Compact Features for Sentiment Analysis

Lisa Gaudette

Thesis Submitted to the Faculty of Graduate and Postdoctoral Studies
In partial fulfilment of the requirements for the degree of Master of Computer Science

School of Information Technology and Engineering
Ottawa-Carleton Institute for Computer Science
University of Ottawa

Abstract

This work examines a novel method of developing features to use for machine learning of sentiment analysis and related tasks. This task is frequently approached using a Bag of Words representation – one feature for each word encountered in the training data (possibly a minimum number of times) – which can easily number in the thousands or tens of thousands. This thesis develops a set of "numeric" features, by learning scores for words, dividing the range of possible scores into a number of bins, and then generating features based on counting how many words in each document have scores in each bin. This allows for effective learning of sentiment and related tasks with 25 features; in fact, performance was very often slightly better with these features.

This vast reduction in the number of features allows for training machine learning classifiers for the problem considerably faster, allowing for the processing of much larger collections of texts than previously attempted.

In addition, we carefully consider the problem of evaluating ordinal problems, as we apply our method to several ordinal datasets and found little information on effective evaluation measures for that problem.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The “Bag of Words” representation is commonly used for machine learning approaches to text classification problems. This representation involves creating a feature vector for every word seen (perhaps some minimum number of times) in the training data and learning based on the words that are present in each document, sentence, or other unit of interest. However, this approach leads to a very large, sparse feature space, as words are distributed such that there are a small set of very frequent, not very informative words, and a great deal of individually rarer words that carry most of the information in a sentence [30]. Thousands of features are required to adequately represent the documents for most text classification tasks.

This research proposes a method of condensing such sparse features into a very compact representation by learning scores for words based on their distribution in positive and negative training instances, and learning from features created based on the distribution of these scores. This approach allows for much faster processing of large data sets.

We focus primarily on the problem of sentiment analysis - identifying and categorizing opinions within text - however, we also briefly consider other text classification type problems such as subjectivity analysis and detecting agreement in political discourse.

1.1 Applications

There are many applications for sentiment analysis techniques. A number of applications are mentioned in a recent survey by Bo Pang & Lillian Lee, including automatically aggregating opinion from blogs and newspapers, improving recommendation and question answering systems by separating fact from opinion or providing perspectives from different viewpoints, allowing businesses to identify problematic features or opinions about their products, and various forms of monitoring opinions of both politicians and voters [38]. Other applications include improving "meta critic" systems to provide an overall score for the product that is more than just an average of various reviewers scores, and by providing a summary view of what "reviewers" (including non-explicit reviewers, such as bloggers) liked and disliked about the product. A great deal of labeled data is available for this problem online in the form of explicitly labeled reviews on various rating scales.

1.2 Motivations

Existing methods for sentiment analysis generally use classifiers trained using bag of words features, the output of a simple scoring method, or use at least some linguistic knowledge such as a manually constructed sentiment lexicon. Bag of words features are slow to train with large amounts of data, simple scoring methods do not generally perform well on their own, and linguistic knowledge is not portable between languages. Even within the same language, the vocabulary used to express sentiment can vary considerably between domains. We would like to develop a method which requires no linguistic resources beyond a labeled corpus and which greatly reduces the set of features required over bag of words representations in order to process large amounts of text quickly.

We accomplish this by producing features based on the distribution of automatically learned word scores appearing in a document, and using these features instead of Bag of Words for machine learning. While this is not the first attempt to combine these two paradigms, previous approaches have still required the training of a complete bag of words based classifier and combining that in some way with the output of word scoring methods.

Requiring only a labeled corpus of data allows for easily transferring between domains or even languages where labeled data is available without carefully reconstructing more advanced linguistic resources. With the wide availability of reviews available online, labeled data that is at least somewhat related to the desired task is often relatively easy to obtain. However, while we will demonstrate language portability of the method by training on a French corpus, it is beyond the scope of this thesis to consider morphologically complex languages or languages where word boundaries are not clearly defined in text, which may require more advanced linguistic preprocessing.

1.3 Approach

This thesis proposes a novel method of generating features for text classification tasks that combines word scores with machine learning. First, the system learns word scores based on how often each word appears in positive documents relative to how often it appears in negative documents. The range of possible scores is divided into some number of bins; for example, the range of 0-1 could be divided into 5 bins of [0,0.2), [0.2,0.4), [0.4,0.6), [0.6,0.8), and [0.8,1]. The system then processes each document by looking up the score for each word, and counting the number of words whose score places them in each bin, ignoring words with an unknown score. Finally, the counts in each bin are normalized by dividing by the total number of scored words in the document. This produces a feature set which is markedly smaller and far less sparse than a bag of words representation.

We will look at performance on several kinds of problems. We will look at binary and ordinal classification of professional and non-professional reviews, binary classification of snippets from movie reviews and “tweets”, and ordinal classification of sentences from online product reviews.

1.4 Evaluation

This thesis also carefully considers the best approach to evaluate the problems being studied; too often researchers merely report accuracy despite its many problems described in [40], among others, for even binary problems. We look at sentiment analysis as an ordinal classification problem in which not all errors are equal; for example, it is far worse to give a rating at the opposite end of the spectrum than it is to give a rating that is off by one point. We perform our own study into the best methods of evaluating ordinal problems, and carefully consider the evaluation measures to use for the binary problem.

Chapter 2

Background

The main focus of this work involves the problem of sentiment analysis, however, we also briefly look into the related fields of text categorization by topic, subjectivity detection, and detection of political viewpoints. In addition, we consider related work on evaluating both binary and ordinal problems.

2.1 Sentiment Analysis

Sentiment analysis is an area of Natural Language Processing (NLP) concerned with identifying and categorizing opinions in text. While much of the research has leveraged the availability of labeled data in the form of online reviews, ultimately the goal is to identify sentiment information in unlabeled, free form sources like blogs, forum postings, newspaper articles, etc. Sentiment analysis can be carried out on various levels - we can analyze the sentiment of a document, paragraph, sentence, or phrase, and we can also analyze the sentiment towards a specific person, organization, or product feature. We can work with determining sentiment from well written sources with an average length of hundreds of words, such as professional reviews or opinion pieces, and we can work with shorter, less well written texts such as online reviews of products or social networking sources like Twitter.

On the surface, sentiment analysis appears similar to text categorization by topic, but it is harder than text categorization for many reasons, as discussed in Pang & Lee's 2008 survey, [38]. First and foremost, with text categorization, it is usually much easier to extract relevant key words, while sentiment can be expressed in many ways without using any words that individually convey sentiment. In topic classification there are undoubtedly red herrings, such as the use of analogies and metaphor, but if a word associated with a given domain is mentioned frequently, it is usually related (although not necessarily the most relevant)[38]. However, in sentiment analysis there are many examples of "thwarted expectations" [1] and comparison to an entity with opposing sentiment [2] such that a positive review can easily have many negative words and vice versa.

Broadly speaking, there are 3 major approaches to sentiment analysis systems: approaches based on manually constructed lexicons, which may include developing complicated rules, approaches based on automatically generated lexicons and rule bases, and machine learning approaches based on Bag of Words feature sets. There is also work which has combined two or more of these approaches together.

2.1.1 Lexicon Based Approaches

Many approaches to sentiment analysis consist of manually or automatically classifying or scoring words or phrases and scoring documents or sentences based on those scores. Here we divide these into approaches using fully manually constructed lexicons and/or rule bases, approaches using automatically generated lexicons, and approaches which go beyond assigning words as simply positive or negative and give them a score.

Two frequently used tools for lexicon based approaches are the General Inquirer

[1] "I was expecting this movie to be great, but it was terrible"
[2] "I loved the first movie, but this sequel is terrible"

(GI) and WordNet. GI, created in 1966 [44], is a “a computer-assisted approach for content analyses of textual data”, which includes both software for analyzing text as well as categorized word listings. Most frequently, it is the word listings which are used in sentiment analysis, as a starting point for a manual or semi-automatic lexicon. WordNet [11] is a lexical database of English which groups nouns, verbs, adjectives and adverbs into “synsets” which reflect groups of synonyms to express a concept. It was created by George A. Miller and colleagues at Princeton. WordNet does not explicitly encode sentiment data, but it does encode links between words, allowing researchers to expand small seed lists of words into a larger list of sentiment words.

Manually Constructed Lexicons and rules

This section looks at approaches that use a manually constructed lexicon, either from an existing source or that is hand constructed by the authors. Broadly speaking, these approaches then try to classify documents by comparing the number of positive and negative features as determined by a pre-determined list of words.

One use of purely manually constructed methods is by Nasukawa and Yi [35]. This work uses a manually constructed lexicon of part of speech tagged sentiment words, with some information about what they transfer the sentiment to, e.g. “gVB admire obj” indicates that admire is a verb that confers favorability on its object; shallow parsing is used to allow the use of such rules. The system assigns scores of +1/-1 to items and decides sentiment based on sum of scores. This work examines the problem of extracting snippets relevant to particular topics; it strives for high precision at the expense of recall.

While these approaches are attractive in that they can easily be applied across many domains, they suffer from several issues. First and foremost, manual lexicons can be very limited in size (e.g. 3000 words) which can prevent these sorts of approaches from being able to score all documents.

In addition, even if there was a perfect sentiment lexicon in existence, a great deal of sentiment information is carried in words that are context specific - either to the domain or the surrounding words. “Small” is not in and of itself is a sentiment term, but it can be a fairly significant positive or negative feature given the context - both in terms of the domain and the context of the sentence. Consider “The laptop is small”, “The keyboard is small”, “The trunk is small”, and “The room is small”. The first sentence would generally be considered as positive, given that it is desirable for laptops and other portable electronics to be small. However, it is not generally desirable to have a small keyboard on a laptop, small trunk on a car, or small hotel room. Words like small can be very important indicators of sentiment but what sentiment they convey depends a great deal on context. A general purpose lexicon can not capture context at all, while one tailored to a specific domain can to some degree.

Finally, people often express sentiment by comparing their opinions of a product to their expectations or experiences with another similar product, and this can result in a great deal of positive words in a negative review and vice versa.

Automatically or Semi-Automatically Constructed Lexicons

This section looks at approaches that automatically or semi-automatically generate lists of positive and negative words. These approaches may start with a manual lexicon and extend it, or, in other cases, with a small set of seed words. This allows the easy creation of larger lexicons, however, they tend to be noisier than a manual lexicon.

One of the first works focused on automatically identifying the polarity of sentiment terms is by Hatzivassiloglou and McKeown [20], which looks at clustering adjectives based on the connections between them in a large text corpus. They hypothesize that adjectives connected by *and* usually have the same polarity, while adjectives connected by *but* have opposing polarity. They construct a log-linear regression model to weight the connections between words, represent the links as a graph, and partition the graph into clusters of positive and negative words. They were able to achieve greater than 90% accuracy on their test set of manually labeled, unambiguous adjectives, and found that the errors were a result of misplacing adjectives in clusters rather than of misidentifying clusters.

The work by Godbole, Srinivasaiah and Skiena [17] focuses on identifying sentiment towards various entities in blogs and news media, in order to track the sentiment toward the entity over a wide range of sources. Their approach is based on using WordNet to generate a sentiment lexicon by extending various domain specific lists of seed terms. Synonyms are given the same sentiment, and antonyms the opposite, while the significance decreases as the length of the path from the seed word increases. The algorithm also takes into account paths that "flip" from positive to negative sentiment. It only looks at the most common (first) sense and filters out words whose scores are in the middle of the distribution. The score for an entity is computed by looking at the number of positive sentiment references relative to the total number of sentiment references, while also making use of negations and intensifiers.

A "holistic lexicon based approach" is used in [9] to determine sentiment at the feature level. They build on previous work to extend a lexicon through WordNet, and devise many rules to transfer sentiment from one term to another. They also manually annotate many idioms. The focus is on individual features of products from a previously introduced dataset. The system assigns words a score of +1/-1, and generates scores on features based on the score of a word divided by its distance from the feature. The system includes manually constructed rules for negation and clauses involving "but" and related expressions. It learns words that are context dependent based on being in sentences with known positive/negative words – "This camera takes great pictures and has a long battery life" confers a positive sentiment to "long" when talking about battery life, while "This camera takes great pictures, but has a short battery life." confers a negative sentiment to "short" in the context of battery life.

In [28], rules and a belief base are used to tackle the problem of extracting snippets from reviews that disagree with the overall sentiment of the review. The system automatically learns a set of rules to identify unexpected clauses using sequential pattern mining algorithms.

Automatically Generated Word Scores

This section differs from the previous two in that we look at approaches which maintain some form of score reflecting how positive or how negative a word is, rather than merely dividing words into positive and negative. Note that some of the work in the previous section generates scores which are thresholded to produce a binary outcome, but which could have been used for work similar to that in this section; the division here is between how the authors chose to use the method, not how it could be used.

A well known early work featuring automatically generated word scores to perform unsupervised sentiment analysis is Peter Turney's PMI-IR approach [46]. In this work, the focus is on two word phrases following certain patterns involving at least one adjective or adverb. Turney uses a search engine to estimate the PMI of each phrase with the words "excellent" and "poor". These two values are then subtracted from each other to find the Semantic Orientation of the phrase; the score for a document is obtained by averaging the scores for terms, and documents with a positive average score are deemed positive and vice versa. This approach achieves between 65% and 84% accuracy over various domains of user reviews.

$$SO(Phrase) = \log_2 \left(\frac{hits(phraseNEAR\text{``}excellent\text{''})hits(\text{``}poor\text{''})}{hits(phraseNEAR\text{``}poor\text{''})hits(\text{``}excellent\text{''})} \right) \tag{2.1.1}$$

Several other authors extend Turney's work, including [14]. This work extends Turney's work with more seed words, and an additional assumption that words with opposing sentiment tend not to co-occur at the sentence level. They use a set of unlabeled in-domain data rather than the Web to compute PMI.

Various word scoring methods which produce word scores from -1 to 1 are used in [7] to classify documents based on sign of the sum of the scores. They use their own dataset constructed from reviews of electronics from Amazon.com and C|net, and test their method in two ways. The first test retains the skews of the original corpus by training on 6 categories and testing on a 7th, while the second uses a balanced selection from the 4 largest categories. The results are compared to machine learning methods similar to [39]. Their best results on the first test are obtained using trigram features with their word scoring method, however, the difference between this approach and the SVM approach using bigrams is small. The best results on Test 2 are obtained by SVM with bigrams, however many of the scoring methods are close.

The problem of identifying sentiments of specific holders towards specific topics at the sentence level is explored in [26]. A small set of words is hand labeled and then WordNet is used to compute sentiment scores for a variety of words, in the range of -1 to 1. The system computes sentiment of a region of a sentence meant to correspond with the holder's sentiment (the window from "Holder" to end of sentence performed best) by averages or counts of words.

An in-domain list of words is automatically mined from the web in [19] to create unsupervised word scores. This work starts with a small seed list with 7 positive and 7 negative words, and searches the web for documents relevant to the domain containing one of the positive (negative) words and none of the negative (positive) words, and keeps adjectives which are highly correlated with the seed words in the retrieved documents. This technique performs better on the positive class than the negative class.

2.1.2 Machine Learning Approaches

Machine Learning approaches generally create a feature set in the form of a Bag of Words (BOW) - that is, the features used as input to the algorithm represent the presence or frequency of various words seen in the training data.

A major early work on using machine learning techniques for sentiment analysis is by Pang and Lee [39]. This paper introduces a publicly available dataset of binary movie reviews. They compare performance of Naive Bayes, Support Vector Machines, and Maximum Entropy classifiers on the task of performing document level binary sentiment analysis with various bag of words feature sets. They find that using an SVM classifier with unigram presence information for features works best; this performs much better than unigram frequency, and trying to add bigrams, part of speech information, or further restrict the unigrams did not help. Unigrams were tagged with negation information by adding NOT_ to all words between certain negation words and the next punctuation mark. They find that sentiment classification is more difficult than text categorization, and hypothesize that this is partly because of "thwarted expectations", where the author contrasts their expectations of the movie with their actual opinion. This work is extended in [36], which looks at combining document level sentiment analysis with sentence level subjectivity analysis in order to score the sentiment of only the subjective parts of the document. They are able to maintain or slightly improve performance while using approximately 60% of the material from the original reviews. Further work in [37], examines the ordinal case, and introduces the concept of metric labeling using positive sentence percentage as a similarity metric in combination with an SVM classifier, which was able to produce small but significant gains in performance.

The bag of words approach is modified in [32] to weight the features by the difference in TFIDF scores in positive and negative documents, such that words which appear more frequently in positive documents have a positive value and words which appear more frequently in negative documents have a negative value, and a word evenly distributed between positive and negative documents would have a value of 0. This weighting improves performance over simple bag of words features.

2.1.3 Combination Approaches

There is also some work which attempts to combine lexicon based approaches with machine learning based approaches. While given sufficient in-domain data, machine learning approaches are generally superior, lexicon approaches have the advantage that they are not necessarily domain specific.

Work by Mullen and Collier explores the use of a variety of numeric features as input to an SVM classifier[33], while also using bag of words features. These features are constructed using “Turney values” as described in section 2.1.1, “Osgood values”, which are computed based on WordNet path length from seed words representing different poles of different aspects of sentiment - potency (strong or weak), activity (active or passive), and evaluative (good or bad). The best results on the movie review dataset from [39] were obtained by combining the Turney values and lemmatized unigrams together with a hybrid SVM. The Osgood values were not found to be particularly useful.

The work in [25] considers simple word counting, word counting with valence shifters (negations, intensifiers, and diminishers), and a hybrid machine learning approach. For the word counting, they use a combination of data sources, including GI and other word and synonym lists, which they ultimately extend using SO-PMI as in [46]. Positive words are given a score of 2 if alone, 1 if preceded by a diminisher, and 3 if preceded by an intensifier, and then negated if preceded by a negation, while negative words have scores of -2 / -1 / -3. For the Machine Learning approach, words are tagged with the information that they were found with (any) negation, intensifier, or diminisher. They find that using the valence shifters, particularly the negations, improves performance over not using them by a small amount. Their final combined system of a meta classifier combining the results of the word counting method and the SVM method, using valence shifters, obtains a small but significant improvement over the basic SVM system.

