size in the experiments conducted.

7.5 Adaptive Pre-filtering

Online adaptive query selection strategies (Baram *et al.*, 2004; Donmez *et al.*, 2007; Osugi *et al.*, 2005; Pandey *et al.*, 2005) have emerged that dynamically switch between a number of different query selection strategies. These online strategies adaptively choose which strategy to use in each iteration of active learning based on their relative performance. In this way, a query selection strategy is applied during the iteration where it is most useful. The adaptive query selection strategy should be as good as the best fixed strategy in hindsight.

7.5.1 ERS-FPL

Similarly to query selection strategies, we suggest that different pre-filtering techniques will be optimal during different phases in the life-cycle of an active learner. We proposed a novel adaptive pre-filtering technique, which will switch among several pre-filtering techniques based on their relative performance, thus extending the idea of online adaptive query selection to the problem of selecting a pre-filtering technique to construct the subset for ERS. An adaptive pre-filtering technique is expected to be as good as the best performing fixed pre-filtering technique in hindsight.

The task of selecting a pre-filtering scheme used to construct the subset in every iteration of active learning is cast as an instance of the Multi-Armed Bandit (MAB) problem (see Section 3.4.6). In the MAB setting, each pre-filtering technique is considered to be a slot machine and in every iteration of active learning just one machine is selected to be played. After playing a machine a reward is received based upon how successful that machine was. The goal is to maximise reward over the finite horizon (the T iterations of active learning). In an active learning setting the goal is to converge to the supervised learning accuracy as quickly as possible.

We employed the *follow-the-perturbed-leader* (Kalai & Vempala, 2005; Pandey *et al.*, 2005) as the MAB algorithm in the adaptive pre-filtering technique. An opaque version of the algorithm was used, where rewards for machines that were not played are not available to the gambler. This mimics the slot machine analogy where we do not know the reward for machine that we did not play.

The reward (s_i) associated when each machine (pre-filtering technique) was played is maintained and used in combination with a perturbation (p_i) factor to choose the optimal machine for the next iteration of active learning.

Algorithm 3: Pseudocode for the opaque Follow-the-Perturbed-Leader algorithm.

```
for $t = 1 \ldots T$ do
    Choose machine $\arg\min_i s_{i_{<t}} + p_i$ ;
    Play machine $i$ and receive reward for selected machine. ;
```

Algorithm 3 details the pseudo-code for the FPL algorithm where $s_{i_{<t}}$ is the total reward associated with machine i obtained in all previous iterations. The reward associated with machines not played in the current iteration is zero. Perturbation values are drawn from the exponential distribution and the horizon (T) in this case is the stopping criterion for active learning. Similar to (Pandey *et al.*, 2005) the perturbation is drawn from the exponential distribution $p_t = e^{-|x|}$ where $x = rand * p_{t-1}$.

Algorithm 4 shows the pseudo-code for an adaptive pre-filtered active learning using ERS. The additional step of selecting the pre-filtering technique is achieved using Algorithm 3 which switches among a number of alternative query selection strategies to use as pre-filters. The subset ($\mathcal{D}'_U$) is constructed using the selected pre-filtering technique and ERS is applied to only the examples within the subset.

Algorithm 4: Adaptive Pre-Filtered Error Reduction Sampling : Active Learner

```
Data: $\mathcal{D}_\ell$: the labelled data.
Data: $\mathcal{D}_U$: the unlabelled data source.
Result: $\hat{P}_{\mathcal{D}_\ell}$: classifier induced from all training data
repeat
    induce classifier from $\mathcal{D}_\ell$ ;
    select pre-filtering technique ;
    construct subset $\mathcal{D}'_U$ ;
    forall $\mathbf{x} \in \mathcal{D}'_U$ do
        foreach $y \in \mathcal{Y}$ do
            Estimate expected error for $\mathcal{D}_\ell \bigcup \{\mathbf{x}, y\}$ ;
            weight each expected error by $\hat{P}_{\mathcal{D}_\ell}(y|\mathbf{x})$ ;
    select query $(\mathbf{x}^*)$ with lowest expected future error;
    update labelled data $\mathcal{D}_\ell \leftarrow \mathcal{D}_\ell \bigcup \{\mathbf{x}^*, y\}$ ;
until stop criterion reached ;
```

7.5.2 Reward

The reward is an important aspect of a reinforcement learning strategy. Without an effective feedback mechanism the learner will not be able to select the optimal pre-filtering technique in every iteration of active learning. Ideally, the best reward to use would be the true error of the classifier. Clearly this is not available. Indeed even an estimate of the true error obtained on a hold-out test set is not strictly available due to the scarcity of labelled examples in an active learning setting.

An obvious metric to use is the error rate achieved on the current set of labelled data. Since the number of labelled examples is small, techniques such as k-fold cross validation or leave-one-out (LOO) validation are often used - this reward structure was used in (Pandey *et al.*, 2005). While this offers an estimate of the error achieved by the current classifier, there are problems, namely in the data used for evaluation. Active learning populates the training data with examples that are inherently difficult to classify. The training examples are therefore generally not representative of the underlying data (not *i.i.d.*). Thus, conducting evaluation on a biased training data, such as one constructed using active learning, may lead to misleading rewards. In the experiment LOO rewards were not used for this reason.

An alternative metric to measure the performance of active learning is Classification Entropy Maximisation (CEM) (Baram *et al.*, 2004), which measures the entropy (**H**) of label predictions given by the current classifier on the remaining unlabelled examples. It is calculated using (7.5).

$$CEM(\hat{P}_{\mathcal{D}_\ell}) = \mathbf{H}(\frac{|\hat{P}_{\mathcal{D}_\ell}(\mathcal{D}_U) = +|}{|\mathcal{D}_U|}) \tag{7.5}$$

where $|\hat{P}_{\mathcal{D}_\ell}(\mathcal{D}_U) = +|$ is the number of positive predictions given by the classifier when run on the unlabelled data. CEM gives higher values to a more even distribution of labels. It was previously used as a reward in (Baram *et al.*, 2004) and was seen to closely resemble the true error estimate. CEM was used as the reward structure in the experiments.

7.6 Empirical Evaluation

In this section we describe empirical evaluations that examine pre-filtered error-reduction sampling. First comparisons between random sub-sampled error-reduction sampling and less complex query selection strategies are made to establish the benefits offered by employing ERS. Secondly, experiments are conducted to establish whether pre-filtering had a positive influence on the accuracy of active learning compared to random sub-sampling. Comparisons are made between ERS applied to subsets constructed using random sub-sampling as well as subsets constructed using pre-filtering. Furthermore we compare the quality of unlabelled examples in a subset constructed by random sub-sampling with a subset constructed using pre-filtering. Third, the proposed adaptive pre-filtering technique was evaluated is respect to each of the fixed pre-filtering techniques. An adaptive pre-filtering technique should be as good as the best fixed pre-filtering technique in hindsight. Finally sensitivity analysis is conducted to discern the impact subset size has on error-reduction sampling. Compared to random sub-sampling, pre-filtering should reduce the number of examples in the subset required to achieve a similar level of accuracy. Smaller subsets can lead to decreased computational overheads for ERS.

7.6.1 Experiment Settings

In this section we describe experimental settings used in the empirical evaluations. Details are given on the datasets, classifier and evaluation metric employed.

Datasets

Experiments were evaluated using two real world benchmark text corpora, namely the 20NG-4 dataset and the R10 dataset. A subset of the 20 newsgroups dataset (as described in Section 4.2.2) was used in the experiments. The dataset consists of four binary text classification problems of varying difficulty. 10-fold cross validation was performed on each of the four sub-problems.

The second corpus used was the R10 dataset (as described in Section 4.2.1). Ten binary text classification problems were constructed for the R10 data in a one-v-rest manner (one for each category). The experimental setup closely follows that of (Tong & Koller, 2001) where the pool of unlabelled examples are formed by randomly sampling 1,000 examples from 9,603 Mod-Apté training set and removing their labels. The test set remained unchanged and consisted of the 3,299 Mod-Apté test examples.

Classifier

Similar to (Roy & McCallum, 2001) an ensemble of multinomial naïve Bayes classifiers were used. However the Weka (Witten & Frank, 2005) implementation was utilised in the experiments. The ensemble consisted of 10 members constructed using Bagging. The aim of the ensemble is to provide smoother probability estimates.

Evaluation Metric

Since there is no standard way in which to report results of active learning experiments, a number of evaluation metrics are used:

- Learning curves (see Section 3.5.2) plot the accuracy of the classifier induced for every iteration of active learning on a hold out test set. They allow for quick visual comparisons to be made between alternative active learning strategies.
- p-value plots (see Section 3.5.4) show at which iterations each strategy is statistically significantly better than random sampling. The confidence level $\alpha = 0.05$ is shown in each plot to help assessment.
- Deficiency values (see Section 3.5.3) are used to report the global performance of each active learning strategy.
- Confidence Intervals as described in Section 3.5.5 are a method of combining the p-values obtained from performing one tail paired t-tests in every iteration of active learning into a single value. Statistical significance is established between two alternative strategies by taking the median confidence level at which they differ across all iterations of active learning.

7.6.2 Error-Reduction Sampling

The first set of experiments was conducted to establish whether error-reduction sampling outperforms uncertainty sampling. The accuracy of active learning using three query selection strategies was compared. Namely random sub-sampled error-reduction sampling (ERS-RS) is compared to the accuracy obtained using both uncertainty sampling (US) and random sampling (RS). For ERS to be considered advantageous it must outperform both RS and US.

Active learning was performed as described in Algorithm 2. Seed data for the 20NG-4 experiments was constructed by randomly selecting 3 positive and 3 negative examples of the class. Seed data for R10 was constructed by randomly selecting 10 positive and 10 negative examples of the class. In order for fair comparison to be made, the same seeds were supplied to each of the strategies under investigation. A single query is selected in every iteration and active learning was stopped after 100 queries were labelled. A subset of size 100 was used for all ERS experiments.

20NG-4 Results

Figure 7.1 shows the learning curves for the 20NG-4 dataset. Error-reduction sampling (ERS-RS) is seen to outperform both random sampling (RS) and uncertainty sampling (US) for all four problems. This is very evident in the aggregate plot of all four sub-problems given in Figure 7.1(a). Surprisingly, Figure 7.1(d) shows random sampling outperforming uncertainty sampling. However, similar results are reported (Tong & Koller, 2001) when the same Usenet articles were used in active learning experiments.

Table 7.1 report the confidence interval for the 20NG-4 dataset. Large confidence interval values are reported when US is compared to RS (except in the WH problem which correlates with the findings from the learning curves).

Error reduction sampling was found to outperform uncertainty sampling with confidence interval ≥ 0.7 on three of the four problems. The poor confidence interval value for the BC sub-problem is due to the fact the problem is particularly easy, hence both ERS and US converge to the supervised learning rate rapidly.

Table 7.1: 20NG-4 Confidence interval values for US and ERS-RS

	AR	GX	WH	BC
RS v US	0.9	0.9925	-0.99	0.995
US v ERS-RS	0.7	0.7	0.99	0.0

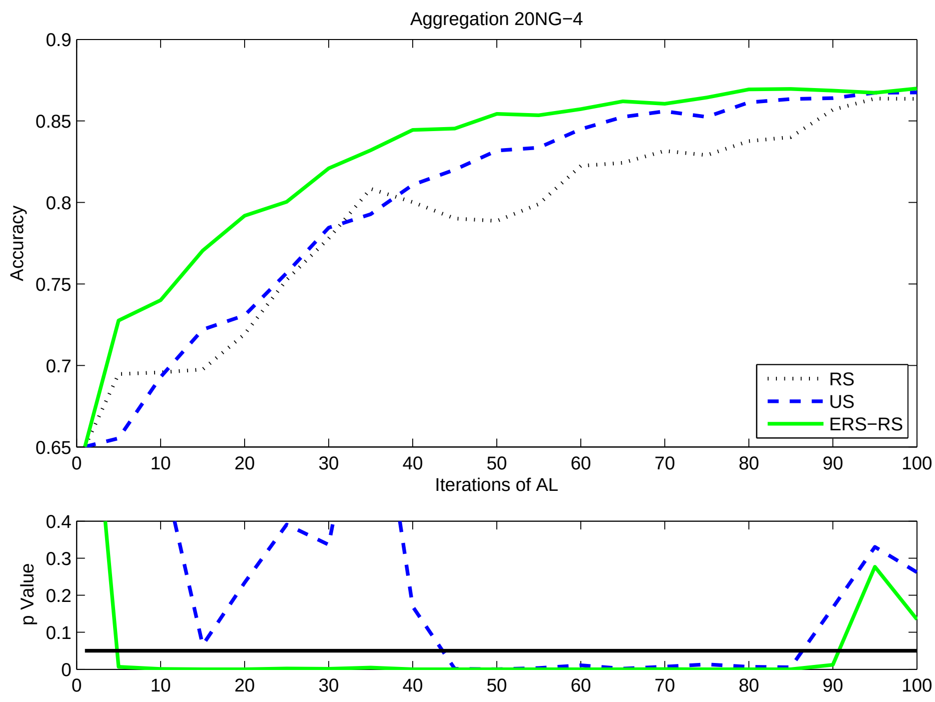
Deficiency values obtained for the 20NG-4 dataset are reported in Table 7.2. The lowest deficiency value was obtained using ERS-RS on all four problems. Error reduction sampling also outperformed random sampling in the WH problem as indicated by the deficiency values being less than one.

Table 7.2: 20NG-4 Deficiency scores (D_A) for US and ERS-RS. Lower values are better. The lowest deficiency values are indicated in bold text

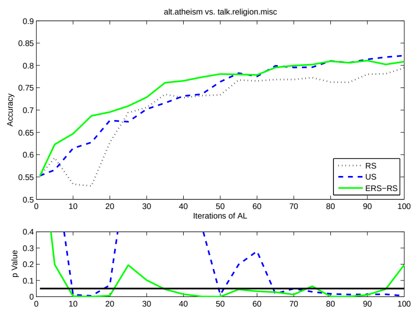
	AR	GX	WH	BC
US	0.8494	0.7257	1.3568	0.5588
ERS-RS	**0.7602**	**0.6176**	**0.9923**	**0.4586**

R10 Results

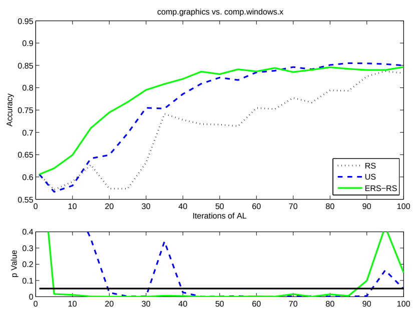
Figure 7.2 shows the learning curves for the R10 dataset. Due to the highly unbalanced data distribution F_1 is used instead of accuracy. The graphs report the F_1 values averaged over all ten categories. Additional learning curves for individual categories on the R10 are available in Appendix C. Both macro and micro-averaged F_1 are reported, where macro-



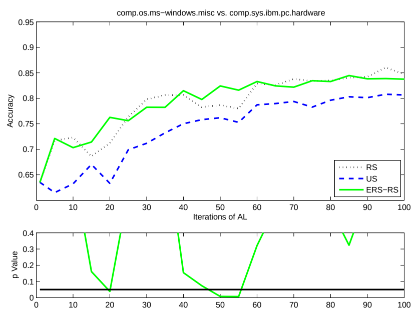
(a) Aggregate 20NG-4



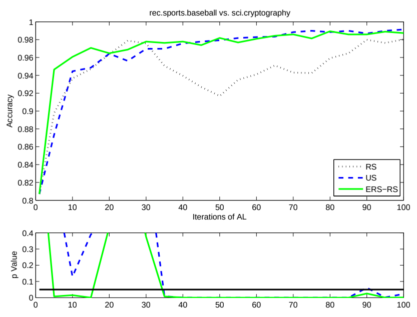
(b) AR



(c) GX



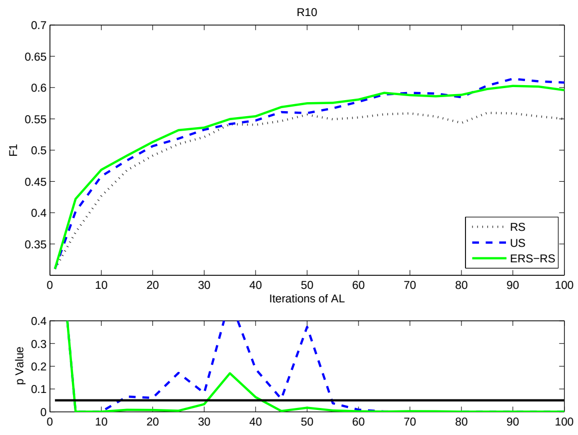
(d) WH



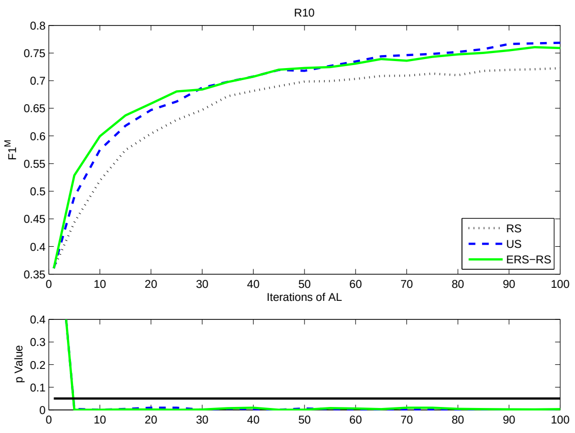
(e) BC

Figure 7.1: 20NG-4 Results. Obtained using a bagged multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)

averaged takes the mean F_1 across categories while micro-averaged F_1 is constructed from the averaged confusion matrix. The learning curves show that uncertainty sampling achieves better F_1 values than random sampling. Furthermore, error reduction sampling was found to be the top-performing query selection strategy achieving higher F_1 values than uncertainty sampling.



