AN EVALUATION OF MACHINE LEARNING ALGORITHMS FOR TWEET SENTIMENT ANALYSIS

by

Joseph D. Prusa

A Thesis Submitted to the Faculty of

The College of Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, FL

August 2015

AN EVALUATION OF MACHINE LEARNING ALGORITHMS FOR TWEET SENTIMENT ANALYSIS

by

Joseph D. Prusa

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Taghi M. Khoshgoftaar, Department of Computer and Electrical Engineering and Computer Science, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

SUPERVISORY COMMITTEE:

Taghi M. Khoshgoftaar, Ph.D.
Thesis Advisor

Mehrdad Nojoumian, Ph.D.

Nurgun Erdol, Ph.D.
Chair, Department of Computer and Electrical Engineering and Computer Science

Dingding Wang, Ph.D.

Mohammad Ilyas, Ph.D.
Dean, The College of Engineering and Computer Science

Deborah L. Floyd, Ed.D.
Dean, Graduate College

7/15/15
Date

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to Dr. Taghi M. Khoshgoftaar, my thesis advisor. His guidance, encouragement and mentorship have aided me throughout my tenure as Master's student. His support of my past and current research efforts has been instrumental to my growth as a student and researcher. Additionally, I would like to thank my thesis committee members, Dr. Mehrdad Nojoumian and Dr. Dingding Wang.

I would also like to thank Mr. Brian Heredia for his help with the preparation of my thesis, and Dr. David J. Dittman for his help and guidance as I have learned to write research and survey papers. Additionally, I would like to thank all members in the Data Mining and Machine learning Laboratory of Florida Atlantic University. Finally, I would like to thank my parents for their love, support and encouragement throughout my studies.

ABSTRACT

Author:	Joseph D. Prusa
Title:	An Evaluation of Machine Learning Algorithms for Tweet Sentiment Analysis
Institution:	Florida Atlantic University
Thesis Advisor:	Dr. Taghi M. Khoshgoftaar
Degree:	Master of Science
Year:	2015

Sentiment analysis of tweets is an application of mining Twitter, and is growing in popularity as a means of determining public opinion. Machine learning algorithms are used to perform sentiment analysis; however, data quality issues such as high dimensionality, class imbalance or noise may negatively impact classifier performance. Machine learning techniques exist for targeting these problems, but have not been applied to this domain, or have not been studied in detail. In this thesis we discuss research that has been conducted on tweet sentiment classification, its accompanying data concerns, and methods of addressing these concerns. We test the impact of feature selection, data sampling and ensemble techniques in an effort to improve classifier performance. We also evaluate the combination of feature selection and ensemble techniques and examine the effects of high dimensionality when combining multiple types of features. Additionally, we provide strategies and insights for potential avenues of future work.

AN EVALUATION OF MACHINE LEARNING ALGORITHMS FOR TWEET SENTIMENT ANALYSIS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The internet has greatly changed how we conduct commerce, consume media, share ideas, and interact with each other. The rise of social media is particularly noteworthy as it has helped to democratize the internet by allowing individuals to freely express and share their opinion, circumventing the need to run ones own website. One common method of sharing ideas, opinions and experiences is through posting microblogs short posts consisting of text, images and links on sites such as Facebook and Twitter. Many of these microblogs are publically available and may contain a users thoughts about a specific topic. The text mining of microblogs can be used to conduct social sensing and opinion mining. By collecting a large numbers of posts relating to a topic of interest and performing sentiment analysis (any of a number of methods that can determine the emotional polarity of a text passage) a statement can be made about the publics opinion on that topic [29].

In its simplest form text sentiment refers to whether a tweet is identified as positive or negative in tone. More complicated classification schemes may have more sentiment categories, such as the addition of neutral sentiment, attributing multiple sentiments to a single instance, or have different polarities of sentiment instead of binary classification. Twitter is a popular social media microblog and an excellent domain for social sensing as it has over 270 million active monthly users who send over 500 million tweets (individual posts on Twitter) per day . Twitter also provides APIs for mining tweets, facilitating the mining of tweets by researchers and app developers. Social sensing of twitter has a myriad of uses, such as predicting elections, helping companies understand the needs of consumers, or prediction of economic trends.

Machine learning algorithms train a classifier to recognize different classes of sentiment and are then used to categorize unseen tweets into a class of sentiment based on features extracted or constructed from the tweet. Once trained a classifier facilitates sentiment analysis of potentially millions of tweets. This provides a powerful tool that can be used in any situation where it is of benefit to understand public opinion, and has been demonstrated to be of use or have potential in a variety of tasks. Prediction of elections [48], product sale [30], movie box office performance [31, 7], and stock market [33, 11, 31] are some of the possible areas where sentiment analysis is of use. For the purpose of prediction, sentiment may be used alongside measures of a topics popularity on twitter. Asur et al. [7] constructed a linear regression model with three parameters to describe a movies popularity on Twitter i.e., the ratio of positive and negative tweets about the movie and a final parameter to account for the movies availability at theaters. Asur et al. claimed their model outperformed other models (such as the Hollywood stock exchange) and demonstrated that sentiment improved predictive performance, especially for days after the initial release.

Machine learning algorithms have been effective in classification of text sentiment in multiple domains, such as movie reviews; however their application to Twitter is an ongoing process due to Twitter having many domain specific problems unique to its format and use. tweets present a challenge when considering features due to their brief nature, as they are limited to 140 characters in length. Additionally the short length is accompanied by colloquialisms, informal and unique language constructs, misspelled words, and entities related to the domain such as hashtags. Go et al. [23] found that the presence of unorthodox spellings and the short text length of tweets results in significantly lower term frequency for words in tweets compared to words in other microblogs such as review forums. Due to the unique nature of tweets, compared to other web-mining domains, current research in the Twitter domain has been primarily concerned with the engineering of different types of features, and the

implementation of different machine learning algorithms. However, there are many other problems such as high dimensionality (the high volume of tweets being used to train classifiers results in the presence of thousands of unique words, many of which may only be contained in a few tweets), noisy data, and class imbalanced data that need to be addressed in an effort to improve classification performance.

1.1 MOTIVATION

Feature selection is a common tool used in machine learning for reducing high dimensionality. High dimensional data means that a very large number of features exist to describe the instances. Often, many of these feature are irrelevant or redundant [17] and may degrade classifier performance due to overfitting. The goal of feature selection is to choose an optimal subset of features by eliminating features that are irrelevant or offer no additional information compared to features within the optimal subset. This reduces the dimensionality of the data, which has the benefit of reducing the computational time associated with tasks such as training a classifier, and can also improve classifier performance. Tweet sentiment classification is a domain where application of feature selection techniques is highly desirable, since common feature engineering methods can result in tens of thousands of features being extracted (as seen in the works of Go et al. [23] and Saif et al. [44]). In combination with the potential need to classify very large numbers of instances (millions of tweets are made each day) the high dimensionality of tweet data can result in the training of classifiers and subsequent classification of new tweets to be computationally expensive. Feature selection techniques dramatically reduce the dimensionality of data, reducing the computational resources needed to train a classifier and perform sentiment classification on new tweets. Additionally feature selection can be used for knowledge discovery. Looking at top features can inform the person responsible for feature engineering of what features should be focused on in future extractions [42].

Imbalanced data, or skewed data, is data in which there is a majority and minority class. Often the minority class is of more interest, but classifiers trained on the data favor the majority class. Many tweet sentiment classifiers have been trained using data with a balanced class distribution. This may be a good representation of some real world applications, such as investigating tweets relating to a close election, but would be a poor representation of sentiment surrounding a natural disaster such as a hurricane. The presence of class imbalance in tweet sentiment datasets should not be a surprise; it follows the observation made by Forman [20] that most text classification problems have substantially skewed class distributions which can impact classifier performance. He also observed that conducting feature selection to reduce dimensionality also could improve performance on skewed datasets. Class imbalance can also be addressed by using techniques such as data resampling to generate data with a more favorable class distribution, or by using ensemble learners which make use of data diversity to construct classifiers more robust than those generated with a single learner [17].

A final data concern is that labeled training data may be noisy. Collecting tweets to train classifiers presents additional challenges for producing an accurate classifier due to the difficulty of procuring a large corpus of accurately labeled tweets. Common solutions are the use of crowd sourcing to have multiple parties manually label large numbers of instances, or the automated labeling of tweets using emoticons (as seen in Go et al. [23]). Unfortunately neither solution has any way to verify the quality of the data. The resulting training set is potentially noisy, having instances with mislabeled classes. Abbasi et al. [1] found noise could have a severe impact on classifier performance in the related domain of web forum sentiment classification; however noise has received little attention for tweet sentiment classification.

One of the main tasks of performing tweet sentiment classification using machine learning algorithms is designing and extracting features from the data. Tweets are

limited to 140 characters, thus choice feature extraction methodology has a strong impact on classifier performance. Research on feature engineering has explored a variety of methods for extracting features; either directly from tweet text, or constructing features using additional resources such as natural language processing APIs, an external sentiment lexicon, or other outside libraries. This can be used to provide features for a tweet not directly found in the text of a tweet. Additional types of features can be extracted by using natural language processing to identify more complicated features such as semantic concepts and topics or parts of speech. It should be noted that feature engineering is distinct from feature selection, but when extracting features simple feature selection techniques are often implemented to limit data dimensionality.

1.2 CONTRIBUTIONS

The sentiment140 corpus and SemEval dataset both exist as labled tweet sentiment datasets available for use by researchers. We created seven datasets from sentiment140 corpus and three from SemEval task 10b dataset. These datasets were constructed with the purpose of testing the impact on tweet sentiment classifiers of three different types of machine learning techniques: feature selection, data sampling and ensemble techniques. Research on these techniques for this application domain is either lacking or inconclusive. This thesis focuses on the application of these techniques to tweet sentiment data and the effects they have on classifier performance. The contributions of this thesis are:

1. A study of the impact of feature selection on tweet sentiment data using unigram features, tested using four classifiers, ten filter-rankers and ten subset sizes.

2. The effect of data sampling for eight sentiment classifiers, trained using different levels of class imbalance ranging from 25% to 1%.

3. How bagging and boosting impact seven base learners when trained on tweet

sentiment data.

4. A comparison of how feature selection, data sampling and ensemble techniques impact classifier performance, determining which techniques are most important on balanced and imbalanced data.

5. The effectiveness of combining both feature selection and ensemble techniques (bagging and boosting) compared to only using feature selection.

6. A study on the impact of adding emoticon features to unigram features, the effects of high dimensionality in this situation, and how the introduction of feature selection impacts combining multiple types of features.

1.3 ORGANIZATION

The remainder of this thesis is organized into the following chapters

- Chapter 2 provides a background on tweet sentiment classification and discusses related work.

- Chapter 3 provides details on the feature selection techniques, data sampling techniques, ensemble techniques, datasets and classifiers used in our studies.

- Chapter 4 presents six case studies using the methodology described in chapter 3.

- Chapter 5 presents the conclusions of our case studies and discussion of future work.

CHAPTER 2

RELATED WORK

The focus of this thesis is on the application of feature selection, data sampling and ensemble techniques to tweet sentiment classifiers; however, the majority of research in this domain has focused on feature engineering and it is important that background is provided on the types of features that may be extracted and research on different methods of feature extraction.

2.1 FEATURE ENGINEERING

Feature engineering (or extraction) is one of the main tasks when performing tweet sentiment classification using machine learning algorithms, since algorithms require features describing instances so that they can learn patterns. As previously mentioned, tweets are limited to 140 characters, naturally choosing what to extract from this short text as features has a strong impact on classifier performance. Numerous methods of feature engineering have been explored creating a variety of methods for extracting features; either directly from tweet text, or constructing features using additional resources such as natural language processing APIs, an external sentiment lexicon, or other outside libraries. This can be used to provide features for a tweet instead of using the actual words or other in text features. Additional types of features can be extracted by using natural language processing to identify more complicated features such as semantic concepts and topics or parts of speech. Feature engineering is distinct from feature selection; however, when extracting features rudimentary heuristic approaches of feature selection are often implemented to limit data dimen-

sionality. These mechanisms generally follow the methods outlined by Forman [20] where features below a minimum frequency are removed prior to implementing more sophisticated feature selection.

The most common types of features are those immediately visible in the text, and generally consist of n-grams. N-grams are sequences of words from the text of length n. Within the domain of tweet classification unigrams (single words as features, also called bag-of-words) are the most frequently used type of n-gram, while higher n-grams, such as bigrams and trigrams (n-grams of two or three words), have received less attention. When discussing how they chose to build features Go et al. [23] suggested that due the short length of tweets longer expressions are of little value. For sentiment classification the use of n-grams as features is sensible starting point as it is expected that words in an emotional tweet reflect the tweets sentiment. N-grams are extracted by parsing text into single words in the case of unigrams, or short combinations of words for bigrams and trigrams. They are then made into binary features by creating a feature vector representing each n-gram as a Boolean object. If an instance contains a given n-gram the corresponding entry in the feature vector reads true, if it is absent it reads false. As previously mentioned, use of n-grams as features results in a high dimensional feature space that grows with training set size. For a large training set the resulting feature vector can potentially contain upwards of 10,000 n-grams (approximately 36,000 for the dataset constructed by Go et al. [23]). In an effort to address the high dimensionality and sparseness of the feature space produced by n-grams, rules based on term frequency may be implemented. Such rules may consist of only extracting features meeting a minimum term frequency throughout all instances, or taking the top K most frequent features, where K is the desired number of features. This approach follows the logic that features appearing in only a few instances are not particularly informative. The problem is that if we extract K features based on the most frequent unigrams there is no guarantee the extracted

features are the K most informative unigrams. Additionally using unigrams (also known as a bag-of-words) has been known to result in over-fitting [46]. Other methods seek to address the variety of spellings exhibited by Twitter users. Approaches demonstrated by Asiaee et al. [6] and Go et al. [23] consist of performing text filtering through letter replacement or deletion, use word lemmatization (finding/replacing a word with its "lemma" or canonical dictionary form), and decomposition of hashtags into their component words in an effort eliminate redundant words from the feature vector.