The General Inquirer lexicon is used in [22] along with a classifier which attempts to learn when the polarity of a word is shifted due to context. A linear discriminative classifier is run on each word to determine if it is shifted, and if it is, the score is negated. Then all the scores in the document are added together. This shift detection improves performance over a basic lexicon method. Finally, this shift detection system is combined with an SVM bag of words model by weighting the output of a score created by SVM with their score created with the lexicon method. The authors claim their method performs better than others, however, their method cannot score all sentences while the other methods they compare to may have been able to score the approximately 10% of sentences that they discarded.

The problem of limited availability of in-domain labeled data is considered by [1]. This approach first creates a general lexicon based on WordNet using the adjectives from [20] as seed terms, and combines the results with a corpus based approach trained on a small training set of in-domain data with weighted voting. The vote for each classifier is based on its precision; the corpus based approach is high precision on positive examples, while the lexicon based approach is high precision on negative examples. This combination approach was able to substantially increase performance in four domains with 800-1200 labeled documents.

2.1.4 Domain Adaptation

Sentiment analysis can be highly domain dependent. While there are many words and expressions that convey a universal sentiment regardless of the domain, many others are specific to a domain or even a particular context within a domain. Domain adaptation is the process of adapting a system from domains with plentiful data to domains with little available labeled data.

One of the early works considering this problem is by Aue, [2], who uses a collection of 4 datasets to consider different potential methods of domain adaptation. These datasets vary considerably in terms of the number of documents and the length of those documents. The methods used include simply training on 3 datasets and testing on the 4th, limiting the feature set of that basic approach using only features that appear in the target domain, training a classifier for each dataset independently and combining the outputs with a meta-classifier trained using a small amount of in-domain labeled data, and finally, making use of unlabeled data in the target domain through the Expectation Maximization (EM) algorithm, starting with a small amount of in-domain labeled data. The best results are obtained by the EM algorithm, using 200 in-domain labeled instances.

Structural correspondence learning (SCL) is a different approach used by Blitzer [4]. SCL involves taking labeled data from a source domain, and unlabeled data from both a source and target domain. The algorithm is looking for features which have high correlations with a set of "pivot features", in order to attempt to find features in the target domain that behave similarly to known features from the source domain. This work uses a combination of unigram and bigram features, and for classification uses linear predictors trained to minimize the Huber loss with stochastic gradient descent. This work also introduces a publicly available dataset of reviews from Amazon.com from 4 different categories which is used in this thesis.

The dataset from [4] is used by Li and Zong [29] to perform document level domain adaptation. First, unigram and bigram features are selected using Bi-Normal Separation. Next, base classifiers for each of the 4 domains are trained using 70% of the data, then these are combined using a meta classifier for each domain trained using 20% of the data, and results are tested on 10%.

2.2 Related Problems

2.2.1 Text Categorization

Text categorization is the general problem of classifying text into various subjective categories. Text categorization can be single-label, where a document needs to be assigned to exactly one category, or multi-label, where a document can be assigned to multiple categories. Frequently, the desired categories are the topic of the document, such as *Sports* or *Politics*, however other distinctions can be considered. For instance, we could also speak of categorizing text in terms of spam or not spam, or by genre, as discussed in a survey of text categorization by Sebastiani [43]. However, there are important differences between topic categorization and other text categorization problems.

The machine learning techniques of Support Vector Machines (SVM) and boosting have been found to be particularly effective for text categorization [43]. The suitability of SVMs for text categorization is discussed in detail by Joachims [24], who develops a theoretical model of the text categorization task based on various properties of the problem, such as the large, sparse feature space, large number of overlapping terms, the distribution of words, and the high level of redundancy within each document. This model enables the development of theoretical results for bounds on the generalization error. Joachmis also determines conditions which make a text categorization problem easier or harder; other parameters being constant, a problem is easier if the discriminative features are more frequent within the corpus, the vocabulary of the two classes is more distinct, and there is a higher level of redundancy within a document.

2.2.2 Subjectivity Analysis

Subjectivity analysis is the process of identifying opinions vs. facts, regardless of the strength or polarity of the opinion.

The approach by Riloff and Wiebe [42] focuses on "extraction patterns" to identify patterns that occur in subjective texts. They use a bootstrapping process which begins with high precision, low recall classifiers created based on subjectivity clues from previous work. The subjective classifier classifies sentences as subjective if it contains at least 2 strong subjectivity clues, while the objective classifier classifies a sentence as objective if it contains at most one weak subjectivity clue in the current, previous, and next sentence combined. Their system is also given a listing of syntactic templates, such as <*subj*> *active-verb dobj*, which are then used to find more specific (but still flexible) extraction patterns, such as <*subj*>*dealt blow*. Extraction patterns that have a high probability of appearing in subjective sentences (and occur frequently enough) are then added to the initial set of subjectivity clues in order to classify more sentences.

This work is extended in [41], which looks at a "subsumption hierarchy" of features to identify complex features that perform better than simple ones - essentially, to identify the complex features like extraction patterns hat are worth keeping over simpler patterns or unigrams. For example, *line* subsumes *the line* because every instance of *the line* includes *line*. The information gain of the two features are compared, and the more complex feature is kept only if it sufficiently improves performance. This work also shows examples of instances where keeping words that are often filtered out as stop words is useful - *the line* is much more subjective than simply *line*. Their best results were obtained using subsumption, however, this difference was not very large.

Subjectivity analysis can also be used a preprocessing step prior to attempting sentiment analysis, as in in [36], where the authors found that by removing objective sentences they were able to maintain or slightly improve on the performance of using the whole reviews. Subjective sentences were identified by combining output of a bag of words SVM on individual sentences with the heuristic that nearby sentences should tend to have the same scores to arrive at a problem of calculating the minimum cost cut of a graph.

2.2.3 Analyzing political discourse

There are various attempts to model different types of political discourse, from analyzing formal debates to identifying political leanings from online postings.

Congressional speeches are combined with voting records as a source of data in order to determine if the speech was in support or opposition of a policy in work by Thomas, Pang, and Lee [45]. In addition to using SVM classifiers on speech segments, as in previous sentiment analysis work, the authors look at the relationships between speakers and model agreement between speakers in the same debate.

A more informal domain of political discourse is examined by Mullen and Malouf, that of online political discussions [34]. Their work attempts to classify users of online political forums into categories of “right” or “left”, using the self selected labels provided by the users. They found even this simple problem to be quite difficult, even after discarding harder to classify users labeled as centrist, independent, green, and libertarian. They use a Naive Bayes classifier, and found that the crude spelling correction they attempted was not helpful, that they were able to obtain slightly better performance when focusing on the most frequent posters and not including the people who posted more rarely and that adding a simple rule that people tend to quote people whom they disagree with improved performance.

Jiang and Argamon work on the problem of first classifying blogs into political or non political, and then classifying the political blogs as “liberal” or “conservative”, using an SVM BOW classifier augmented with link information [23]. They also used various feature selection methods to reduce the dimensionality of their original dataset considerably, from nearly 400,000 features to under 7,000 without hurting performance; however, it should be noted that their original set simply used all words and did not filter out rare words (e.g. those appearing less than 3 times in the training data) as is often the case. They found that using out-link information in addition to BOW was helpful, and also found that adjusting the "cost factor" was beneficial to account for the imbalance in their dataset between liberal and conservative blogs.

2.3 Evaluation

In recent years, there has been much discussion on the flaws of accuracy as a metric for comparing performance on machine learning tasks (see [40], among others). In addition to the flaws inherent in using accuracy for binary problems, in the ordinal case accuracy tells us nothing about the severity of the error and in many applications this is important. We thus feel that we need to carefully consider the evaluation methods to be used in this research, for both binary and ordinal problems.

2.3.1 Binary Metrics

There has been extensive examination of performance measures for the binary problem. Accuracy is the simplest measure, but one measure that has been used increasingly frequently is the Area Under the ROC Curve (AUC). An ROC curve is the plot of the True Positive vs. False Positive rate, and can be constructed using all possible thresholds of the classifier output to show the possible trade offs in error. AUC is the area under this curve, and ranges from 0 to 1.

A comparison of a variety of evaluation metrics for binary classification is performed in [6] by looking at performance over a wide range of algorithms and datasets, and looking at how the measures relate through visualization methods and their correlation with each other. They found that root mean squared error (RMS) was a good all purpose measure that was well correlated with other metrics; they also propose a new metric called SAR that averages RMS, AUC, and accuracy, which they found to represent different categories of metrics. SAR provided very modest improvement over RMS.

Another similar comparison, this time including some multiclass datasets is performed in [12]. They compare the correlations of a variety of metrics on a variety of problems, and then they test the metrics for sensitivity to different kinds of noise on an artificial binary problem. Their intent is to see if results across some metrics are comparable; however they find that with the exception of the various methods for computing multi-class AUC they are not, and that even seemingly similar metrics are measuring different things.

We chose to present results for AUC and Accuracy for the binary problem, and focus on AUC, as we feel that the ability to rank the output is an important facet for many applications of this problem. We include Accuracy as it is an intuitive measure and is most frequently used in related work.

2.3.2 Ordinal Metrics

Most papers on ordinal classification that we have found simply use accuracy as an error measure, without considering whether or not it is an appropriate measure, such as [37, 18, 31, 13]. A measure called the normalized distance performance measure is used in [51] for work with image retrieval. An AUC type measure for the ordinal case is introduced in [48], representing a volume under a surface. The authors of [37] conducted a small test to determine human performance on the task of which of two reviews is more positive, and described the results of this in terms of "rating difference", and were, for instance, correct for 83% of reviews with a rating difference of 1 star on a 4 or 5 star scale, however they simply use accuracy for describing the performance of their classifiers. We feel that this problem has not been adequately studied, and so we will undertake our own examination of this problem.

2.3.3 Establishing Significance of Results

Another important component of evaluation is determining if the results represent a significant difference.

Cross validation is a technique whereby the data is split into some number of folds - commonly 3, 5, or 10 - with the data being trained on all but one fold and the final fold being used as the test set. This is repeated with each fold appearing as the test set once, and then the results are combined over all folds. We can then use the result on each fold to calculate statistics like the standard deviation and to determine significance between two experiments through methods like a t-test. However, cross validation results can be greatly affected by the initial data split; for this reason, [5] suggests repeating the cross validation multiple times, and performing 10x10 fold cross validation.

The Student's t-Test is a very commonly used measure of statistical significance, however it is not without its problems, as discussed by Drummond in [10]. For this reason, we will supplement use of t-tests by also providing confidence intervals for our results.

2.4 Tools & Datasets

2.4.1 WEKA

WEKA [50] is a machine learning framework written in Java. Weka implements many machine learning algorithms, as well as modules for preprocessing and evaluation. It provides graphical interfaces as well as an API. We make extensive use of the WEKA API and use its implementations of machine learning algorithms.

2.4.2 Datasets

This thesis makes use of a number of datasets that have been collected by other researchers and made publicly available. Table 2.1 presents some quantitative information on the datasets, and their distribution for the binary problem. Table 2.2 presents the distributions for the ordinal divisions on the datasets which have them, while qualitative features are discussed in the following paragraphs.

Amazon Reviews Online user reviews posted to Amazon.com, labeled by the user from 1 to 5 stars. This data was processed to balance it for the binary problem, and only contains examples of classes 1, 2, 4, and 5. These reviews vary widely in length and writing quality. This dataset was introduced in [4].

Large Amazon Reviews A large set from which the previous reviews were extracted [4]. The entire dataset includes many categories, some with only a few hundred entries; we use the three which have over 100,000 reviews. This set is not manually balanced, however, it still only contains examples of classes 1, 2, 4, and 5. Note that there are nearly 1 million reviews available for the Books dataset; we found that dealing with data of this size is not trivial.

Ordinal Movie Reviews This dataset consists of subjective extracts of movie reviews from four separate authors. These reviews are professionally written, and presumably the style does not vary as much between the same author as between different authors. [37]

Binary Movie Reviews This dataset was originally created in 2002, and has been widely used since then. It consists of movie reviews taken from IMDB.com, and is known by the authors as the Cornell Polarity Dataset 2.0 [39]. This dataset contains the longest documents on average of the sentiment analysis datasets.

Table 2.1: Overview of Datasets; Related datasets are grouped together

Name	#	Distribution		Number Of Words			
		Neg	Pos	Min	Max	Median	Avg
Amazon Reviews							
Electronics	2000	50.0%	50.0%	5	1065	79.5	112.3
DVD	2000	50.0%	50.0%	7	1510	113	172.0
Books	2000	50.0%	50.0%	1	3566	114	177.1
Housewares	2000	50.0%	50.0%	8	1002	70	94.1
Large Amazon							
Music	172180	8.0%	92.0%	1	3362	97	144.1
Books	615415	13.2%	86.8%	1	5878	108	164.7
DVD	122438	13.5%	86.5%	1	4303	106	171.4
Ordinal Movie Reviews							
Rhodes	1770	40.5%	59.5%	66	1136	347	368.9
Renshaw	902	45.5%	54.5%	127	764	462	461.1
Berardinelli	1307	25.2%	74.8%	146	1201	424	440.7
Schwartz	1027	59.5%	40.5%	6	2366	252	294.9
Binary Movie Reviews	2000	50.0%	50.0%	16	2366	606	647.6
Snippets	10662	50.0%	50.0%	1	51	18	18.5
Product Reviews	1235	39.9%	60.1%	1	100	15	16.9
Twitter							
Train	40000	50.0%	50.0%	1	37	13	14.2
Test	216	50.0%	50.0%	2	30	12	13.7
Congress							
Speaker Train	1174	50.5%	49.5%	4	11690	442	693.1
Speaker Test	410	44.1%	55.9%	4	18824	430	721.4
Segment Train	2740	47.6%	52.4%	1	6065	168.5	296.5
Segment Test	860	41.6%	58.4%	1	6880	241.5	345.4
Subjective Sentences	10000	50.0%	50.0%	5	106	20	21.2
French Sentences	702	53.3%	46.7%	5	74	24	26.5

Table 2.2: Distribution of Ordinal Datasets

Dataset	Size	Classes	Distribution					
Amazon Reviews			1	2	4	5		
Electronics	2000	4 (5)	33.3%	16.7%	16.0%	34.0%		
DVD	2000	4 (5)	26.5%	23.5%	14.3%	35.7%		
Books	2000	4 (5)	26.9%	23.1%	13.5%	36.6%		
Housewares	2000	4 (5)	34.4%	15.7%	12.3%	37.7%		
Large Amazon Reviews			1	2	4	5		
Music	172180	4 (5)	4.3%	3.7%	21.4%	70.6%		
Books	615415	4 (5)	7.2%	6.0%	22.6%	64.2%		
DVD	122438	4 (5)	7.6%	6.0%	25.5%	61.0%		
Ordinal Movie Reviews			0	1	2	3		
Rhodes	1770	4	10.8%	29.7%	43.3%	16.2%		
Renshaw	902	4	12.7%	32.7%	37.0%	17.5%		
Berardinelli	1307	4	10.6%	22.3%	45.6%	21.5%		
Schwartz	1027	4	16.7%	42.8%	29.4%	11.1%		
Product Reviews			-3	-2	-1	1	2	3
All	1235	6	9.3%	22.2%	8.4%	8.6%	34.6%	16.9%

Snippets Snippets from movie reviews mined from RottenTomatoes.com, a website which aggregates movie reviews, labeling them as “fresh” and “rotten”. Each review has a snippet of usually a sentence or two associated with it; these were gathered and assigned the label of their source review [37].

Product Reviews Reviews from Amazon.com manually labeled with opinions for particular features within a review. The data used here represented reviews for 5 different products [21].

Twitter Entries from Twitter, a social networking website which allows users to post “tweets” which are limited to 140 characters. The training data is manually labeled based on the presence of :) (positive) and :((negative), while the test data is hand labeled [16]. This dataset contains the shortest documents, and there is also a tendency for them to be poorly written with spelling errors and abbreviations.

Congress This dataset consists of transcripts of US Congressional debates, matched with the vote of the speaker on the issue being debated [45]. These documents are divided into a separate training and test set which use different debates. "Speaker" groups all segments by the same speaker in the same debate together; material is labeled through using the vote of the speaker on the issue. These speeches range from single words to the longest documents in all of the datasets we look at.

Subjective Sentences Sentences taken from movie reviews (subjective) and movie plot summaries (objective) [36].

French Sentences Sentences from the Belgian French newspaper *Le Soir*, scored by several raters from -3 to 3 in terms of being unpleasant, neutral, or pleasant [3].

2.5 Situating this Research Among Previous Work

We see two major groups of approaches to sentiment analysis: those approaches that use scores for words or phrases to score larger textual units, and those approaches that use machine learning algorithms based on bag of words type feature sets for machine learning algorithms. We have found that in general, bag of words classifiers appear to perform better than purely lexical methods when sufficient training data is available. In particular, word scoring based methods report accuracies below 80% on the Pang & Lee movie review dataset ([28, 14, 19], while BOW based methods easily exceed 80%, with the best results exceeding 90% accuracy. In addition, there is work in combining the results of a BOW classifier with the output of a lexical system, which is often able to produce results that are better than either classifier independently.

This method combines machine learning methods with automated scoring methods, but in a different way from previous combinations. Previous combinations required training a complete BOW based classifier, and combining that with some word score based features. Our work uses scoring methods to help produce novel inputs to a machine learning classifier in order to condense the feature set, which has the advantage of being much faster to learn.

We examine our method in comparison to BOW, not because simple BOW is the best system, but because it forms a key component to the best systems, and could easily be replaced by our method in such combination systems.