(a) Macro F_1



(b) Micro F_1

Figure 7.2: R10 Results, obtained using a bagged multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 10 examples from each class and selecting 1 query per iteration. F_1 is used as the evaluation metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)

Confidence interval values obtained for the R10 dataset are reported in Table 7.3. Significant improvements are observed when uncertainty sampling is used in place of random sampling. Moreover, ERS-RS was found to outperform US during the first number of iterations. In the case of macro F_1 it continues to outperform US for the majority of the 100 iterations. However, in micro F_1 the initial gains obtained using ERS-RS are quickly recovered by US. Individual category learning curves in Appendix C show that while in six of the R10 categories ERS-RS was observed to improve obtained F_1 values, in three of the top five most popular categories ERS-RS obtained lower F_1 values than US.

Table 7.3: R10 Confidence interval values for US and ERS-RS

	F_1	F_1^M
RS v US	0.99	0.995
US v ERS-RS	0.7	-0.65

Deficiency values for the R10 dataset are reported in Table 7.4. Values less than 1 show improved performance compared to random sampling. Uncertainty sampling is seen to be better than random sampling. Furthermore, error-reduction sampling was found to be the best performing query selection strategy with lower deficiency values than those obtained from uncertainty sampling.

Table 7.4: R10 Deficiency scores (D_A) for US and ERS-RS. Lower values are better. The lowest deficiency values are indicated in bold text.

	F_1	F_1^M
US	0.8905	0.8222
ERS-RS	**0.8741**	**0.8094**

Summary

Error reduction sampling was shown to achieve better results than alternative commonly used query selection strategies. The results presented here are consistent with those found in (Roy & McCallum, 2001). This motivates the use of ERS when computational expenditure is favoured in place of labelling effort.

7.6.3 Pre-Filtered ERS

The following experiments were conducted to compare alternative pre-filtering techniques. There are many query selection strategies which select queries based on different heuristics (see Chapter 3). We establish if alternative query selection strategies, when used as pre-filters will have an impact on quality of queries selected by ERS.

The pre-filtering techniques described in Section 7.4 were used to construct the subset, from which a query is subsequently selected using ERS. Active learning was conducted following Algorithm 2, where the set of unlabelled examples $\mathcal{D}_U$ is the subset constructed using the three techniques: ERS-RS which constructs the subset using random sampling, ERS-US which constructs the subset using uncertainty sampling and ERS-FF which constructs the subset using furthest first sampling. A subset of size 100 was used for all ERS query selection strategies.

Seed data for the 20NG-4 experiments was constructed by randomly selecting 3 positive and 3 negative examples of the class. Seed data for R10 was constructed by randomly selecting 10 positive and 10 negative examples of the class. In order for a fair comparison to be made, the same seeds were supplied to each of the strategies under investigation. A single query is selected in every iteration and active learning was stopped after 100 queries were labelled.

20NG-4 Results

Learning curves for the 20NG-4 dataset are reported in Figure 7.3. Of the three[1] techniques under investigation, pre-filtering using uncertainty sampling (ERS-US) was found to be the top performing technique. Examining the individual sub-problems this is borne out by ERS-US being the top-performing technique on three of the four 20NG-4 problems. However, in the GX problem, pre-filtering with furthest first (ERS-FF) was the top performing technique.

p-value plots are reported directly under each learning curve plot in Figure 7.3 and show statistical significance against ERS-RS. Comparing the three techniques overall, it can be seen that ERS-US comes much closer to achieving statistically significantly ($p = 0.05$) difference to ERS-RS than the other two techniques. Examining the individual problems: in the AR and BC problems ERS-US is seen to be better than ERS-RS while ERS-FF does not offer a statistically significant advantage. Roles are reversed in the GX problem where ERS-FF is seen to be best and ERS-US is not statistically significantly better than ERS-RS. In the WH problem ERS-RS performs well hence pre-filtering strategies do not outperform it.

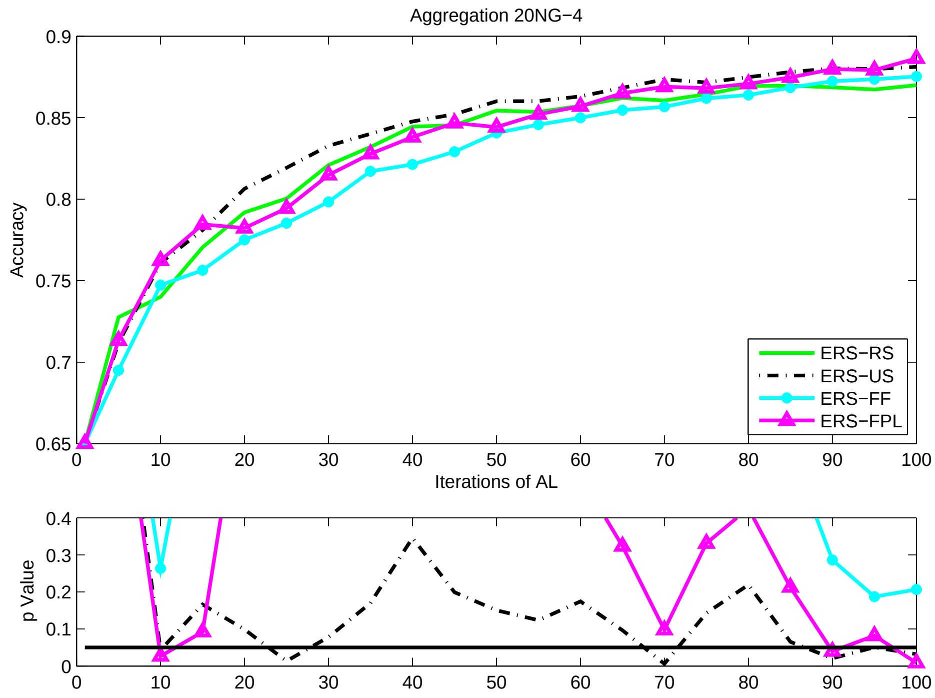
Confidence interval values for the 20NG-4 dataset are reported in Table 7.5. The two pre-filtering techniques are compared against the random sub-sampling technique over all 100 iterations of active learning. Statistically significant increases in accuracy are achieved when pre-filtering using uncertainty sampling is used in the AR, WH and BC problems. Similarly significant increases compared to random sub-sampling are observed with pre-filtering using furthest first is employed in the GX problem.

Table 7.5: 20NG-4 Confidence interval values for Pre-filtering. Pre-filtering techniques are compared against ERS-RS

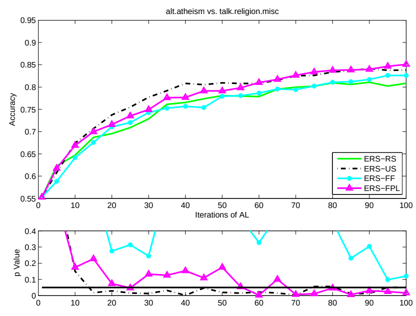
	AR	GX	WH	BC
ERS-RS v ERS-US	0.95	0.0	0.6	0.7
ERS-RS v ERS-FF	0.0	0.8	-0.95	-0.975
ERS-RS v ERS-FPL	0.95	0.75	-0.9	0.6

Table 7.6 reports the deficiency values for each of the subset construction techniques. Pre-filtering using uncertainty sampling achieved the lowest deficiency value on three of the

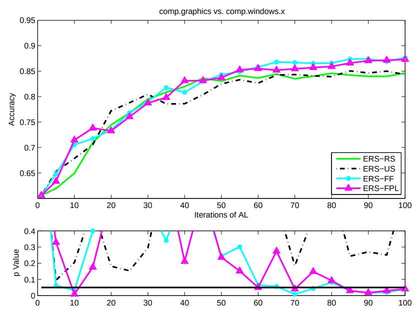
[1]For brevity the figure also includes the learning curve for ERS-FPL which is discussed in §7.6.5



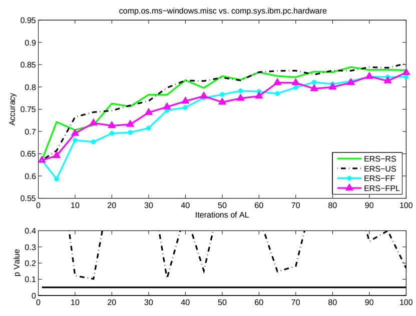
(a) AR



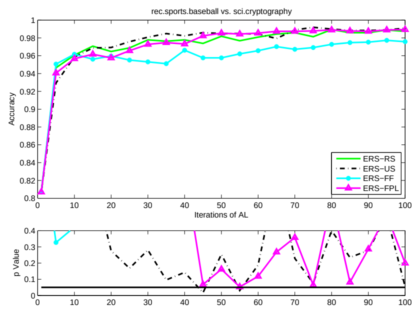
(b) AR



(c) GX



(d) WH



(e) BC

Figure 7.3: 20NG-4 Results, obtained using a bagged multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling ERS-RS is given ($\alpha = 0.05$ is shown as a solid line)

four problems, while ERS-FF achieved the lowest deficiency value in the GX problem.

Table 7.6: 20NG-4 Deficiency scores (D_A) for Pre-filtering. Lower values are better. The lowest deficiency values are indicated in bold text

	AR	GX	WH	BC
US	0.8494	0.7257	1.3568	0.5588
ERS-RS	0.7602	0.6176	0.9923	0.4586
ERS-US	**0.6272**	0.6110	**0.9513**	**0.4039**
ERS-FF	0.7685	**0.5388**	1.2716	0.7210
ERS-FPL	0.6876	0.5705	1.1825	0.4682

R10 Results

Learning curves for pre-filtering techniques on the R10 dataset are given in Figure 7.4 where the average F_1 calculated over all categories is reported. Individual learning curves for the ten categories of the R10 are given in Appendix C.

Pre-filtering using uncertainty sampling (ERS-US) achieved the largest F_1 values, outperforming random sub-sampling. Pre-filtering using furthest-first performed significantly worse than random sub-sampling. Furthest first has a tendency to select outliers which do not help in active learning since they offer little information.

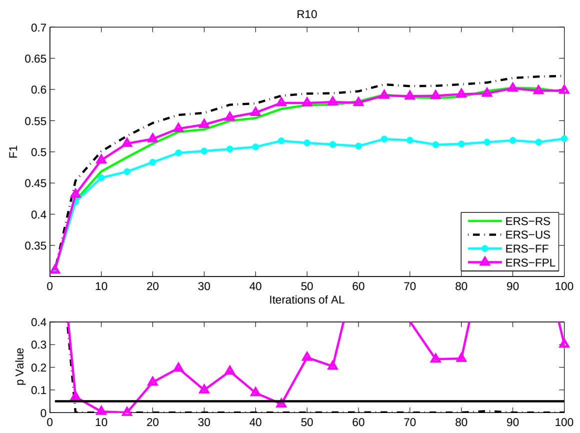
p-value plots in Figure 7.4 compare statistical significance of each pre-filtering technique against random sub-sampling. The plots show that ERS-US is statistically significantly better than ERS-RS while ERS-FF is not. This is consistent with the learning curves observed.

Confidence intervals for the R10 dataset are given in Table 7.7. Pre-filtering with uncertainty sampling was found to give the greatest statistically significantly improvement over random sub-sampling. Subsets constructed using furthest first sampling were found to be significantly worse than random sub-sampling.

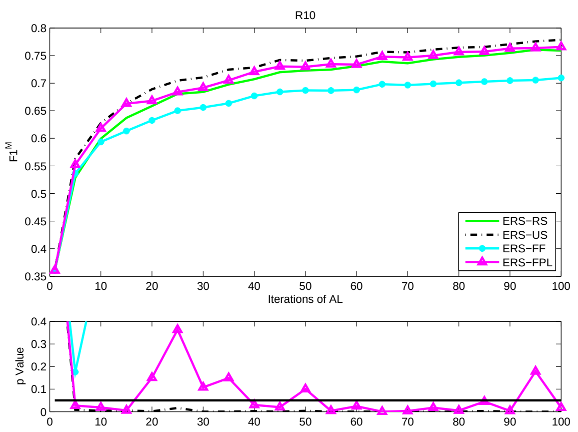
Table 7.7: R10 Confidence interval values for Pre-filtering. Pre-filtering techniques are compared against ERS-RS

	Accuracy	F_1	F_1^M
ERS-RS v ERS-US	0.995	0.995	0.995
ERS-RS v ERS-FF	-0.995	-0.995	-0.995
ERS-RS v ERS-FPL	0.975	0.7	0.95

Table 7.8 displays the deficiency values for the R10 dataset. The lowest deficiency value



(a) Macro F_1



(b) Micro F_1

Figure 7.4: R10 Results, obtained using a bagged multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 10 examples from each class and selecting 1 query per iteration. F_1 is used as the evaluation metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against ERS-RS is given ($\alpha = 0.05$ is shown as a solid line)

was achieved by pre-filtering with uncertainty sampling. Pre-filtering using furthest first had the highest deficiency values which confirms its as a sub-optimal technique.

Table 7.8: R10 Deficiency scores (D_A) for Pre-filtering. Lower values are better. The lowest deficiency values are indicated in bold text

	Accuracy	F_1	F_1^M
US	0.7988	0.8905	0.8222
ERS-RS	0.8182	0.8741	0.8094
ERS-US	**0.6683**	**0.7831**	**0.7062**
ERS-FF	0.9127	1.0576	0.9250
ERS-FPL	0.7449	0.8517	0.7614

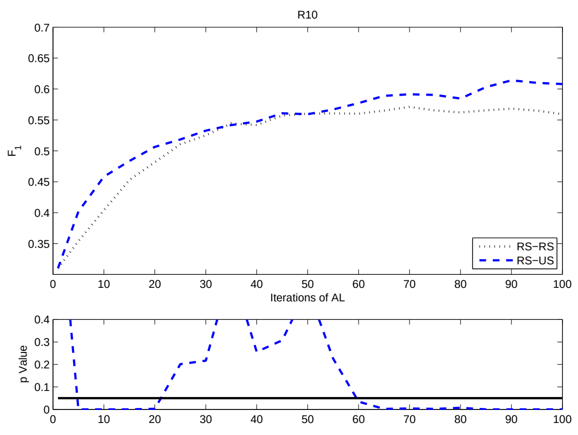
7.6.4 Subset Quality

Pre-filtering should populate the subset with more informative examples than random sub-sampling. The next set of experiments was conducted to examine the quality of the subset constructed. Comparisons are made by selecting queries at random from a subset constructed using random sub-sampling as well as a subset constructed using the best performing pre-filtering technique. In this way, we can determine if the unlabelled examples in the pre-filtered subset are intrinsically more informative than those in the randomly sub-sampled subset.

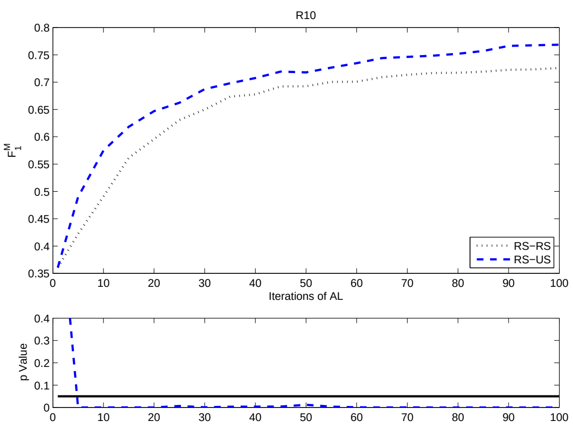
Active learning was performed as described in Algorithm 2. Seed data for the 20NG-4 experiments was constructed by randomly selecting 3 positive and 3 negative examples of the class.