While the above methods of limiting features may help achieve manageable computational complexity they also result in a loss of information due to excluding features. This may negatively impact performance. An example is the elimination of words believed to either contain no value or be too common for use in classification. Removal of these words, called stop-words, has been demonstrated to be of benefit in other Natural Language Processing domains, but hurt tweet sentiment classification. As a precursor to their main study on semantic features seeking, Saif et al. [44] conducted an experiment using Naïve Bayes and unigram features to measure the effects of removing stop words when conducting sentiment classification. Classifier accuracy for both types of feature sets was measured across three datasets, the Stanford Twitter Sentiment corpus and two with specific topics generated for the experiment (Health Care Reform and Obama-McCain debate). Removal of stop-words resulted in a decrease in accuracy of around 3% across all data sets. They speculated that this is due to their stop-word lists containing important words relating to negation.

Due to the limitations of n-grams, particularly the high dimensionality and sparseness, researchers have developed alternative approaches to engineer features to describe tweets. These methods are more complicated in their construction, often requiring external tools to aid in preparing datasets, but offer potential reductions in the computational cost of training a classifier or improvements in classifier performance by

reducing dimensionality and data sparseness. More advanced morphological features that have been considered as features include part of speech (POS) tags, sentiment-topic, semantic, word polarities (sentiment lexicon features [19]), and Twitter specific features (often a list of specific punctuations, words or emoticons [27]). Another potential source of Twitter specific feature are hashtags, words proceeded by # in a tweet, used to tag the topic(s) of the tweet. POS tagging classifies words as being a particular part of speech, such as a noun, verb or adjective [22]. Sentiment-topic features are implemented in a similar fashion as word lemmatization. An external tool is used to parse text and determine the closest match to a word, but instead of matching words against a canonical form they are grouped into higher level concepts with the goal of reducing data sparseness [44]. Semantic features refer to concepts associated with a word. The disadvantages of using these types of features stem from their implementation (replacing n-grams is a loss of data while being used in conjunction with n-grams increases the size of the feature set) and novelty their construction is often unique to a particular research group making it difficult to establish benchmarks [44].

The effect of using different types of features has been extensively studied, frequently using unigram features as a baseline method. N-grams, sentiment lexicon features, morphological features and semantic features have been investigated for their use in augmenting or replacing unigrams. Go et al. [23] compared unigrams, bigrams and part of speech tagging. Using a collection of tweets with automatically labeled sentiments (using emoticons), four training sets with different features were prepared: the first with only unigram features, the second with only bigram features, the third with both unigram and bigram features, and the final set with unigrams as features augmented with POS tags. They trained classifiers using three learners, Naïve Bayes (NB), Max Entropy (MaxEnt), and Support Vector Machine (SVM). Testing each classifier they found that there was no best or worst feature set for all learners. They

also found that the performance difference between feature sets for a single learner was not statistically different than using unigrams. Their results showed that no feature set yielded higher performance for all learners, and that no learner is superior for all feature sets. The addition of POS tags had a uniformly negative performance impact compared to unigrams. They conclude POS tags and larger n-grams are of little to no value in the domain of tweet classificaion. Instead classifier performance can be improved through consideration of semantics, training domain specific classifiers, and the possible use of emoticons as features. Their findings matched results of a similar experiment conducted by Pang and Lee [34] on IMDb (also known as Internet Movie Database) user movie reviews. Once again four training sets with different feature spaces were constructed using unigrams, bigrams, unigrams and bigrams, and unigrams and POS tags. Comparing performance across NB, MaxEnt and SVM found no clear improvement over unigrams by any of the alternative feature sets for the movie reviews. A major limitation of the study conducted by Go et al. is that only one dataset was considered and performance was evaluated using a single performance metric, accuracy. Testing each method of feature engineering across multiple datasets and evaluating performance with multiple metrics could provide a more conclusive result. Implementing additional learners would also be of benefit. Saif et al. [44] conducted an experiment to compare semantic features against unigrams, unigrams and POS tags, and sentiment-topic features. Semantic features were used to replace, augment or interpolate words in a tweet with the overall semantic concept of the word. It was determined that interpolation with unigrams was the best implementation of semantic features. These features were constructed using the Alchemy API , a cloud platform machine learning tool. Sentiment/topic features were generated using the Joint Sentiment/Topic (JST) Model developed by Lin and He [43] to group words of similar sentiment and topic. For POS tagging Twitter NLP and POS tagger were used. Each feature set was tested on three data sets using NB as a learner.

For all three data sets Sentiment-Topic features and semantic feature interpolation outperformed unigrams and unigrams plus POS tags, improving F-measure by 4-9% depending on the dataset. Semantics outperformed sentiment-topic features for the Stanford Twitter Sentiment Corpus (STS), while sentiment-topic features yielded higher performance for specific topics (in their experiment a data set on health care reform and a second on the Obama-McCain debate). The performance increase due to incorporating either sentiment-topic or semantic features was varied with dataset size. Additionally only one learner was used, possibly because other learners were not economical due to the presence of 37,054 features. Both methods of feature extraction should be tested on a wider variety of datasets using multiple learners, incorporating feature selection to reduce data dimensionality if needed.

A third experiment by Kouloumpis et al. [27] compared n-gram features, lexicon features, POS features and microblogging features. Microblogging features consist of emoticons, hashtags, specific abbreviations and intensifiers such as all-caps or character repetition. They tested different combinations of features on two datasets (one with and one without emoticons) constructed from three different corpora (Edinburgh Twitter corpus, tweetsentimentappspot.com and iSieve) using AdaBoost.MH, a multiclass version of the boosting algorithm AdaBoost, with 500 iterations. They found using microblogging features in combination with unigrams and lexicon features resulted in the highest F-measure, approximately 10% higher than unigrams alone for both datasets. Microblogging features appear to be the most useful addition; however it is difficult to assess their impact as they are never tested with only unigrams. In same experiment Kouloumpis et al. found using all types of features to be inferior to using only unigrams, lexicon and microblogging features, and using unigram and POS features is better than using unigram and lexicon features. Additionally adding lexicon features to the unigram baseline decreases F-measure for the dataset containing emoticons. Based on how adding lexicon features in other instances within the

experiment decreased performance, it could be expected that their best model would be outperformed by using unigram and microblog features or unigram, microblog and POS features. While their study demonstrated that microblogging features positively impact classifier performance, the extent of their impact is unclear. Agarwal et al. [4] also investigated the impact of tweet specific features. They created a set of 100 "senti-features" through a combination of POS tags, specific word polarities, emoticons, and hashtags. Using a manually annotated dataset of 5127 tweets (split evening into positive, negative and neutral tweets) they tested both binary (positive negative) and multi-class classification using a unigram baseline, the senti-features alone and the senti-features in combination with unigrams. They found that using the senti-features alone to yield similar performance to the unigram baseline, despite using only 100 features compared to over 10,000 unigrams. Using both together performed 4% better than either alone. Looking at the improvements due to the different component types within the senti-features they found that most of the improvement was due to the addition of POS tags and that twitter specific features improved classifier performance by less than 1%. Their results do not match the observations of Kouloumpis et al. [27], who found twitter specific features to be the most beneficial type of feature for improving a unigram baseline model. Their results also showed that a small feature set could achieve similar performance to a large feature set. This may indicate that most unigram features contribute little or no useful information for sentiment classification. The relative performance improvement observed when combining both types of features is fairly small, this indicates there is likely overlap in the information given by both unigram and senti-features. Feature selection could eliminate overlapping features and remove useless unigram features, possibly improving classifier performance.

While it has been demonstrated that the unigram baseline method can be improved upon by augmentation or transformation of the feature space the benefits of

these actions across multiple tweet domains is unclear. The main difficulty in determining if a particular feature space is superior to another is the lack of consistent results across different datasets, and that most experiments use different data sets and/or feature extraction methods, resulting is different feature spaces.

One problem is that unigram feature sets vary wildly in size between experiments and datasets. Some experiments use all available unigram features, while other experiments impose limits on the number of unigrams considered. Go et al. [23] is an example of the former, using over 36,000 features. Kouloumpis et al. [27], the later, as they selected the top 1000 most frequent features for use in classification. The variety of approaches can be problematic within the context of a single experiment. Saif et al. [44] used 37,054 unigrams to construct a baseline classifier for STS data and found adding addition types of features (semantic features resulted in the greatest gain) increased F-measure by 4%. However for the other two data sets only 2,060 and 2,464 unigrams were extracted. Adding additional features (either sentiment-topic or semantic) resulted in F-measure being increased by 5% to 9%, with the 9% gain being on the set with the least unigram features. In this case adding additional features to a smaller feature set has a greater impact than adding additional features to a larger feature set. This complicates comparison of feature extraction methods. Using a very large number of features can also be computationally expensive; this is a problem that can be addressed through feature selection.

Adding additional types of features, either POS tags or lexicon features, have displayed inconstant results in different studies. Saif et al. [44] found that augmenting unigrams with POS features increased F-measure across three datatsets. Go et al. [23] found their addition to either decrease, or have little to no effect on classifier performance depending on the classifier used. Adding further ambiguity Kouloumpis et al. [27] found that the addition of POS features to unigrams increases performance, but adding POS features to unigrams, word polarities and microblogging features

decreased performance. Sentiment lexicon (word polarity) features also have an inconsistent impact on classifier performance. Kouloumpis et al. found their addition to unigrams to have a modest improvement on one dataset, while this same addition decreased performance when the dataset had been augmented with instances containing emoticons. An additional experiment using sentiment lexicon features was conducted by Agarwal et al. [5] who constructed 100 features from prior probabilities of words expressing a particular sentiment. Addition of these features resulted in a 4% improvement in classifier accuracy over the unigram baseline. It is unclear in many cases if adding additional types of features is beneficial. This is due to different research groups using different methods of extracting features similar features, and also because experiments have only been conducted on a small number of datasets, with a small selection of learners. More conclusive results could be established by expanding studies to include more datasets, and standardizing which natural language APIs, dictionaries, etc. are used for feature extraction.

Sentiment-topic features and semantic features interpolated with unigrams are still of interest as both have been demonstrated to outperform unigrams on multiple datasets, and each appear suited to different domains [44]. Comparison of these two feature sets against unigrams combined with domain specific features is also of interest as all three methods have demonstrated similar performance gains over unigrams, but have not been directly compared. More rigorous studies should be conducted of these types of features for both the purpose of direct comparison and also to address the limited number of datasets used in individual studies and other discrepancies (inconsistent number of unigrams extracted, differences in learner selection, and lack of evaluation using multiple performance metrics).

2.2 ENSEMBLE TECHNIQUES

Bagging and boosting are ensemble techniques that combine multiple learners in an effort to improve classifier performance. Bagging, also known as bootstrap aggregating, uniformly samples data instances with replacement (the same instance can be sampled multiple times). This creates multiple, newly sampled bootstraps of the same size as the original training dataset. Each bootstrap is used to train a classifier using a single base learner. When classifying a new instance the individual classifiers are combined through voting. Bagging has been shown to improve unstable or weak learners, but can degrade the performance of stable learners such as K-Nearest Neighbor [14].

Unlike bagging, which trains multiple classifiers in parallel, boosting creates an ensemble of classifiers iteratively, training and evaluating a classifier in each iteration and altering how the subsequent iteration's classifier is trained based on the previous classifiers performance. One of the most popular families of boosting algorithms is AdaBoost. AdaBoost and its variants have been shown to improve classifier performance for many learners [10].

AdaBoost functions by training and evaluating a new classifier every iteration. On the next iteration, weights are assigned to instances based on the classification results, weighting misclassified instances more heavily. This results in the subsequent iteration being more likely to correctly classify previously misclassified instances. AdaBoost combines the classifiers using weighted voting to classify new instances. Re-weighting is not compatible with some base learners; however, an alternative approach allows boosting to be used with these learners by resampling instances to match the weights assigned to each instance [10]. In this implementation of AdaBoost, data sampling with replacement is used to create a new training dataset where the probability of selecting an instance is proportional to its assigned weight.

The case studies presented in this thesis use AdaBoost.M1, one of many variants of

AdaBoost with data resampling. We use 10 iterations, selected based on preliminary experiments which found using more iterations did not significantly improve classification performance. This is denoted as AdaBoost-r for the remainder of this thesis. We combine bagging and Adaboost-r with feature selection, performed during each iteration of AdaBoost-r or on each bootstrap created by bagging, to create the combined ensemble and feature selection techniques, Select-Bagging and Select-Boost. A framework for each technique can be found in Figures 3.1 and 3.2. Performing Feature Selection alone, with no ensemble technique, is denoted as "None" in all tables and figures in Results.

2.3 HIGH DIMENSIONALITY

Feature selection is a common tool used in machine learning for reducing the dimensionality of a data set. It has received little attention in the domain of tweet sentiment classification despite being a potentially powerful solution to the large number of features extracted for sentiment classification. Forman [20] demonstrated many available feature selection techniques that can be used to reduce dimensionality while improving classifier performance for a wide range of text classification problems. A similar claim was made by Guyon et al. [24] who expressed that this performance increase was in part due to reduction of overfitting. As mentioned earlier feature engineering for tweets can result in tens of thousands of features being extracted (as seen in the works of Go et al. [23] and Saif et al. [44]). Thus, tweet sentiment classification is a domain where application of feature selection techniques is highly desirable.

Feature selection techniques can be broken into several categories: filter based, wrapper based, hybrid filter/wrapper based and embedded feature selection techniques. Filter based feature selection techniques use statistical metrics to select features. They are fast, scalable to large datasets and independent of classifiers [42]. Filter based techniques can be used as rankers or for subset selection. Rankers select

a subset made of the top ranked features, while subset selection evaluates subsets of features to find an optimal subset. This second approach is computationally time intensive as the number of tests grows exponentially with the number of features. Due to the large number of features used for tweet sentiment classification (potentially over 10,000) subset evaluation is not advised. Wrapper based techniques build classification models to evaluate subsets of important features [45]. For every feature subset tested a new classifier must be trained making this method require a prohibitive amount of computational resources as it could require an exponentially large number of classifiers to be trained. Hybrid approaches combine filter and wrapper techniques. Finally, embedded feature selection techniques are associated with specific learners, such as decision tree models, and are not applicable when considering other learners such as Naïve Bayes [42]. Due to the computational complexity or classifier dependence of other techniques, filter based feature selection techniques are arguably best suited for tweet sentiment classification.