We found the work by Martineau and Finin, [32] weighting the inputs to an SVM BOW classifier by TFIDF scores, to be most related to our method. However, while both methods modify the features based on word scores, they are still very different - the TFIDF weighting doesn't decrease the size of the feature set, while our method condenses the input to the classifier into a much more compact representation that can be learned quickly by a machine learning algorithm.

Chapter 3

Reducing the Feature Space through Word Scores

The Bag of Words (BOW) approach to features for machine learning of sentiment analysis and other similar tasks involves a very large, sparse set of features. This feature set is slow to train, and the sparsity may make it more difficult to learn from rarer but still important words. A different approach to learning sentiment involves automatically learning word scores, and then determining scores for documents based on the sum or average of those scores. This scoring approach can be very quick to learn, but is not generally as effective at learning sentiment as machine learning approaches. This chapter describes a method of using word scores to develop a "numeric" feature set for machine learning based on the distribution of word scores in a document across a number of 'bins'. This feature set is much smaller than a BOW feature set, while providing more information than a single overall score produced by a word scoring method.

3.1 Generating Numeric Features from Scored Words

The approach to reducing the feature space involves 3 basic steps, illustrated in Figure 3.1: computing a score for each word representing the strength and polarity of its sentiment, counting the number of words in each document in each "bin" of scores, and learning from these counts using a machine learning algorithm. This greatly condenses the feature set, making it much less sparse.

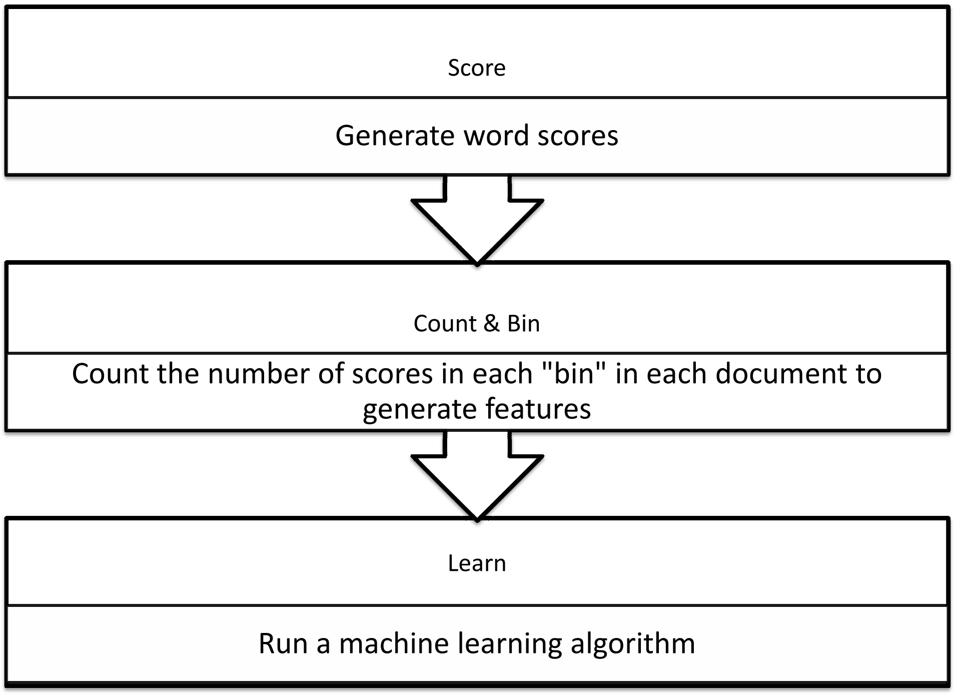

Score
Generate word scores
Count & Bin
Count the number of scores in each "bin" in each document to generate features
Learn
Run a machine learning algorithm

Figure 3.1: The basic process

The documents used to compute the word scores may or may not be the same as the documents used to train the machine learning algorithm. In addition, these docu-

Table 3.1: Binning the words

Bin range	Words in range	Feature
$[0.0, 0.2)$	1	0.01
$[0.2, 0.4)$	5	0.05
$[0.4, 0.6)$	70	0.70
$[0.6, 0.8)$	20	0.20
$[0.8, 1.0]$	4	0.04
Total:	100	1

ments do not have to be scored on the same scale - the scoring methods presented here only require that the word scores can be split into a binary problem. This allows us to combine multiple sources of data which may have been labeled differently from our desired output; this can facilitate the acquisition of larger datasets. While a number of papers feature attempts to score words (or use a manually created lexicon) and then classify documents by summing or averaging the scores of words in a document, to the best of our knowledge, using the scores in this way as features for a machine learning algorithm is novel.

Once we have learned scores for words, we generate features by going through each document and counting the number of words whose scores place them in each bin. The counts in each bin are normalized by the number of scored words in each document, so that each feature represents the proportion of words with scores in that range. This is demonstrated in Table 3.1.

3.1.1 Simple Scoring Methods

We explore three simple ways of computing a score for words based on counting their frequencies in positive and negative documents. In all of the metrics below, a word which appears only once or twice but in one kind of document by chance could receive a very strong score; for this reason, we filter out words which do not occur some minimum number of times in some minimum number of documents. The following notation will be used to describe the scoring metrics:

P, N the total number of tokens (words) in all positive (negative) documents

wP, wN the number of occurrences of word w in positive (negative) documents

Other methods of computing scores, including unsupervised methods, are also possible, but one advantage of the three presented here is that they are simple to compute.

Normalized Positive Percentage (NPP) This scoring metric amounts to computing the percentage of the time that a word appears in positive documents, normalized to account for differences in the number of words in positive and negative documents. This produces a value between 0 and 1.

$$npp = \frac{\frac{wP}{P}}{\frac{wP}{P} + \frac{wN}{N}} \tag{3.1.1}$$

This method was developed independently but was later learned to be a special case of the method introduced by [27], as part of an approach to determine sentiment through average word scores of a document.

Normalized Difference (ND) This scoring method is similar to NPP, but instead represents the difference between the percentage of positive and negative words. This method is used in [7], with good results for the binary problem as a classifier based on the sum of scores in a document. This produces a value between -1 and 1.

$$nd = \frac{\frac{wP}{P} - \frac{wN}{N}}{\frac{wP}{P} + \frac{wN}{N}} \tag{3.1.2}$$

Precision This method was inspired by its use in [49] for extracting potential subjective words. It represents the proportion of occurrences of the word which were in positive documents, but does not account for differences in the number of words in the sets of positive and negative documents. This produces a value between 0 and 1.

$$precision = \frac{wP}{wP + wN} \tag{3.1.3}$$

3.1.2 Preprocessing

One other detail is how the data is preprocessed and tokenized. All data is preprocessed to remove non-word characters, excess whitespace, and to convert all words to lower case.

3.1.3 Example of Generating Numeric Features

Generating the word scores

Figure 3.2: Example of scoring the word “good”

$$\begin{aligned} goodP &= 15 \\ P &= 3000 \\ goodN &= 4 \\ N &= 2500 \end{aligned}$$

$$\begin{aligned} npp &= \frac{\frac{15}{3000}}{\frac{15}{3000} + \frac{4}{2500}} \\ &= 0.758 \\ nd &= \frac{\frac{15}{3000} - \frac{4}{2500}}{\frac{15}{3000} + \frac{4}{2500}} \\ &= 0.515 \\ precision &= \frac{15}{4+15} \\ &= 0.789 \end{aligned}$$

As an example of how to compute the scores for a word, assume we have counted the occurrences of words in positive and negative documents and have 15 instances of the word *good* out of 3000 total words in positive documents, and 4 instances of *good* out of 2500 total words in negative documents, as shown in Figure 3.2.

Scoring a document

This section shows an example of scoring a document after we have scored the words. We show the original text of the review in Table 3.2, the text as our system sees it with the word scores in (or '-' where the system has not learned a score) in Table 3.3, and the counts and normalized counts for each bin in Table 3.4. The normalized counts then get used as the features for a machine learning algorithm.

Table 3.2: Example 5 star Review in Original Form

Excellent sound system, reminds you of being in a movie theatre. Great, Great system

Table 3.3: Example Review Preprocessed and Annotated with Word Scores

0.860	0.655	0.533	-	0.569	0.503	0.548
excellent	sound	system	reminds	you	of	being
0.501	0.518	0.889	-	0.728	0.728	0.533
in	a	movie	theatre	great	great	system

Table 3.4: Features Generated from Example Review; Words: 14 Scored Words: 12

Range	Count	Feature
0.00-0.10	0	0.000
0.10-0.20	0	0.000
0.20-0.30	0	0.000
0.30-0.40	0	0.000
0.40-0.50	0	0.000
0.50-0.60	7	0.583
0.60-0.70	1	0.083
0.70-0.80	2	0.167
0.80-0.90	2	0.167
0.90-1.00	0	0.000

After going through this process for a set of documents, we have a set of numeric features based on the distribution of the word scores that is much more compact than the bag of words representation and can be used with any machine learning algorithm.

Chapter 4

Evaluating Ordinal Problems

The evaluation of machine learning systems is an important aspect that is often overlooked. While many researchers have examined how to evaluate binary problems, and some have looked at multiclass problems, there is very little existing work focusing on evaluating ordinal problems. We consider a number of metrics that we have found in other studies, along with two novel approaches, and perform our own experiments to determine which measure we should use in the ordinal case. In order to facilitate the discussion, we narrow our focus slightly from ordinal problems as a whole to ordinal problems with certain restrictions on relative costs of errors, which we feel are appropriate for a great deal of ordinal domains, including sentiment analysis, but not all. We examine how the different error metrics relate to each other, and also analyze a scenario where we believe three of the measures do not perform as intended.

4.1 Defining the Type of Domain

In order to facilitate discussion of the sort of domain we are interested in, we present some constraints on the costs associated with different errors, as shown in the inequalities of (4.1.1). The cost of misclassifying an example of class k as class j is denoted by $cost(k,j)$. The first two conditions indicate that it is always preferable to chose a class that is closer to the optimal class given that the direction of the error is the same, while the second two impose additional constraints on the size of cost differences which can exist if the direction of the error differs. As an example, in a 5 class ordinal problem where the target class is 3, the first two conditions mean that it is always preferable to chose class 4 over class 5, or class 2 over class 1, while the last two conditions mean that it would always be preferable to chose class 2 over class 5, or class 3 over class 1. Note that we do not require that $cost(k,k+n) = cost(k,k-n)$, we only require that differences between costs of errors that are the same number of classes away from the target not be greater than differences between costs for errors with a larger class difference.

$$\begin{aligned}
cost(k,i) &< cost(k,i+n), i >= k, n > 0 \\
cost(k,i) &< cost(k,i-n), i <= k, n > 0 \\
cost(k,k-n) &< cost(k,k+n+1), n > 0 \\
cost(k,k+n) &< cost(k,k-n-1), n > 0
\end{aligned} \tag{4.1.1}$$

We feel that these are suitable constraints for the text analysis domains that we examine in this thesis, along with a large subset of all ordinal domains. However, one can imagine an ordinal scenario where these constraints may not be met. For instance if a classifier was determining if patients complaining of chest pain should be sent for emergency surgery, sent for further tests, kept for observation, or sent home, the ideal classifier should probably generally err on the side of more treatment rather than less, while the cost of being off by one is very different when the difference is emergency surgery vs. more tests instead of keep for observation vs. sent home.

4.2 Measures

We selected a variety of measures to examine through reviewing previous work on ordinal problems in a variety of domains. We again note that the majority of work has simply reported accuracy without considering other measures.

Accuracy (ACC) is frequently used as an evaluation metric. It is simple, intuitive, and represents the proportion of correctly classified examples. However, beyond problems inherent in even binary problems, in the ordinal case it does not provide any indication of the severity of the error. Accuracy ranges from 0 to 1, with 1 being ideal.

Accuracy within n (ACC1, ACC2, etc) represents a family of measures which are similar to accuracy, however, they allow for a wider range of outputs to be considered "correct". In the case where the correct output is 4 stars, outputs of 3 stars and 5 stars would be considered accurate within 1, along with the correct output of 4 stars. When there are k classes, accuracy within $k - 2$ includes all outputs as correct except for the worst possible kind of error, that is, mistaking class k with class 1 or vice versa. These measures, like accuracy, range from 0 to 1 with 1 being ideal.

These measures provide a more qualitative picture of the performance of the classifier and the severity of its errors, while greatly summarizing the information in a confusion matrix. It is not, however, the intention that they be used as stand alone performance measures in most cases. While this concept is not entirely novel ([8] uses the concept of "percentage correct within 1" for ordinal classification of credit scores), it is not used frequently.

Like all measures, this family provides a summary, and like all summaries, some information is lost. However, given our constraints on costs, we feel that these measures preserve the most important information. If detailed costs are known, it could be possible to distinguish between classifiers that are equal at all accuracies within n, however, this is generally not the case.

Mean Absolute Error (MAE) and Mean Squared Error (MSE) are frequently used for regression type problems. These measures are based on the absolute or squared difference between the desired output and the system output for each example. Strictly speaking, they make more sense when the problem is not truly ordinal and the distances between values are meaningful, however, they do distinguish between correct output, slightly wrong output, and very wrong output. In this work we use these measures on the final output of the classifier for numbered categories, and not on some internal representation of the classifier, such as an output between 0 to 1 that is divided in to different classes based on thresholds. MAE and MSE are in the range from 0 to infinity, with 0 being ideal.

When trying to devise new measures, we found that if we compute the average of the accuracies within n, $\left(\frac{a_0+a_1+\cdots+a_{k-2}}{k-1}\right)$ where k is the number of classes and a_i represents accuracy within i, with a_0 representing simple accuracy, the result is a measure that is perfectly inversely correlated with MAE but in the range from 0 to 1, with 1 being ideal. We can also express MAE and MSE in terms of a weighted sum of the accuracies within n, as shown in equation (4.2.1), where k represents the number of classes, $a_{k-1} = 1$, and a_0 represents simple accuracy.

$$\begin{aligned} MAE &= \sum_{n=1}^{k-1} n(a_n - a_{n-1}) \\ MSE &= \sum_{n=1}^{k-1} n^2(a_n - a_{n-1}) \end{aligned} \tag{4.2.1}$$

Linear Correlation (Correl) measures a linear relationship between two sets of numbers. A perfect classifier would have a correlation of 1. However, given that correlation measures the relationship between values regardless of the actual values, it is also possible for a classifier to be well correlated but to have output which is incorrect, but in this unlikely scenario the incorrect output could be linearly scaled in some way to produce correct output. Correlation ranges from -1 to 1, although will usually be from 0 to 1.

Normalized Distance Performance Measure (NDPM) is a measure introduced by [52] that is designed for information retrieval in cases where a user has established relative preferences for documents. This can be applied to ordinal classification when we assume that the user is indifferent towards documents appearing in the same class, and prefers documents from a higher class to those of a lower class. In this respect, this measure seems appropriate for purely ordinal situations where we do not know the magnitude of the difference, only that there is one.

NDPM is calculated based on comparing each pair of "documents" in the set. $d \succ d'$ means that the user prefers d to d', while $d \sim d'$ means that the user has no preference between d and d'. If $d \succ d'$ in both the user and system rankings, the pair agrees. If $d \succ d'$ in one ranking, but $d \sim d'$ in the other, the pair is compatible. If $d \succ d'$ in one ranking, and $d' \succ d$ in the other, the pair contradicts. The number of contradictory pairs between user (u) and system (s) rankings is denoted as C^-, and the number of compatible pairs for which the user assigns a preference but the system is indifferent is denoted as C^u. The DPM is then defined as $dpm(\succ_u, \succ_s) = 2C^- + C^u$.

The DPM is then normalized by comparing it to the worst case, which, as shown in [52] is the converse of the user ranking - that is, the ranking where the lowest ranking is assigned whenever the user assigns the highest ranking, the 2nd lowest ranking is assigned whenever the original assigns the 2nd highest ranking, and so forth, and not the reverse list of the rankings.

$$ndpm(\succ_u, \succ_s) = \frac{dpm(\succ_u, \succ_s)}{dpm(\succ_u, \succ_u^c)} \tag{4.2.2}$$

NDPM ranges from 0 to 1, with 0 being ideal.

ROC for Ordinal Classification (VUS) has been proposed by [48] using the volume under a 3d surface (VUS) as opposed to the area under a curve as is used in the AUC measure for binary classification. [47] introduces an efficient dynamic programming algorithm to calculate the VUS. The VUS, like AUC, ranges from 0 to 1.

A major disadvantage of this approach is that even the efficient algorithm still takes some time to compute when being calculated on a few thousand examples, while the other measures are trivial to compute. This is significant when many fairly large experiments are being conducted. Another issue is that there is an intermediate value which consists of the product of the number of examples from each class for which overflow can be a concern. A 10 class problem with 100 examples per class is enough to cause overflow in a Java `long` variable, while a 5 class problem can have over 5000 examples per class. We found that this method performed poorly in initial experiments and did not continue to use it.

Accuracy + Correlation (A+C) is a measure constructed simply by averaging the accuracy and correlation. We were motivated to try this combination in an attempt to overcome the shortcomings of accuracy by combining it with a measure that provides some information about the severity of the errors. In theory this result could range from -1 to 1, but in practice, negative correlations are only seen for very poorly performing classifiers, so the range can easily be truncated to 0 to 1, by setting any negative values as 0.

4.3 Desirable Qualities of an Evaluation Measure

Comparing evaluation measures is a task made difficult by the virtue of it being hard to define exactly when a measure is better than another, particularly in a general sense.

We feel that the accuracies within n summarize the overall picture and preserve the most important information for this type of ordinal problem. For this reason, we analyze the correlations between the other metrics and the accuracies within n, and we are looking for a measure that correlates well with all of the accuracies within n. In the particular domain of ordinal sentiment analysis, we feel that "off by one" type errors are much less serious than errors that are at the opposite end of the scale, and so we emphasize correlation with the higher accuracies within n.

In addition, given a domain where the inequalities presented in (4.1.1) are satisfied, we can define a scenario where one classifier is indisputably better than another. If accuracy within n is higher for each n (or higher for at least one and equal for others), the system with the higher accuracies within n is superior. By definition, accuracy cannot distinguish between two classifiers that have the same accuracy but are different at some higher n, likewise for each of the individual accuracies within n.