RS-RS randomly selects examples from the subset constructed using random sub-sampling, while RS-US randomly selects examples from the subset constructed by pre-filtering using uncertainty sampling. Random sampling is used to select queries to ensure no bias is introduced from the query selection strategy.

Active learning was conducted on the R10 corpus where seed data was constructed by randomly selecting 10 positive and 10 negative examples of the class. In order for fair comparison to be made, the same seeds were supplied to each of the strategies under investigation. A single query is selected in every iteration and active learning was stopped after 100 queries were labelled. A subset of size 100 was used for both RS-RS and RS-US.



(a) Macro F_1



(b) Micro F_1

Figure 7.5: Comparison of random sampling operating in a subset constructed from sub-sampled subset and pre-filtered subset. A bagged multinomial naïve Bayes classifier was used and Active Learning was run for 100 iterations. Evaluation on the test set was done after every 5 iterations with F_1 used. Below each graph the p-value showing statistical significance against random sampling (RS) on the full pool is given ($\alpha = 0.05$ is shown as a solid line)

Figure 7.5 shows the learning curves obtained. Increased F_1 values were reported for active learning performed using the pre-filtered subset (RS-US). Given that the only difference between the two active learners is the manner in which the subset was constructed, these results show that the unlabelled examples selected by pre-filtering to populate the subset are more informative than those selected by random sub-sampling.

7.6.5 Adaptive Pre-Filtering

In this section we empirically evaluate the adaptive pre-filtering technique proposed in Section 7.5. In every iteration of active learning ERS-FPL dynamically switches between a number of pre-filtering techniques used to construct the subset. The goal of adaptive pre-filtering is to be as good as the best fixed pre-filtering technique in hindsight.

Experiments followed Algorithm 4 where the *follow-the-perturbed-leader* was used to switch between random sampling, uncertainty sampling and furthest first sampling. Rewards were constructed using CEM on the entire unlabelled pool. In each iteration of active learning, a single pre-filtering strategy is selected by the FPL algorithm to construct the subset. ERS selects the query from the constructed subset. The same seed data used in Section 7.6.3 was supplied to the adaptive pre-filtering experiments to allow for fair comparison.

20NG-4 Results

The learning curves obtained from the adaptive pre-filtering technique have been included in Figure 7.3 to allow for direct comparison. Visual inspection reveals that the adaptive pre-filtering technique works well in the early and late iterations of active learning. Examining the individual sub-problems of the 20NG-4: ERS-FPL is competitive with the best performing technique on three of the four problems. Furthermore, p-value plots which show statistical significance against ERS-RS, report that ERS-FPL tracked the performance of the best fixed pre-filtering technique in AR, GX and BC problems.

The deficiency values in Table 7.6 and confidence interval values in Table 7.5 correlate with the results observed in the learning curves. In the AR and GX problems, ERS-FPL performed as expected and was runner-up to the best fixed pre-filtering strategy, namely ERS-US and ERS-FF respectively. ERS-FPL demonstrated adaption in the selection of pre-fitlering technique by tracking ERS-FF in the GX problem.

In the BC problem ERS-US was the best fixed pre-filtering technique. While ERS-RS was runner-up there is only marginal difference between it and ERS-FPL. The BC problem is particularly easy, hence all techniques (with the exception of ERS-FF) perform well.

In the WH problem ERS-US is the best fixed pre-filtering technique with ERS-RS being the runner-up. ERS-FPL was ranked third outperforming ERS-FF. This result can be attributed to the strong performance of random sampling on this problem, as observed in Section 7.6.2 where random sampling outperformed uncertainty sampling.

R10 Results

Figure 7.4 plots the learning curves for the adaptive pre-filtering technique. Both macro and micro F_1 values are reported. In the case of macro F_1 ERS-FPL is initially seen to closely follow the learning curve of the top performing uncertainty sampling pre-filtering technique before tracking the learning curve of the random sub-sampled error reduction sampling in subsequent iterations. In the micro F_1 plots, the learning curve for the adaptive pre-filtering technique resides between the top performing uncertainty sampling pre-filtering technique and the learning curve given by random sub-sampled error reduction sampling. p-value plots which show statistical significance against ERS-RS are consistent with the learning curves. ERS-FPL is initially significantly better than ERS-RS in the macro F_1 before tailing off. In the micro F_1 plot ERS-FPL is seen to be significantly better than ERS-RS and tracks the performance of ERS-US.

Confidence intervals for the adaptive pre-filtering technique are reported in Table 7.7. It was found that statistically significant improvements in F_1 values were obtained using the adaptive pre-filtering technique when compared with random sub-sampling. The top performing technique was ERS-US while the worst was ERS-FF.

Table 7.8 reports the deficiency value obtained for ERS-FPL. Both the macro and micro F_1 variants show that lower deficiency is obtained by the adaptive technique than random sub-sampling and pre-filtering using furthest first. Across all evaluation metrics ERS-FPL was found to be runner-up to the top performing ERS-US. I

Friedman Test

Demšar (2006) proposes the use of the Friedman test to compare classifiers over multiple datasets. The Friedman test is a non-parametric test which is equivalent to the repeated-measures ANOVA. Table 7.9 reports the rank of each subset construction technique. The average rank of each technique is given at the bottom of the table. From these values alone we can see that pre-filtering using uncertainty sampling is the best technique on the given datasets. The adaptive technique is runner up and is seen to outperform both random sub-sampling and pre-filtering using furthest first.

The Friedman test compares the average rank of each of the k subset construction techniques over the N datasets as shown in (7.6). From the table it can be seen that the adaptive pre-filtering technique is runner up to ERS-US.

$$R_j = \frac{1}{N}\sum_i r_i^j \tag{7.6}$$

Table 7.9: Rank for subset construction techniques deficiency values. A rank of one indicates the lowest deficiency value

	ERS-RS	ERS-US	ERS-FF	ERS-FPL
AR	3	1	4	2
BC	2	1	4	3
GX	4	3	1	2
WH	2	1	4	3
Acq	3	1	4	2
Corn	3	1	4	2
Crude	3	1	4	2
Earn	3	1	4	2
Grain	3	1	4	2
Interest	2	1	4	3
Money-Fx	3	1	4	2
Ship	3	1	4	2
Trade	2	1	4	3
Wheat	3	1	4	2
Average	2.7857	1.1429	3.7857	2.2143

The Friedman statistic (7.7) is distributed according to χ_F^2 with $k-1$ degrees of freedom. However, the statistic is undesirably conservative and an alternative statistic derived from the Friedman is used. It is calculated as shown in (7.8) and is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom.

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{7.7}$$

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{7.8}$$

From Table 7.9 the Friedman statistic is $\chi_F^2 = 27.72857$ and $F_F = 25.25826$. The critical value for $F(3,39) = 2.85$ for $\alpha = 0.05$ hence the null hypothesis is rejected. This means there is a statistically significant difference between the four subset construction techniques.

Post-hoc analysis can determine which of the techniques are statistically different from each other. The Nemenyi test was used to establish if the difference between each technique was significant. The differences between each technique was found to be statistically significant.

7.6.6 Computational Expense

If pre-filtering is to be regarded as a practical alternative to sub-sampling the computational cost of applying pre-filtering should not outweigh the accuracy benefits it brings. We performed CPU timing experiments in order to establish that the various pre-filtering strategies do not incur significantly larger computational overhead than sub-sampling.

Pre-filtering techniques such as ERS-US requires extra computational expense compared to random sub-sampling, namely in the construction of a classifier, obtaining predictions on unlabelled examples and sorting the predictions to find the queries. The experiments conducted utilised a multinomial naïve Bayes classifier.

Training of a multinomial naïve Bayes classifier requires that the conditional probabilities for each feature is calculated which is achieved using just one run through the training data and has the time complexity of $O(|\mathcal{Y}||\tau| + |\mathcal{D}_\ell|L_{avg})$ where $|\mathcal{Y}|$ is the number of categories, τ is the vocabulary of the corpus, $|\mathcal{D}_\ell|$ is the number of training documents and L_{avg} is the average length of a document (see (Manning *et al.*, 2008, Chapter 13)). Similarly, testing a document requires the combination of the conditional probabilities for those features which appear in the test document.

Both training and testing a multinomial naïve Bayes are linear in the time it takes to scan the data (the cost incurred when alternative classifiers are used is discussed in (Yang *et al.*, 2003)). Moreover, sorting the predictions can be achieved using an efficient sorting algorithm, such as Quicksort, with time complexity of $O(N \log N)$ where N is the number of items to sort. Therefore the computational expense of performing pre-filtering can be justified if an improvement in active learning is achieved.

We also empirically timed the pre-filtering and compared it to random sub-sampling. Experiments were conducted using the AR sub-problem of the 20NG-4 dataset and performed on an Intel Core 2 Duo 2GHz processor with 2GB of RAM. A subset of size 10 constructed using sub-sampling and pre-filtering was used in these experiments. Pre-filtering led to an increase in computation of approximately 0.4 seconds per iteration of active learning using ERS when compared to sub-sampling. Given the running time per iteration of ERS is approximately 16 seconds this is an increase in computation expense of 0.02 percent. Similar results were found for the adaptive pre-filtering technique. The marginal increase in computational expense acceptable given the potential improvements obtained in active learning.

7.6.7 Subset Size

In the following experiments the effect of subset size on the performance of ERS using both sub-sampling and pre-filtering was investigated. Subset size has a direct influence on the running time of ERS therefore smaller subsets are preferred.

Comparisons are made by performing active learning using ERS with subset sizes of size $k \in \{10, 50, 100\}$. Furthermore, we also compared subsets constructed using both random sub-sampling and pre-filtering using uncertainty sampling.

Active learning was performed as described in Algorithm 2. Seed data for the 20NG-4 experiments was constructed by randomly selecting 3 positive and 3 negative examples of the class. Seed data for R10 was constructed by randomly selecting 10 positive and 10 negative examples of the class. In order for fair comparison to be made, the same seeds were supplied to each of the strategies under investigation. A single query is selected in every iteration and active learning was stopped after 100 iterations.

Table 7.10 shows the deficiency values for the various subset sizes. Each row reports the deficiency value obtained by performing active learning using ERS on a subset of the specified size ($\{10, 50, 100\}$) and constructed using either random sub-sampling (RS) or pre-filtering using uncertainty sampling (US). Random sub-sampling is seen to benefit from a larger subset, where lower deficiency values reported for subsets of size 50 and 100. Conversely, pre-filtering using uncertainty sampling did not to the same degree from the use of larger subsets. Decrease in deficiency values in larger subsets were not as pronounced as those reported for random sub-sampllng.

Table 7.10: Deficiency for various subset sizes for ERS on both 20NG-4 and R10 (rightmost columns). Lower values are better and the lowest deficiency value is indicated using bold text

	AR	GX	WH	BC	F_1	F_1^M
US	0.850	0.726	1.357	0.559	0.891	0.744
ERS-RS10	0.830	0.737	1.136	0.629	0.896	0.673
ERS-RS50	0.733	0.716	0.961	0.529	0.885	0.746
ERS-RS100	0.760	0.618	0.992	0.459	0.874	0.712
ERS-US10	0.748	0.643	1.036	0.451	0.786	**0.566**
ERS-US50	0.665	**0.571**	1.020	0.418	**0.777**	0.608
ERS-US100	**0.627**	0.611	**0.951**	**0.404**	0.783	0.611

The benefit of pre-filtering is also evident in the results obtained. Deficiency values reported for pre-filtered ERS are significantly lower than those reported for random sub-sampling using the same subset size. Furthermore, pre-filtering using a subset of size 10 reports lower deficiency than random sub-sampling with a subset of size 100 on the majority of the problems.

7.7 Discussion

Error reduction sampling (ERS) is a top performing query selection strategy. Results obtained from Section 7.6.2 demonstrate the ability of ERS to increase the accuracy achievable by active learner. However, error reduction sampling is a computationally expensive process. In order for ERS to be used more frequently in real world active learning problems, the computational overhead must be reduced.

Pre-Filtering

Pre-filtering is proposed as a way to both increase the quality of queries selected by ERS and simultaneously reduce the computational expense. We examined two pre-filtering techniques which are based around exploration and exploitation. ERS-US, which constructs a subset using uncertainty sampling, is an exploitation-based pre-filtering technique. It leverages the current knowledge to populate the subset with those unlabelled examples which are believed to be close to the decision boundary. Conversely, ERS-FF constructs a subset using furthest first sampling, which is an exploration-based pre-filtering technique. The subset is populated with those examples most dissimilar to all known labelled examples.

Comparisons were made between ERS applied to subset constructed using random sub-sampling and subsets constructed using the two pre-filtering techniques. Results from Section 7.6.3 demonstrated that overall ERS-US was capable of increasing the learning rate of active learning using ERS in all problems, with the exception of GX problem where ERS-FF was best. We believe that this correlates with the finding that exploration is optimal only in a small number of iterations at the beginning of the active learning process (Xu *et al.*, 2003; Donmez *et al.*, 2007). ERS-FF was forcing exploration when exploitation was advantageous.

Results obtained in Section 7.6.4 showed that a subset constructed via pre-filtering using uncertainty sampling contained more informative examples than one constructed using random sub-sampling. The only difference between the two is the manner in which the subset was constructed, therefore it is rational to say that the pre-filtered subset contained more informative examples.

Adaptive Pre-Filtering

The fact that there was no globally optimal pre-filtering technique motivated the need for an adaptive pre-filtering technique, which can balance exploration and exploitation. Section 7.6.5 reports the results of the proposed adaptive pre-filtering technique (ERS-FPL) that uses the follow-the-perturbed-leader to switch among three subset construction techniques namely, random sub-sampling, pre-filtering using uncertainty sampling and pre-filtering using further first sampling. In every iteration of active learning ERS-FPL dynamically switches to the relatively best performing technique where performance is tracked using CEM since the lack of labelled data limits forming a holdout test set.

The goal of the adaptive pre-filtering technique was to be as good as the best fixed pre-filtering technique in hindsight. Overall, comparisons between the various subset optimisation techniques explored in this chapter ranked ERS-FPL as second best with ERS-US ranked first. Moreover, in the individual case where ERS-FPL followed the best performing pre-filtering technique. There is clearly some benefit to performing an adaptive pre-filtering strategy. As part of our future work we intend to explore more advanced reinforcement learning techniques and reward metrics.

Subset Size

We also examined the effect of subset size on ERS. Section 7.6.7 found that ERS-RS benefited from larger subsets while larger subsets did not offer as much improvement for ERS-US. We believe this is due to the fact that larger randomly sub-sampled subset have a greater chance of containing informative examples. Conversely, a pre-filtered subset will have a higher proportion of informative examples at small subset size, hence the addition of more unlabelled examples to the subset will generally be of inferior quality.

It was found that higher accuracy was obtained by ERS-US with a subset of size 10 than by ERS-RS with a subset of size 100. As stated previously, the computational expense of ERS is directly related to the size of the subset. Pre-filtering allows for significant reductions in the size of the subset without adversely affecting accuracy.

Summary

Results obtained in this chapter found that pre-filtering offers a way to both reduce computational overhead of ERS as well as increase accuracy of the classifier produced from active learning. The proposed adaptive pre-filtering technique was shown to be ranked second while ERS-US was shown to be the overall winner.

7.8 Summary

This chapter examined the application of error reduction sampling (ERS) which is a high-performing query selection strategy for active learning. ERS is a computationally expensive process which limits its application in real world settings. Subset optimisation has been proposed as a way to reduce the computational expense of error reduction sampling. Rather than applying ERS to the entire pool of unlabelled examples it is only applied to a small fraction.

The main focus of this chapter was on the manner in which the subset is constructed. The most popular technique is random sub-sampling where the subset is populated with randomly chosen unlabelled examples. We investigated pre-filtering which is an alternative technique where a subset is constructed by filtering unlabelled examples from the pool using a less expensive query selection strategy.

Comparisons were made between random sub-sampling and two fixed pre-filtering techniques. Empirical evaluations found that pre-filtering using uncertainty sampling increases the rate of active learning leading to higher accuracy and lower labelling effort. The increase in the learning rate of active learning was attributed to the intrinsically more informative examples found in subsets constructed using pre-filtering.

While pre-filtering using uncertainty sampling was the overall winner there were problems where pre-filtering using alternative query selection strategies was not optimal. We proposed an adaptive technique that dynamically switches among a number of pre-filtering techniques in every iteration based on the relative performance of each. The goal of the adaptive technique was to be as good as the best technique in hindsight and empirical evaluations showed that the adaptive technique was runner-up to the optimal strategy in the majority of problems.

Finally, sensitivity analysis conducted on the size of the subset revealed that subsets constructed using uncertainty sampling achieved higher rates of learning than larger subsets constructed using random sub-sampling. Pre-filtering allows for smaller subsets to be used without sacrificing potential improved learning rates.