Filter based feature selection techniques have been used to reduce dimensionality for tweet sentiment classification; though, only a single technique and its impact on classifier performance has been studied. When comparing the performance of sentiment topic features and semantic features Saif, He and Alani [43] observed that semantic features only offered better performance when trained from larger datasets, which allow more unigram features to be extracted. To investigate this they conducted an experiment using different numbers of features to determine the effect on the performance of NB for sentiment-topic and semantic features. Feature selection was conducted using Information Gain (IG), between 42 and 34,855 features were selected for the comparison. Classifiers were tested on 1,000 instances from the Stanford Twitter Corpus. Saif, He and Alani found that sentiment/topic features offered high performance, above 80% accuracy with only 500 features selected. Semantic features performed poorly up until 30,000 features were selected. At this point classifier

accuracy started to increase with the number of features, eventually surpassing the performance of sentiment/topic features once 34,000 or more features were selected. They concluded that sentiment/topic features are preferred over semantic features, and that sentiment/topic features using only 500 features is not significantly different than using a much larger number of features. IG has also been used for knowledge discovery. Agarwal et al. [4] used IG to select the top 15 unigram feature in an effort to better understand what types of unigrams are valuable for sentiment classification. They found most of the top features have a strong positive or negative polarity; however some neutral words were present.

Apart from the above mentioned experiments, research on the impact of feature selection on tweet sentiment classification is unavailable; however, feature selection techniques in combination with unigrams have been studied in closely related domains. In a classification task similar to sentiment analysis Chamlertwat et al. [15] reported optimal performance for classification of tweets as subjective or objective was achieved by combining SVM with IG. Narayanan et al. [32] conducted an experiment showing the benefit of applying feature selection in the related domain of movie review sentiment classification. Using Naïve Bayes classification with unigram features as a baseline they were able to demonstrate a 15% gain in accuracy through the use of Mutual Information feature selection. It should be noted that this experiments conclusion may not hold for tweets, as they are much shorter than movie reviews. Additionally accuracy may not be the best performance metric for tweet sentiment as real-world data is frequently skewed in class distributions.

A study of filter ranker based techniques for a variety of text classification problems was conducted by Forman [20] using 11 statistical metrics as filter rankers. Each technique was tested on 19 multi-class datasets which represented 229 binary text classification problems. Using SVM as a learner, each feature selection techniques impact on classifier performance was evaluated using F-measure, precision, recall and

accuracy. Forman found there was no clear best metric for all classification problems, but feature selection techniques improved performance while decreasing dimensionality. For class imbalanced data, using F-measure as a performance metric, Bi-normal separation (BNS) had the best performance followed by IG. BNS in combination with Odds Ratio was determined most likely to maximize precision while BNS paired with F-measure (as a ranker, not performance metric) was found most likely to maximize recall.

Current research on feature selection and tweet sentiment classification has shown it to be an important technique for addressing the high-dimensionality of tweet data. Work by Saif, He and Alani [43] highlights the importance of feature selection in the domain of tweet sentiment classification. IG could be used to reduce the feature subset size from over 30,000 features to 500 features without significant impact to classifier performance. This is not sufficient as different feature selection techniques should be compared across a wide range of datasets with multiple learners, subset sizes and performance metrics in an effort to establish best practices for reducing data dimensionality. Additionally combining feature selection techniques and different feature engineering methods is of interest as optimal subset size may vary for different types of features, making some feature engineering methods more desirable than others.

2.4 CLASS IMBALANCE

Imbalanced data, or skewed data, is data in which there is a majority and minority class. Often the minority class is of more interest, but classifiers trained on the data favor the majority class. Many tweet sentiment classifiers have been trained using data with a balanced class distribution. This may be a good representation of some real world applications, such as investigating tweets relating to a close election, but would be a poor representation of sentiment surrounding a natural disaster such as

a hurricane. The presence of class imbalance in tweet sentiment datasets should not be a surprise; it follows the observation made by Forman [20] that most text classification problems have substantially skewed class distributions which can impact classifier performance. He also observed that conducting feature selection to reduce dimensionality also could improve performance on skewed datasets. Class imbalance can also be addressed by using techniques such as data resampling to generate data with a more favorable class distribution, or by using ensemble learners which make use of data diversity to construct classifiers more robust than those generated with a single learner [17].

Hassan et al. [25] approached addressing class imbalance by using Bootstrap aggregation with stepwise iterative model selection. This model trains many diverse models by altering three parameters — datasets, features and classifiers — generating a diverse set of learners. Subsets of the resulting models are tested to find an optimal classifier. The resulting model was found to improve classifier accuracy by 3% across all datasets compared to the best single model incorporated in the ensemble. This study compares and demonstrates stepwise iterative model selection to be superior to the genetic algorithm, but fails to evaluate the proposed ensemble technique against popular techniques, such as boosting. Hassan et al. also provides little detail regarding what the best single model consisted of and provide no measure of a unigram baseline, making it difficult to compare their performance against other approaches. Use of ensemble learning methods (boosting in combination with either NB or SVM) was used to handle imbalanced data in experiments by Kouloumpis et al. [27] and Silva et al. [46]; however, Kouloumpis et al. made no comparison to non-ensemble methods and neither study compared boosting against other ensemble techniques. Silva et al. found that boosting improved NB, but decreased the performance of SVM, however the limited scope of their study does not allow meaningful conclusion can be made about the benefits of using ensemble learners, necessitating

future work investigating this issue more thoroughly.

Data sampling is a common machine learning technique that alleviates the problems associated with imbalanced data by producing a more balanced dataset. This can be achieved by under-sampling (removing majority instances) or oversampling (adding minority instances). Both techniques can be applied randomly or intelligently by being informed to add or remove instances based on identifiable parameters. Under-sampling can be expected to work well for tweet sentiment as the large volume of instances allows majority class instances to be deleted. Data sampling techniques have been implemented in some studies, but different data resampling techniques have not been compared on imbalanced tweet sentiment data. Acknowledging the potential negative impact class imbalance can have on classifier performance Li et al. [28] used Random under-sampling (RUS) to create a roughly balanced dataset to train their classifier. Unfortunately, they did not directly measure the impact using RUS has on classifier performance, as they failed to compare it against using no sampling technique. Agrawal and An [5] implemented Synthetic Minority Over-sampling TechniquE (SMOTE, for details see [12]) in an effort to alleviate the problem of their model favoring the recall of positive instances due to negative and neutral instances being minority classes. They increased the number of negative instances using SMOTE in an effort to improve negative recall; however this did not improve recall of negative instances. This showed that for their specific dataset and classifier SMOTE was not effective, but should not be considered conclusive as only one dataset, imbalance level and data sampling technique was tested.

2.5 NOISY DATA

Data collected from real world sources are rarely ideal. Depending on the domain, collection method and intended use noise present in the data can have a noticeable impact on classifier performance. In text classification, such as tweet sentiment,

noise is often present in the attributes as a result of words being misspelled or auto-corrected. Attribute noise can be accidentally created by overly aggressive word or character replacement when attempting to recognize and replace misspelled words while extracting features. Noise can also be present in an instance's class label if it has been incorrectly labeled. Abbasi et al. [1] found noise could have a severe impact on classifier performance in the related domain of web forum sentiment classification; however noise has received little attention for tweet sentiment classification. Other sources of noise include redundant data, instances that are either outliers (being too distant from other examples), or instances that cannot be reliably distinguished as belonging to a class.

Due to the short length of tweet text, constructing an adequate set of features may use thousands to upwards of ten thousand labeled instances as each individual tweet contains few useful features. The volume of instances needed makes it difficult for humans to manually assign class labels. Due to the labor intensity of manual labeling many research groups have instead implemented a method of automatically labeling tweets. The approach outlined by Go et al. [23] was used to construct the Sentiment140 corpus. This approach searches Twitter for tweets containing emoticons associated with positive or negative sentiment, then assigns the tweet a sentiment based on that emoticon. Unfortunately there is no guarantee that the sentiment of emoticons will always match the sentiment of a tweet and this method could introduce significant class noise to the training set. Additionally many tweets do not contain emoticons, using this method of acquiring and labeling data results in a biases data set that is not representative of all tweets. A further concern is that this method removes emoticons from being considered by the learner as their presence would trivialize classification. This may be detrimental to classifier performance as emoticons may be among the most useful features for determining sentiment. This has been demonstrated by Kouloumpis et al. [27] who compared multiple feature sets consist-

ing n-grams in various combinations with lexicon, part of speech and microblogging features (Twitter specific features includes emoticons). They observed the addition of microblogging features had the greatest impact on classifier performance, increasing F-measure significantly in comparison to lexicon and part of speech features.

Davidov et al. [16] proposed a method of automatically labeling data using use hashtags in addition to emoticons. Like emoticons, Hashtags that have been judged to express or be associated with strong sentiment can be used to label instances. This allows a wider range of tweets to be used as training instances as they can contain either emoticons or common hashtags with known polarities. Additionally, more complex sentiment labels can be extracted from hashtags as they can express a wider range of ideas. This method of labeling suffers from the same concerns as emoticon labeling without hashtags, specific noisy class labels. In testing against human judges, Davidov et al. found 84% agreement (between their algorithm and the judges) for emoticon labels and 77% agreement for hashtag labels. An alternative approach, crowd sourcing, is described by Agarwal [4]. This method labels tweets by splitting the dataset into many small batches and having many people manually label each batch. Crowd sourcing provides a solution to generating large manually labeled data sets with the added benefit of preserving emoticons. This approach was used for SemEval-2014 to generate data for Task 9 (Sentiment Analysis in Twitter). Despite being manually labeled it is likely that some class noise is still present due to reader error, negligence or bias. Additionally it is difficult to evaluate if crowd sourcing generates higher quality data than automatic labeling methods as no study exists comparing both methods.

The impact noise has on tweet sentiment has largely been ignored, dismissed on the grounds that many of the learners are designed to be resilient to noise. Current research has not specifically investigated the impact of different levels of noise on tweet sentiment classification. Noise is an issue needing further attention. Evaluation of the

effects of noise on classifier performance could determine if additional actions need to be taken to address this issue. In the event that noise is detrimental to classifier performance several machine learning techniques can be implemented. Data can be cleaned of noisy instances through supervised undersampling and/or removal of noisy instances with the aid of clustering algorithms such as K-NN or similar reduction strategies [37]. Instances suspected of containing noise can be tagged and removed from the set, provided there are sufficiently many instances that this is not a major loss of information. An alternative approach for removing noise was demonstrated by Barbosa et al. [8] in the related domain of classifying tweets as subjective or objective. They developed a process for cleaning the data of noisy instances by having instances labeled by multiple sources and removing conflicting instances. This reduced classification error for subjectivity vs objectivity from 19.9% to 18.1%. Alternatively, or in the event data cannot be sufficiently cleaned of noisy instances, ensemble learners can be used. Ensemble learners aggregate multiple learning algorithms or multiple instances of an algorithm trained on different datasets, resulting in a classifier that is less susceptible to noise [17]. The disadvantage of ensemble learners is that because multiple learners are used the computational cost of training a classifier is increased.

CHAPTER 3

METHODOLOGY

This section serves to provide full information on how our datasets were created and how our experiments were conducted. In the case studies presented in this thesis, we study feature selection, data sampling and ensemble techniques. The following subsections provide further information on our datasets, feature selection techniques, random undersampling, bagging and boosting (our chosen feature selection techniques), and the framework we have used to combine them into Select-Bagging and Select-Boost. We also provide information on our learners, what parameters we selected when training them, and how they are trained and evaluated.

3.1 FEATURE SELECTION

In this thesis we evaluate ten filter-based feature ranking methods. These can be divided into three categories: commonly used feature selection, Threshold Based Feature Selection (TBFS), and feature selection using First Order Statistics (FOS) techniques. For commonly used we have elected to use Chi-Squared. First, TBFS techniques normalize attributes to have values between 0 and 1. These values are treated as posterior probabilities and each at tribute's value is compared against a threshold using the two following classification rules (P and N refer to positive and negative class labels).

1. instances with $value > t$ are classified as P while values $< t$ as N.

2. instances with $value > t$ are classified as N while values $< t$ as P.

A metric is used to evaluate the attribute across all possible t, with the most relevant attributes having the highest value. In our case studies, we employ six TBFS techniques based on the following metrics: gini-index, KolmogorovSmirnov statistic, Mutual Information, Probability Ratio, area under the precision-and-recall curve, and area under the receiver operating characteristic curve. FOS techniques make use of first order statistics such as mean and standard deviation. This family of techniques includes: Signal-to-Noise ratio, Significance analysis of microarrays, and Wilcoxon Rank Sum. Each of these techniques is used to perform feature selection with ten feature subset sizes: 5, 10, 15, 20, 25, 50, 75, 100, 150 and 200.

3.1.1 Commonly Used Feature Selection techniques

The chi-squared (CS) test [35] is used to measure class/attribute dependence. CS tests the divergence from an expected distribution against the null hypothesis (a feature is equally likely to be in either class). For very small expected counts, common in text classification, CS may behave erratically (Forman 2003).

3.1.2 Threshold Based Feature Selection techniques

Gini-Index (GI) measures the impurity of each feature towards categorization [3].

$$GI = \min_{t\in[0,1]} [2\mathrm{P}\,(\mathrm{t})\,(1-\mathrm{P}\,(\mathrm{t})) + 2\mathrm{NPV}\ \ (\mathrm{t})\,(1-\mathrm{NPV}\ \ (\mathrm{t}))] \tag{3.1}$$

The KolmogorovSmirnov statistic (KS) is a nonparametric test that measures a features relevance by dividing data into clusters based on class and is defined by the maximum distance between true positive rate (TPR) and false positive rate (FPR) for a given feature.

$$KS = \max_{t\in[0,1]} |TPR\,(t) - FPR\,(t)| \tag{3.2}$$

Mutual Information (MI) measures the dependence between a given feature and class by computing the number of times a feature and class co-occur and occur without the class [3].

$$MI = \max_{t \in [0,1]} \sum_{\hat{y}^t \in \{P,N\}} \sum_{y \in \{P,N\}} p\left(\hat{y}^t,\ y\right) log \frac{p\left(\hat{y}^t,\ y\right)}{p\left(\hat{y}^t\right) p\left(y\right)} \tag{3.3}$$

Probability ratio (PR) is defined as the sample estimate probability of the feature given the positive class divided by the sample estimate probability of the word given the negative class [20].

$$PR = \max_{t \in [0,1]} \frac{TPR\left(t\right)}{FPR\left(t\right)} \tag{3.4}$$

Area under the Precision-and-recall curve (PRC) is a measure of the trade-off between precision and recall. Precision and recall are calculated as the decision threshold is varied.

The area under the receiver operating characteristic (ROC) curve measures the trade-off between True Positive Rate (TPR) and False Positive Rate (FPR) computed by calculating the TPR and FPR as the threshold is varied from 0 to 1 similar to PRC.