Finally, we need a measure that does not require a careful consideration of all costs, as these costs are not precisely known. In some ordinal domains, and even for some particular applications in text analysis domains, it may be possible to define such costs in detail, however, given that this thesis does not focus on a particular application it is not feasible to do so here. We note that MAE and MSE are effectively simple cost weightings on the error.

4.4 Examining Correlations Between Measures

We build classifiers using the DVD reviews in the dataset from [4]. These experiments make use of an early attempt to reduce the feature size of bag of words based sentiment analysis; in this case, we used word scoring techniques to identify the most positive and most negative words to use as features; these results are only used to judge the correlations between measures. We use 2500 reviews drawn at random from the collection of 34,741 reviews included in the "unlabeled" files to filter out the top and

Table 4.1: Correlations Between Error Measures (absolute values), 4 classifiers, imbalanced dataset

	MAE	MSE	Correl	NDPM	A+C	ACC	ACC1	ACC2	ACC3
MAE	1	0.944	0.256	0.065	0.384	0.523	0.904	0.874	0.775
MSE	0.944	1	0.234	0.014	0.272	0.226	0.923	0.944	0.898
Correl	0.256	0.234	1	0.821	0.955	0.307	0.006	0.122	0.364
NDPM	0.065	0.014	0.821	1	0.819	0.366	0.285	0.117	0.222
A+C	0.384	0.272	0.955	0.819	1	0.576	0.069	0.138	0.343
ACC	0.523	0.226	0.307	0.366	0.576	1	0.205	0.104	0.096
ACC1	0.904	0.923	0.006	0.285	0.069	0.205	1	0.924	0.696
ACC2	0.874	0.944	0.122	0.117	0.138	0.104	0.924	1	0.744
ACC3	0.775	0.898	0.364	0.222	0.343	0.096	0.696	0.744	1

bottom 25% of words that are most associated with positive and negative documents to help reduce the number of features, and another set of 2500 reviews also drawn at random from the same set to generate a bag of words feature set using the words selected in the first step as features. We then train classifiers on the features as implemented by WEKA with default settings [50] to perform 10 fold cross validation. We used 2 base classifiers, SMO and J48, and we use each of those alone and wrapped with the OrdinalClassClassifier method discussed in [13], for a total of 4 distinct classifiers. This experiment has been published in [15].

The absolute value of the linear correlations between the measures [1] are shown in Table 4.1. When we examine classifiers constructed on this dataset, MSE clearly correlates most highly with all of the accuracies within n, while MAE correlates better with simple accuracy but not as well with the accuracies within n, particularly for higher n. However, all of the other measures are very far behind.

When looking at the results of individual folds, there are cases where all accuracies within n (including simple accuracy) for one fold agree on the better classifier, yet correlation, NDPM, and accuracy + correlation are best for the worse classifier. This suggests a problem with these measures - they are not choosing the superior

[1]As calculated using the CORREL function in Microsoft Excel 2007

Table 4.2: Demonstrating flaws in metrics; A > B means that A is superior by the metric to B

Target	Output A	Output B	Measure	A	B	
1	1	1	ACC	0.5	0.5	A = B
1	4	4	ACC1	0.7	0.6	A > B
2	2	2	ACC2	0.7	0.7	A = B
3	**4**	**1**	ACC3	0.9	0.9	A = B
3	3	3	MSE	3.6	3.9	A > B
4	3	3	MAE	1.2	1.3	A > B
4	4	4	Correl	0.12	0.15	**A < B**
4	1	1	A+C	0.31	0.32	**A < B**
5	5	5	NDPM	0.45	0.44	**A < B**
5	1	1				

classifier.

4.5 Examining Agreement with Accuracies Within n

As established previously we view the set of accuracies within n as summarizing the most important information about a classifier for ordinal applications given our constraints on the costs of different kinds of errors. In particular, if each accuracy within n for one classifier is equal to or greater than the equivalent accuracy within n for a second classifier (with at least one being greater), the first classifier is superior. A measure which does not always agree with this assesment is not a good measure for this task.

In Table 4.2, we demonstrate that some metrics can produce results in which an inferior classifier is deemed to be superior by some of the measures through presenting a simple example with 10 data points, with one difference between the two classifiers. Classifier A meets our definition for a superior classifier, as in a case where the target is 3, it predicts 4 and B predicts 1. Both MAE and MSE agree with this assessment, while the individual accuracies within n find no difference except for ACC1. Classifier B is deemed superior by correlation, accuracy + correlation, and NDPM. Given that

MSE and MAE can be expressed as weighted averages of the accuracies within n, it is not possible for this flaw to apply to them.

4.6 Conclusions

Both the experimental results examining the correlations of the metrics, and the example of how correlation, accuracy + correlation, and NDPM can at least sometimes find that an inferior classifier is better, show that of these measures, MAE and MSE are most desirable for the subset of ordinal problems meeting the constraints we defined for the magnitude of the costs. Given our particular domain, we feel that MSE is most appropriate as we want to absolutely minimize the largest errors at the expense of potentially having more small errors. This is despite the fact that neither of these two measures is truly ordinal by design.

Chapter 5

Parameter Settings

In this chapter, we test the three proposed scoring methods, experiment with varying numbers of bins, and finally, test several alternative machine learning algorithms to optimize speed and performance. We perform these tests using two different datasets, each of which is used as both a binary and ordinal problem.

5.1 Datasets

In order to determine reasonable settings for our system, we use two different publicly available sentiment analysis datasets, and use each as a binary and as an ordinal problem. The first set is the electronics reviews from [4], which we will refer to as the electronics dataset. This set consists of user reviews collected from Amazon.com, and rated on a 5 star scale. The authors eliminated reviews of 3 stars, as their focus was on the binary problem; however, for ordinal tests, we treat this as a 5 class classification problem that has no examples of class 3 in order to preserve the larger gap between a 2 star and 4 star rating for the evaluation methods, while in the binary case, less than 3 stars is negative and over 3 stars is positive. The second dataset consists professional movie reviews by one author (Steve Rhodes, also referred to as author A) compiled by Pang & Lee [37], which we will refer to as the Pang dataset. The reviews in this dataset consist of only the subjective elements and are scored both as a 4 class ordinal problem and on a scale from 0-1. For the ordinal problem, we use the 0-3 scale, while for the binary problem, we treat reviews with a score of 0.5 or less as negative and the rest as positive.

5.2 Methodology

In order to test a wide variety of options relatively quickly, we use 5 fold cross validation except where otherwise noted. Results on ordinal problems will be compared using Mean Squared Error (lower is better), while results on binary problems will be compared using AUC (higher is better).

For the machine learning algorithms, we use the implementations found in WEKA 3.6.1 [50] with default settings except where otherwise noted, and access these through the Java API. One major exception is that for SMO classifiers, the option to "fit logistic models" was found to substantially improve AUC and so we use it in all SMO classifiers. Running times reported in this chapter include the time taken to load all of the files and perform a 5 fold cross validation, and are run on a laptop with a Pentium M 1.73 GHz processor, and a memory limit of 1200mb for the Java virtual machine.

5.3 Experiments

5.3.1 Determining appropriate scoring methods and numbers of bins

The purpose of this experiment is to determine which scoring method(s) are best and to examine the effects of varying the numbers of bins. We use the SMO classifier

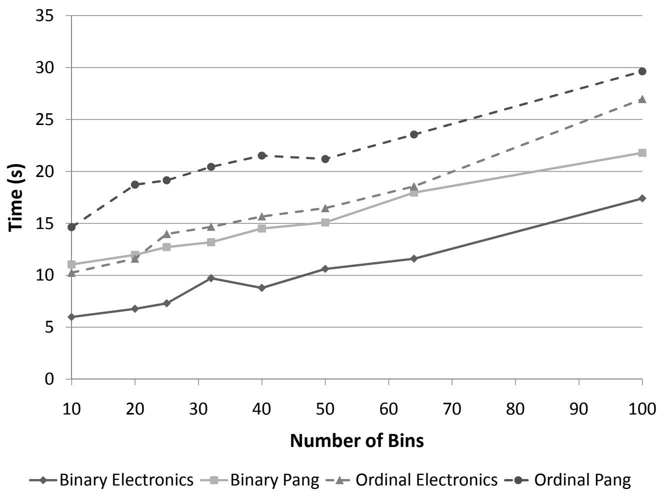


Figure 5.1: Influence of the number of bins on total time

(WEKA's implementation of SVM) as it has shown promise in preliminary work. We test the three scoring methods with a variety of different numbers of bins on the ordinal and binary versions of the two datasets, using 5 fold cross validation. The influence of the number of bins on the time required to train the classifiers is shown in 5.1, and the performance results are shown in figures 5.2 and 5.3.

Overall, we find that the differences between different numbers of bins and scoring methods are quite small over a range of possibilities. While error bars are not shown in these graphs, generally most of the results are within the 95% confidence range of the best result. Precision is most often the best scoring system.

For the binary problems, shown in figure 5.2, using as few as 10 bins produces good results on one problem, but not on the other. Overall, 25 bins seems to be a good choice for a default setting as the results are good on both datasets. As the best performance is at or below 25 bins in both cases, and increasing the number of bins slows the classifier down, there seems little reason to consider using more than

25 bins for binary problems.

Results are more mixed for the ordinal problems, shown in figure 5.3, but generally, they appear to benefit from using more bins than previously suggested for the binary problems. The electronics dataset seems to benefit from increasingly large numbers of bins, however, this produces a slower classifier. 50 bins seems to be a good first choice considering both datasets; it produces close to optimal results on both datasets, while still being relatively fast.

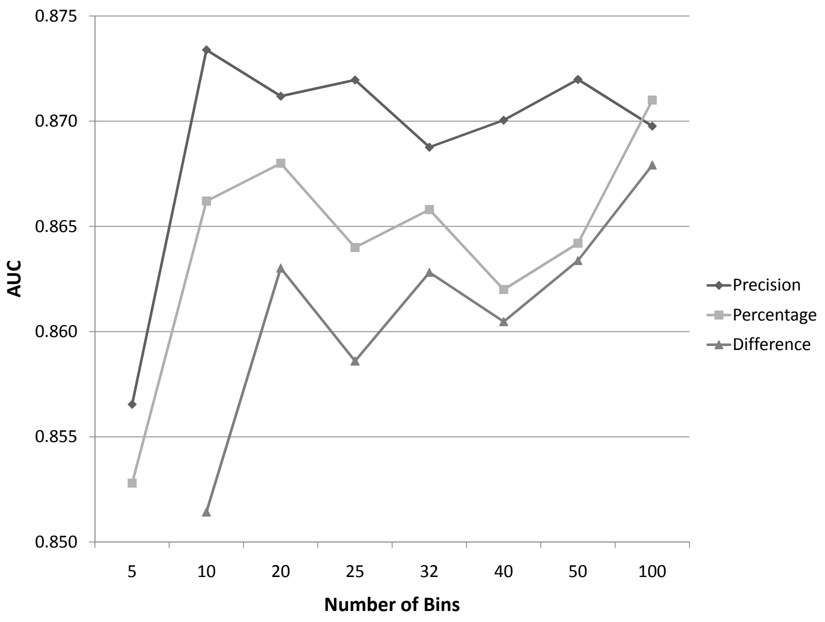
The precision scoring method nearly always produces the best results overall. The only exception is on the ordinal Pang dataset, where all 3 scoring methods produce incredibly close results at 32 bins.

Thus, the recommendations from this study would be to use precision scoring, and use 25 bins for a binary problem and 50 bins for an ordinal problem. If there is time for further optimization of the parameters, there seems no point in searching above 50 bins for the binary problem, while the ordinal problem may have improved performance above 50 bins, and probably would not benefit from going lower than 32. However, while it is possible to make bad choices for the number of bins for a particular problem, most choices in the range of 10–100 bins will be very close to the best.

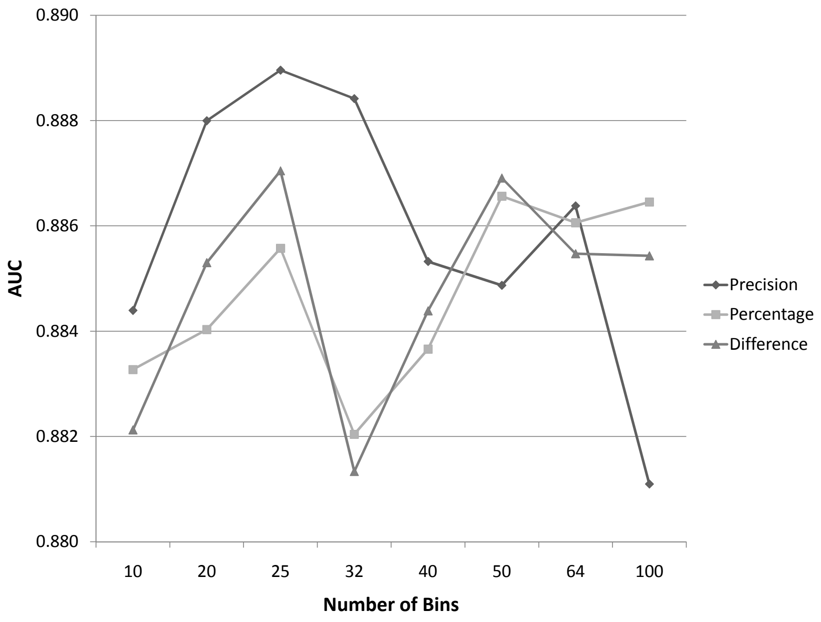
5.3.2 Testing a wide range of algorithms

In this section we will investigate whether SMO is a good choice of algorithm, or whether there are other algorithms we should consider. The times reported here are for a single run of 5 fold cross validation. Differences within a few seconds may not be significant, however, as shown in Table 5.1 most of the time differences are quite large.

We select a number of very different algorithms and use them with the default implementations in WEKA in order to search for alternatives that compete with SMO

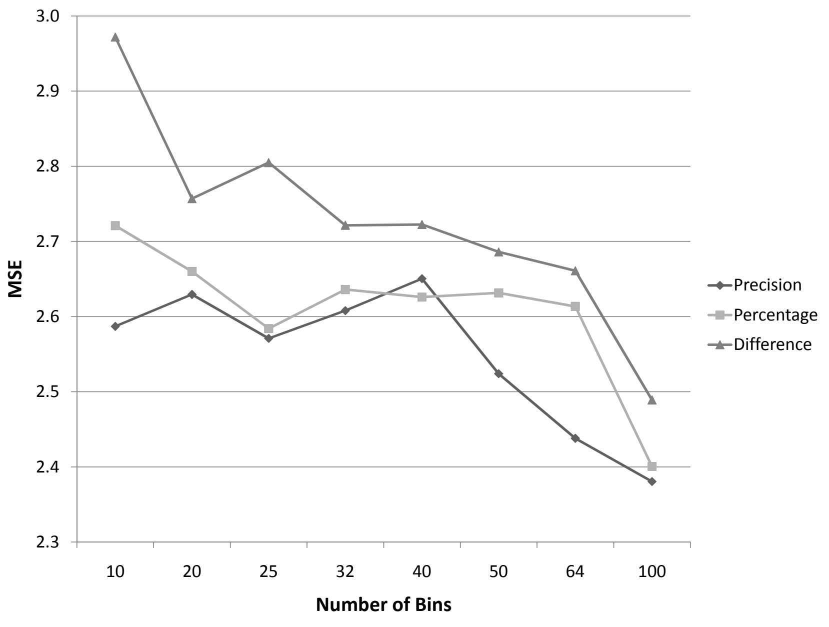


(a) binary electronics

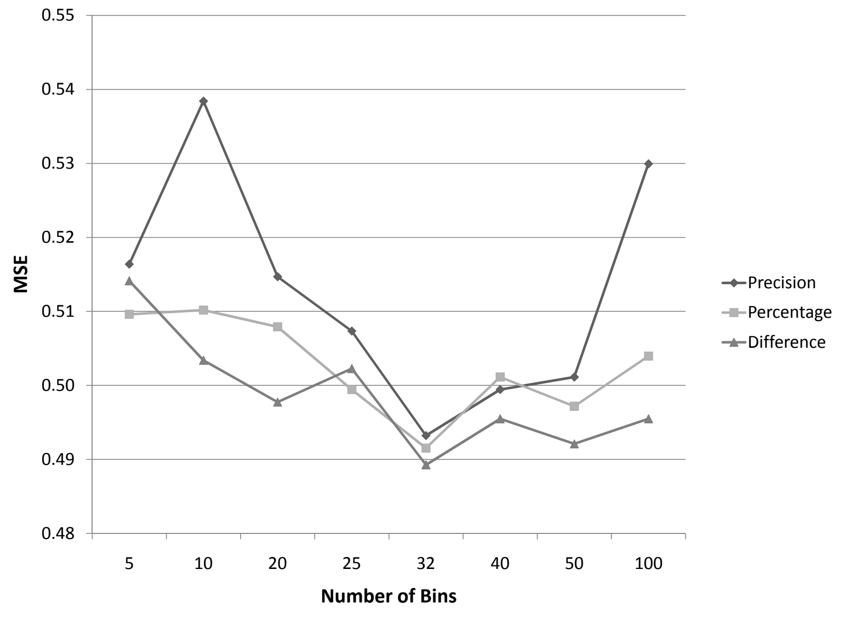


(b) binary Pang

Figure 5.2: AUC, varying numbers of bins and scoring methods, binary datasets



(a) ordinal electronics



(b) ordinal Pang

Figure 5.3: MSE, varying numbers of bins and scoring methods, ordinal datasets

Table 5.1: Several algorithms on the Ordinal Pang Dataset with 50 bins

Algorithm	MSE	Accuracy	Time (S)
SMO (Poly kernel)	0.501	0.558	20.1
SMO (PUK kernel)	0.453	0.595	112.5
SMO (RBF kernel)	0.480	0.566	123.1
BayesNet	0.519	0.559	13.5
MultilayerPerceptron	0.556	0.547	525.4
RandomForest	0.666	0.506	18.2
Kstar	0.702	0.499	592.5
NaiveBayes	0.705	0.489	15.7
REPTree	0.755	0.467	13.9
J48	0.833	0.472	17.3

with respect to both speed and performance. We compare all classifiers using 50 bins, as this was a good choice for SMO; while this number of bins may not be optimal for all classifiers, given the previous results which imply that the number of bins does not make a large difference within a wide range, we feel that a classifier that is substantially worse than SMO with this setting is not likely to be competitive with SMO at any setting.