Results obtained in this chapter support the use of pre-filtering as a way to reduce the cost of applying active learning by reducing both computational expense and labelling effort.

Chapter 8

Conclusions

8.1 Introduction

A limiting factor in the widespread application of machine learning solutions has been the cost of gathering sufficient training data for supervised learning. The cost of labelling is often overlooked and in certain domains can be extremely high (e.g. performing expensive biological tests). Active learning has been shown to significantly reduce the number of labelled examples required to induce an accurate classifier using supervised learning. In this thesis we addressed two fundamental problems in active learning, namely the identification of informative unlabelled examples and the computational overhead of active learning.

In this thesis we primarily focused on the *query selection strategy*, which is responsible for selecting informative unlabelled examples. We proposed novel non-Markovian query selection strategies, which identify informative unlabelled examples by incorporating information from prior iterations of active learning. Reducing labelling effort significantly lowers the cost of producing an accurate classifier.

The computational expense of active learning must be considered in practical applications. Efficient and scalable solutions to reduce computational expense in active learning were investigated in this thesis. Some of the best-performing query selection strategies are under-utilised due to their high computational overhead. We investigated optimisation techniques which lowered the computational expense and helped select more informative queries. In addition, high-dimensional text categorisation data is a contributing factor to computational expense. Unsupervised dimensionality reduction techniques were also investigated as a way to lower the computational expense of active learning.

In Section 8.2 we review the main findings of the thesis. Section 8.3 discuss the contributions of the thesis and Section 8.4 outlines general directions of future research.

8.2 Thesis Findings

8.2.1 History-Based Query Selection

Query selection strategies typically select queries based on predictions from a classifier induced in the current iteration. This is a Markov process, which selects queries using ephemeral predictions. We hypothesised that valuable information is being discarded in every iteration of active learning and developed novel query selection strategies that store and incorporate this information.

In Chapter 5 we investigated non-Markovian query selection strategies, which incorporate information from past iterations of active learning into the selection of queries in the current iteration. In each iteration, predictions from the current classifiers are stored in a data structure we call *History*. Given a particular unlabelled example, labelling predictions from every iteration of active learning can be retrieved from the History.

Novel *History-based* query selections strategies were developed. History uncertainty sampling (HUS) selects queries based on the cumulative uncertainty over the past d iterations of active learning. History Kullback-Leibler Divergence (HKLD) selects queries based on disagreement within a committee formed using the classifiers produced in the last d iterations of active learning. Empirical evaluation revealed that incorporating information from the History did benefit active learning, producing higher accuracy classifiers than Markovian query selection strategies in certain situations.

HUS and HKLD placed equal emphasis on all predictions, including those predictions from the History. The current classifier is likely to be the most accurate, therefore the most recent predictions should receive greater emphasis. Filtered variants of the History-based query selection strategies were proposed (HUSF and HKLDF respectively) to address this problem. A subset of the pool is formed by filtering the N most uncertain examples using uncertainty sampling. History-based query selection is applied only to the subset. Empirical evaluation revealed that both HUSF and HKLDF demonstrated significant improvement over a Markovian query selection strategy.

History-based query selection offers a way to reduce the labelling effort of active learning by aiding the selection of more informative examples as queries. Furthermore, the computational expense of History-based query selection strategies are extremely low since they reuse predictions made in previous iterations of active learning.

8.2.2 Pre-Filtering Error Reduction Sampling

Error reduction sampling (ERS) has been shown to be a top-performing query selection strategy. However, practical application is limited by the excessive computational expense of the strategy. Several optimisation techniques have been proposed to reduce the computational overhead, with subset optimisation being the most popular. The computational expense of ERS is proportional to the number of unlabelled examples it must consider. Significant computational savings can be obtained by applying ERS only to a subset of the unlabelled pool. The most popular method for constructing the subset is random sub-sampling, which selects unlabelled examples at random from the pool.

In Chapter 7 an alternative subset optimisation technique known as *pre-filtering* was extensively investigated. Pre-filtering constructs the subset by filtering examples from the pool using a query selection strategy. Comparisons were made between random sub-sampling and pre-filtering using both an exploration-based strategy (furthest first) and an exploitation-based strategy (uncertainty sampling). Pre-filtering ERS using uncertainty sampling (ERS-US) was found to be the best performing technique overall. The accuracy of classifiers produced using ERS-US were found to achieve significantly higher accuracy than classifiers produced using either random sub-sampling (ERS-RS) or pre-filtering using furthest first (ERS-FF). In general, it was found that exploitation is preferable for ERS. However, this is not always the case and in certain problems exploration is optimal.

We proposed a novel *adaptive pre-filtering* technique, which dynamically chooses either an exploration-based or an exploitation-based query selection strategy to form the subset in every iteration of active learning. The adaptive technique attempts to balance exploration and exploitation. Empirical evaluation showed that the adaptive technique was ranked second overall, outperforming ERS-RS and ERS-FF, while still tracking the performance of ERS-FF in problems where exploration was found to be optimal.

Pre-filtering allows for potential reductions in the computational overhead. ERS-US was found to achieve similar levels of deficiency to ERS-RS using subsets an order of magnitude smaller. Smaller subsets allow for significant savings in computational expense for ERS.

8.2.3 Dimensionality Reduction for Active Learning Settings

As well as increasing computational and storage overheads, high-dimensional data can limit the choice of classifier used in active learning. The k-NN classifier exhibits state-of-the-art performance when operating in a reduced feature set given by an appropriate supervised feature selection (Yang & Liu, 1999). However, it performs poorly in high-dimensional data as it is susceptible to the *curse of dimensionality.* In order to use a k-NN classifier in an active learning setting, dimensionality reduction must first be applied.

The most commonly used dimensionality reduction techniques employed in text categorisation are supervised feature selection techniques. These have been shown to significantly reduce computational overhead and in some cases increase the accuracy of the classifier produced (Yang & Pedersen, 1997). However, the majority of data supplied to active learning is unlabelled, which poses a significant problem for the application of supervised dimensionality reduction techniques.

Chapter 6 investigated the effect of unsupervised dimensionality reduction techniques on active learning. Document frequency applied with a global policy (DFG), was shown to reduce the dimensionality and improve the accuracy of the k-NN classifier in active learning. Principal Components Analysis (PCA) was found to be the best-performing unsupervised dimensionality reduction technique. The k-NN operating in the PCA-reduced data achieved similar levels of accuracy as a k-NN operating in dimensionally reduced data given by a top performing supervised feature selection technique.

8.3 Thesis Contribution

This thesis reviewed both automated text categorisation and active learning, with particular emphasis placed on query selection strategies in active learning. We identified two fundamental problems in active learning, namely the computational expense of performing active learning and the selection of informative unlabelled examples. Novel techniques and strategies were proposed as possible solutions to these problems.

8.3.1 Computational Efficiency of Active Learning

The computational expense of active learning must be considered in practical applications. Some of the most promising query selection strategies are under-utilised due to their computational overhead. We developed novel, efficient and scalable solutions to reduce the computational overhead of active learning. Unsupervised dimensionality reduction was found to increase the performance of the k-NN classifier in an active learning setting. Pre-filtering error reduction sampling demonstrated both considerable saving in computational overhead, by using smaller subsets and reduced labelling effort, by selecting more informative unlabelled examples using error reduction sampling.

8.3.2 Incorporating Past Information

We investigated ways to incorporate information from prior iterations of active learning into the selection of queries in the current iteration. History-based query selection did this explicitly by incorporating the predictions from previous iterations of active learning into the selection of queries in the current iteration. Adaptive pre-filtering did this indirectly by selecting the pre-filtering technique based on the performance of each fixed pre-filtering technique chosen in previous iterations. Empirical evaluations found that incorporating information from previous iterations selects more informative examples than traditional Markovian query selection strategies.

8.4 Future Work

8.4.1 Active Learning Performance

The performance of supervised learning is measured using a hold-out test set. Given the scarcity of labelled data in an active learning setting, withholding labelled data to form a test set is impractical. An accurate way of measuring the performance of active learning would be of major benefit in a host of applications. One obvious use for such a metric would be as a reward in an adaptive query selection strategy. The ability to accurately monitor the progress of active learning would allow for the optimal strategy to be selected in every iteration.

8.4.2 Distributed Computing

Significant computational *speed-up* is possible through the use of distributed processing in machine learning. A number of the algorithms presented in this thesis are *embarrassingly parallel* (Foster, 1995). Error reduction sampling (Roy & McCallum, 2001) in particular could see significant computational speed-up if implemented in a distributed manner.

8.4.3 Early Stopping

An interesting phenomenon reported in the baseline analysis in Chapter 4 and which has been previously reported in the literature (Schohn & Cohn, 2000) is that active learning can achieve higher accuracy than that of a passive learner trained on the entire labelled pool. These results motivate the need for an *early stopping* mechanism for active learning.

Section 3.3.4 reviewed stopping criteria for active learning. However, early stopping active learning remains an open problem since an effective generic solution has not been proposed. The ability of active learning to detect and stop at a global maxima could result in improved accuracy and reduce labelling effort if the maxima is reached at an early iteration.

8.4.4 Alternative Domains

The novel techniques proposed in this thesis can be ported to alternative domains. In particular we believe that the construction of classifiers from small amounts of training data is paramount to many labour saving tasks emerging as part of social networking and online presence. Reducing labelling effort makes classifiers more ephemeral. They can be created quickly and efficiently in an *ad-hoc* manner and applied to many more domains.

Media is generally labelled by the content producer into a broad set of categories. However, consumers are defining custom labels, which are used to categorise media in a personalised way. Categories are subjective and the availability of large annotated corpora of user-defined labels are difficult to obtain. Active learning offers a tailor made solution to the problem of automating categorisation of new media using personalised labels or tags.

Singh *et al.* (2008) examines ways of annotating images using active learning to train a classifier. We believe that the techniques proposed in this thesis to lower the cost of active learning could be directly applied to such tasks. Tagging all manner of multimedia is a growing trend. An accurate classifier capable of labelling many thousands of items after being trained with only a small number of examples is an ideal solution for automating such tasks.

Bibliography

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, 319–342.

Apté, C., Damerau, F. & Weiss, S. (1994). Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, **12**, 233–251.

Auer, P., Cesa-Bianchi, N., Freund, Y. & Schapire, R. (2003). The nonstochastic multiarmed bandit problem. *SIAM JOURNAL ON COMPUTING*, **32**, 48–77.

Balcan, M., Beygelzimer, A. & Langford, J. (2006). Agnostic active learning. *Proceedings of the 23rd international conference on Machine learning*, 65–72.

Baram, Y., El-Yaniv, R. & Luz, K. (2004). Online choice of active learning algorithms. *The Journal of Machine Learning Research*, **5**, 255–291.

Baum, E. & Lang, K. (1992). Query learning can work poorly when human oracle is used. *International Joint Conference in Neural Networks*.

Becker, M. (2008). *Active Learning An Explicit Treatment of Unreliable Parameters*. Ph.D. thesis, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh.

Bingham, E. & Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 245–250, ACM New York, NY, USA.

Blum, A. & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 92–100.

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, **24**, 123–140.

Campbell, C., Cristianini, N. & Smola, A. (2000). Query learning with large margin classifiers. *Proceedings of the Seventeenth International Conference on Machine Learning*, 111–118.

Chapelle, O., Weston, J. & Scholkopf, B. (2003). Cluster kernels for semi-supervised learning. *Advances in Neural Information Processing Systems*, **15**, 1.

Cohn, D., Atlas, L. & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, **15**, 201–221.

Cohn, D., Ghahramani, Z. & Jordan, M. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, **4**, 129–145.

Comon, P. (1994). Independent component analysis, a new concept. *Signal Processing*, **36**, 287–314.

Culver, M. (2006). *Active Learning to Maximize Area Under the ROC Curve*. Master's thesis, University of Nebraska, Department of Computer Science and Engineering, Avery 256.

Culver, M., Kun, D. & Scott, S. (2006). Active learning to maximize area under the ROC curve. *Proceedings of the Sixth International Conference on Data Mining*, 149–158.

Dagan, I. & Engelson, S. (1995). Committee-based sampling for training probabilistic classifiers. *Proceedings of the Twelfth International Conference on Machine Learning*, 150–157.

Dasgupta, S. (2005). Analysis of a greedy active learning strategy. *Advances in Neural Information Processing Systems 17: Proceedings Of The 2004 Conference*.

Debole, F. & Sebastiani, F. (2005). An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, **56**, 584–596.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T. & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, **41**, 391–407.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, **7**, 1–30.

Donmez, P., Carbonell, J. & Bennett, P. (2007). Dual strategy active learning. In *ECML 07: Proceedings of the 18th European Conference on Machine Learning*, vol. 4701, 116, Springer.

Dumais, S., Platt, J., Heckerman, D. & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of the seventh international conference on Information and knowledge management*, 148–155.

Forgy, E.W. (1965). Cluster analysis of multivariate data: efficiency vs interpretability of classifications. *Biometrics*, **21**, 768–769.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, **3**, 1533–7928.

Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Freund, Y., Seung, H., Shamir, E. & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, **28**, 133–168.

Guo, Y. & Greiner, R. (2007). Optimistic active learning using mutual information. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Hayes, P. & Weinstein, S. (1990). Construe/tis: A system for content-based indexing of a database of news stories. *Proceedings of the The Second Conference on Innovative Applications of Artificial Intelligence*, 49–64.

Hochbaum, D. & Shmoys, D. (1985). A best possible heuristic for the k-center problem. *Mathematics of operations research*, **10**, 180–184.

Hoi, S., Jin, R. & Lyu, M. (2006). Large-scale text categorization by batch mode active learning. *Proceedings of the 15th international conference on World Wide Web*.

Hsu, C. & Lin, C. (2002). A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, **13**, 415–425.

Iyengar, V., Apte, C. & Zhang, T. (2000). Active learning using adaptive resampling. *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 91–98.

Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features. Tech. Rep. 23, Universität Dortmund, LS VIII-Report.

Kalai, A. & Vempala, S. (2005). Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, **71**, 291–307.

Kirby, M. & Sirovich, L. (1990). Application of the Karhunen-Loève Procedure for the Characterization of Human Faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 103–108.

Lewis, D. & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. *Proceedings of the Eleventh International Conference on Machine Learning*, 148–156.

Lewis, D. & Gale, W. (1994). A sequential algorithm for training text classifiers. *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 3–12.

Lewis, D., Yang, Y., Rose, T. & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, **5**, 361–397.

Lindenbaum, M., Markovitch, S. & Rusakov, D. (1999). Selective sampling for nearest neighbor classifiers. *Machine Learning*, 366–371.

Manning, C., Raghavan, P. & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

McCallum, A. & Nigam, K. (1998). Employing EM in pool-based active learning for text classification. *Proceedings of the 15th International Conference on Machine Learning*, 350–358.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Nguyen, H.T. & Smeulders, A. (2004). Active learning using pre-clustering. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, 79, ACM, New York, NY, USA.

Osugi, T., Kun, D. & Scott, S. (2005). Balancing exploration and exploitation: A new algorithm for active machine learning. *Proceedings of the Fifth IEEE International Conference on Data Mining*, 330–337.

Pandey, G., Gupta, H. & Mitra, P. (2005). Stochastic scheduling of active support vector learning algorithms. *Proceedings of the 2005 ACM Symposium on Applied computing*, 38–42.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, **2**, 559–572.

Porter, M. (1980). An algorithm for suffix stripping. *Program*, **14**, 130–137.

Roweis, S. & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**, 2323.

Roy, N. & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *Proceedings of the 18th International Conference on Machine Learning*, 441–448.

Salton, G. & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, **24**, 513–23.

Salton, G., Wong, A. & Yang, C. (1975). A vector space model for automatic indexing. *Communications of the ACM*, **18**, 613–620.

Saunier, N., Midenet, S. & Grumbach, A. (2004). Stream-based learning through data selection in a road safety application. *STAIRS 2004, Proceedings of the Second Starting AI Researchers' Symposium*, **109**, 107–117.

Schapire, R. (1999). A brief introduction to boosting. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, **2**, 1401–1406.

Schohn, G. & Cohn, D. (2000). Less is more: Active learning with support vector machines. *Proceedings of the 17th International Conference on Machine Learning*, 839–846.

Schölkopf, B., Burges, C. & Smola, A. (1999a). *Advances in Kernel Methods: Support Vector Learning*. MIT Press.

Schölkopf, B., Smola, A. & Muller, K. (1999b). Kernel principal component analysis. *Advances in Kernel Methods-Support Vector Learning*, 327–352.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, **34**, 1–47.

Seung, H., Opper, M. & Sompolinsky, H. (1992). Query by committee. *Proceedings of the fifth annual workshop on Computational learning theory*, 287–294.