3.1.3 First Order Statistics

Signal-to-Noise ratio (S2N) is a value describing how well a feature is discriminated between two classes (c1 and c2). It is calculated from the ratio of the difference of class means to the sum of the standard deviations of each class.

$$S2N = \frac{(\mu_{c1} - \mu_{c2})}{(\sigma_{c1} + \sigma_{c2})} \tag{3.5}$$

Significance analysis of microarrays (SAM) is a statistical technique initially developed for performing feature selection of DNA micro arrays

Wilcoxon Rank Sum (WRS) is a non-parametric hypothesis test that can be used to rank features by measuring the difference between two classes for a given feature.

3.2 DATA SAMPLING

Random undersampling is a form of data sampling that randomly selects majority class instances and removes them from the dataset until a desired class distribution is achieved [9]. This means that for a dataset containing 100 positive and 400 negative instances, RUS must remove 300 negative instances in order to achieve a 50:50 post-sampling positive:negative class ratio. As tweet sentiment datasets are large, this loss of instances should not be problematic as sufficient instances will remain from which to train the classifier. Oversampling techniques are less desirable as they increase the size of already high dimensional datasets (tweet datasets have thousands of instances and thousands of features). In this thesis we compare the impact of two different post-sampling class distribution ratios, 50:50 (denoted as RUS50) and 35:65 (denoted as RUS35). The first is selected as it creates a perfectly balanced dataset from which to train learners. The second post-sampling ratio is selected as it has been shown in other domains that sampling to a perfectly balanced class distribution is not always optimal in cases of high class imbalance, since it eliminates less majority instances [47]. These sampling techniques were implemented in the Weka data-mining toolkit [50].

3.3 ENSEMBLE TECHNIQUES

Bagging and boosting are ensemble techniques that combine multiple learners in an effort to improve classifier performance. Bagging, also known as bootstrap aggregation, uniformly samples data instances with replacement (the same instance can be sampled multiple times). This creates multiple, newly sampled bootstraps of the same size as the original training dataset. Each bootstrap is used to train a classifier using a single base learner. When classifying a new instance the individual classifiers

are combined through voting or averaging posterior probabilities. Bagging has been shown to improve unstable or weak learners, but can degrade the performance of stable learners such as K-Nearest Neighbor [14].

Unlike bagging, which trains multiple classifiers in parallel, boosting creates an ensemble of classifiers iteratively, training and evaluating a classifier in each iteration and altering how the subsequent iteration's classifier is trained based on the performance of the classifier in the previous iteration. One of the most popular families of boosting algorithms is AdaBoost. AdaBoost and its variants have been shown to improve classifier performance for many learners [26].

AdaBoost functions by training and evaluating a new classifier every iteration. On the next iteration, weights are assigned to instances based on the classification results, weighting misclassified instances more heavily. This results in the subsequent iteration being more likely to correctly classify previously misclassified instances. AdaBoost combines the classifiers using weighted voting (or averaging posterior probabilities) to classify new instances. Re-weighting is not compatible with some base learners; however, an alternative approach allows boosting to be used with these learners by resampling instances to match the weights assigned to each instance [26]. In this implementation of AdaBoost, data sampling with replacement is used to create a new training dataset where the probability of selecting an instance is proportional to its assigned weight.

We use AdaBoost.M1, one of many variants of AdaBoost with data resampling, with 10 iterations, selected based on preliminary experiments which found using more iterations did not significantly improve classification performance. This is denoted as AdaBoost-r for the remainder of this thesis. We combine bagging and Adaboost-r with feature selection, performed during each iteration of AdaBoost-r or on each bootstrap created by bagging, to create the combined ensemble and feature selection techniques, Select-Bagging and Select-Boost. These algorithms were recently pro-

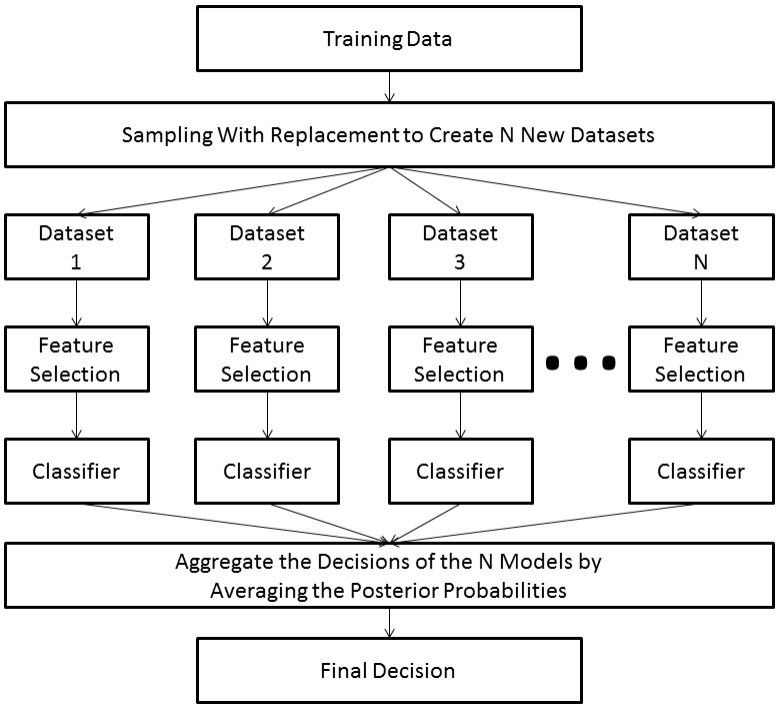


Figure 3.1: Select-Bagging Framework

posed and implimented by our research team. A framework for each technique can be found in Figures 3.1 and 3.2. Performing Feature Selection alone, with no ensemble technique, is denoted as "None" in all tables and figures when accompanying Select-Bagging and Select-Boost.

3.4 MACHINE LEARNING ALGORITHMS

We created inductive models using eight different classification algorithms, described with accompanying parameters. These classifiers were selected as they are commonly used in machine learning, and several are commonly used for sentiment classification. All learners were implemented using the WEKA toolkit [50] with default values unless otherwise noted. All changes were found to improve classification results by preliminary research or previous work [47].

We train and evaluate the performance of K Nearest Neighbor (KNN), two variants

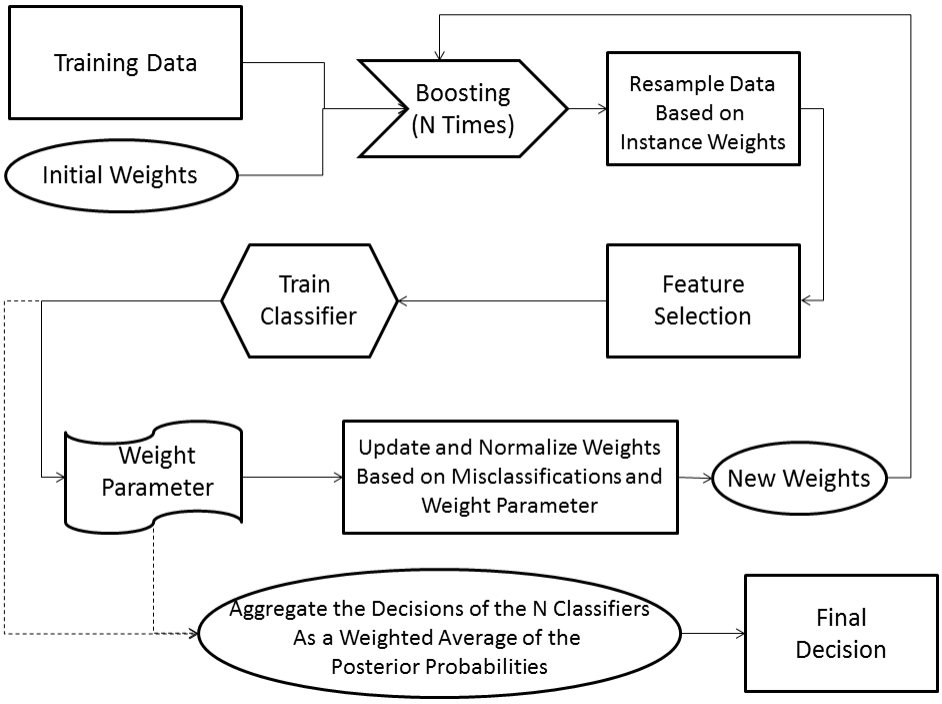


Figure 3.2: Select-Boost Framework

of the C4.5 decision tree algorithm, Support Vector Machines (SVM), Multilayer Perceptron (MLP), Radial Basis Function Network (RBF), Logistic Regression (LR), and Naïthey (NB).

KNN, denoted IBk in Weka, predicts the class of a new, unseen instance by conducting a vote using its k nearest neighbors. In our experiments we elected to use "$k = 5$" and set the "$distanceWeighting$" parameter to "$Weight\ by\ 1/distance$" to use inverse distance weighting when voting on the class of a new instance. Inverse distance weighting gives greater weight to neighbors closer to the instance being classified. We denote this learner as 5NN.

We construct two variants of the C4.5 decision tree using J48 in Weka. C4.5 uses information entropy to create a decision tree. Each node uses the attribute with the highest normalized information gain to split the data by class. Our first version is labeled as C4.5D. and uses default parameters, while C4.5N uses no pruning and Laplace Smoothing as this has been shown to improve performance [49].

SVM, called SMO in WEKA, is a machine learning algorithm that maps instances into n dimensional space (n is the number of features) and finds the hyperplane creating the greatest distance between the two classes. New instances are projected into this space and assigned a class based on which side of the hyperplane they fall on. The instances closest to the plane are known as "Support Vectors". For our experiments, the complexity constant "c" was changed from 1.0 to 5.0, and the "$buildLogisticModels$" parameter set to "$true$" to allow proper probability estimates to be obtained [50]. Additionally, we used a linear kernel.

MLP is a forward feed artificial neutral network that uses back propagation to train the network. We changed two parameters from their default values. The "$hiddenLayers$" parameter was changed to "3" to define a network with 1 hidden layer containing three nodes and the "$validationSetSize$" parameter was set to "10", so that 10% of the training data would be left aside to use as validation to determine when to stop the training process.

A second type of artificial neural network, RBF, works by outputting a linear combination of radial basis functions of the inputs and neuron parameters. Our RBF networks was trained with the parameter "$numClusters$" set to "10."

Our last learner, NB, is based on conditional probabilities, and uses the assumption (naïvely) that all attributes are independent of each other to determine the probability of an instance belong to a class based on the features it contains. Despite this assumption generally being false NB is still an effective and popular classifier.

3.5 DATASETS AND FEATURE EXTRACTION

For our case studies, we construct datasets from two sources. The first set of datasets were constructed from the 2015 SemEval Task 10, subtask B dataset [41]. The SemEval dataset is manually labeled, and thus expected to contain few mislabeled instances; however, it is much smaller in size than the sentiment140 corpus. For this

Table 3.1: Datasets

Name	# Minority	Total #	%minority	#attribute
SemEval unigram	1399	4819	29%	2500
SemEval emoticon	1399	4819	29%	45
SemEval both	1399	4819	29%	2545
1:99 Sentiment140	30	3000	1%	2380
5:95 Sentiment140	150	3000	5%	2371
10:90 Sentiment140	300	3000	10%	2375
15:85 Sentiment140	450	3000	15%	2362
20:80 Sentiment140	600	3000	20%	2370
25:75 Sentiment140	750	3000	25%	2368
50:50 Sentiment140	1500	3000	50%	2388
Sentiment140 manual	177	359	49%	675

dataset, we used all available positive and negative instances, choosing to exclude neutral and objective (having no sentiment) instances to facilitate better comparison with the results for the Sentiment140 dataset. The resulting dataset is imbalanced, having roughly three times as many positive instances compared to negative. Using this dataset we constructed three training sets with different features (unigrams, emoticons or unigrams and emoticons). All three of our training sets contain the same 3420 positive sentiment and 1399 negative sentiment tweets.

Our second set of datasets were constructed from the sentiment140 corpus [23]. This corpus is a collection of 1.6 million tweets with positive or negative sentiment labels, constructed in a semi-supervised manner. Tweet collection and labeling was automated by searching Twitter for tweets with emoticons. Emoticons are combinations of letters, punctuation marks and symbols that seek to create a face conveying an emotion. For example ":-)" is a smiley face and ":-(" is a frown. This process

is expected to result in some tweets being mislabeled, since an emoticon may not always match the sentiment of the tweet, making it a noisy dataset. From this corpus we created datasets by sampling (without replacement) a set number of positive and negative instances in order to construct a dataset with a desired size and class distribution ratio. We constructed six datasets in this manner, each with 3000 instances but a different class distribution ratio. Additionally, we made a dataset using the positive and negative instances from the small manually labeled "test" portion of the corpus. This dataset consists of 498 manually labeled tweets that are either positive, negative or neutral. Removing the neutral instances leaves 177 negative and 182 positive instances.

For both datasets, we extracted unigrams (individual words within the text of the tweet) as features. Unigrams have been used successfully for sentiment classification in related domains, such as movie review sentiment [34], and many studies on tweet sentiment classification. Before we extracted features, symbols, punctuation marks and URLs were removed from the text, and all letters were made lower case. Additionally letters repeated more than 2 times in a row were reduced to 2 repetitions. We extracted the top 2500 most frequent features, provided that the feature appears in multiple tweets. As the datasets are different in size, origin or class ratio, they are made from different instances and contain different unigram features.

In the construction of the three SemEval datasets we extracted a second feature type, emoticons. They are valuable in predicting sentiment as it can be expected that the emotion expressed by an emoticon should roughly match the sentiment of the tweet containing the emoticon. This has been demonstrated by the use of emoticons to automatically label training instances, as was conducted by Go et al. [23] when creating the sentiment140 corpus. When extracting features for our dataset we used a dictionary of 133 emoticons, compiled from multiple websites listing emoticons and associated emotions. We did not include the emotional polarity or sentiment of

the emoticons as part of the extracted features as our machine learning algorithms determine the relationship between features and sentiment. 45 of the 133 emoticons contained in our dictionary were present in the SemEval dataset and were extracted as features. The use of the training set generated with this feature extraction method is denoted by "emoticons". Using both emoticons and unigrams is denoted as "both". Additional details on our datasets can be found in Table 3.1.