Table 5.1 shows the performance of a selection of classifiers along with two different kernels for the SMO algorithm, compared with the default SMO polynomial kernel which was used in the previous experiments, sorted by MSE.

The two alternate SMO kernels perform well, however, they are considerably slower than the polynomial kernel. The PUK kernel performs best of the two and also appears to be faster than RBF. The BayesNet classifier is of interest as its performance is quite similar to our baseline, while being considerably faster. MultilayerPerceptron's performance is not dramatically worse than the baseline, but its speed is and we will not consider it further. All remaining classifiers are markedly inferior in performance.

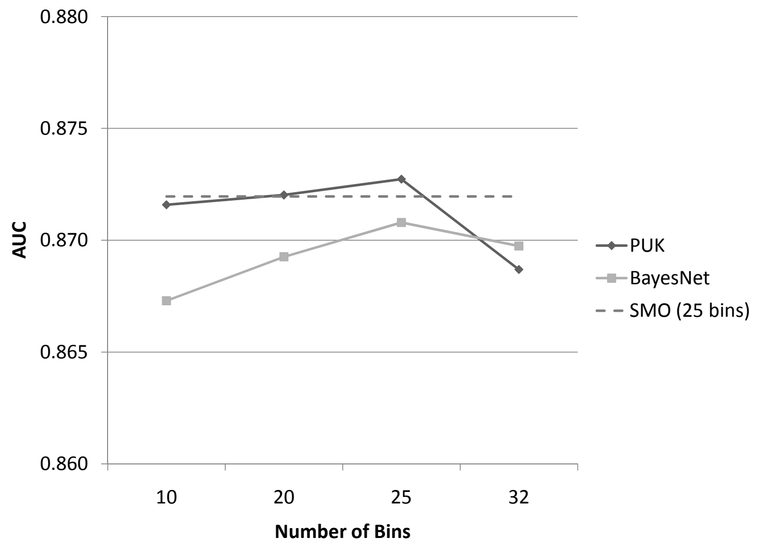
5.3.3 Further testing of BayesNet and SMO with PUK Kernel

In this experiment, we compare BayesNet and SMO with PUK with varying numbers of bins to our baseline SMO results, in order to determine the impacts of these choices on speed and performance and whether either of these algorithms is a worthwhile choice.

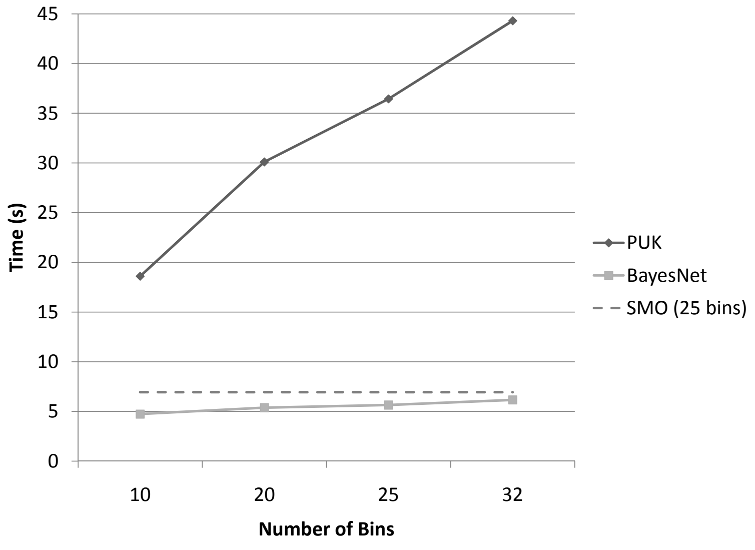
On the binary electronics dataset, as shown in figure 5.4, PUK requires far longer to obtain results that are not significantly better than the baseline, while BayesNet is faster than the baseline, but slightly worse performance wise.

The results on the ordinal Pang dataset are shown in Figure 5.5. On this dataset, SMO with the PUK kernel remains substantially slower than SMO with the linear kernel on the ordinal Pang dataset, and remains so even with small numbers of bins. The initial selection of 50 bins is best in terms of performance, while 25 bins appears to be a reasonable compromise between between performance and time. However, even with 10 bins, this algorithm is considerably slower, and with 10 bins, the performance advantage that we see with 25 or more bins disappears. BayesNet is faster than the SMO baseline on this dataset, while also achieving good performance with small numbers of bins.

Given the gap in training times, and the fact that this method's primary benefit is speed, we will not consider the PUK kernel further, however, there may be situations where speed is less of a concern where this option would be useful. BayesNet, on the other hand, remains promising.

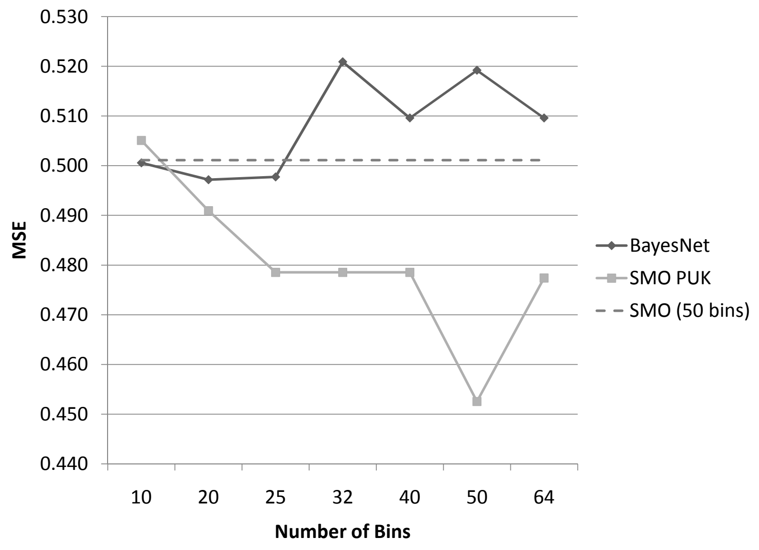


(a) AUC

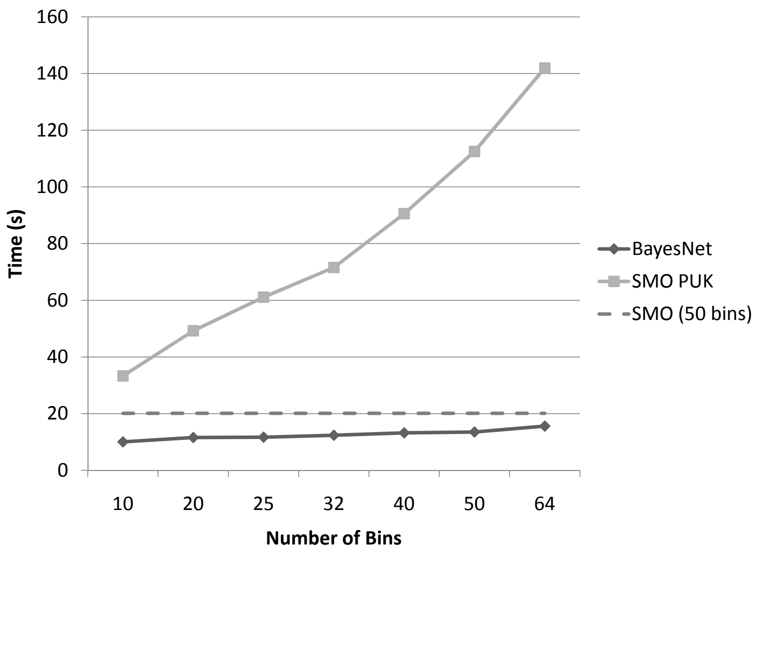


(b) Time

Figure 5.4: BayesNet and SMO with PUK kernel, with SMO 25 bin baseline, binary electronics dataset



(a) AUC



(b) Time

Figure 5.5: BayesNet and SMO with PUK kernel, with SMO 50 bin baseline, ordinal Pang dataset

5.3.4 BayesNet vs. SMO on all test datasets

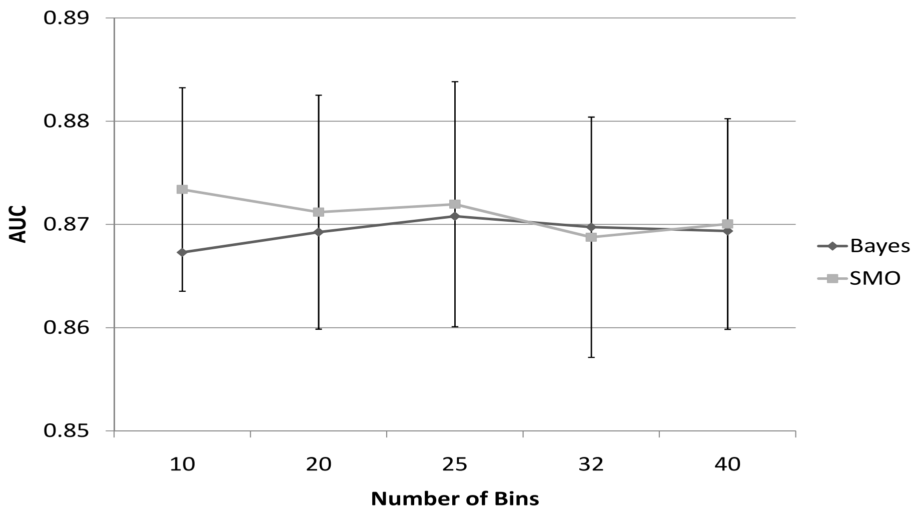
Now that we have determined that BayesNet is competitive on performance and is fast, we will look at BayesNet in comparison to SMO on all datasets and with a wide range of bins, as well as showing how the differences compare with the 95% confidence range of the SMO results.

On the ordinal Electronics dataset, shown in figure 5.7a, BayesNet is consistently performing better than SMO although both are close. All of the BayesNet results fall within the 95% confidence range of the best SMO result, however, the best overall results are with BayesNet and many of the BayesNet results perform significantly better than many of the SMO results. On the ordinal Pang dataset, shown in figure 5.7b, both the best and the worst results are obtained with SMO, and 4 out of 6 of the BayesNet results are within the 95% confidence range of the best result. BayesNet appears to have a small advantage over SMO in terms of performance on the ordinal datasets, but not enough to establish a significant difference.

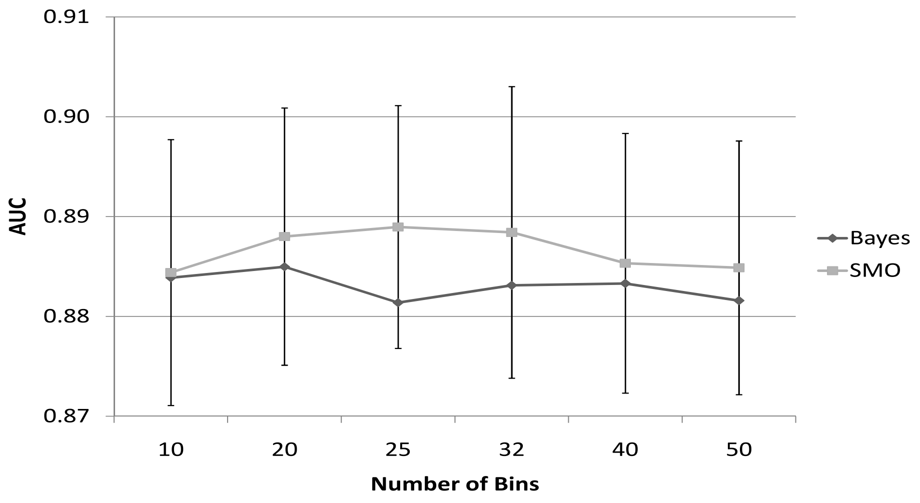
On the binary problems, shown in figure 5.6, the two classifiers are very similar in terms of performance, with all results falling in the 95% confidence range of the best. However, SMO is consistently producing better results.

5.3.5 BayesNet vs. SMO speed analysis

Finally, we would like demonstrate the time differences between these algorithms in a somewhat more rigorous, controlled fashion than the previous tests. In order to do this, we perform 3 runs of 5 fold cross validation using varying numbers of bins on the ordinal Pang and binary electronics datasets. For SMO, we select the smallest number of bins that is reasonable, and then the optimal number of bins for the dataset in the case of the ordinal Pang set, and the overall recommendation of 25 bins in the case of the binary electronics set (as 10 bins was both the optimal for the dataset and the smallest reasonable value). In order to minimize the effects of other CPU use, these

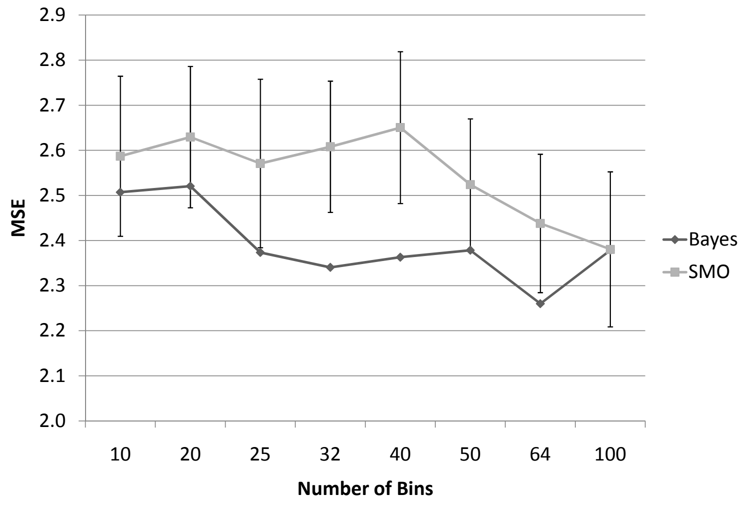


(a) Electronics

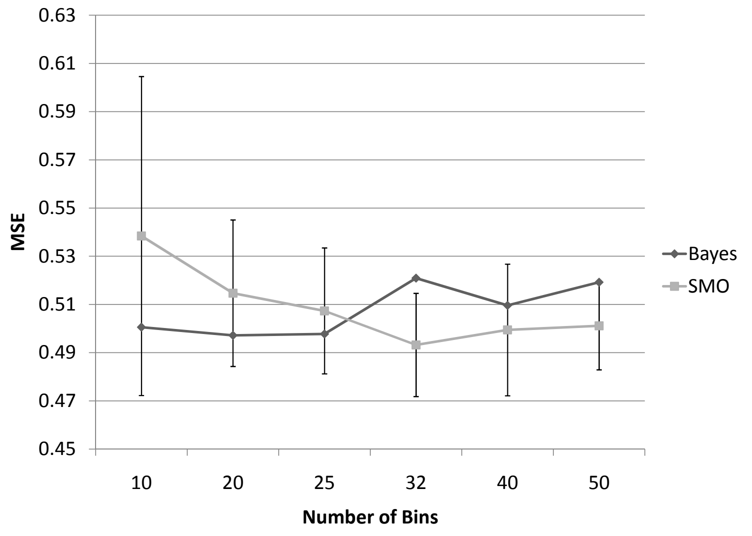


(b) Pang

Figure 5.6: BayesNet vs. SMO, with 95% confidence range for SMO, binary datasets



(a) Electronics



(b) Pang

Figure 5.7: BayesNet vs. SMO, with 95% confidence range for SMO, ordinal datasets

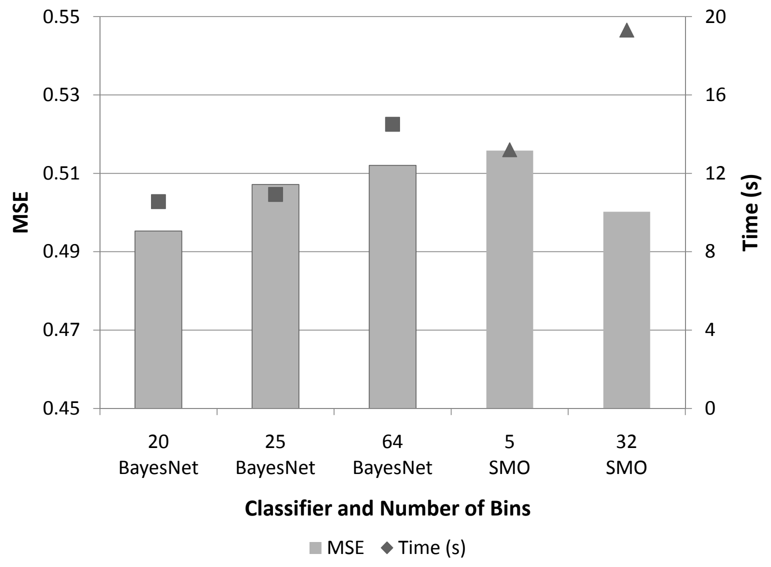
experiments were done within a short time period, and with the same background conditions. We chose these two test datasets as they had the smallest time differences in preliminary work. Error bars are not shown here as they are very small; the times recorded were quite consistent from run to run, varying by 0.5s using SMO with 32 bins on the ordinal Pang dataset, and less than 0.25s in all other cases.

Time differences are very significant in the Ordinal case, as shown in Figure 5.8a. SMO with 32 bins takes nearly twice as long as BayesNet with 20 bins - and a substantial part of the time recorded here is the time to read in datasets. This is presumably because SVM handles ordinal classification by constructing multiple binary classifiers; in addition, BayesNet performs best with fewer bins. In terms of performance, the BayesNet classifiers compare well to the results with SMO; while the differences are not large enough to be significant, BayesNet is performing best.