Singh, M., Cunningham, P. & Curran, E. (2008). Active learning for multi-label image annotation. In *Proc. 19th Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2008)*, Dublin, Ireland.

Sutton, R. & Barto, A. (1998). *Reinforcement learning: An introduction*. MIT press.

Tong, S. & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, **2**, 45–66.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, **27**, 1134–1142.

Vapnik, V. (1995). The nature of statistical learning theory.

Vermorel, J. & Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation. *Proceedings of the 16th European Conference on Machine Learning*, 437–448.

Vlachos, A. (2007). A stopping criterion for active learning. *Computer Speech & Language*.

Witten, I.H. & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd edn.

Wolpert, D. (1995). The relationship between pac, the statistical physics framework, the bayesian framework, and the vc framework. *The Mathematics of Generalization*, 117–214.

Xu, Z., Yu, K., Tresp, V., Xu, X. & Wang, J. (2003). Representative sampling for text classification using support vector machines. *Advances in Information Retrieval: 25th European Conference on IR Research, ECIR 2003*.

Yang, Y. & Liu, X. (1999). A re-examination of text categorization methods. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 42–49.

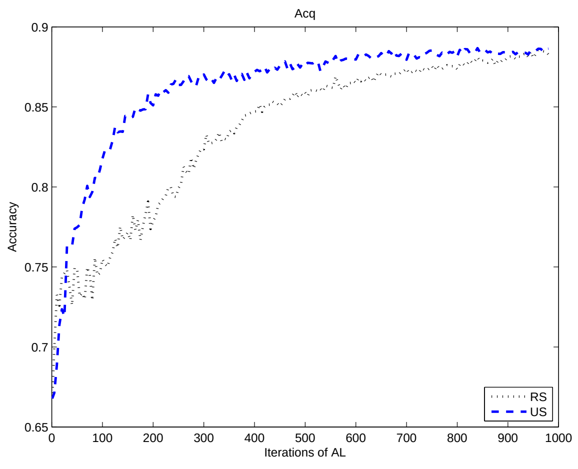
Yang, Y. & Pedersen, J. (1997). A comparative study on feature selection in text categorization. *Proceedings of the 14th International Conference on Machine Learning*, **97**.

Yang, Y., Zhang, J. & Kisiel, B. (2003). A scalability analysis of classifiers in text categorization. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, 96–103.

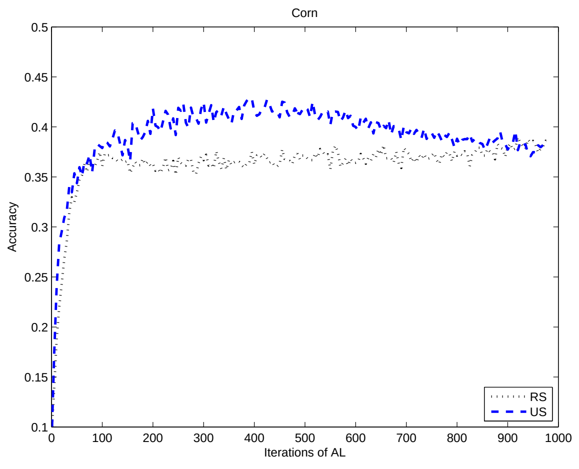
Appendix A

Supplemental Baseline Analysis Results

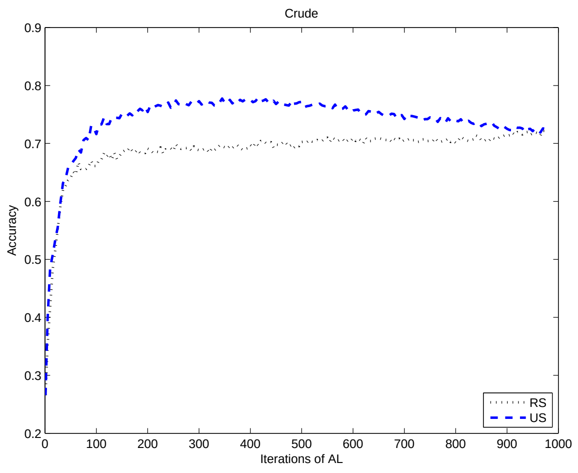
We present additional learning curves for the baseline analysis experiments in Chapter 4 on the R10 dataset. Figures reports the learning curves for each category where F_1 is averaged over all 10 trails of the experiment.



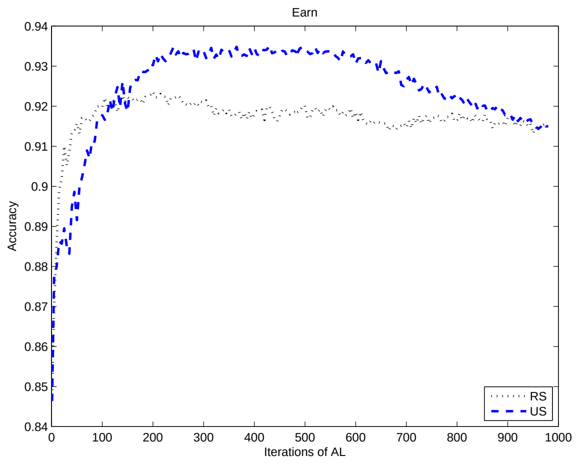
(a) F_1



(b) F_1

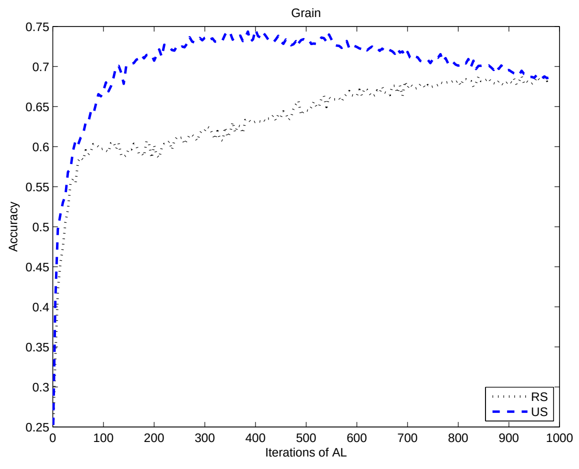


(c) F_1

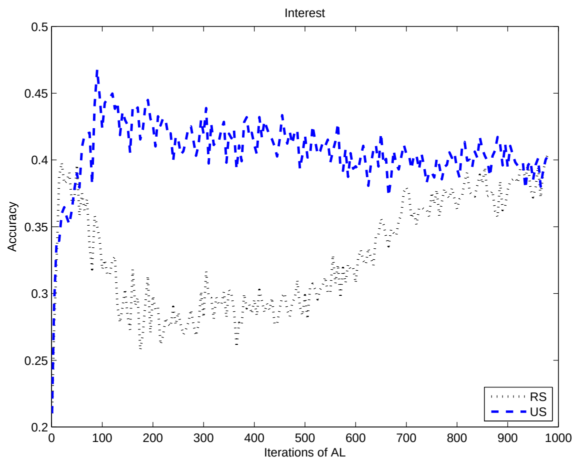


(d) F_1

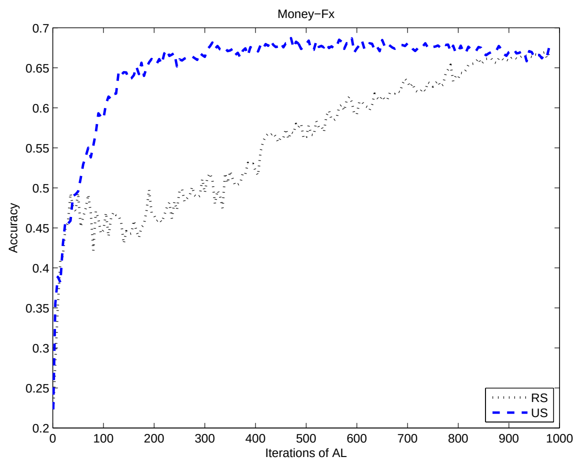
Figure A.1: Baseline Analysis Learning Curves on Acquisitions, Corn, Crude and Earn categories. A committee of 10 multinomial naïve Bayes classifiers (constructed using bagging) were used. F_1 was used as the performance metrics



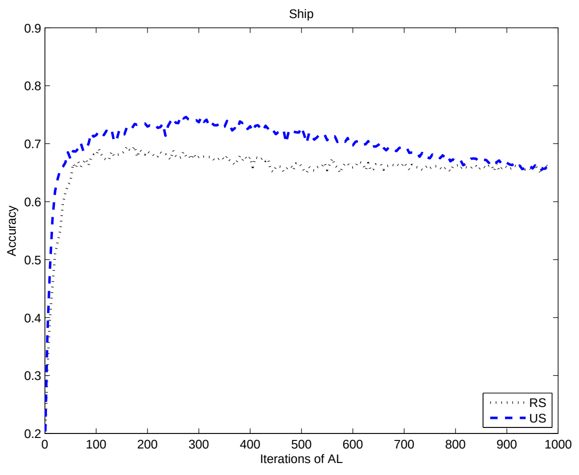
(a) F_1



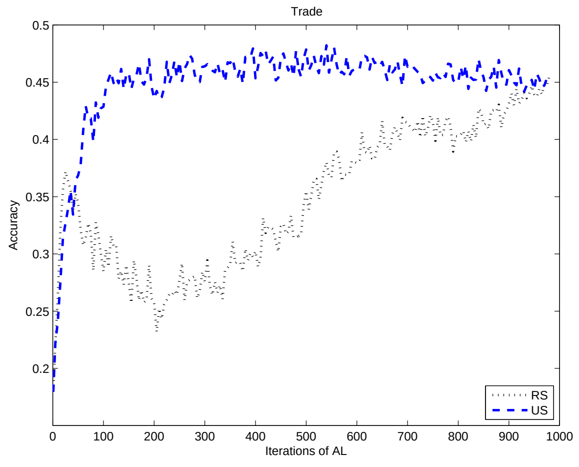
(b) F_1



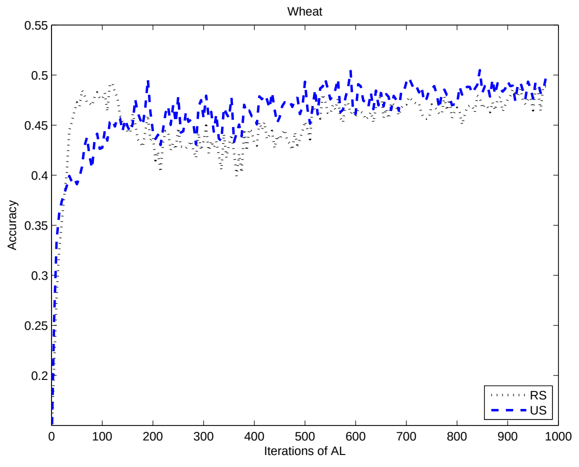
(c) F_1



(d) F_1



(e) F_1



(f) F_1

Figure A.2: Baseline Analysis Learning Curves on Grain, Interest, Money-Fx, Ship, Trade and Wheat categories. A committee of 10 multinomial naïve Bayes classifiers (constructed using bagging) were used. F_1 was used as the performance metrics

Appendix B

Supplemental History-based Query Selection Results

We present additional learning curves for the history-based query selection as desribed in Chapter 5. Analysis of different subset sizes ($t = \{10, 25, 50, 100\}$) and different history depths ($d = \{3, 5, 7\}$) was conducted.

B.1 HBQS

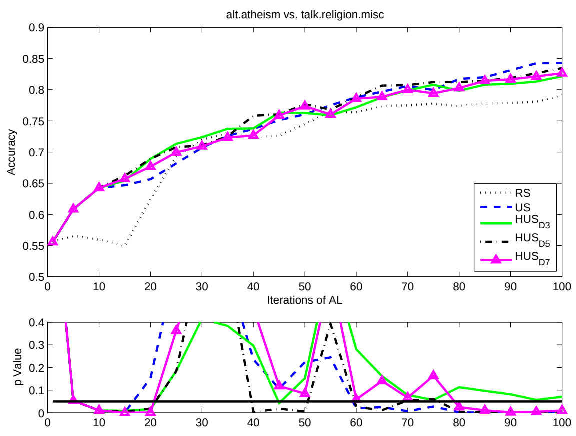
Figure B.1 displays the individual learning curves obtained when HUS was applied to each of the 20NG-4 sub-problems. Similarly, Figure B.2 plots the individual learning curves obtained when HKLD was applied to each of the four problems in the 20NG-4 dataset.

Results obtained by performing HUSF on the 20NG-4 are given in Figure B.3 where a subset of size 10 was used. Figure B.4 where a subset of size was used 25, Figure B.5 where a subset of size 50 was used and Figure B.6 where a subset of size 100 was used.

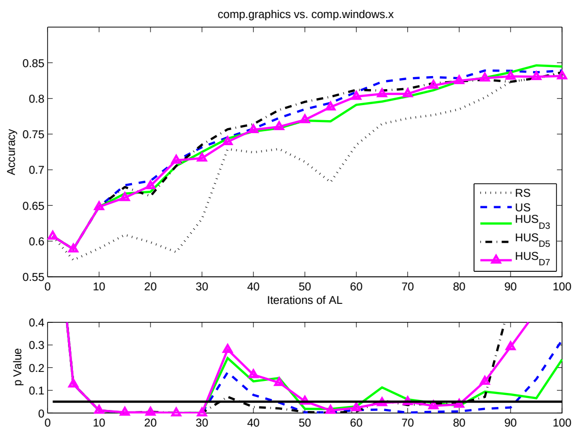
B.2 Filtered HBQS

Results obtained by performing HUSF on the 20NG-4 are given in Figure B.3 where a subset of size 10 was used. Figure B.4 where a subset of size was used 25, Figure B.5 where a subset of size 50 was used and Figure B.6 where a subset of size 100 was used.

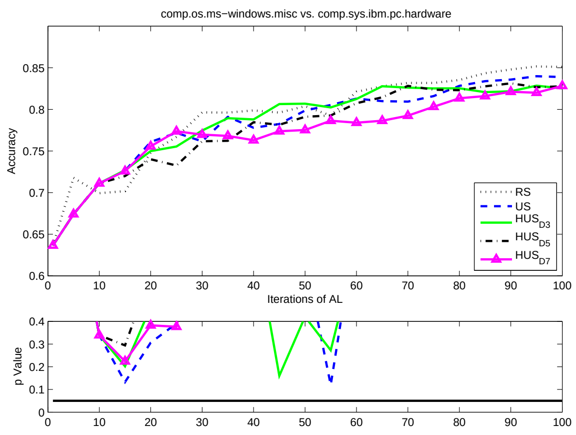
Similarly, results obtained by performing HKLDF on the 20NG-4 are given in Figure B.7 where a subset of size 10 was used. Figure B.8 where a subset of size was used 25, Figure B.9 where a subset of size 50 was used and Figure B.10 where a subset of size 100 was used.