3.6 CROSS-VALIDATION AND PERFORMANCE METRIC

All models were trained using five-fold cross-validation. Cross-validation partitions the dataset into N equal folds and uses $(N-1)$ parts at a time to train a model, while using the remaining fold for validation [40]. This is repeated N times, so that each fold is used for validation once. By cycling through which fold is used for validation, the entire dataste can be used for both training and validation. In our experiments, cross-validation was repeated four times to eliminate bias due how the data was split when the partitions were created.

As some of our datasets are imbalanced, classification performance of our models is evaluated using Area Under the receiver operating characteristic Curve (AUC) [13]. The recevier opperating characteristic curve plots the trade-off between TPR and FPR across the entire range of class distributions and error costs the classifier may encounter. Thus, AUC provides a numeric representation of how well a classifier performs in a variety of situations [36]. This makes it an appropriate metric for imbalanced data, where there is a majority and minority class. It is important to note that AUC is used to denote a performance metric, while ROC is used to denote a feature selection technique in this thesis.

CHAPTER 4

CASE STUDIES

The following sections provide results for our case studies testing the impact of feature selection techniques, data sampling and ensemble techniques. The methodology and datasets discussed in Chapter 3 were used for all experiments. First, feature selection, data sampling and ensemble techniques are presented separately. This is followed by a comparison of all three types of techniques in an effort to determine which are most important. Next a study combining feature selection and ensemble techniques is presented, showing the effectiveness of combining multiple techniques. Finally, we evaluate adding emoticon features to unigram features for classifiers trained on the SemEval data and investigate the importance of combating the high dimensionality resulting from the use of unigram features when adding additional types of features.

4.1 FEATURE SELECTION

In this case study, we seek to observe the impact of using feature selection on tweet sentiment classification. The 50:50 Sentiment140 dataset was used for this study. We use a combination of ten feature rankers, ten subset sizes, and four classifiers. Table 4.4 presents the results of our experiments. In each column the best model for each feature subset size is indicated in **boldface**. It is important to note that "None", meaning no feature selection was performed, is included as if it was an additional ranker in the bottom row of each table. This allows the impact of feature selection to be evaluated against not using feature selection. It should be noted that the feature subset sizes listed in the tables are not relevant for None as it uses all

Table 4.1: Classification Results for Feature Selection

Classifier	Ranker	Feature Subset Size									
		5	10	15	20	25	50	75	100	150	200
5-NN	CS	0.60775	0.64098	0.65854	0.67309	0.68049	**0.69613**	0.69427	0.69469	0.69896	0.69301
	GI	0.49942	0.50075	0.49894	0.49928	0.50188	0.50675	0.52898	0.53729	0.54867	0.55433
	KS	0.60736	0.63384	0.64033	0.64413	0.64825	0.65781	0.65978	0.65665	0.65643	0.65340
	MI	0.60646	0.64009	**0.65905**	**0.67379**	**0.68150**	0.69461	**0.69712**	**0.69614**	0.68692	0.68131
	PR	0.50634	0.50511	0.50550	0.50773	0.50839	0.51378	0.51862	0.51914	0.52354	0.52428
	PRC	0.60461	0.63410	0.63417	0.64056	0.64595	0.66168	0.66232	0.65977	0.65771	0.65813
	ROC	0.60881	0.63532	0.63852	0.64495	0.65134	0.65528	0.65928	0.65642	0.65600	0.65308
	S2N	0.49933	0.49875	0.50092	0.49917	0.49958	0.55302	0.66389	0.69105	**0.70387**	**0.70086**
	SAM	0.50175	0.50193	0.49671	0.49540	0.50272	0.50353	0.50709	0.52114	0.52563	0.55014
	WRS	0.50050	0.50058	0.49899	0.49843	0.50204	0.50371	0.50521	0.50888	0.52105	0.51907
	None	**0.65191**	**0.65191**	0.65191	0.65191	0.65191	0.65191	0.65191	0.65191	0.65191	0.65191
C4.5	CS	0.60174	0.61949	0.64452	0.66184	**0.67591**	**0.69836**	**0.70404**	0.69819	0.69663	0.69898
	GI	0.50000	0.50000	0.50000	0.50000	0.50000	0.50371	0.52396	0.53270	0.53947	0.54231
	KS	0.59454	0.62584	0.64234	0.65454	0.66062	0.68084	0.68529	0.68440	0.68773	0.68710
	MI	0.59967	0.61746	0.64399	0.66013	0.67168	0.69624	0.70214	**0.70071**	0.69312	0.69241
	PR	0.50635	0.50509	0.50548	0.50773	0.50839	0.51351	0.51771	0.51678	0.52067	0.52099
	PRC	0.59718	0.62004	0.63001	0.64336	0.65224	0.68605	0.69424	0.68937	0.68781	0.68129
	ROC	0.59501	0.62160	0.64240	0.65360	0.65827	0.68091	0.68444	0.68246	0.68714	0.68470
	S2N	0.50000	0.50000	0.50000	0.50000	0.50000	0.54960	0.65122	0.67474	**0.69817**	**0.70288**
	SAM	0.50000	0.50000	0.50000	0.50000	0.50000	0.50062	0.50803	0.51765	0.51629	0.53831
	WRS	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000	0.50000
	None	**0.66392**	**0.66392**	**0.66392**	**0.66392**	0.66392	0.66392	0.66392	0.66392	0.66392	0.66392
LR	CS	**0.60817**	0.64454	**0.66514**	0.68230	0.69105	0.72512	0.73821	0.74207	0.74192	0.73558
	GI	0.50000	0.50000	0.50000	0.50000	0.50000	0.50394	0.52875	0.53704	0.55014	0.55611
	KS	0.60696	**0.64312**	0.66167	0.67285	0.68257	0.72518	0.73915	0.74672	**0.75226**	**0.74872**
	MI	0.60718	0.64160	0.66640	**0.68391**	**0.69256**	**0.72599**	**0.74144**	**0.74761**	0.74088	0.74120
	PR	0.50637	0.50501	0.50549	0.50755	0.50827	0.51427	0.51907	0.51940	0.52598	0.52584
	PRC	0.60507	0.64172	0.65185	0.66468	0.67479	0.72395	0.73593	0.74296	0.74014	0.73485
	ROC	0.60804	0.64496	0.66029	0.67196	0.68357	0.72451	0.73833	0.74559	0.75048	0.74623
	S2N	0.50000	0.50000	0.50000	0.50000	0.50000	0.55141	0.67275	0.70292	0.71912	0.72685
	SAM	0.50000	0.50000	0.50000	0.50000	0.50000	0.50102	0.50772	0.52294	0.52397	0.55192
	WRS	0.50000	0.50000	0.50000	0.50000	0.50000	0.50242	0.50610	0.51223	0.52453	0.52641
	None	0.59623	0.59623	0.59623	0.59623	0.59623	0.59623	0.59623	0.59623	0.59623	0.59623
MLP	CS	**0.61039**	**0.64024**	0.66022	**0.67063**	**0.67943**	**0.70027**	**0.71318**	0.71308	**0.72117**	**0.72731**
	GI	0.50015	0.49979	0.49985	0.49964	0.49959	0.50125	0.52047	0.52463	0.54167	0.53968
	KS	0.60306	0.63145	0.64126	0.64734	0.65995	0.68358	0.69684	0.70518	0.71540	0.71398
	MI	0.60743	0.63865	**0.66215**	0.66980	0.67784	0.69958	0.71123	**0.71832**	0.71876	0.71565
	PR	0.50629	0.50699	0.50772	0.50785	0.50803	0.50684	0.50832	0.50636	0.50170	0.50298
	PRC	0.60028	0.62780	0.63258	0.64674	0.65543	0.68334	0.70139	0.70390	0.71708	0.72097
	ROC	0.60380	0.63004	0.64131	0.65454	0.65688	0.68255	0.69936	0.70027	0.71606	0.71591
	S2N	0.50010	0.50041	0.50018	0.50010	0.49962	0.54384	0.65915	0.68680	0.70451	0.70605
	SAM	0.49998	0.49930	0.49941	0.49968	0.49933	0.49968	0.50233	0.50749	0.50843	0.51215
	WRS	0.50006	0.49979	0.49984	0.50072	0.50053	0.50062	0.50083	0.50198	0.50427	0.50490
	None	0.53133	0.53133	0.53133	0.53133	0.53133	0.53133	0.53133	0.53133	0.53133	0.53133

2388 features available from the dataset and is repeated for each subset size.

First, we look at the results of 5-NN. It can be observed that 5-NN requires at least 15 features to improve upon None. MI achieves the highest AUC for subset sizes between 15 and 100 features, excluding using 50 features, where CS is the best ranker. S2N achieves the highest AUC values for 150 and 200 features, with S2N and 150 feature being the best performing model trained with 5-NN. This last result is interesting as S2N performs poorly for small subset sizes, being the worst performer when selecting 5, 10, 20 or 25 features. In general GI, PR, SAM and WRS perform poorly for all subset sizes and never outperform None.

When we observe the results for the decision tree learner C4.5, we see that None achieves higher AUC values than rankers for models trained with 5, 10, 15 or 20 features. Using CS yields the highest performance for models trained with 25 to 75 features, while MI is best for 100 features. Again S2N yields the best performance for 150 and 200 features. The highest AUC value for an individual model was achieved by CS with 75 features. GI, PR, SAM and WRS perform poorly for all subset sizes. S2N is again observed to be a worst performer for small subset sizes, achieving lower AUC values than None for 75 or less features.

From the results for classifiers trained with LR, we observe that all top performing options used feature selection, and again GI, PR, SAM and WRS with any subset size fail to outperformed None. CS is best for 5 or 15 features and ROC for 10 features. For subset sizes between 20 and 100 features MI achieves the highest AUC values. The best performance for 150 and 200 features was achieved with KS, using KS and 150 features achieving the highest AUC value among the models trained with LR. S2N is never the best ranker for a subset size and is among the worst rankers for small subsets.

Lastly, the results for MLP show that CS is the best performing ranker for 8 out of 10 subset sizes; MI is the best for 15 and 100 features. GI, PR, SAM, WRS perform

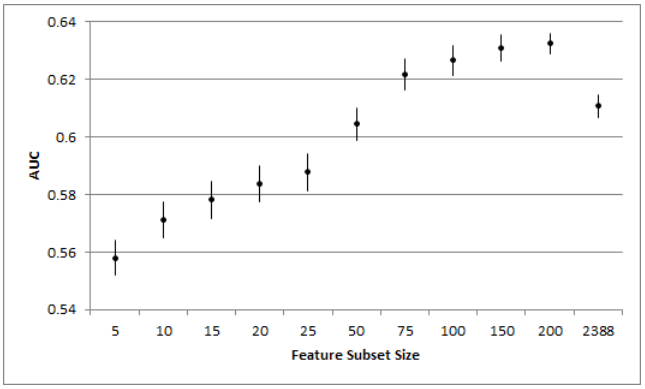
Table 4.2: ANOVA: Features subset size and Rankers

Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
# Features	6.453267	9	0.71703	884.7823	0
Ranker	49.86691	10	4.986691	6153.352	0
Error	13.15177	8780	0.001498		
Total	69.47194	8799			

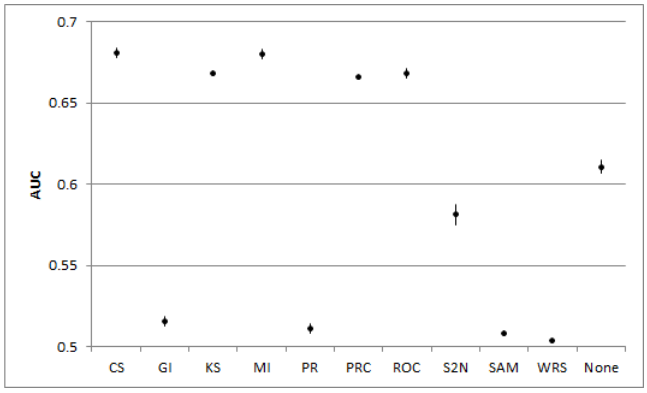
poorly for all subset sizes and S2N performs poorly for small subset sizes. Like LR, the top classification performance with any of the tested subset sizes occurs when using a ranker. The highest AUC value was achieved by the model trained using CS and 200 features.

In summary filter-based feature selection improved classification performance for all learners tested; though, using a small feature subset was inferior to None for both 5-NN and C4.5. The highest AUC values were achieved for 5-NN using S2N with 150 features, for C4.5 using CS with 75 features, for LR using KS with 150 features, and finally for MLP using CS with 200 features. The best performing model was the result of training a classifier using LR with 150 features selected using KS. In general models performed better when rankers selected larger numbers of features, best performing models for each learner had 75 or more features. Other results of note include: S2N performs poorly with small subsets for all learners, but performs well for larger subsets (it was the best ranker for 5-NN and C4.5 for 150 and 200 features); PR, SAM and WRS performed worse than using no feature selection for all learners and subset sizes, while GI managed improved performance compared to "None" only when selecting 150 or 200 features with C4.5; and PRC is never the best performing ranker, and is generally inferior to CS, KS, MI and ROC, but better than PR, SAM, and WRS.

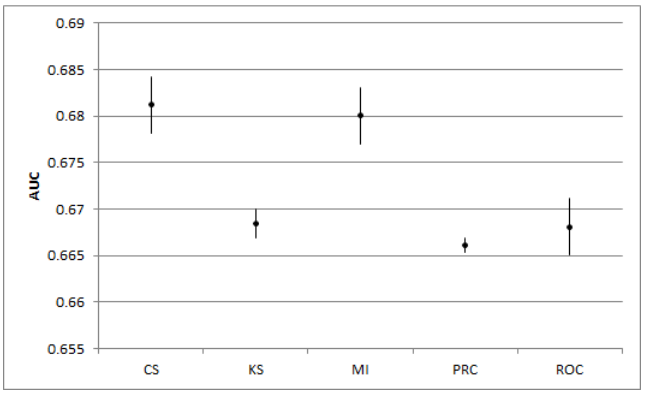
Figure 4.1: Tukey HSD Results: Feature Selection



(a) Subset Sizes



(b) Rankers



(c) Top Rankers

4.1.1 Statistical Analysis

Statistical significance of the results presented in this work was tested by performing two-way ANOVA with a 5% confidence level using Microsoft Excel. The results are presented in Table 4.9 and show both choices of ranker and feature subset size to be significant factors in determining classifier performance. In addition, to the ANOVA values we present the Tukey's Honestly Significant Difference test to compare the factors. Figures 4.1a through 4.1c present plots of mean AUC values across all learners and the four runs of five-fold cross-validation associated with each subset size or ranker with accompanying confidence intervals.