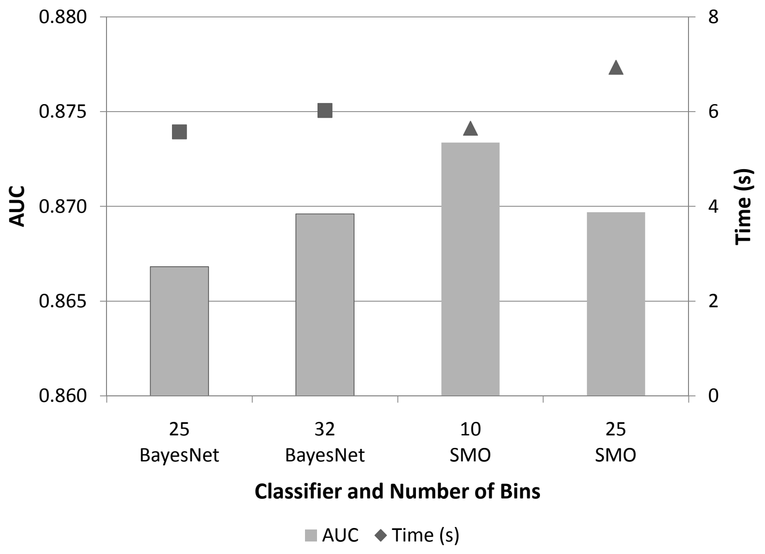
In the Binary case, shown in Figure 5.8b, the time differences are much smaller, with the differences between SMO with 10 bins and BayesNet with 25 bins not being significant using a T-test. Given that the time differences here are much smaller than in the ordinal case, performance slightly favours SMO, and SMO is a bit less sensitive to the number of bins chosen, we will continue to recommend SMO with 25 bins in the binary case.

5.4 Conclusions

Given that the primary motivation of using this feature extraction method is speed, we will focus on using BayesNet with 25 bins for the Ordinal case as performance is slightly better while being much faster, and SMO with 25 bins for the binary case, as performance seems slightly better with only a small speed penalty.



(a) Pang ordinal



(b) electronics binary

Figure 5.8: Speed and error for SMO and BayesNet classifiers

Chapter 6

Experiments

This chapter presents the results of comparing the numeric scoring method described previously with a baseline Bag of Words method (BOW). In order to do so, we test the two methods on a variety of datasets from different sources. The numeric method is performed using 25 bins in all cases, with a BayesNet classifier for ordinal problems and SMO classifier for binary problems.

We consider a wide selection of datasets. We group them here into document level sentiment analysis, short sentiment analysis, and other related problems. Roughly speaking, the short texts represent up to a few sentences, while the documents are generally longer; but while there is a clear distinction in terms of the average size of elements in these datasets, there is overlap for particular instances. The datasets are described in detail in section 2.4.2.

In order to thoroughly test the numeric method, we will perform 10 x 10 Cross Validation on the smaller datasets, however, this is not possible on all datasets due to their size or the availability of test data. As discussed in Chapter 4, we use Mean Squared Error (MSE) to compare error for ordinal cases, and Area under the ROC curve (AUC) for binary cases, and also report accuracy on all cases. We include 95% confidence intervals where repeated tests were run, and when these are close, we use a T-Test to demonstrate statistical significance. We will report p-values where they fall in the borderline of significance, between 0.1 and 0.01 so that the reader may make their own judgements in these more borderline cases.

Two different computers were used to run these experiments. One machine is a laptop with a Pentium M 1.73Ghz processor and 4400 RPM hard drive. The other is a desktop computer with an AMD Athlon 64 X2 5200+ Dual Core processor running at 2.7Ghz. The Java Virtual Machine on the laptop was given 1200mb of RAM, while on the desktop it was given 1500mb. Unless otherwise noted, experiments in the same group are run on the same machine, with the laptop generally being used for shorter experiments. The reported times are meant to highlight large differences.

6.1 Document Level Sentiment Analysis

6.1.1 Amazon Reviews

This dataset consists of online user reviews across 4 categories and was used in [4]. As shown in Table 6.1, the numeric scoring method is always slightly more accurate with substantially higher AUC, while also being considerably faster. The Electronics portion of this dataset was used for parameter tuning.

In this case, we also chose to look at how a classifier trained and tested on all datasets together performed in comparison to the average of all classifiers. We found that both can train a classifier using all of the data that is slightly better than the average performance of the individual classifiers, however, training this classifier is very, very slow for the BOW method - so much slower we only performed 2x10 CV, and used the faster desktop computer, rather than 10x10 CV and it still took many times longer to train. On the other hand, the numeric method trains a combined classifier slightly faster than the sum of the individual numeric classifiers.

Table 6.1: Amazon reviews, BOW vs. Numeric, Accuracy, AUC, and Time

Dataset	Type	Accuracy		AUC		Time (h:mm:ss)
Electronics	Numeric	0.801	± 0.005	0.874	± 0.004	0:01:45
	BOW	0.791	± 0.005	0.791	± 0.005	0:37:00
DVD	Numeric	0.797	± 0.005	0.865	± 0.005	0:02:24
	BOW	0.775	± 0.006	0.776	± 0.006	1:03:04
Books	Numeric	0.768	± 0.005	0.839	± 0.005	0:02:21
	BOW	0.754	± 0.006	0.754	± 0.006	1:11:31
Kitchen	Numeric	0.814	± 0.006	0.896	± 0.004	0:01:39
	BOW	0.809	± 0.005	0.809	± 0.005	0:39:11
All Data [a]	Numeric	0.796	± 0.003	0.874	± 0.002	0:07:41
	BOW	0.791	± 0.002	0.791	± 0.002	16:40:12
Average [b]	Numeric	0.795	± 0.005	0.869	± 0.005	0:08:10
	BOW	0.782	± 0.006	0.782	± 0.006	3:30:46

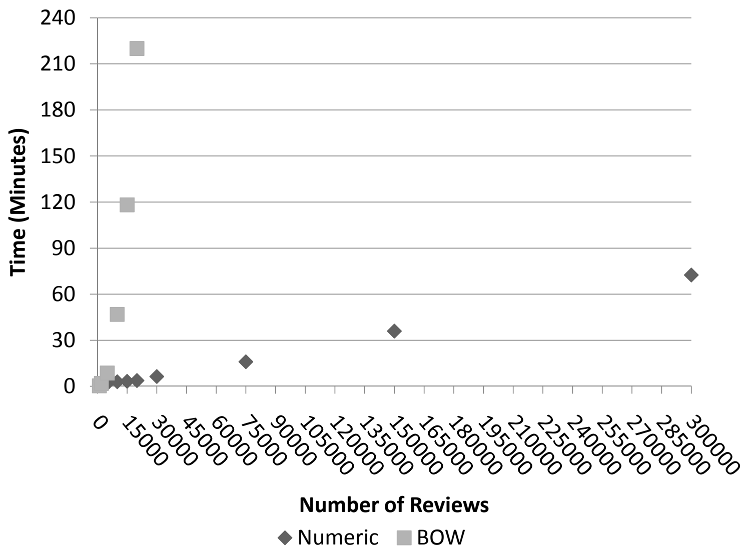
[a]One classifier trained on all datasets. BOW classifier is 2x10 CV, all other classifiers 10x10 CV

[b]Average performance of the individual classifiers over all datasets and total time to train the individual classifiers

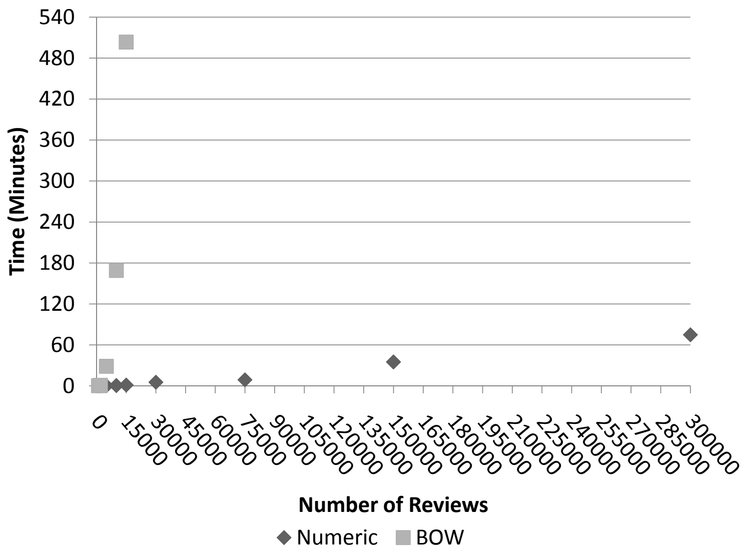
6.1.2 Large Amazon Datasets

This data is a superset of the smaller Amazon reviews used in the previous section. It maintains the original skewed distribution of online user reviews, however, these too do not include 3 star reviews. We use 3 categories for which over 100,000 reviews are available, and in all cases, these results use a test set of 10,000 reviews, which is larger than the total number of documents in most of the other data sets.

Varying Dataset Size The first stage presents a wide cross section of dataset sizes with both the numeric method and the bag of words method in order to demonstrate how both methods react to increasing amounts of training data, on both ordinal and binary datasets. We demonstrate that the time taken to train using the Bag of Words dataset grows much faster than the time taken to train with the numeric method. These experiments were performed by running the numeric classifier for a given number of documents and then the BOW method. Due to some common



(a) Binary



(b) Ordinal

Figure 6.1: Run Time Comparison, Books Dataset

Table 6.2: Books Large data set, Ordinal

	Numeric			BOW		
# of reviews	MSE	Accuracy	Time (h:mm:ss)	MSE	Accuracy	Time (h:mm:ss)
1000	1.826	0.603	0:00:07	1.897	0.579	0:00:19
2000	1.638	0.624	0:00:09	1.784	0.581	0:00:54
5000	1.530	0.630	0:00:19	1.640	0.607	0:28:39
10000	1.393	0.633	0:00:37	1.439	0.642	2:49:02
15000	1.315	0.624	0:01:22	1.277	0.670	8:23:31
30000	1.263	0.622	0:05:26			
75000	1.200	0.628	0:08:58	Majority Classifier		
150000	1.162	0.634	0:35:10	2.034	0.630	
300000	1.056	0.653	1:14:54			

preprocessing of the data, the method which is run first has an advantage in terms of time, which is often significant when comparing the smallest datasets; this situation is reversed in the case of the ordinal DVD reviews, and the effect of this reversal will be examined in the next section.

All combinations are run with a range of numbers of training documents. In all cases, training sizes of 1000, 2000, 5000, and 10,000 reviews are used, along with several larger sets based on fractions of the available data. We did not train BOW on the largest numbers of documents as it was already very time consuming to train those classifiers using 10,000-20,000 documents, and the speed was increasing rapidly. Figure 6.1 shows the time differences graphically on the Books dataset, for which we had the largest amount of data to use.

For comparison, we include the results of a simple majority class classifier on the same test set.

In the case of ordinal book reviews, shown in Table 6.2, the best overall accuracy is obtained with the BOW classifier developed with 15,000 reviews; however, this classifier took over 8 hours to train, and is also considerably worse by MSE than many of the numeric classifiers. With small training sets (1000-5000), the numeric

Table 6.3: Books Large data set, Binary

	Numeric			BOW		
# of reviews	Accuracy	AUC	Time (h:mm:ss)	Accuracy	AUC	Time (h:mm:ss)
1000	0.865	0.812	0:01:18	0.846	0.637	0:00:16
2000	0.872	0.825	0:00:41	0.863	0.675	0:01:48
5000	0.884	0.863	0:01:54	0.876	0.722	0:08:34
10000	0.889	0.877	0:02:58	0.891	0.757	0:46:48
15000	0.900	0.892	0:03:16	0.899	0.776	1:58:09
20000	0.901	0.897	0:03:45	0.906	0.797	3:39:58
30000	0.905	0.900	0:06:22			
75000	0.914	0.919	0:15:60	Majority Classifier		
150000	0.922	0.935	0:35:56	0.859	0.500	
300000	0.929	0.950	1:12:30			

classifier is more accurate with a modestly improved MSE, and also faster (particularly relevant with the dataset of 5000). With moderate training sets (10000-15000), BOW performs better by accuracy and is close with MSE; however, BOW requires hours to train on these datasets. The best overall performance by MSE is obtained using the numeric features.

For the binary book reviews, shown in Table 6.3, at a given amount of data the accuracy for both features is similar, while the AUC for the numeric features is considerably better. In this case, we are able to obtain the best accuracy using large datasets with the numeric method. The numeric method can process large datasets very quickly.

In the case of ordinal music reviews, shown in Table 6.4, the imbalance is so severe that it becomes difficult to improve over a majority class classifier; with 2000 or fewer training examples available, the majority class classifier is the best choice by both measures. However, with more data available, the classifiers trained with numeric features are clearly the better choice by MSE, and accuracy remains very close to that of a majority class classifier. Examining a confusion matrix shows that with more and more data, the classifier improves on identifying the minority classes.

Table 6.4: Music Large data set, Ordinal

	Numeric			BOW		
# of reviews	MSE	Accuracy	Time (h:mm:ss)	MSE	Accuracy	Time (h:mm:ss)
1000	1.272	0.700	0:00:55	1.350	0.637	0:00:17
2000	1.270	0.695	0:00:13	1.352	0.639	0:00:53
5000	1.148	0.699	0:00:29	1.290	0.657	0:27:58
8109	1.078	0.699	0:00:52	1.206	0.662	1:42:40
10000	1.040	0.698	0:01:01	1.218	0.662	3:08:09
16218	1.027	0.694	0:01:26			
40545	0.995	0.687	0:03:53	Majority Classifier		
81090	1.002	0.691	0:07:54	1.259	0.704	
162180	0.956	0.694	0:31:55			

Table 6.5: Music Large data set, Binary

	Numeric			BOW		
# of reviews	Accuracy	AUC	Time (h:mm:ss)	Accuracy	AUC	Time (h:mm:ss)
1000	0.917	0.791	0:00:10	0.904	0.572	0:00:13
2000	0.918	0.809	0:00:12	0.903	0.619	0:00:24
5000	0.923	0.848	0:00:27	0.906	0.675	0:02:44
8109	0.926	0.857	0:00:23	0.915	0.707	0:08:11
10000	0.929	0.869	0:00:49	0.921	0.717	0:13:58
16218	0.932	0.879	0:01:11			
40545	0.936	0.900	0:03:39	Majority Classifier		
81090	0.941	0.919	0:06:44	0.918	0.500	
162180	0.943	0.925	0:26:47			

Table 6.6: DVD Large data set, Ordinal

	Numeric			BOW		
# of reviews	MSE	Accuracy	Time (h:mm:ss)	MSE	Accuracy	Time (h:mm:ss)
1000	1.765	0.597	0:00:06	1.870	0.567	0:00:18
2000	1.489	0.608	0:00:08	1.720	0.580	0:00:52
5000	1.346	0.590	0:00:11	1.531	0.602	0:34:40
5621	1.299	0.600	0:00:12	1.458	0.606	0:39:26
10000	1.338	0.597	0:00:18	1.376	0.623	3:44:20
11243	1.332	0.600	0:00:35			
28109	1.276	0.605	0:03:25	Majority Classifier		
56219	1.223	0.602	0:03:28	1.970	0.611	
112438	1.217	0.613	0:16:05			

Table 6.7: DVD Large data set, Binary

	Numeric			BOW		
# of reviews	Accuracy	AUC	Time (h:mm:ss)	Accuracy	AUC	Time (h:mm:ss)
1000	0.876	0.826	0:00:14	0.860	0.645	0:00:17
2000	0.884	0.841	0:00:16	0.870	0.691	0:00:31
5000	0.889	0.849	0:00:35	0.885	0.735	0:04:48
10000	0.895	0.867	0:01:11	0.891	0.757	0:33:33
20000	0.907	0.901	0:02:11	0.908	0.799	3:01:31
25000	0.911	0.905	0:04:38			
50000	0.917	0.915	0:10:40	Majority Classifier		
100000	0.920	0.924	0:24:37	0.867	0.500	

The binary Music dataset, shown in Table 6.5, is by far the fastest for the BOW method to learn with 10,000 data points. Like in the case of the ordinal version of this dataset, accuracy hovers around that of a majority class classifier, and the BOW classifier only beats the majority classifier in terms of accuracy with 10,000 examples. Again, the two methods are similar in terms of accuracy, with the numeric method having a slight edge with a given amount of data. As noted in other sets, with 10,000 documents, the performance difference by accuracy is quite small.

On the ordinal DVD dataset, shown in table 6.6, we notice that the accuracy stays within a fairly small range for the numeric method, while it varies more widely for the BOW classifier - both the best and the worst accuracies come from the BOW method. We note that on this dataset the classifiers were trained in the opposite order as on the other datasets, meaning that the BOW method has a disadvantage in terms of time due to its time recording some of the common preprocessing steps.

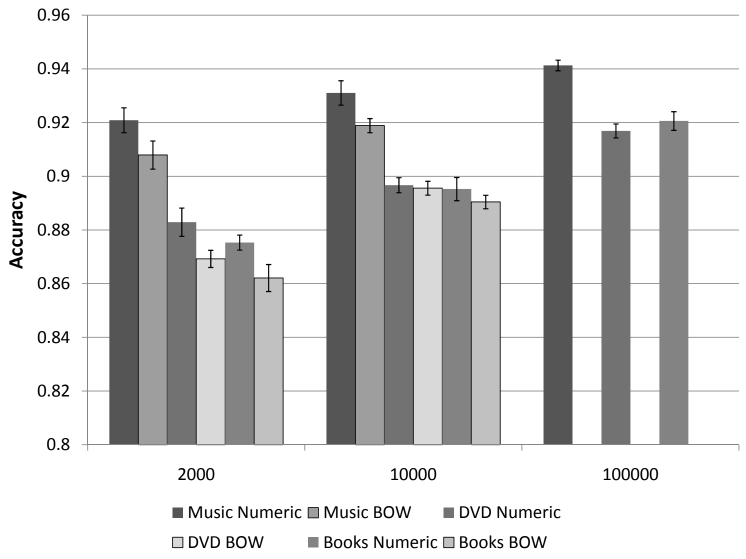
On the binary DVD reviews, shown in Table 6.7, with a given amount of data the numeric method is always much better in terms of AUC, similar but generally slightly better in terms of accuracy, but, as with the other datasets, much faster.