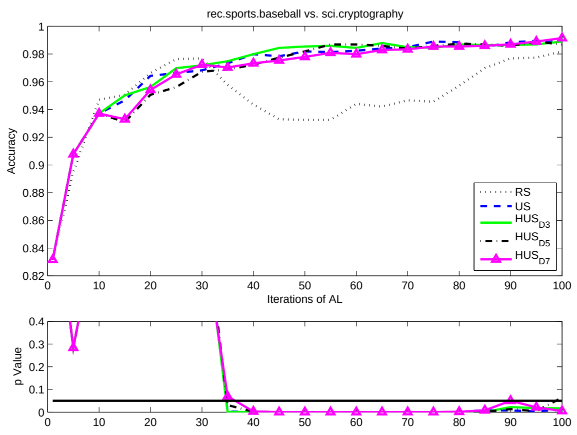
(a) AR



(b) GX

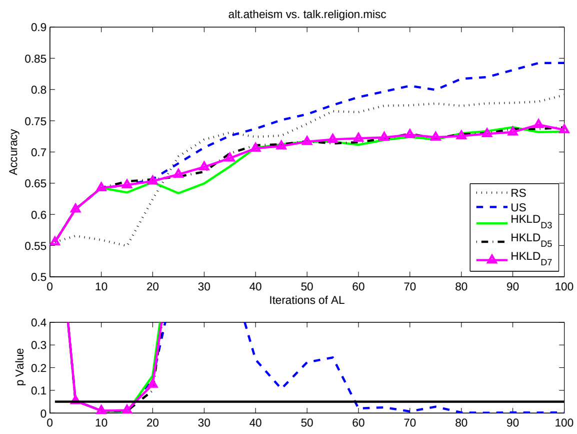


(c) WH

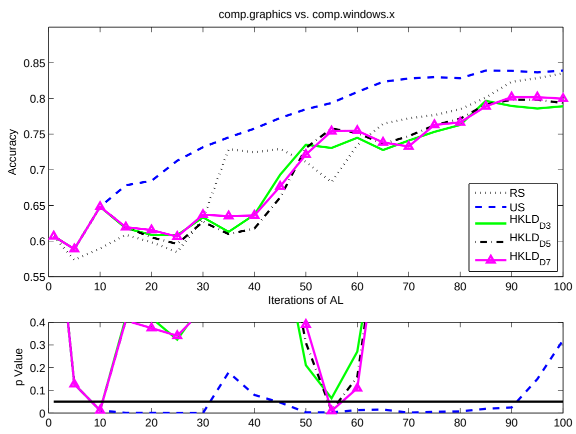


(d) BC

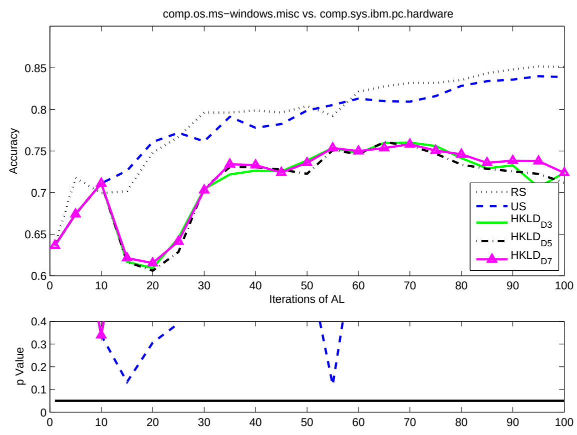
Figure B.1: HUS results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



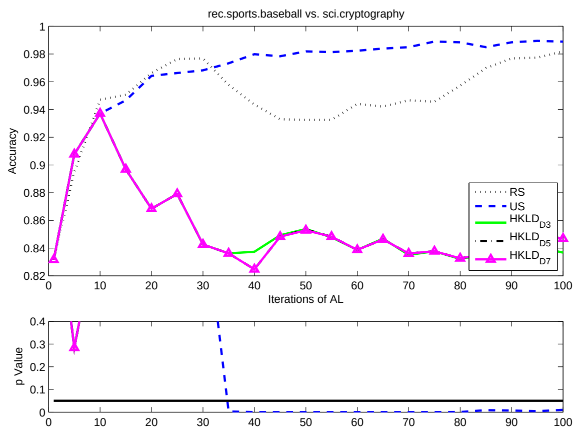
(a) AR



(b) GX

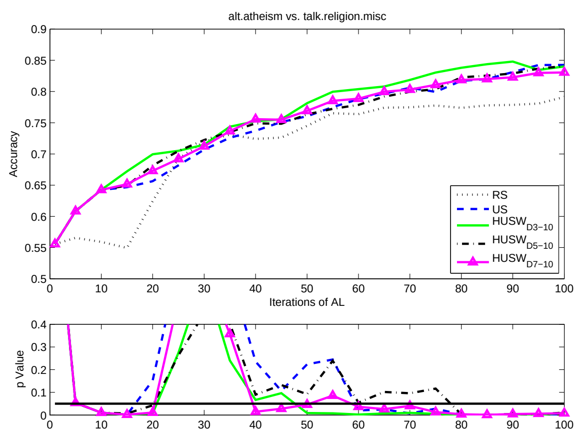


(c) WH

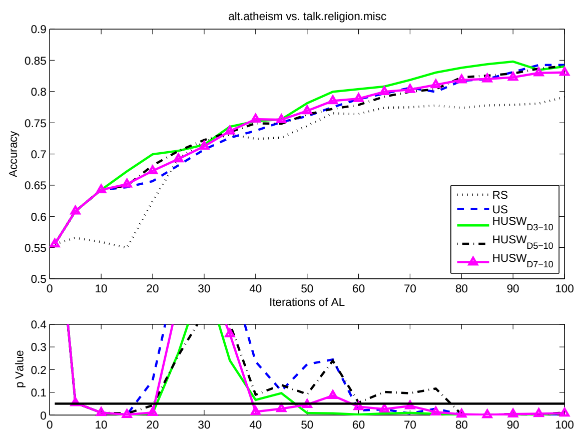


(d) BC

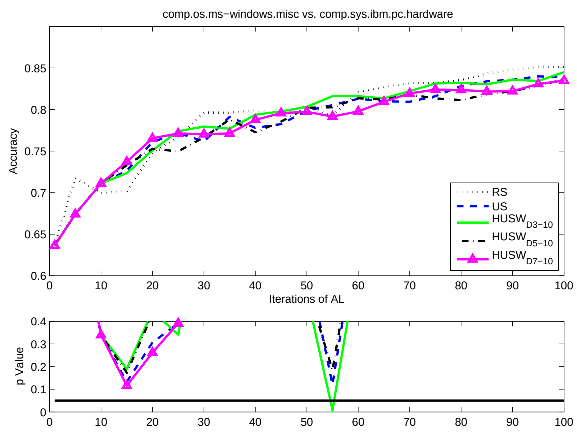
Figure B.2: HKLD results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



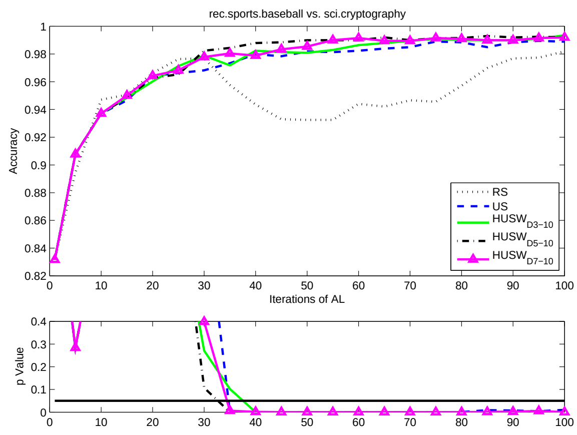
(a) AR



(b) GX

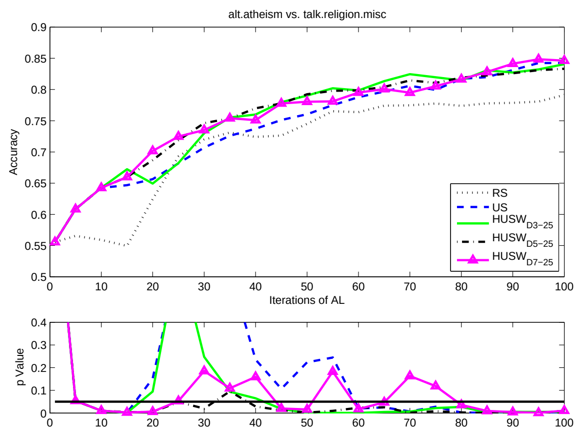


(c) WH

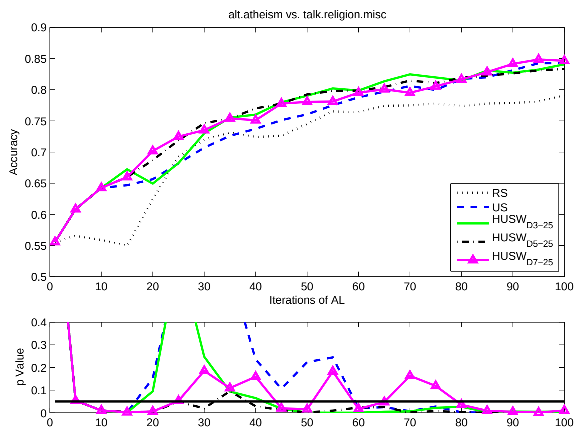


(d) BC

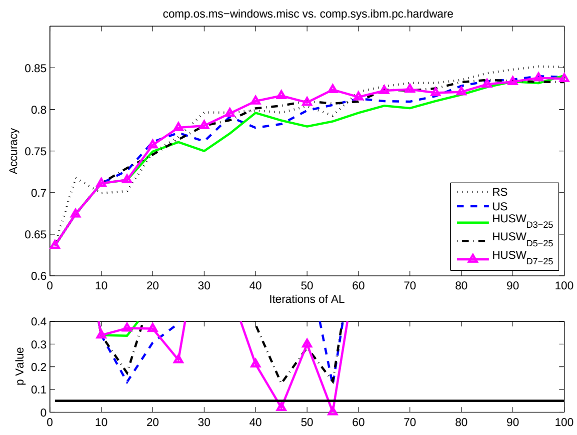
Figure B.3: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



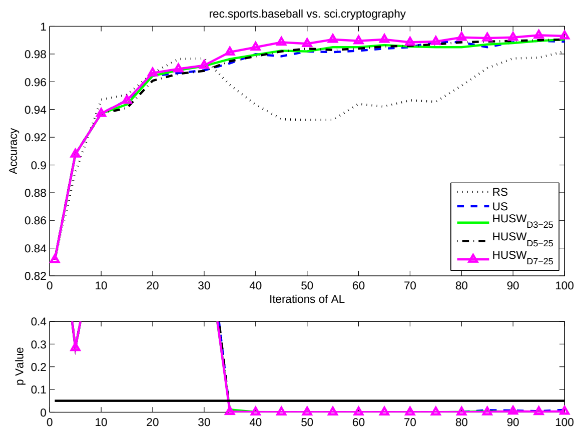
(a) AR



(b) GX

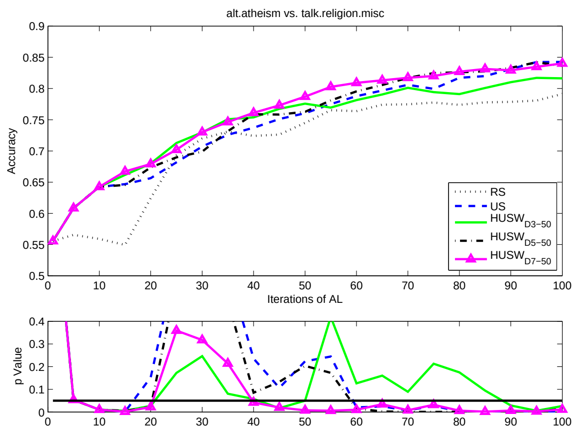


(c) WH

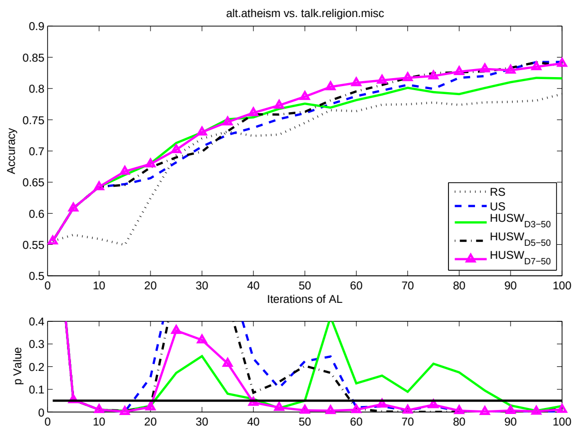


(d) BC

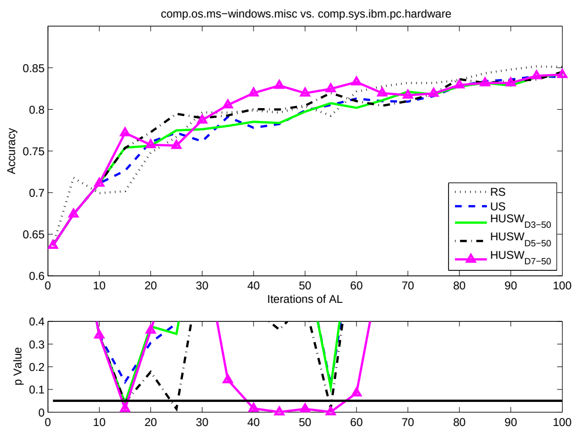
Figure B.4: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



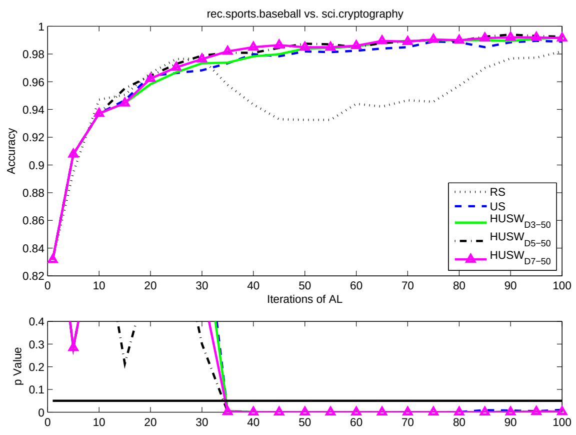
(a) AR



(b) GX

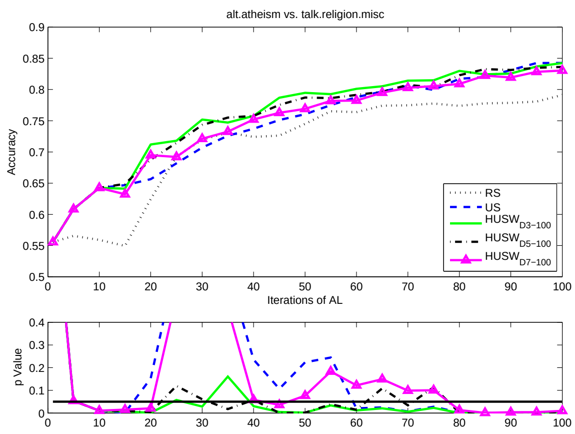


(c) WH

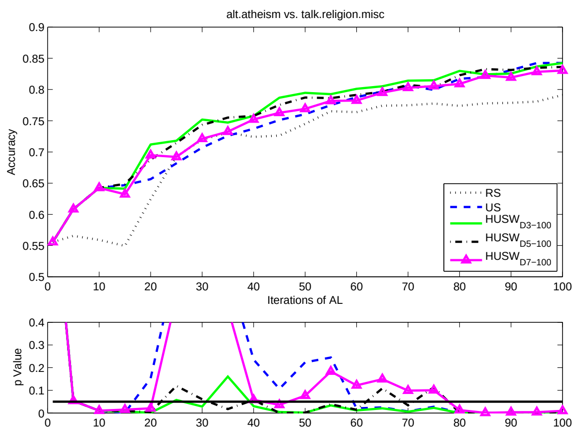


(d) BC

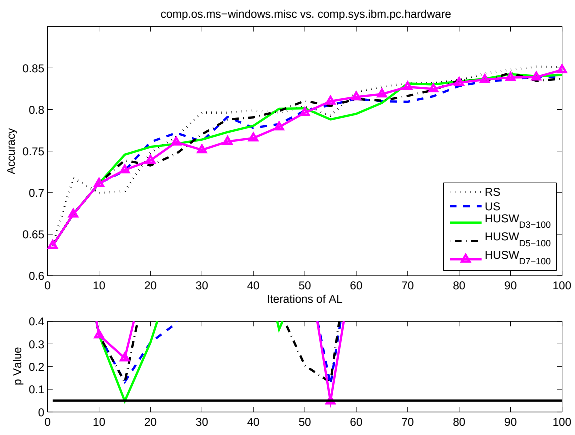
Figure B.5: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



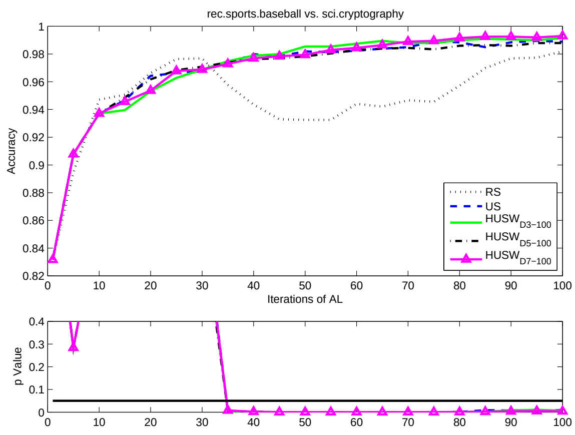
(a) AR



(b) GX

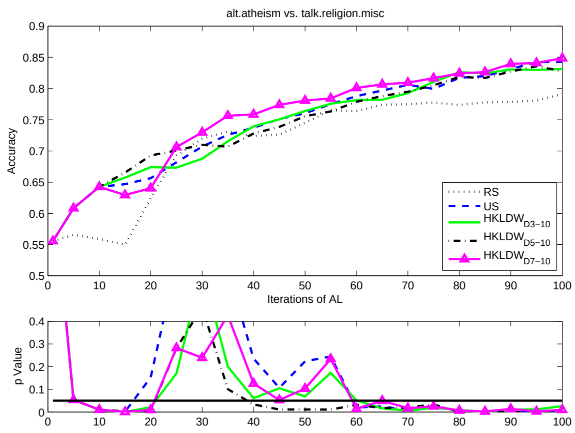


(c) WH

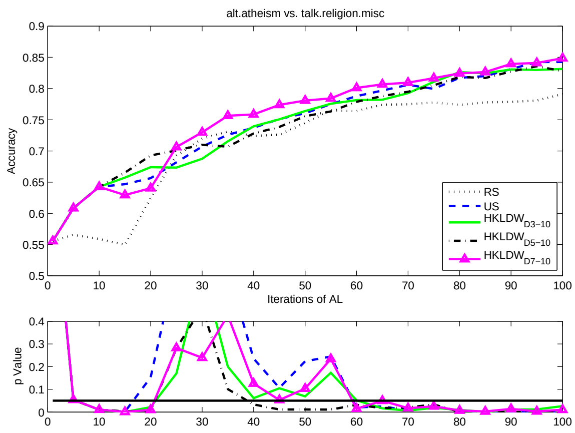


(d) BC

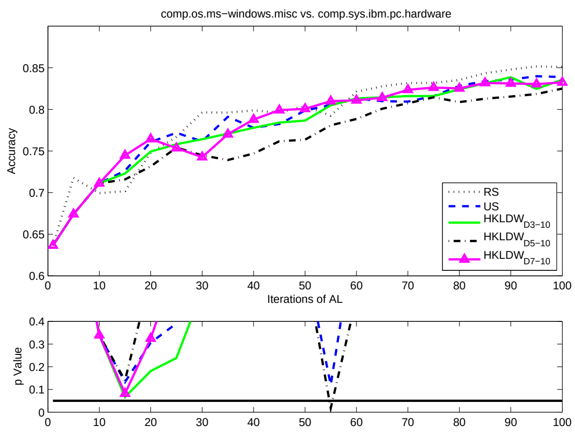
Figure B.6: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