Figure 4.1a presents mean AUC values for each feature subset size, including 2388 features, the result of using no feature selection. It can be observed that using 200 features is best on average, but not significantly different from 150, or 100 features. 75 features significantly outperforms None but is significantly less than 100 to 200. Most importantly, Figure 1 shows that the performance gains due to selecting subsets of features are significant. Additionally, using 50 or less features is confirmed to perform worse than None (though 50 is not significantly lower than None).

Figure 4.1b and figure 4.1c show mean AUC values for rankers, including using no ranker (labeled None). Figure 2 displays all rankers while figure 3 is a close up of the 5 rankers that achieve significantly better classification performance to None. CS achieves the highest mean AUC value, but is not significantly different than MI. KS, PRC and ROC form the next grouping, still performing significantly better than using None. The remaining rankers are significantly worse than None. S2N achieves a similar, but lower mean AUC value compared to None. As expected GI, PR, SAM and WRS are significantly worse than other rankers.

Table 4.3: Classification Results for Data Sampling

Dataset	Sampling	Learner							
		C4.5N	NB	MLP	5NN	SVM	RBF	LR	C4.5D
1:99	None	0.49987	**0.57909**	0.49827	0.50065	0.49195	0.51096	**0.49149**	0.50013
	RUS35	**0.53112**	0.53480	**0.50901**	0.47253	**0.50852**	**0.51951**	0.47283	**0.53384**
	RUS50	0.50942	0.50446	0.48117	**0.51708**	0.50234	0.51629	0.47981	0.51820
5:95	None	0.54387	0.58657	0.53729	0.54737	**0.61706**	0.50579	0.57273	0.50449
	RUS35	**0.58652**	**0.62940**	**0.60126**	**0.57961**	0.61026	0.57978	0.59752	**0.57137**
	RUS50	0.57822	0.62724	0.54270	0.57238	0.59635	**0.58449**	0.58499	0.56491
10:90	None	0.57065	0.64153	**0.60915**	**0.62295**	**0.67913**	0.57695	0.60407	0.53129
	RUS35	0.60481	0.67986	0.58922	0.61377	0.66126	**0.61741**	0.62639	0.55944
	RUS50	**0.61586**	**0.69630**	0.56964	0.61186	0.64100	0.60507	**0.63744**	**0.58800**
15:85	None	0.58273	0.67723	**0.62227**	**0.65847**	**0.70903**	0.62565	0.56310	0.52342
	RUS35	0.61696	0.70754	0.59079	0.63835	0.69660	0.63933	0.65109	0.56861
	RUS50	**0.62177**	**0.72115**	0.61437	0.61002	0.67912	**0.64404**	**0.65245**	**0.60978**
20:80	None	0.58823	0.71100	**0.64772**	**0.66428**	**0.71254**	0.63209	0.57199	0.52496
	RUS35	0.61825	0.72785	0.59648	0.64750	0.70166	**0.64519**	0.64727	0.56832
	RUS50	**0.62117**	**0.73636**	0.63212	0.62888	0.69120	0.64168	**0.66339**	**0.59795**
25:75	None	0.58446	0.72613	**0.65955**	**0.65428**	**0.69683**	0.65034	0.56000	0.53693
	RUS35	0.63248	0.73410	0.61938	0.64154	0.69416	**0.65115**	0.62884	0.57592
	RUS50	**0.64257**	**0.73790**	0.62511	0.62289	0.68194	0.65079	**0.64195**	**0.62524**

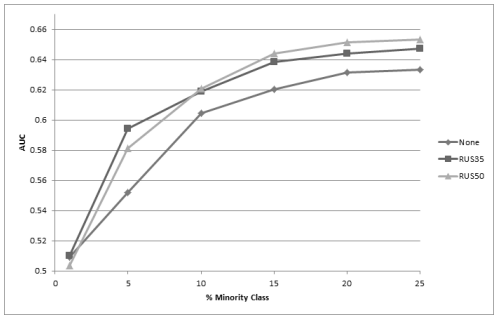
4.2 DATA SAMPLING

Using the methodology and six imbalanced sentiment140 datasets described above, we measured the effect using RUS has on the performance of tweet sentiment classifiers. We compare the performance of eight learners on six imbalanced datasets, testing RUS50 and RUS35 on every dataset, and compare the performance of classifiers trained using RUS against classifiers trained with no sampling technique. Table 4.3 presents the results of our experiments, divided by dataset. AUC score for each learner is presented with no data sampling (None), RUS35, and RUS50. In each column the model with the highest AUC is indicated in **boldface**.

The 10:90, 15:85, 20:80, and 25:75 imbalanced datasets exhibit similar patterns. for all of these datasets, RUS improves performance of five of the eight learners,

while SVM, 5-NN, and MPL were not improved. Also, RBF network performed best with RUS35, while the remaining four learners performed better using RUS50. The best performing classifier on these datasets was NB with RUS50 and best performing classifier with no data sampling was SVM, which was only outpeformed by NB with RUS50. Averaged across all learners, both RUS35 and RUS50 perform better than using no undersampling. RUS50 achieves higher AUC scores than RUS35 for these datasets; however, this difference is small and narrows as the level of class imbalance rises.

Figure 4.2: AUC vs. Level of Class Imbalance Averaged Across All Learners



The 5:95 imbalanced dataset represents data with more severe class imbalance, containing only 5% (or 150) positive sentiment instances. Due to the high level of class imbalance, all learners have lower performance than is observed on less imbalanced datasets, since there are less available negative instances. Once again SVM achieves the highest AUC for learners when no sampling is used, and does not benefit from sampling. Unlike previous datasets, performance for all seven remaining learners is improved by both RUS50 and RUS35. RUS50 achieves higher performance for MLP, while all other learners perform better with RUS35. The higher level of class imbalance accounts for improvements observed for MLP and 5-NN when using RUS. Again NB is the best performing learner when using either RUS50 or RUS35, and achieves the highest overall AUC when using RUS35.

All learners performed poorly on the 1:99 imbalanced dataset and the results for

this dataset are inconsistent with the trends and patterns observed for the other five. While RUS35 performs slightly better on average than using no data sampling, RUS50 performs slightly worse than using no data sampling. The inconsistency is likely due to an insufficient number of minority instances being present with which to train classifiers and very few instances remaining after resampling. Only 48 instances are used to train classifiers using RUS50, and 66 for those using RUS35. Due to this, no conclusions can be drawn from the results of this dataset.

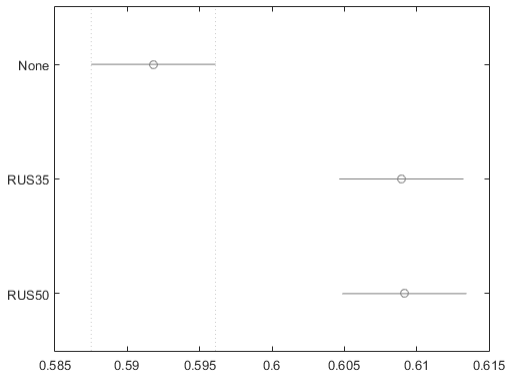
Figure 4.2 plots the AUC (averaged across all learners) of each sampling technique, for each level of class imbalance. It can be observed that increasing the level of class imbalance has a greater impact on classifiers trained with no data sampling technique compared to those trained using RUS. For datasets with 15-25% minority class ratios, RUS50 performs better than RUS35, while for 5% minority class RUS35 performs better. RUS35 appears to overtake RUS50 once the minority class falls below 10%. For these five datasets there is little difference in performance between RUS35 and RUS50. As discussed above, the results for 1% minority class are not meaningful since there are insufficient minority instances with which to train the classifiers.

4.2.1 Statistical Analysis

The statistical significance of the different sampling techniques was tested by performing two-way ANOVA [21] with a 5% confidence level using MATLAB. The results are presented in Table 4.9 and show that choice of RUS50, RUS35 or no sampling technique is a significant factor in determining classifier performance for both datasets. Tukey's HSD test [2] was also conducted to compare the performance of each technique averaged across all learners, the results of which are shown in Figure 4.3. In the figure, the average performance for the three sampling techniques (None, RUS50, and RUS35) are shown, along with their confidence interval. When comparing two techniques, if there is no overlap of the confidence intervals, then the averages are sig-

nificantly different. It can be observed that that both RUS35 and RUS50 significantly improve classifier performance on class imbalanced data; however, while RUS50 has slightly higher performance than RUS35, this difference is not significant. Additional Tukey's HSD test (not presented due to space constraints) were conduced for each level of imbalance. No significant difference was found between RUS35 and RUS50 for any level of imbalance.

Figure 4.3: Tukey's HSD Results: Data Sampling



4.3 ENSEMBLE TECHNIQUES

Using the above methodology and "50:50 Sentiment140" and "Sentiment140 manual"datasets, we compared the performance of seven base learners with bagging, AdaBoost-r, or no ensemble technique (denoted at "None" in this section). The results of our experiment are presented in Table 4.4, divided by dataset. AUC scores for each learner and ensemble technique are presented. In each column, the model with the highest AUC is indicated in **boldface**.

For the first dataset,"50:50 Sentiment140", six out of seven learners achieved the highest AUC values when used with bagging, with RBF Network and bagging achieving the highest AUC for the dataset. AdaBoost-r improves performance for five of seven learners, but has slightly lower performance than bagging for every learner.

Table 4.4: Classification Results for Ensemble Techniques

Dataset	Sampling	Learner						
		5NN	C4.5D	C4.5N	LR	MLP	RBF	SVM
50:50 Sentiment140	None	**0.66963**	0.66276	0.67056	0.59692	0.60230	0.69243	0.72182
	AdaBoost-r	0.66128	0.71513	0.71431	0.68164	0.60596	0.74696	0.71731
	Bagging	0.66644	**0.72133**	**0.72164**	**0.69326**	**0.64876**	**0.76319**	**0.74341**
Sentiment140 manual	None	0.69991	0.73495	0.78245	0.81856	0.74785	0.81053	0.79966
	AdaBoost-r	**0.71984**	**0.82797**	**0.84396**	0.81635	0.81580	0.84879	0.80777
	Bagging	0.70843	0.81721	0.81933	**0.85853**	**0.83276**	**0.86107**	**0.82729**

5NN was the only learner to not be improved by using an ensemble technique. Instead, using either technique resulted in a small decrease in performance. The highest performance for each base learner (apart from 5NN) was observed when using bagging.

On the second dataset, "Sentiment140 manual", Four learners (LR, MLP, RBF Network and SVM) achieve the highest performance using bagging, and the remaining three using AdaBoost-r. Both ensemble techniques improved performance for all learners, apart from LR which was only improved by AdaBoost-r. Again, the highest performance was observed when training a classifier using RBF Network in combination with bagging. Interestingly, the AUC scores for LR and MLB with bagging are substantially closer to the top performing classifier for this dataset compared to their results on "50:50 Sentiment140". Once again, ensemble techniques clearly improve the performance of the base learners.

Overall, 5NN is not greatly affected by using either ensemble technique; however, it performs poorly on both datasets in any configuration. This is not entirely unexpected, as bagging is known to have minimal impact on the performance of K-Nearest Neighbor [14]. The remaining learners are all improved by ensemble techniques on both datasets. Both variants of C4.5 benefit from both ensemble techniques, with bagging yielding higher performance on "50:50 Sentiment140" and AdaBoost-r higher

Table 4.5: ANOVA: Ensemble Techniques

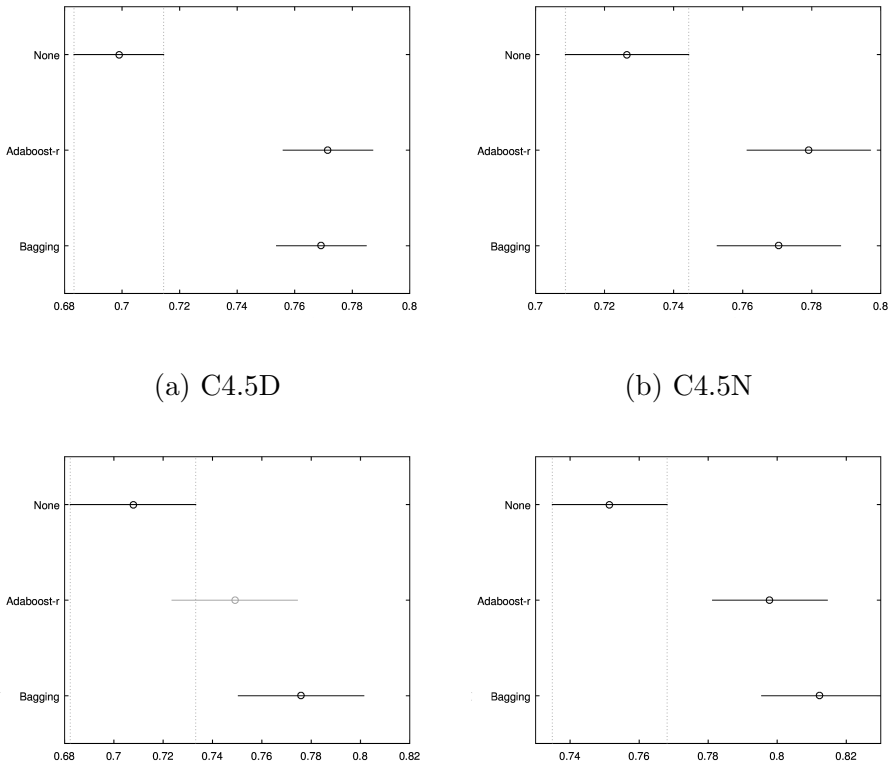
Learner	Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
5NN	Ensemble Technique	0.00067	2	0.00034	0.13	0.874
	Error	0.29126	117	0.00249		
	Total	0.29193	119			
C4.5D	Ensemble Technique	0.13665	2	0.06832	19.84	3.80295e-08
	Error	0.40289	117	0.00344		
	Total	0.53954	119			
C4.5N	Ensemble Technique	0.06372	2	0.03186	7.04	0.0013
	Error	0.52985	117	0.00453		
	Total	0.59357	119			
LR	Ensemble Technique	0.09427	2	0.04713	5.13	0.0073
	Error	1.07564	117	0.00919		
	Total	1.16991	119			
MLP	Ensemble Technique	0.08652	2	0.04326	2.68	0.0727
	Error	1.88731	117	0.01613		
	Total	1.97383	119			
RBF	Ensemble Technique	0.08046	2	0.04023	10.25	7.93216e-05
	Error	0.45934	117	0.00393		
	Total	0.53979	119			
SVM	Ensemble Technique	0.01505	2	0.00753	2.93	0.0576
	Error	0.301	117	0.00257		
	Total	0.31605	119			

performance on "Sentiment140 manual". LR and MLP are both greatly improved by ensemble techniques on both datasets, but still offer poor performance on the larger dataset. Using RBF Network as a base classifier in conjunction with bagging yields the highest performance on both datasets. SVM is improved by ensemble techniques (though not by as much as other learners). It also has the second highest performance observed on the larger dataset, and used with no ensemble technique offers performance comparable to or better than the other learners used in an ensemble, excluding RBF Network.