Repeated Runs In order to further demonstrate the time differences in the range of dataset sizes where they are comparable we will focus on a smaller selection of dataset sizes and look at the effects of altering the order in which the experiments are performed. This also provides repetition of the results. We chose dataset sizes with an intent to cover a wide range of allowable training times up to approximately 30 minutes, however note that the time required to train using a particular number of reviews varied considerably between some of the datasets, with music being particularly fast.

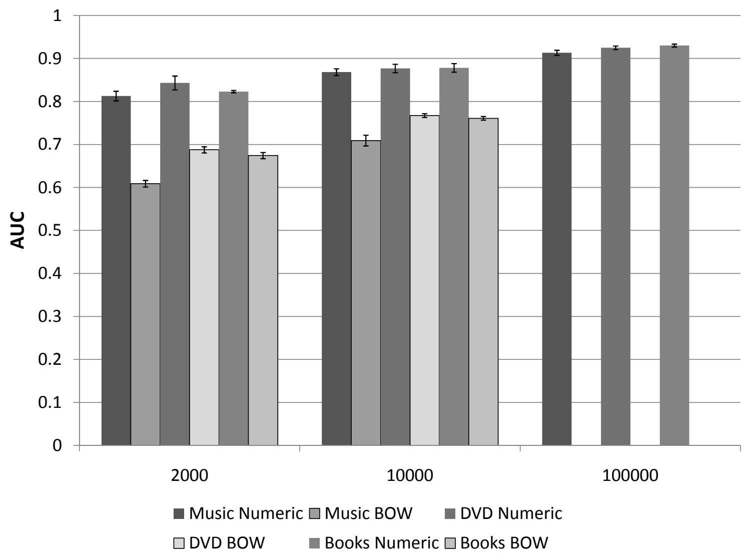
We perform 4 repetitions of the following process, with training sets of 2000, 5000 (ordinal datasets only), 10000 (binary datasets only), and 100,000 (numeric features only).

```
for i = 1..4
    Randomly select a 10000 document test set
        For each training set size
            if i is even
                train and test numeric method
                train and test BOW method
            else
                train and test BOW method
                train and test numeric method
```

The performance of these tests is shown in Figure 6.2 for the Binary case and



(a) Accuracy



(b) AUC

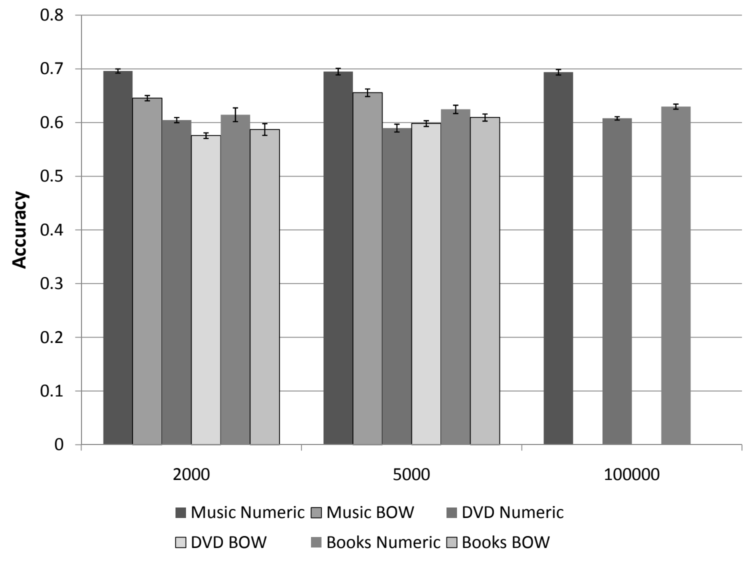
Figure 6.2: Performance, Binary Datasets, with 95% Confidence Interval

6.3 for the Ordinal case. We notice that results are closest with 5,000 and 10,000 document sets, and performed T-Tests on the closest results. The accuracy difference between the two methods is not significant for the Books and DVD datasets with 10,000 documents in the binary case, or for Books with 2000 or DVD with 5000 in the ordinal case. For 5000 Book reviews in the ordinal case, we found borderline significance with $p = 0.068$. In terms of MSE, we found the difference between the two measures with 2000 Music reviews to be significant with $p = 0.012$, and for 2000 DVD reviews we found borderline significance at $p = 0.071$.

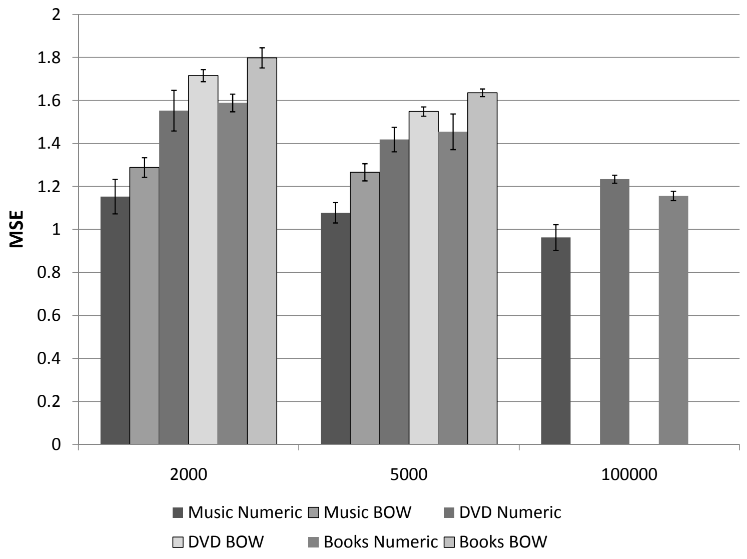
When discussing time here, we divide the time into the "setup" time and the "classifier" time. The setup time includes the time taken to read in the documents from text files on disk and perform all processing to convert them into the features given to the machine learning algorithm (in this case, WEKA Instances). As the system was not implemented with this sort of time comparison in mind, there are a few particulars that need to be considered. Documents are only read in as needed, and, in addition, the first time a particular document is used, some initial calculations common to both representations are performed. The test documents are accessed for the first time during the very first run of a set, which greatly affects the set up time of the first run given the large size of the text set.

Because of this, the time for the first run of 2000 is greatly inflated, as it includes the time to process the 10000 test documents, while the time for the second run is optimistic. The most realistic setup times would be for the first runs of the larger tests; furthermore, the "first" runs for the higher numbers of documents are also slightly faster.

There was a large amount of variance in the setup times, as shown in Figure 6.4. While the numeric features appear to be slightly faster to set up than the equivalent BOW scenarios, given this variance it is difficult to make a claim stronger than stating that this part of the training time does not take any longer for the numeric features than it does for the BOW features.

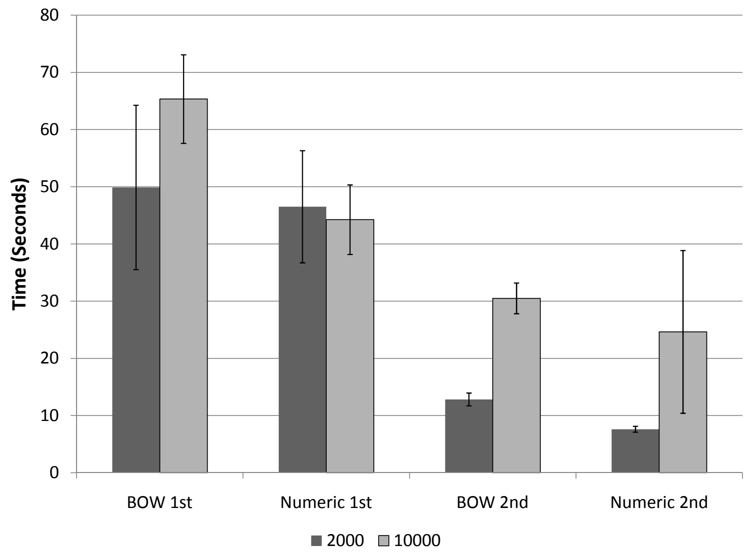


(a) Accuracy

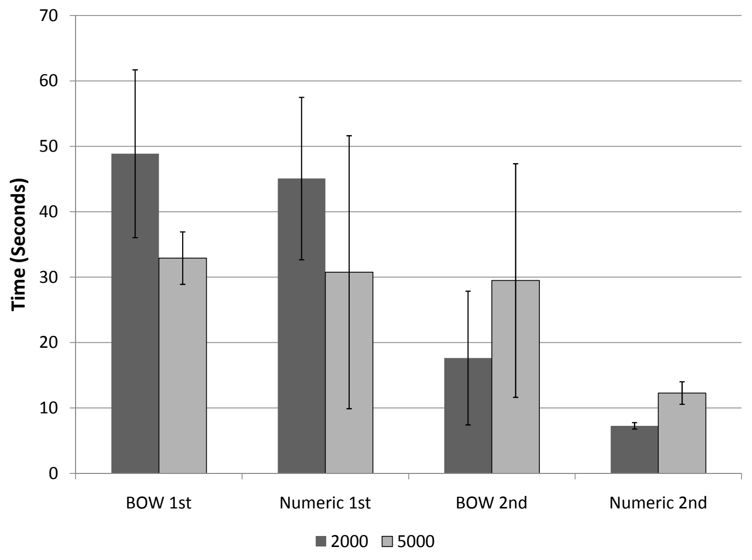


(b) MSE

Figure 6.3: Performance, Ordinal Datasets, with 95% Confidence Interval

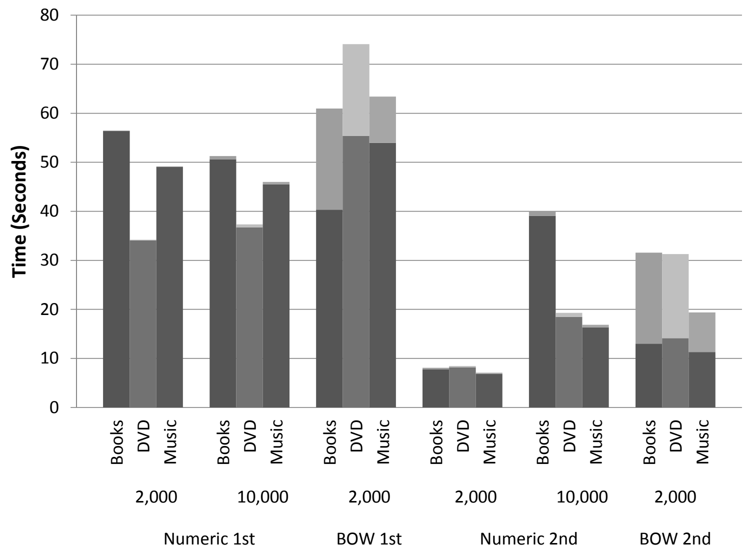


(a) Binary

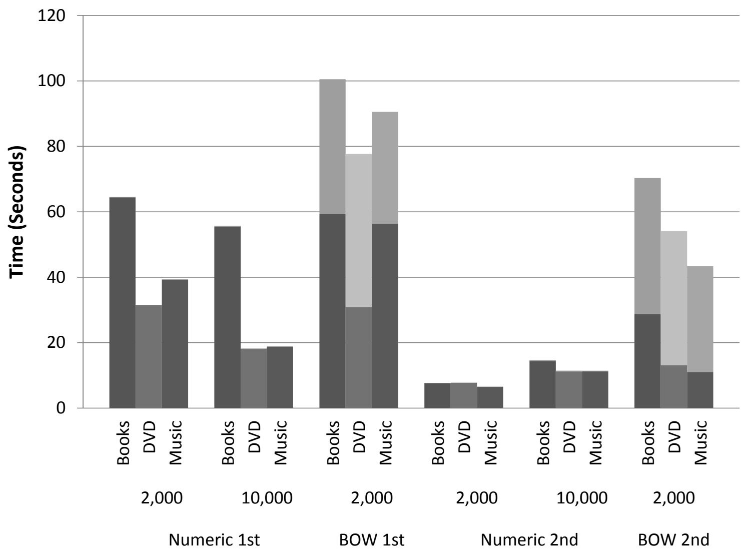


(b) Ordinal

Figure 6.4: Setup Time, with 95% Confidence Interval

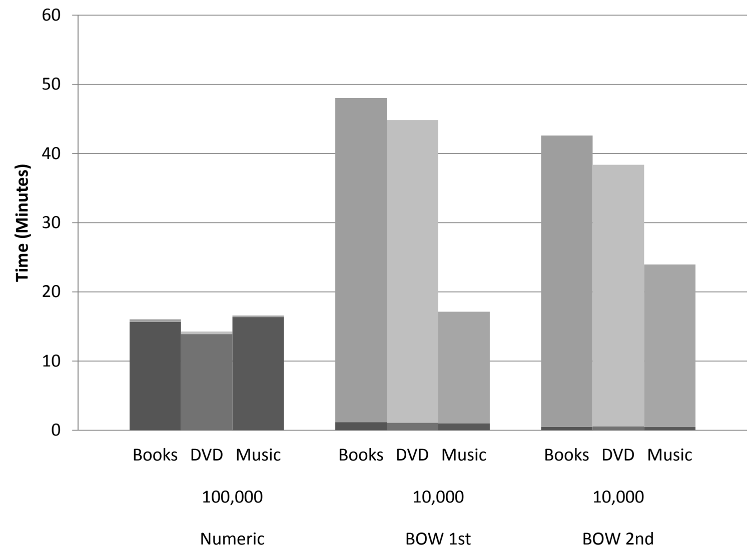


(a) Binary

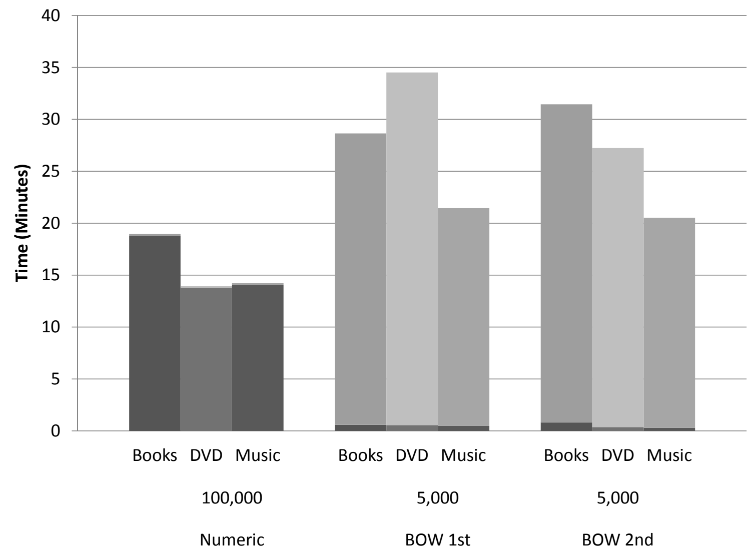


(b) Ordinal

Figure 6.5: Time, Faster Classifiers: The darker portion of the bars is the setup time, while the lighter portion is the classifier time



(a) Binary



(b) Ordinal

Figure 6.6: Time, Slower Classifiers: The darker portion of the bars is the setup time, while the lighter portion is the classifier time

Table 6.8: Ordinal Movie Reviews, BOW vs. Numeric, Accuracy, MSE, and Time

Author	Type	MSE		Accuracy		Time (h:mm:ss)
Schwartz	Numeric	0.580	± 0.013	0.518	± 0.009	0:01:23
	BOW	0.691	± 0.020	0.510	± 0.009	0:22:00
Berardinelli	Numeric	0.478	± 0.010	0.557	± 0.008	0:02:28
	BOW	0.443	± 0.012	0.644	± 0.007	0:59:24
Renshaw	Numeric	0.634	± 0.016	0.468	± 0.011	0:01:45
	BOW	0.696	± 0.020	0.496	± 0.009	0:22:19
Rhodes	Numeric	0.490	± 0.010	0.566	± 0.007	0:02:45
	BOW	0.478	± 0.011	0.609	± 0.006	1:12:59

However, the difference in the overall training time, shown in Figures 6.5 and 6.6 is striking when we move past 2,000 documents. Setup time for both methods is similar with the same amount of data, however, the classifier time for the numeric method is trivial while it grows very quickly for the BOW method. In the ordinal case, the numeric classifier trains on 100,000 documents in approximately 1/3 the time the BOW method takes to train on 2,000, while in the Binary case, the BOW method on 2,000 documents is only slightly faster.

6.1.3 Ordinal Movie Reviews

The Pang Scale dataset [37] consists of automatically determined subjective extracts from professional movie reviews by 4 different authors; one author, Rhodes, was used to determine the parameter settings, while the others were not.

Results for this experiment are presented in Table 6.8. In the case of Accuracy, BOW is significantly better in three of the cases, however, on the reviews by Schwartz, the numeric method is better with borderline significance per a T-Test ($p = 0.075$). However, looking at MSE, the numeric features are significantly better for the Schwartz and Renshaw reviews, while BOW is better on the Rhodes reviews with borderline significance ($p = 0.035$). In the case of the Berardinelli reviews, BOW

Table 6.9: Pang Polarity Dataset 2.0

	Accuracy	AUC	Time (h:mm:ss)
Numeric	0.824 ± 0.005	0.896 ± 0.005	0:05:18
BOW	0.850 ± 0.005	0.850 ± 0.005	2:09:35

is significantly better. While the performance results are mixed on this data set, it does show how much faster the numeric method is.