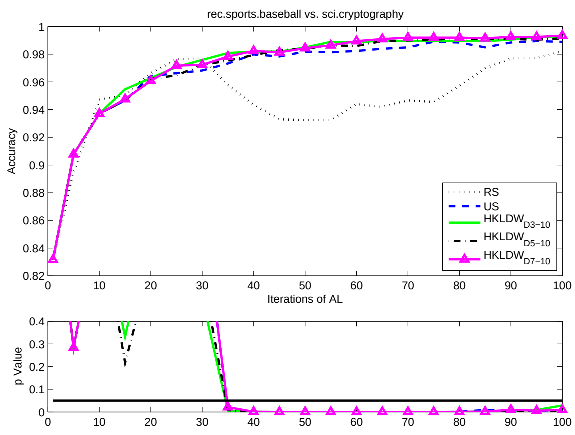
(a) AR



(b) GX

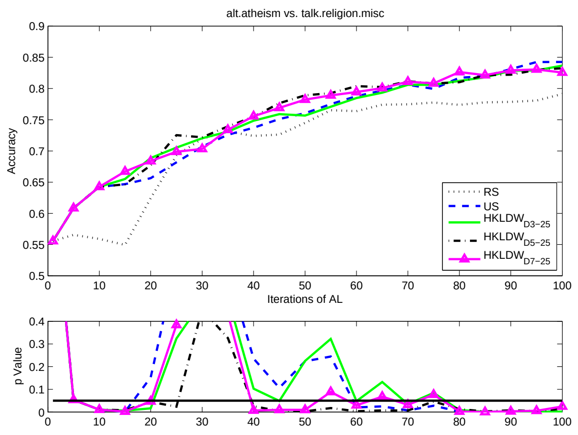


(c) WH

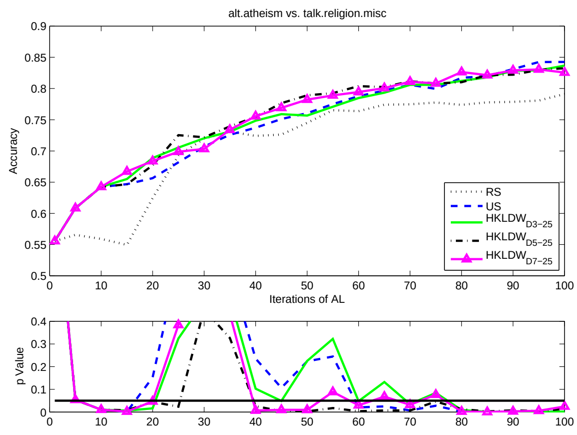


(d) BC

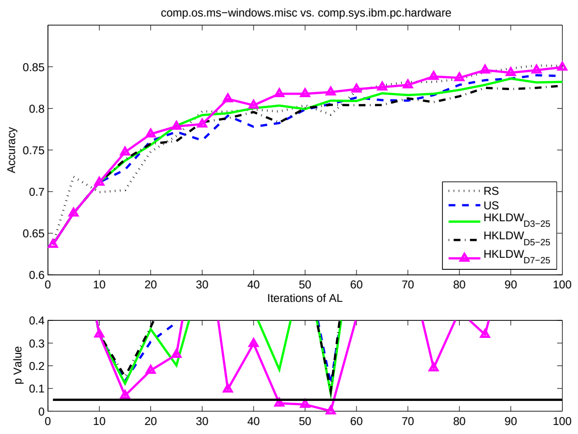
Figure B.7: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



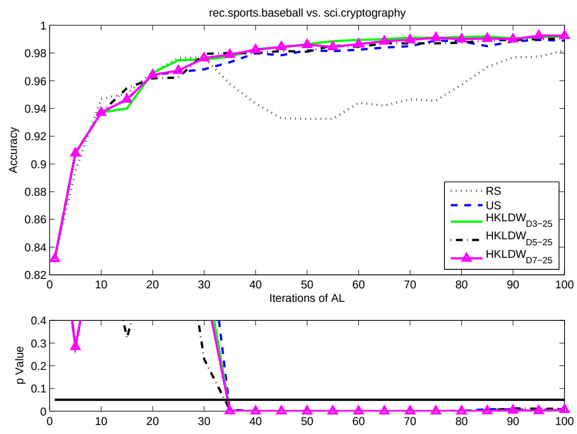
(a) AR



(b) GX

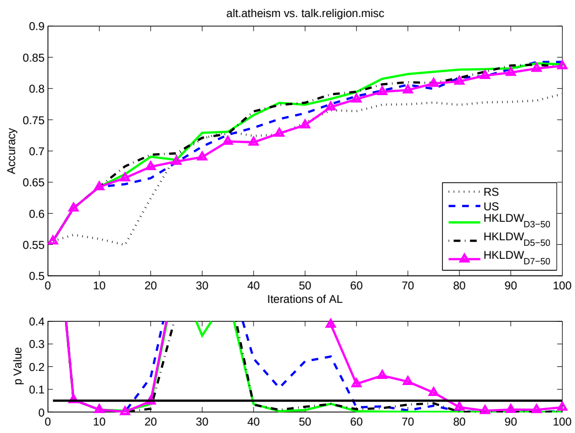


(c) WH

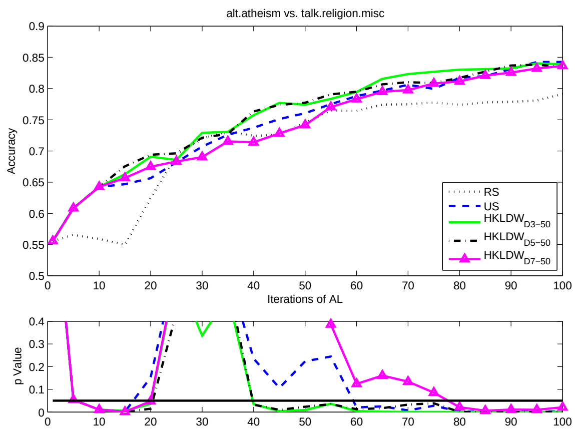


(d) BC

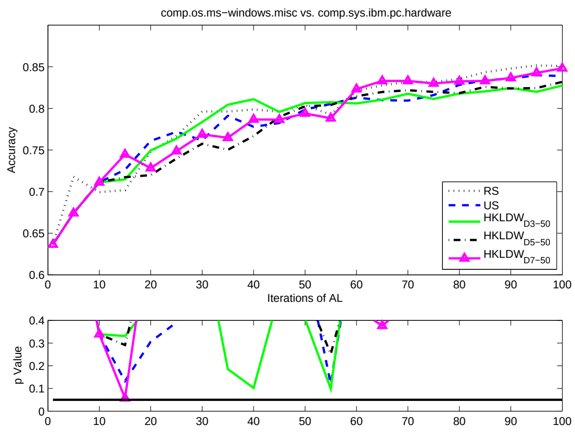
Figure B.8: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



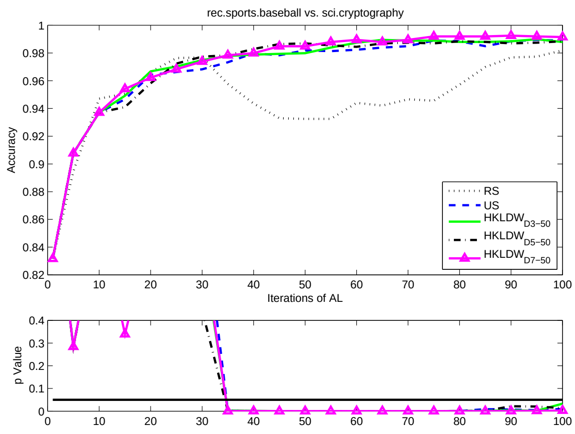
(a) AR



(b) GX

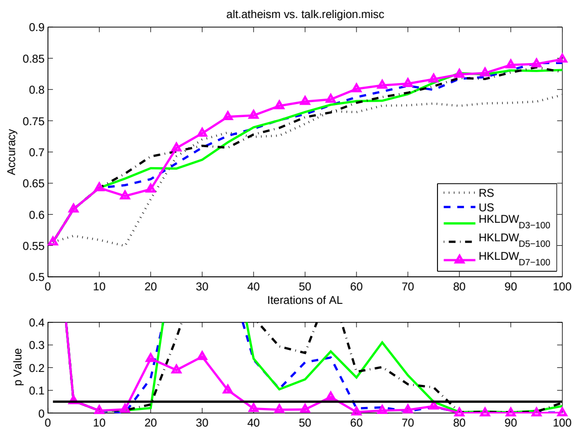


(c) WH

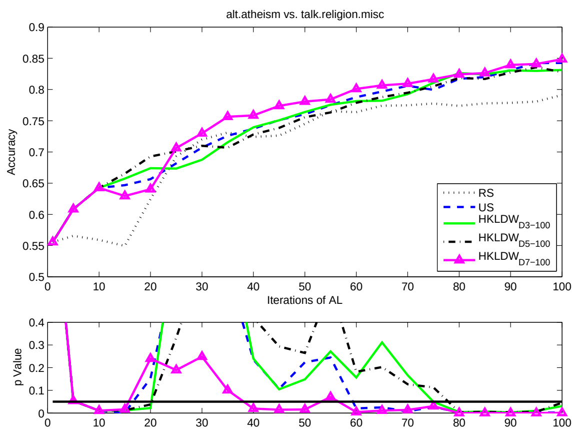


(d) BC

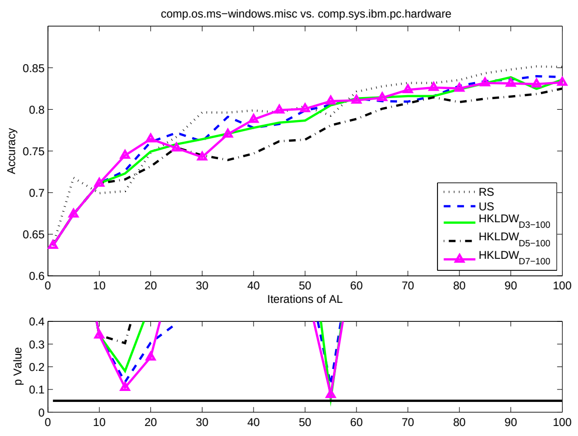
Figure B.9: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)



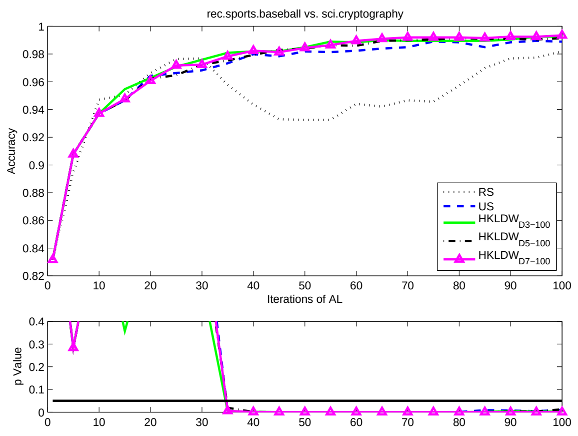
(a) AR



(b) GX



(c) WH



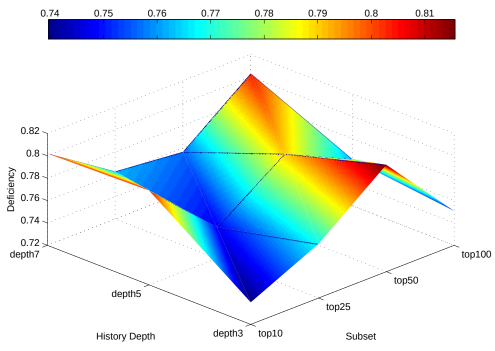
(d) BC

Figure B.10: Results obtained using a multinomial naïve Bayes classifier with Active Learning run for 100 iterations, seeded with 3 examples from each class and selecting 1 query per iteration. Accuracy was used as the performance metric and evaluation on the test set was done after every 5 iterations. Below each graph the p-value showing statistical significance against random sampling (RS) is given ($\alpha = 0.05$ is shown as a solid line)

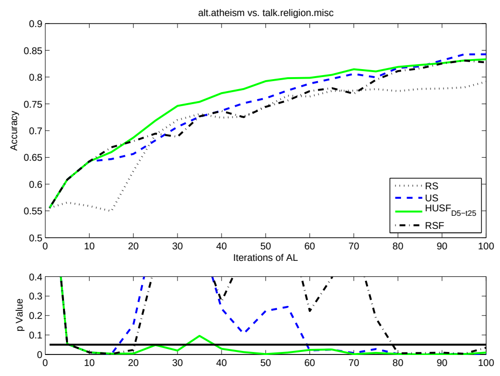
B.2.1 Optimal Filtered HBQB

Figure B.11 plots the learning curves for the optimal HUSF. Optimal parameters for history depth and subset size were taken from the minimum deficiency value. A surface plot of the deficiency values obtained is given beside each learning curve plot.

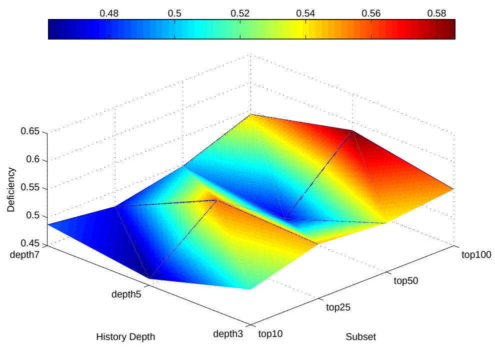
Figure B.12 plots the learning curves for the HKLDF. Optimal parameters for history depth and subset size were taken from the minimum deficiency value. A surface plot of the deficiency values obtained is given beside each learning curve plot.