4.3.1 Statistical Analysis

We tested the statistical significance of our results by performing ANOVA [21] with a 5% significance level on each learner using MATLAB. The results are presented in Table 4.5 and show that the performance differences between using bagging, AdaBoost-r, or no ensemble technique is significant for C4.5D, C4.5N, LR and RBF Network. Additionally, Tukey's HSD tests [2] were conducted on these four learners, comparing the performance of the different ensemble learning methods. The results can be found in Figure 4.4. In each figure, the performance for ensemble learning techniques (None, Bagging, and AdaBoost-r) are shown, along with their confidence interval. If there is no overlap of the confidence intervals of two techniques, then their averages are significantly different.

Figure 4.4: Tukey's HSD Results: Ensemble Techniques



(a) C4.5D

(b) C4.5N

(c) LR

(d) RBF Network

From these figures it can be observed that using bagging offers significantly better performance than not using an ensemble learner for all four learners. For three of these learners, RBF Network and both C4.5 variants, AdaBoost-r also significantly improved classifier performance. While bagging significantly improved performance for LR, the improvement observed when using AdaBoost-r was not significant. Additionally, the performance difference between bagging and AdaBoost-r was found to not be significant for any of the learners.

4.4 COMPARISON OF FEATURE SELECTION, DATA SAMPLING AND ENSEMBLE TECHNIQUES

In this case study we compared the performance of our base learners using feature selection, RUS, bagging, boosting, or no additional technique (denoted as "None"). Each combination of learner and technique was trained and evaluated on three datasets: "50:50 Sentiment140", "20:80 Sentiment140" and "5:95 Sentiment140". The results of our experiment are presented in Tables 4.6, 4.7 and 4.8, divided by dataset. Tables are subdivided by technique type. RUS is not performed on the 50:50 dataset as it is balanced. AUC scores for combination of learner and technique are presented. In each column, the model with the highest AUC is indicated in **boldface** and the model with the lowest AUC is indicated in *Italic*.

In Table 4.6, it can be observed that all ensemble and feature selection techniques improve classification performance on the 50:50 dataset, with the exception of both ensemble techniques for SVM, and AdaBoost-r on 5NN. "None" achieves the worst AUC scores for three of the learners, while feature selection techniques are best for four learners, and bagging is best for C4.5. Using bagging results in higher performance than AdaBoost-r for all learners, though the difference is generally small. Looking at the eight combinations of feature ranker and subset sizes, using CS results in higher performance than using ROC for all learners other than LR, and larger subset sizes

Table 4.6: Classification results for 50:50 dataset

Technique		Learner				
		C4.5	MLP	5NN	SVM	LR
None		*0.67056*	*0.602304*	0.669631	0.721823	*0.596924*
Ensemble	AdaBoost-r	0.71431	0.60596	0.661275	*0.71431*	0.681644
	Bagging	**0.721639**	0.648757	0.66644	0.721639	0.693261
FS	CS200	0.710988	**0.745096**	0.695357	**0.749974**	0.736259
	CS150	0.710166	0.740526	**0.700501**	0.748969	0.742577
	CS100	0.709372	0.729272	0.696206	0.744436	0.742455
	CS75	0.711903	0.72208	0.695755	0.737134	0.739175
	ROC200	0.696418	0.736777	*0.655395*	0.745414	0.746489
	ROC150	0.694797	0.734769	0.657209	0.748458	**0.751043**
	ROC100	0.68116	0.715936	0.657464	0.742642	0.746367
	ROC75	0.681458	0.709907	0.659736	0.736406	0.739091

Table 4.7: Classification results for 20:80 dataset

Technique		Learner				
		C4.5	MLP	5NN	SVM	LR
None		0.588225	0.647716	0.664278	0.71254	0.571986
Ensemble	AdaBoost-r	0.676073	0.6183	0.657858	0.708658	0.662432
	Bagging	0.623823	0.656645	0.659531	**0.721739**	0.66132
Sampling	RUS35	0.618247	0.596475	0.647505	0.701657	0.647275
	RUS50	0.621167	0.632123	0.628884	0.691205	0.66339
FS	CS200	0.64065	0.705485	0.681275	0.671443	0.684238
	CS150	0.647537	0.707857	**0.683605**	0.677977	0.693158
	CS100	0.650459	0.687075	0.671211	0.669913	0.687378
	CS75	0.631063	0.66964	0.667373	*0.65229*	0.67577
	ROC200	0.658818	**0.717549**	0.630131	0.710648	0.705106
	ROC150	0.663009	0.708113	0.636124	0.714556	0.715675
	ROC100	0.677814	0.69946	0.634292	0.712392	**0.722319**
	ROC75	**0.67835**	0.689247	0.639714	0.7028	0.719005

Table 4.8: Classification results for 5:95 dataset

Technique		Learner				
		C4.5	MLP	5NN	SVM	LR
None		*0.543873*	0.537294	0.547368	0.617057	*0.572728*
Ensemble	AdaBoost-r	0.587488	*0.528431*	0.559868	0.604284	0.590516
	Bagging	0.590401	0.595484	*0.538604*	**0.635942**	0.594519
Sampling	RUS35	0.586517	0.601256	0.579605	0.610259	0.597522
	RUS50	0.578224	0.542705	0.572383	0.596347	0.584994
FS	CS200	0.542705	0.606431	0.598481	0.615545	0.601234
	CS150	0.572383	0.576715	0.566931	0.592436	0.590975
	CS100	0.596347	0.576073	0.594632	0.61425	0.617396
	CS75	0.584994	0.582104	**0.599709**	*0.587247*	**0.619327**
	ROC200	0.586184	**0.637335**	0.563809	0.612325	0.578196
	ROC150	0.592059	0.619966	0.56088	0.629167	0.596238
	ROC100	0.612363	0.615295	0.563498	0.599719	0.608265
	ROC75	**0.631965**	0.625367	0.563466	0.592012	0.616825

generally offer higher performance. LR with ROC150 had the highest AUC observed on this dataset, followed by SVM and MLP with CS200. All three perform best when using feature selection instead of an ensemble technique.

Results for the 20:80 imbalanced dataset are presented in Table 4.7. Again feature selection techniques result in the highest observed AUC values for four learners; however, SVM now achieves its highest AUC when using bagging and does not benefit from most of the ranker/subset combinations. RUS35 and RUS50 both improve the performance of C4.5 and LR; however, only RUS50 improves SVM and the performance of MLP and 5NN are degraded by the use RUS50 and RUS35. The highest observed AUC is again achieved by using LR and ROC, but with a subset size of 100 rather than 150. Again, SVM followed by MLP are the next best performing learners; however, as noted SVM now performs best with bagging instead of feature selection. Unlike the balanced data, ROC appears to generally perform better than CS on this dataset, likely because it is a threshold based feature selection technique designed to work with imbalanced data [18]. Additionally, when using ROC, there is

little performance variation due to choice of subset size for the five learners.

Our final set of experiments was conducted using the 5:95 imbalanced dataset, results are presented in Table 4.8. Compared to the 20:80 imbalanced dataset, using RUS has a more noticeable impact on performance, with both RUS35 and RUS50 improving performance for all learners appart from SVM. Again, feature selection was the best technique for four of the learners. The exception, SVM, performed best with bagging as was observed in for the 20:80 dataset. RUS35 acheives higher performance than RUS50 for all learners. While feature selection and ensemble techniques are still effective at improving classifier performance, the impact of these techniques is highly dependent on choice of learner. SVM is minimally impacted by feature selection, and in some cases their performance is degraded by its use. MLP with ROC200 is the best performing classifier, followed by SVM with bagging and C4.5 with ROC75.

4.4.1 Statistical Analysis

We tested the statistical significance of our results using two-factor ANOVA with a 5% significance level on each dataset. The ANOVA results, presented in Table 4.9, show that choice of technique and choice of learner are significant for all three datasets. As we found significant difference between techniques and learners, we performed Tukey's HSD tests, comparing the performance of the different techniques or learners on each dataset. The results can be found in Figure 4.5. In Figures 4.5a, 4.5b and 4.5c, the performance of the techniques on each dataset is presented, along with their confidence interval. If there is no overlap of the confidence intervals of two techniques, they are significantly different. Though not the focus of this study, Figures 4.5d, 4.5d and 4.5f show the performance and confidence intervals of each learner for the three datasets.

From Figure 4.5a, it can be observed that the improvements observed when using feature selection or ensemble techniques on balanced data are statistically significant.

Table 4.9: ANOVA: Feature Selection, Data Sampling, Ensemble Techniques

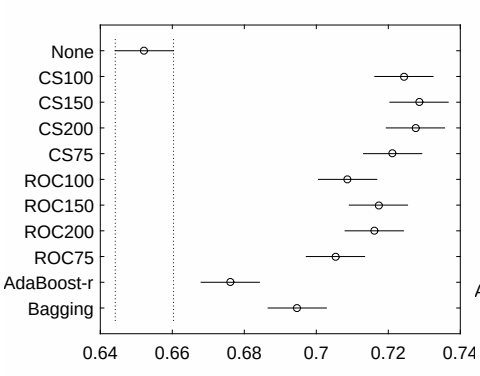
Dataset	Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
50:50	Technique	0.56842	10	0.05684	44.96	5.72053e-75
	Learner	0.5305	4	0.13262	104.9	1.41303e-75
	Error	1.37171	1085	0.00126		
	Total	2.47062	1099			
20:80	Technique	0.38647	12	0.03221	20.15	7.67218e-41
	Learner	0.4288	4	0.1072	67.07	1.41469e-51
	Error	2.05074	1283	0.0016		
	Total	2.86601	1299			
5:95	Technique	0.19025	12	0.01585	6.02	2.47777e-10
	Learner	0.25916	4	0.06479	24.62	1.16803e-19
	Error	3.37665	1283	0.00263		
	Total	3.82606	1299			

Additionally, the difference between rankers and feature subset sizes is not significant, except when comparing CS with 150 or 200 features to ROC with 100 or 75 features. Bagging performs significantly better than AdaBoost-r, and all versions of CS, ROC150 and ROC200 are significantly better than either ensemble technique.

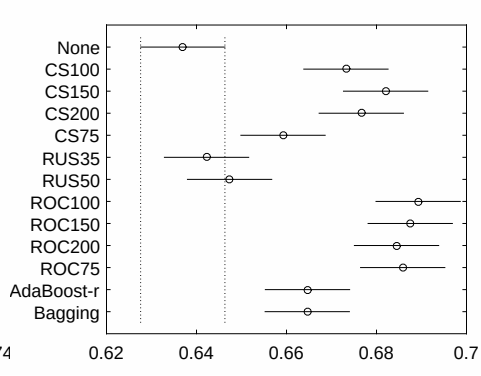
Figure 4.5b shows that using RUS35 or RUS50 are not significantly better than using "None". Again, ensemble techniques and feature selection techniques are significantly better than "None". There is no significant difference between subset sizes when using ROC; however, CS150 is significantly better than CS75. All four subset sizes in combination with ROC are significantly better than either ensemble technique.

The HSD test for the 5:95 imbalanced data is presented in Figure 4.5c. AdaBoost-r, RUS50 and CS150 do not significantly improve classifier performance, compared to "None". The remaining techniques significantly improve performance, but there is no significant difference between them.

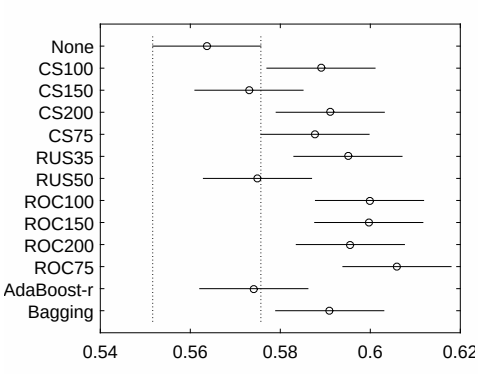
Figure 4.5: Tukey's HSD Results: Comparison of Techniques and Learners