The documents in this dataset are relatively long, well written, by a limited set of authors, and, as mentioned, contain only the subjective elements. It seems likely that one or more of those first three factors makes the problem easier for BOW, or perhaps the numeric method derives more useful information from the objective portions of the text. However, it should be noted, that if it is indeed one of those factors that affects the results here, many of the applications of sentiment analysis involve shorter, less well written texts by a wide range of authors, and the subjectivity extracts are an extra processing step.

6.1.4 Binary Movie Reviews

This test uses the Pang Polarity dataset [36]. Note that this is not the same dataset as the binary split of the ordinal dataset used to determine parameter settings; it is a set of reviews by many authors divided into positive and negative reviews.

In this case, as shown in Table 6.9, the numeric features outperform BOW in terms of AUC but not accuracy. While the numeric features are faster, we note that per classifier trained here this is a difference of 1m 18s to 3s, which while still considerable may not be relevant to an application on this size of data set.

Table 6.10: Movie Review Snippets

	Accuracy	AUC	Time (h:mm:ss)
Numeric	0.760 ± 0.002	0.841 ± 0.002	0:07:40
BOW	0.739 ± 0.002	0.739 ± 0.002	33:12:50

6.2 Short texts

6.2.1 Movie Review Snippets

These snippets were quite challenging to learn considering that this is a balanced binary problem.

Looking at the training data, we notice that some of the snippets use very creative or sophisticated language; there are also several examples of snippets clearly from borderline reviews that would be difficult for a human to place. For example, "effective but too-tepid biopic" comes from a positive snippet, while "interesting, but not compelling." comes from a negative snippet. We came across at least one snippet which was in Spanish.

However, while the numeric method outperforms the BOW method the difference is comparable to that in many other datasets.

6.2.2 Twitter

The Twitter dataset contains a large amount of noisy training data automatically labeled based on the presence of the smileys :) (positive) and :((negative), and a small amount of hand labeled test data. As such it is not possible to perform cross validation for randomized experiments.

The numeric method provides a small boost to accuracy, a larger boost to AUC, and a very substantial decrease in time.

Table 6.11: Twitter Results

	Accuracy	AUC	Time (h:mm:ss)
Numeric	0.759	0.798	0:00:40
BOW	0.741	0.739	3:30:19

Table 6.12: Product Reviews, Ordinal

	MSE		Accuracy		Time (mm:ss)
Numeric	2.635	± 0.076	0.421	± 0.009	0:28
BOW	3.083	± 0.090	0.405	± 0.008	10:28

6.2.3 Product Reviews

This dataset consists of identifying opinions with regards to specific features of a product using a 6 class ordinal scale (-3, -2, -1, 1, 2, 3). There is relatively limited training data, and the classifications are made based on a small amount of text.

The numeric method improves over BOW in both the binary and ordinal cases, as shown in tables 6.13 and 6.12.

6.3 Performance on Other Problems

6.3.1 Detecting Subjective Sentences

This dataset consists of 5000 each of subjective sentences taken from movie reviews and objective sentences taken from movie plot summaries, resulting in a 10000 element set. As previously noted the time required for BOW increases mainly with the number of documents, not with the number of words. The numeric method is superior in all ways to the BOW classifier on this dataset.

Table 6.13: Product Reviews, Binary

	Accuracy		AUC		Time (m:ss)
Numeric	0.770	± 0.007	0.839	± 0.007	0:43
BOW	0.746	± 0.007	0.731	± 0.008	6:30

Table 6.14: Subjective Sentences

	Accuracy		AUC		Time (hh:mm:ss)
Numeric	0.910	± 0.002	0.967	± 0.001	0:07:39
BOW	0.880	± 0.002	0.880	± 0.002	15:24:33

Table 6.15: Congress, by Segment

	Accuracy	AUC	Time (m:ss)
Numeric	0.634	0.675	0:5.6
BOW	0.612	0.606	4:22.2

6.3.2 Detecting agreement in Congressional Debate Transcripts

This dataset links the transcripts of debates surrounding various pieces of legislation with the vote of the speaker on that legislation. The division by 'segment' treats each speech segment independently, while by 'speaker' groups all of a speaker's speeches about a particular piece of legislation together. This dataset provided separate training and test sets, and so this represents training on the training set and testing on the testing set.

One interesting thing to note is that even though the same amount of text is processed in both cases, the BOW classifier takes many times longer to train when the training data is divided into 2740 speech segments rather than 1174 speakers, while the numeric classifier is barely affected by this.

Table 6.16: Congress, by Speaker

	Accuracy	AUC	Time (m:ss)
Numeric	0.695	0.780	0:5.3
BOW	0.654	0.652	0:17.5

Table 6.17: French Sentences, Binary

	Accuracy	AUC	Time (m:ss)
Numeric	0.594 ± 0.011	0.614 ± 0.011	0:30
BOW	0.561 ± 0.011	0.561 ± 0.011	3:14

6.3.3 Detecting Pleasantness of French Sentences

This dataset consists of 702 sentences from a French language newspaper, rated in terms of how "pleasant" they are - effectively, distinguishing good news from bad news.

As shown in Table 6.17, this proved to be a difficult problem for both the numeric and BOW approaches, however, the numeric approach did perform best and much more quickly.

6.4 Conclusions

The experiments indicate that the numeric features perform better than basic BOW features in many, but not all, instances across a wide range of datasets, and do so much faster. This section discusses where the numeric features performed best and worst as compared with BOW.

The numeric features performed better than BOW in all respects on all of the datasets with short texts, both sentiment and not, and on ordinal and binary problems. This covered datasets with a wide range of numbers of documents and quality of writing, and even different languages, from 702 total sentences for French Sentences to 40000 training documents for the Twitter dataset.

In terms of datasets with longer documents, the numeric features performed best in all instances on the smaller sets of Amazon reviews. On the larger Amazon sets, we found that with 1000 and 2000 documents, the best performance was achieved through using the numeric features, while in some instances with 5000-20000 documents, sometimes the BOW features performed best in terms of accuracy and in one instance (Books 15000) in terms of MSE as well. In the Books and DVD datasets in the ordinal case the best overall accuracy at any amount of data was obtained with the BOW features; however, the best overall MSE was from the numeric features. These accuracy differences were less than 2%, while the amount of time taken to train the BOW classifiers was several hours. Our features also performed well on the Congress dataset.

On the document level movie review datasets, the results are mixed. The numeric features perform slightly better on accuracy on one of the authors, and we see the worst relative performance overall on two of the authors. However, we note that the MSE differences are relatively large in favor of the numeric features on two of the datasets, and relatively small in favor of BOW on the other two - on average, MSE prefers the numeric features. In the binary movie reviews, BOW has higher accuracy, and while the numeric features retain their advantage in terms of AUC, it is the smallest gap we see on that measure. These two datasets contain both relatively long and relatively well written material. While not conclusive, it may be that the numeric features are particularly adept at dealing with the shorter, less well written material found in all manner of less formal online discourse including online user reviews.

Chapter 7

Discussion And Conclusions

7.1 Summary

This thesis explored a method of developing a small numeric feature set based on the distribution of word scores for machine learning of various text analysis tasks, in particular sentiment analysis. The problem of evaluating ordinal problems in this type of domain was also considered.

This method was thoroughly tested, comparing its performance to a basic bag of words feature set over a wide range of ordinal and binary datasets. The machine learning algorithms were able to learn from the feature set produced by our method significantly faster than from the bag of words features, particularly with large datasets, and, in addition, the numeric features nearly always had a performance advantage.

Furthermore, because of this speed advantage, these numeric features allow for using datasets which are orders of magnitude larger while still training in minutes. In the experiments, the time required to read in and process the files was the bottleneck, and we feel that a more refined system (both in terms of the software implementation and the hardware available) could greatly improve over our simple implementation for processing large amounts of text files.

7.2 Discussion

7.2.1 Why does this method work?

This work decomposes the problem of learning the sentiment of documents into two parts: scoring the strength of different words based on their distribution in positive and negative documents, and then learning document sentiment based on the distribution of those scores. This produces a compact representation, of around 25 features, compared with thousands for an effective "Bag of Words" approach.

While this decomposition results in the loss of some information – for instance, if two words appearing together in a document is significant – it appears as if the BOW representation may be too sparse for such relationships to be learned meaningfully. It seems as though the machine learning algorithms for the BOW representation are mainly learning which words are significant indicators of sentiment, but are much slower at this than simple word scoring methods. It should be noted that Bag of Words is, of course, a simplification of the original document as well, which loses information on word order.

This idea can be generalized to the concept of learning something about the attributes in a simple way and then using that knowledge to help with more complicated learning approaches. In a slightly less general way, the numeric features group attributes together (in this case, through their scores as determined by the word scoring methods) and the machine learning algorithm learns based on the distribution of the attributes over the groups.

In the case of sentiment analysis, simple word scoring methods can learn the sentiment to a large extent, even when just computed as a simple threshold, but not as well as machine learning approaches.

While we have attempted to apply this feature selection method to other non-text data sets with limited results, we believe that this exact approach would only work in limited contexts, where the attributes behave similarly to words, in that there is a large number of sparse attributes and the moderately infrequent ones are most useful.

7.2.2 Ratio of word count to document count

The idea of trying to filter out less useful words was one avenue that this research explored. While this did not improve performance, an interesting finding is that words which appear, on average, several times in the documents in which they do appear do not seem to be good indicators of sentiment. However, sometimes they receive a strong score. This includes non-relevant stop words ("the" is virtually neutral and occurs the most times per document), and subjects whose reviews are skewed one way or another. In the books dataset, this includes references to well known authors ("twain", "tolkien", "chrichton" are all very positive) as well as topics or genres ("iraq", "poem", "diet" all appear about 3 times per document, and are moderately positive, very positive, and moderately negative respectively). The words "pan" and "cash" were also both highly positive and appeared close to 3 times per document; it turns out that "pan" is from references to "Peter Pan" as well as "pan-something", while "cash" referred to the person W.J. Cash.

This helps to explain why presence has been found to perform better than frequency in BOW sentiment analysis; a word being more frequent does not make it more meaningful to the overall sentiment of the document, while it may help indicate the topic.

7.3 Future Work

While this thesis tested 3 different word scoring techniques, there are undoubtedly many other possibilities. One particularly interesting avenue would be the application of unsupervised or lightly supervised word scoring techniques to make use of unlabeled data.

One idea which we did not have time to fully explore is to learn word scores from either a broad range of datasets or a more targeted selection for which there is lots of labeled data available, and adapt it to a new dataset with limited training data through the process described below. This may also be promising when combined with machine learning algorithms that can be adapted from one dataset to another related dataset - and it is much easier to use the numeric representation than a BOW representation for this sort of adaptation.

1. Generate word scores for each of n similarly sized domains individually.

2. Combine the counts from the n domains to create a master list of word scores, with constraint that a word must appear in at least d domains (to simplify computation, we do not also compute a document threshold). Retain counts.

3. Remove "domain dependent" words whose polarity goes from positive to negative or vice versa with a difference of greater than t (e.g. if $t = 0.1$, discard a change from 0.55 to 0.44) between the master list and any of the sub lists. Possibly define t in terms of bin increments.

4. Count words in a small set of documents from the target domain. Weight counts by w, and combine weighted counts with master counts to generate new scores for words. e.g. a word appears 10 times in positive documents and 5 times in negative documents in the master list, while it appears 3 times in positive documents and 1 time in negative documents in the new domain, and our weight

is 5. The counts are then computed as $10+3*5 = 25$ positive and $5+1*5 = 10$ negative.

As mentioned earlier in this work, this feature extraction method could replace the BOW component to many of the better systems for sentiment analysis and related tasks, resulting in a much faster combined system.

7.4 Conclusions

This thesis has shown that it is possible to greatly condense the features used for machine learning of sentiment analysis and other related tasks for large speed improvements. Second, we have shown that these features often improve performance over a simple BOW representation, and when they do perform worse than BOW it is not a large amount by MSE. These speed improvements make it possible to process data sets orders of magnitude larger than previously attempted for sentiment analysis, which in turn generally leads to further performance improvements.

Bibliography

[1] Alina Andreevskaia and Sabine Bergler. When specialists and generalists work together: Overcoming domain dependence in sentiment tagging. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, 2008.

[2] Anthony Aue and Michael Gamon. Customizing sentiment classifiers to new domains: A case study. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, 2005.

[3] Yves Bestgen, Cédrick Fairon, and Laurent Kevers. Un baromètre affectif effectif. In *Actes des 7es Journées internationales d'Analyse statistique des données textuelles*, 2004.

[4] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, 2007.

[5] Remco R. Bouckaert. Chosing between two learning algorithms based on calibrated tests. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[6] Rich Caruana and Alexandru Niculescu-Mizil. Data mining in metric space: An empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD04)*, Seattle, Washington, USA., August 2004.

[7] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: Opinon extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, 2003.

[8] Harmen J. Dikkers. Support vector machines in ordinal classification. Master's thesis, Delft University of Technology, August 2005.

[9] Xiaowen Ding, Bing Liu, and Philip S. Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the international conference on Web search and web data mining*, 2008.

[10] Chris Drummond. Machine learning as an experimental science (revisited). In *Proceedings of the Twenty-First National Conference on Artificial Intelligence: Workshop on Evaluation Methods for Machine Learning*, 2006.

[11] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[12] C. Ferri, J. Hernndez-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30:27–38, 2009.

[13] Eibe Frank and Mark Hall. A simple approach to ordinal classification. Technical Report 01/05, Department of Computer Science, University of Waikato, 2001.

[14] Michael Gamon and Anthony Aue. Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In *Proceedings of*

the ACL-05 Workshop on Feature Engineering for Machine Learning in Natural Language Processing, 2005.

[15] Lisa Gaudette and Nathalie Japkowicz. Evaluation methods for ordinal classification. In *Proceedings of the twenty-second Canadian Conference in Artificial Intelligence (AI'2009)*, 2009.

[16] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment. `http://twittersentiment.appspot.com`, June 2009.

[17] Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. Large scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media*, 2007.

[18] Andrew B. Goldberg and Xiaojin Zhu. Seeing stars when there arent many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, 2006.

[19] Ali Harb, Michel Plantié, and Gerard Dray. Web opinion mining: How to extract opinions from blogs? In *International Conference on Soft Computing as Transdisciplinary Science and Technology*, 2008.

[20] Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181, Morristown, NJ, USA, 1997.

[21] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2004.

[22] Daisuke Ikeda, Hiroya Takamura, Lev-Arie Ratinovy, and Manabu Okumura. Learning to shift the polarity of words for sentiment classification. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP-08)*, 2008.

[23] Maojin Jiang and Shlomo Argamon. Finding political blogs and their political leanings. In *Text Mining 2008, workshop at the SIAM International Conference on Data Mining*, 2008.

[24] Thorsten Joachims. A statistical learning learning model of text classification for support vector machines. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 128–136, New York, NY, USA, 2001. ACM.

[25] Alistair Kennedy and Diana Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 32(2):223–262, June 2006.

[26] Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *Proceedings of the 20th international conference on Computational Linguistics*, 2004.

[27] Andrew Lacey. A simple probabilistic approach to ranking documents by sentiment. In *Proceedings of the Class of 2005 Senior Conference*. Swarthmore College, 2005.

[28] Dong (Haoyuan) Li, Anne Laurent, Mathieu Roche, and Pascal Poncelet. Extraction of opposite sentiments in classified free format text reviews. In *Proceedings of the 19th international conference on Database and Expert Systems Applications*, 2008.

[29] Shoushan Li and Chengqing Zong. Multi-domain sentiment classification. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies, Short Papers*, 2008.

[30] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* Massacheusetts Institute of Technology, 1999.

[31] Yi Mao and Guy Lebanon. Sequential models for sentiment prediction. In *Proceedings of the ICML Workshop on Learning in Structured Output Spaces*, 2006.

[32] Justin Martineau and Tim Finin. Delta TFIDF: An improved feature space for sentiment analysis. In *Third AAAI Internatonal Conference on Weblogs and Social Media*, 2009.

[33] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

[34] Tony Mullen and Robert Malouf. A preliminary investigation into sentiment analysis of informal political discourse. In *Proceedings of the AAAI Workshop on Analysis of Weblogs*, 2006.

[35] Tetsuya Nasukawa. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, 2003.

[36] Bo Pang and Lillian Lee. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 271, Morristown, NJ, USA, 2004. Association for Computational Linguistics.

[37] Bo Pang and Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[38] Bo Pang and Lillian Lee. *Opinion mining and sentiment analysis*, volume Vol. 2 of *Foundations and Trends in Information Retrieval*. Now, 2008.

[39] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[40] Foster Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML'98)*, 1998.

[41] Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. Feature subsumption for opinion analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, 2006.

[42] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, 2003.

[43] Fabrizio Sebastiani. Text categorization. In *Encyclopedia of Database Technologies and Application*, pages 683–687. 2005.

[44] Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. *The General Inquirer: A Computer Approach to Content Analysis.* MIT Press, 1966.

[45] Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (Corrected December 2006)*, 2006.

[46] Peter Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, 2002.

[47] Willem Waegeman, Bernard De Baets, and Luc Boullart. On the scalability of ordered multi-class ROC analysis. *Computational Statistics & Data Analysis*, 52:3371–3388, 2008.

[48] Willem Waegeman, Bernard De Baets, and Luc Boullart. ROC analysis in ordinal regression learning. *Pattern Recognition Letters*, 29:1–9, 2008.

[49] Janyce Wiebe, Theresa Wilson, and Matthew Bell. Identifying collocations for recognizing opinions. In *Proceedings of the ACL 01 Workshop on Collocation*, 2001.

[50] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[51] Hong Wu, Hanqing Lu, and Songde Ma. A practical SVM-based algorithm for ordinal regression in image retrieval. In *Proceedings of the eleventh ACM international conference on Multimedia (MM 03)*, 2003.

[52] Y. Y. Yao. Measuring retrieval effectiveness based on user preference of documents. *Journal of the American Society for Information Science*, 46:133–145, 1995.