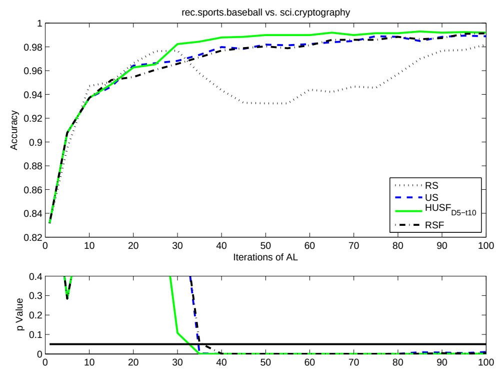
(a) AR



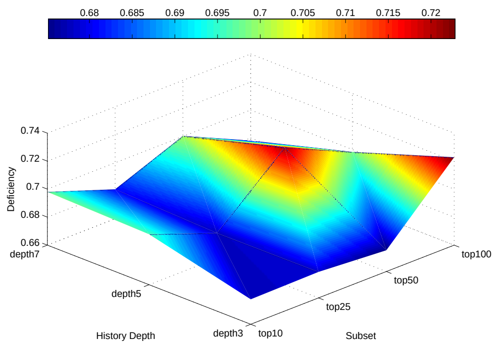
(b) AR



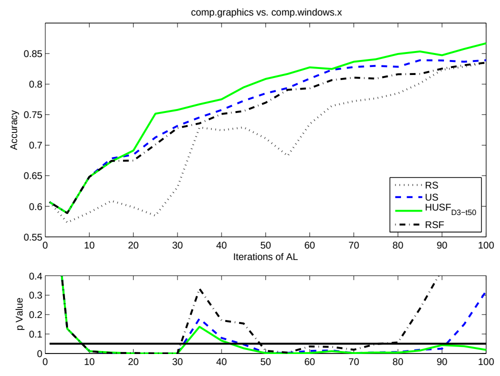
(c) BC



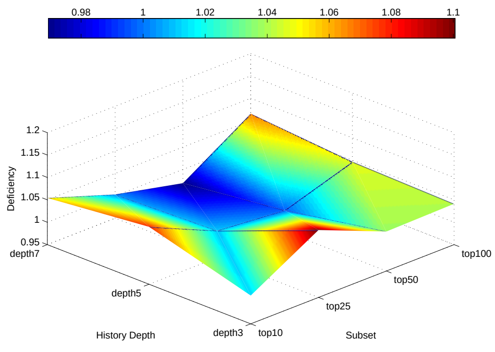
(d) BC



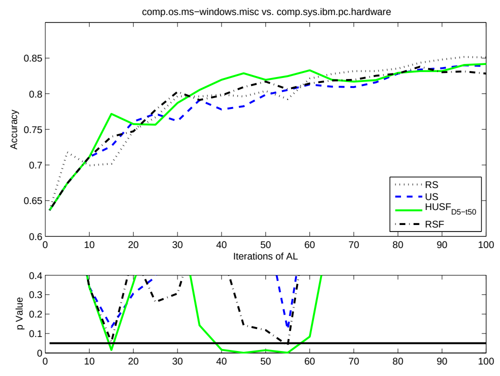
(e) GX



(f) GX

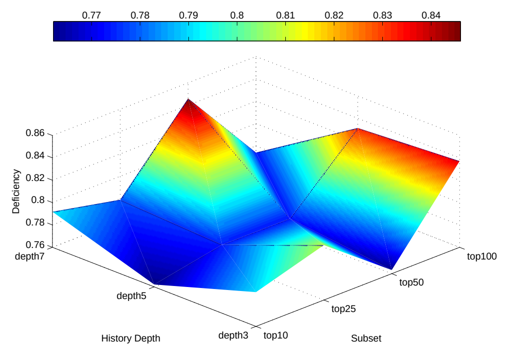


(g) WH

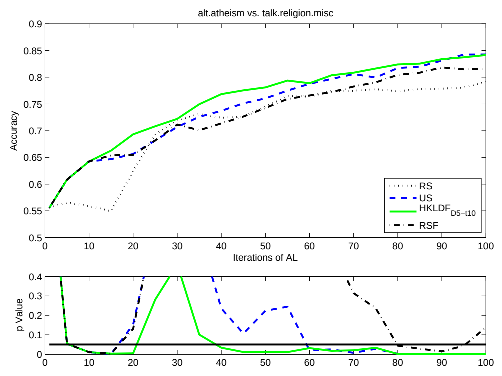


(h) WH

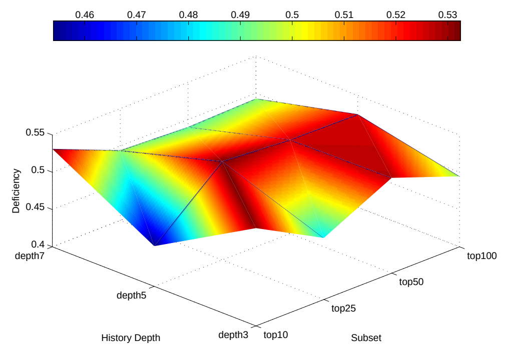
Figure B.11: HUSF learning curves for optimal history depth and subset size in each of the four 20NG-4 problems. A surface plot of the deficiency values for both parameters is shown next to each learning curve



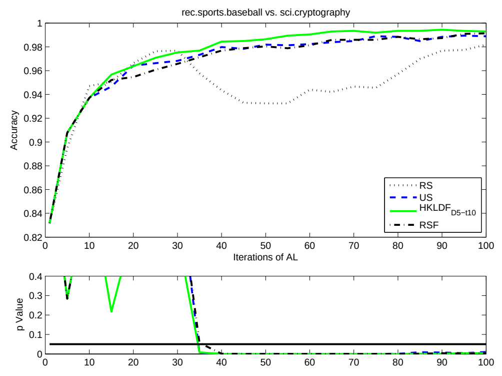
(a) AR



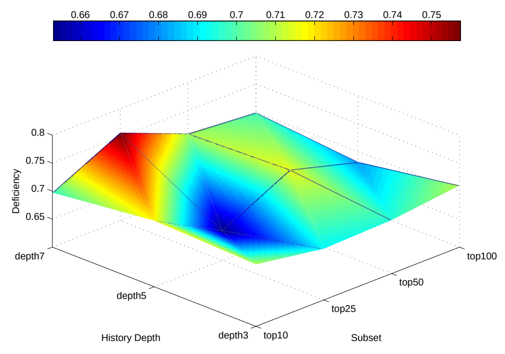
(b) AR



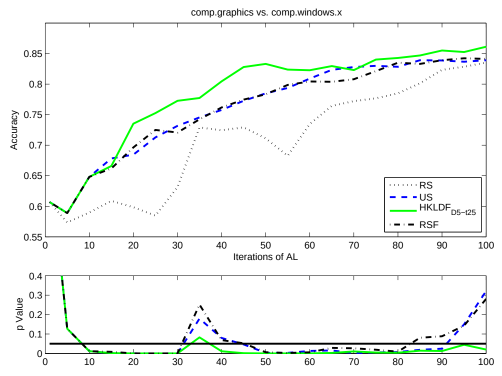
(c) BC



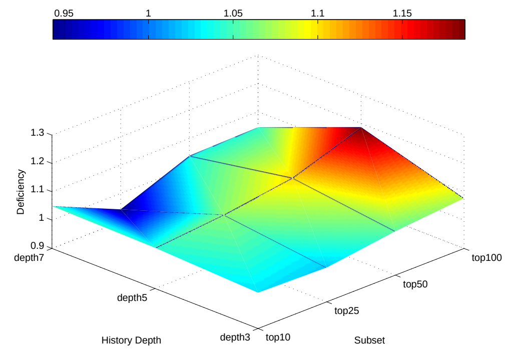
(d) BC



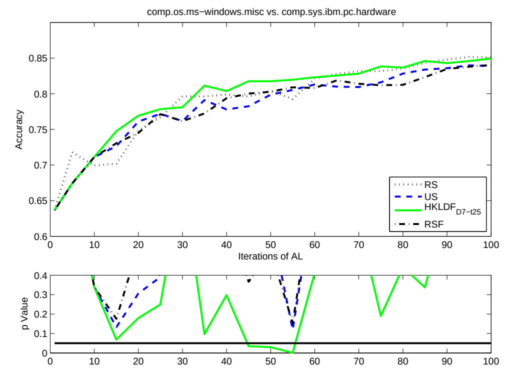
(e) GX



(f) GX



(g) WH



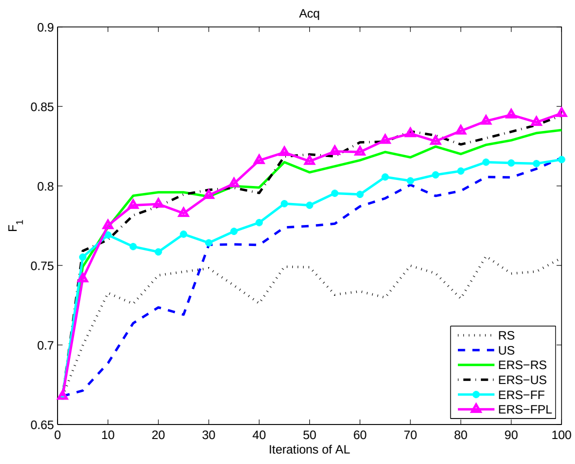
(h) WH

Figure B.12: HKLDF learning curves for optimal history depth and subset size in each of the four 20NG-4 problems. A surface plot of the deficiency values for both parameters is shown next to each learning curve

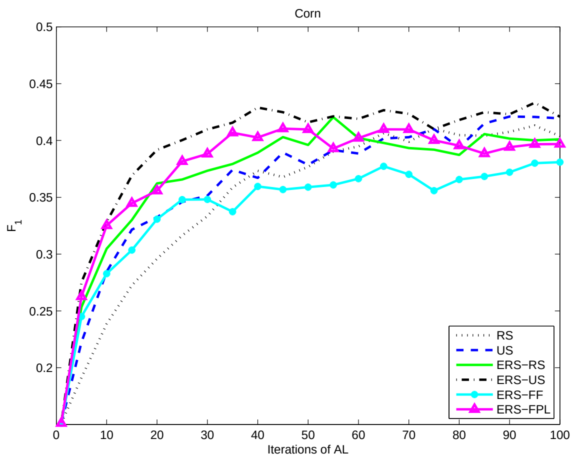
Appendix C

Supplemental Pre-Filtering Results

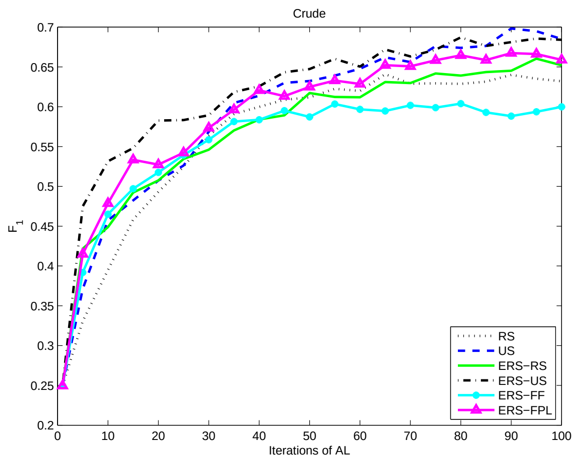
We present the results obtained from the pre-filtering experiments conducted in Chapter 7 performed on the individual categories of the R10 dataset. F_1 was used as the evaluation metric.



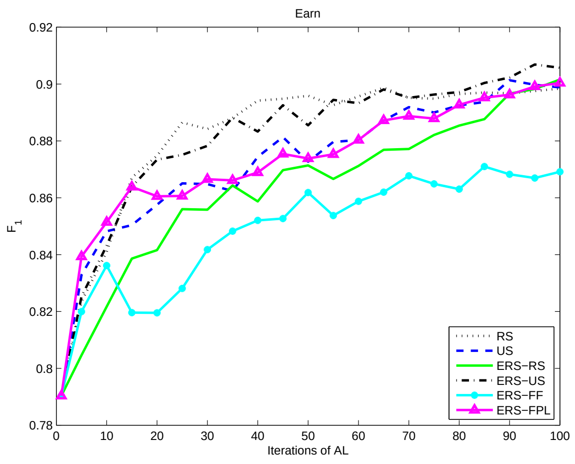
(a) Acq



(b) Corn

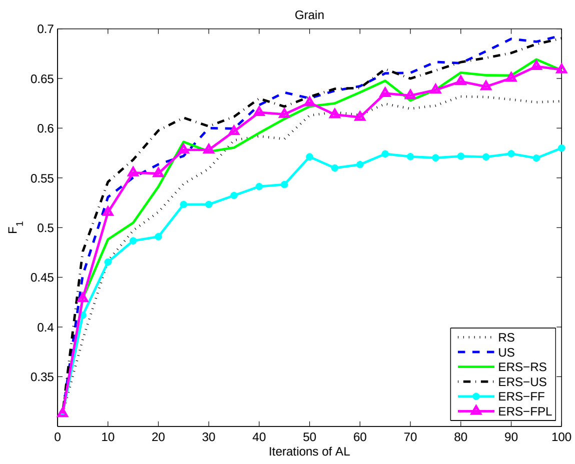


(c) Crude

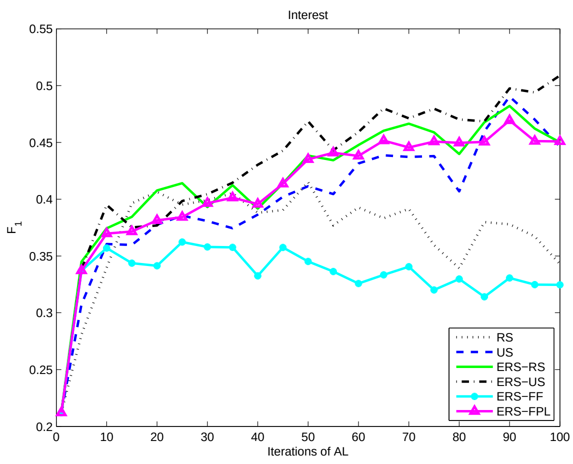


(d) Earn

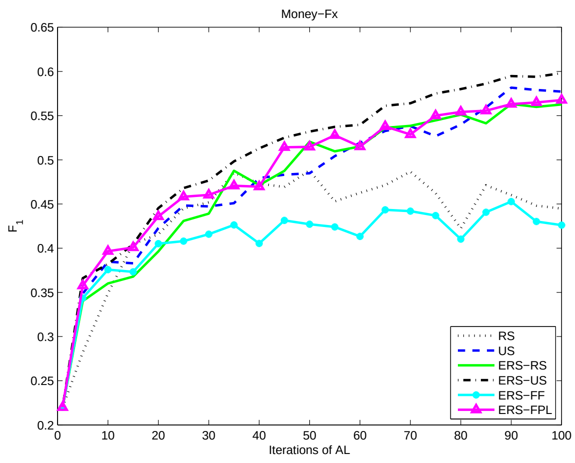
Figure C.1: Results obtained by pre-filtering error reduction sampling on the Acquisitions, Corn, Crude and Earn categories. A committee of ten multinomial naïve Bayes classifier (constructed using bagging) was used. F_1 was used as the evaluation metric



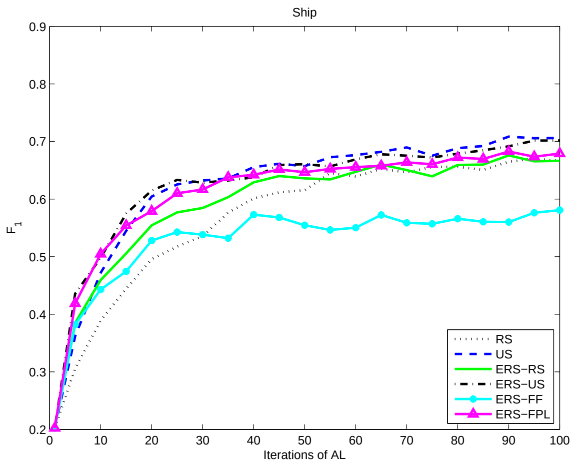
(a) Grain



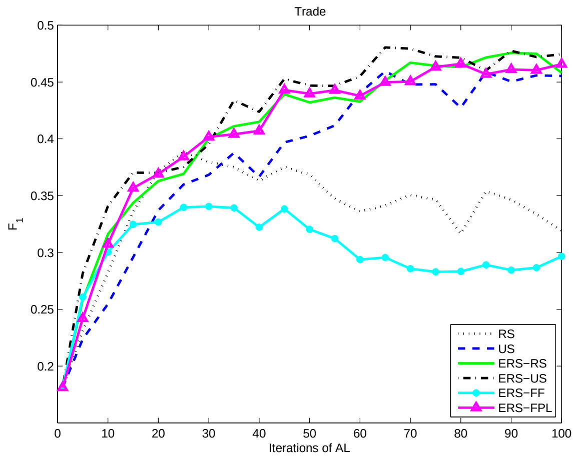
(b) Interest



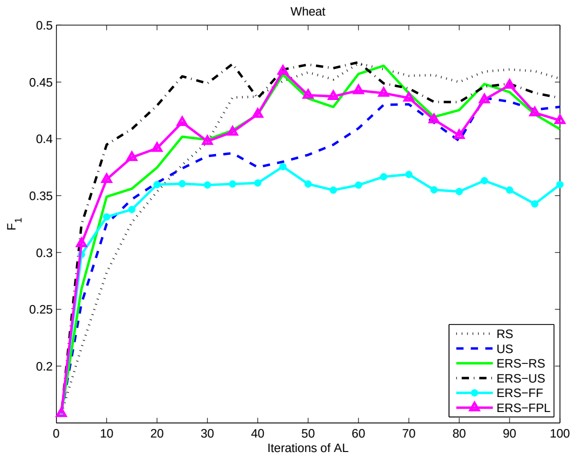
(c) Money-Fx



(d) Ship



(e) Trade



(f) Wheat

Figure C.2: Results obtained by pre-filtering error reduction sampling on the Grain, Interest, Money-Fx, Ship, Trade and Wheat categories. A committee of ten multinomial naïve Bayes classifier (constructed using bagging) was used. F_1 was used as the evaluation metric