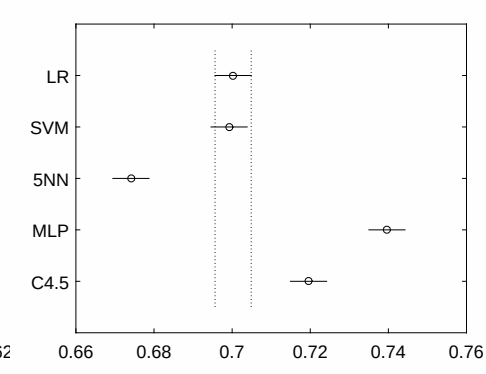
(a) Techniques on 50:50 dataset



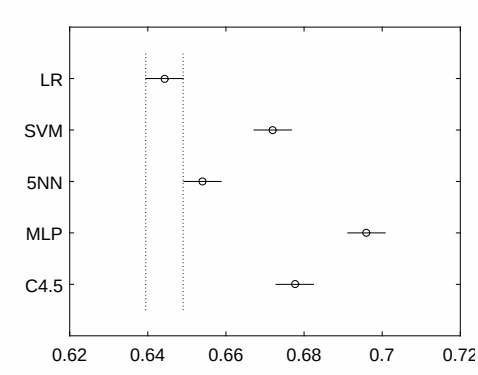
(b) Techniques on 20:80 dataset



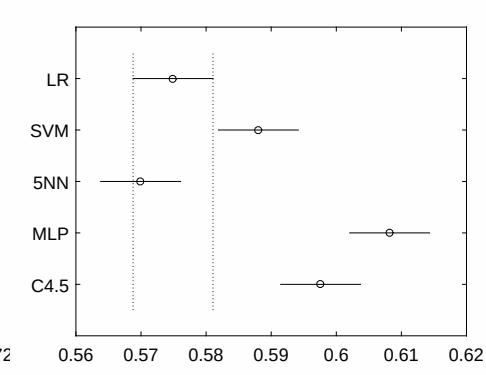
(c) Techniques on 5:95 dataset



(d) Learners on 50:50 dataset



(e) Learners on 20:80 dataset



(f) Learners on 5:95 dataset

Table 4.10: Select-Bagging and Select Boost Results: SemEval Dataset

Learner	Ensemble Technique	Feature Subset Size									
		200	150	100	75	50	25	20	15	10	5
C4.5	None	**0.77558**	0.77083	0.75395	0.74308	0.72853	0.70950	0.70122	0.68649	0.64752	0.57091
	Select-Boost	**0.82046**	0.80977	0.79736	0.78642	0.77414	0.75110	0.74746	0.73883	0.72237	0.67497
	Select-Bagging	**0.81125**	0.80187	0.78652	0.77390	0.75844	0.73466	0.72446	0.71039	0.69110	0.62366
MLP	None	**0.80413**	0.79516	0.76714	0.75915	0.74735	0.72330	0.71465	0.70019	0.67038	0.62316
	Select-Boost	**0.82653**	0.82175	0.80680	0.80200	0.78968	0.76317	0.75204	0.74002	0.72635	0.69279
	Select-Bagging	**0.82278**	0.81644	0.80172	0.79108	0.77322	0.74323	0.73227	0.71528	0.69260	0.65345
5-NN	None	0.65437	0.65847	0.66254	0.67081	0.68187	0.68394	**0.68455**	0.68086	0.66290	0.62319
	Select-Boost	0.69554	0.68789	0.69097	0.69801	0.70993	0.73375	**0.73523**	0.73350	0.72533	0.69660
	Select-Bagging	0.69022	0.69536	0.70227	0.70651	**0.71331**	0.71193	0.70858	0.70194	0.68993	0.65295
LR	None	**0.82529**	0.81657	0.80232	0.79186	0.77673	0.73925	0.72465	0.70628	0.67231	0.62216
	Select-Boostr	**0.84406**	0.83875	0.82854	0.81989	0.80436	0.76972	0.75982	0.74541	0.72907	0.69703
	Select-Bagging	**0.83458**	0.82597	0.81183	0.80021	0.78038	0.74678	0.73344	0.71650	0.69222	0.65189

4.5 SELECT-BAGGING AND SELECT-BOOST

Using the methodology described above, the "50:50 Sentiment140" dataset and the "SemEval unigram" dataset, we compared the performance of each base learner in combination with feature selection (using one of ten feature subset sizes) and either bagging, boosting, or no ensemble technique. Each combination of feature selection subset size and ensemble technique was trained and tested on both datasets separately. The results of our experiment are presented in Table 4.10 and Table 4.11, divided by dataset. AUC scores for each learner and ensemble technique are presented, with each score being the average across the models trained through CV. In each row the model with the highest AUC is indicated in **boldface**.

On the SemEval dataset, both Select-Bagging and Select-Boost improve classifier performance over "None" using any of the four base learners, with Select-Boost resulting in a greater increase in performance. Classifiers trained with LR, followed by MLP, had the best performance. The worst performing base learner was 5-NN. C4.5, MLP and LR all perform best with a feature subset size of 200 features, regardless of the ensemble technique chosen. For these three learners performance drops as the

Table 4.11: Select-Bagging and Select Boost Results: Sentiment140 Dataset

Learner	Ensemble Technique	Feature Subset Size									
		200	150	100	75	50	25	20	15	10	5
C4.5	None	0.68470	**0.68714**	0.68246	0.68444	0.68091	0.65872	0.65360	0.64240	0.62160	0.59501
	Select-Boost	0.73366	0.73471	**0.73485**	0.73181	0.71682	0.70548	0.69930	0.68592	0.66825	0.64798
	Select-Bagging	0.73674	**0.73756**	0.73198	0.72560	0.71579	0.69028	0.67776	0.66577	0.65616	0.63743
MLP	None	0.71591	**0.71606**	0.70027	0.69936	0.68255	0.65688	0.65454	0.64131	0.63004	0.60380
	Select-Boost	**0.75281**	0.75059	0.74757	0.74152	0.73249	0.70862	0.70144	0.68795	0.66892	0.64466
	Select-Bagging	**0.75811**	0.75317	0.74880	0.73809	0.72613	0.69521	0.68165	0.67134	0.65610	0.63763
5-NN	None	0.65308	0.65600	0.65642	**0.65928**	0.65528	0.65134	0.64495	0.63852	0.63532	0.60881
	Select-Boost	0.68478	0.68442	0.68551	**0.69701**	0.68821	0.69376	0.68722	0.68141	0.66804	0.64677
	Select-Bagging	0.68549	0.68822	0.69010	**0.69064**	0.68723	0.67440	0.66779	0.65901	0.65169	0.63464
LR	None	0.74623	**0.75048**	0.74559	0.73833	0.72451	0.68357	0.67196	0.66029	0.64496	0.60804
	Select-Boost	0.76138	**0.76479**	0.76169	0.75700	0.74431	0.71874	0.70604	0.69276	0.67345	0.64649
	Select-Bagging	0.76040	**0.76115**	0.75617	0.74765	0.73432	0.70074	0.68754	0.67284	0.66016	0.63619

Table 4.12: ANOVA: Select-Bagging and Select Boost for SemEval Dataset

Learner	Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
C4.5	Ensemble	.29148	2	0.14574	48.2	3.92546e-20
	Error	1.80504	597	0.00302		
	Total	2.09651	599			
MLP	Ensemble	0.17461	2	0.08731	31.92	6.78179e-14
	Error	1.63311	597	0.00274		
	Total	1.80772	599			
5-NN	Ensemble	0.20674	2	0.10337	168.5	9.54825e-59
	Error	0.36625	597	0.00061		
	Total	0.57299	599			
LR	Ensemble	0.13438	2	0.06719	19.17	8.49585e-09
	Error	2.09201	597	0.0035		
	Total	2.2264	599			

number of feature is reduced. 5-NN performs best with 20 features when using no ensemble technique or when using Select-Boost, and performs best with 50 features when using Select-Bagging. The best performing classifier was created by using LR with Select-Boost with 200 features; however, in this instance, using Select-Boost improved the classifiers AUC by less than 2% compared to using no ensemble technique. The greatest increase in AUC due to using an ensemble technique was observed when using Select-Boost with 5-NN. In this scenario, using an ensemble technique improved AUC by over 5% compared to using no ensemble technique.

The sentiment140 dataset yielded slightly different results than were observed on the SemEval dataset. Again using an ensemble technique, either Select-Bagging or Select-Boost, improved performance in comparison to using no ensemble technique for all base learners; however, on this dataset Select-Boost did not always achieve higher performance than Select-Bagging. For both C4.5 and MLP best performance was observed when using Select-Bagging, while for 5-NN and LR using Select-Boost achieved the best performance. Additionally, while the best feature subset size was uniform (excluding results for 5-NN) on the SemEval data, the best subset size for the sentiment140 dataset varied with both ensemble technique and learner. For C4.5, best performance using either no ensemble technique or bagging was achieved with a feature subset size of 150, while best performance using Select-Boost was achieved using 100 features. For classifiers trained using MLP, both ensemble techniques performed best with 200 features, while using 150 features achieved the highest AUC when using no ensemble technique. The best subset size for 5-NN was observed to be 75 features and the best subset size for LR was found to be 150 features. The highest AUC was achieved by using LR in combination with a 150 features and Select-Boost. In this instance, the use of an ensemble technique improved the AUC score by 1.5% in comparison to the best classifier using no ensemble technique and LR.

For both datasets, we observed that using ensemble techniques can improve clas-

Table 4.13: ANOVA: Select-Bagging and Select Boost Sentiment140 Dataset

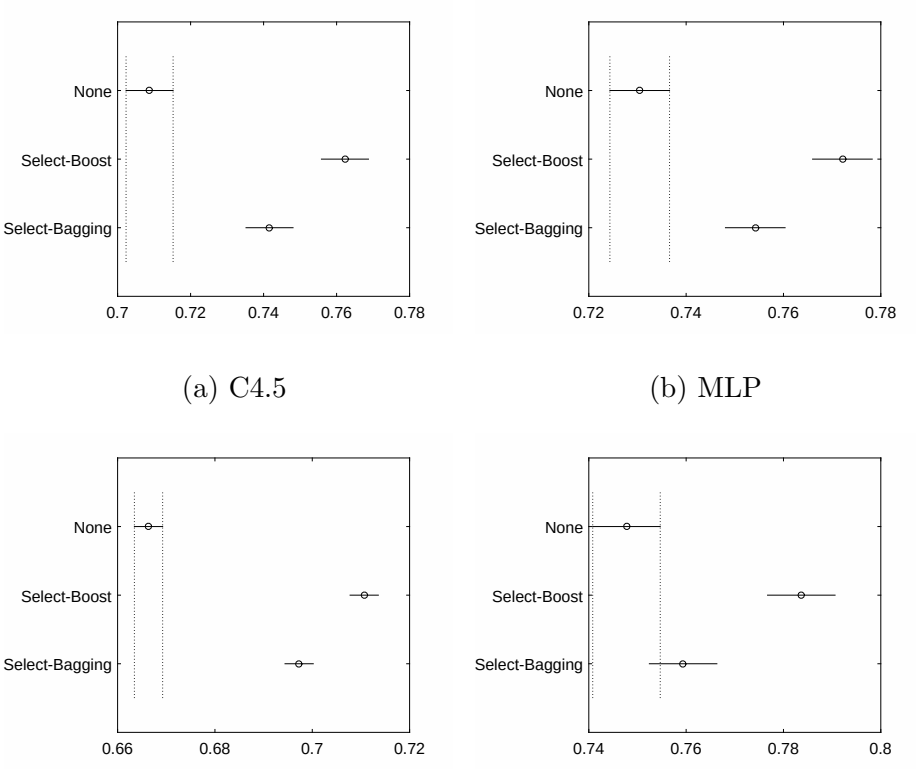
Learner	Source	Sum Sq.	d.f.	Mean Sq.	F	Prob>F
C4.5	Ensemble	0.22101	2	0.1105	79.56	2.34648e-31
	Error	0.82924	597	0.00139		
	Total	1.05025	599			
MLP	Ensemble	0.10553	2	0.05276	26.55	8.98525e-12
	Error	1.18644	597	0.00199		
	Total	1.29196	599			
5-NN	Ensemble	0.11371	2	0.05686	82.8	1.83524e-32
	Error	0.40995	597	0.00069		
	Total	0.52366	599			
LR	Ensemble	0.06051	2	0.03026	12.69	4.02047e-06
	Error	1.42379	597	0.00238		
	Total	1.4843	599			

sifier performance by as much as 5%; however, we also observed that there is less of a performance improvement when using the best performing base learner, LR. Both datasets show a similar performance difference between learners. LR was the best learner, followed by MLP then C4.5, while 5-NN is the worst performing learner. Using either Select-Bagging or Select-Boost decreases the performance gap between learners, however it never overcomes the differences between base learners.

4.5.1 Statistical Analysis

The statistical significance of our results was verified by performing ANOVA [10] with a 5% confidence level on each learner using MATLAB. The results are presented in Tables 4.12 and 4.13, and show the performance differences observed on both datasets are statistically significant. Additionally, we conducted Tukey's HSD tests [10] for Select-Boost, Select-Bagging and "None" on both datasets, divided by learner, allowing us to compare the performance of each technique. The results can be found in Figures 4.6 and 4.7. In each figure, the performance for each ensemble technique is shown, along with their confidence interval. If there is no overlap between the

Figure 4.6: Tukey's HSD Results: Select-Bagging and Select-Boost on SemEval



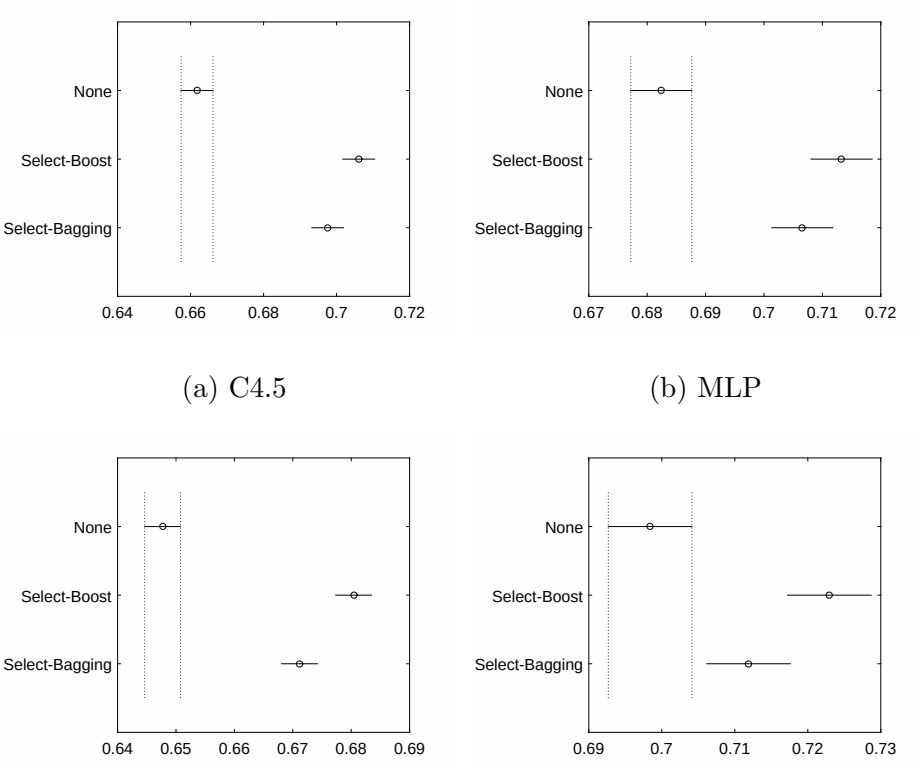
(a) C4.5 (b) MLP

(c) 5-NN (d) LR

confidence intervals of two techniques, then their averages are significantly different. Otherwise, they are not significantly different.

The HSD tests for each dataset and learner show that there is always an ensemble technique that is significantly better than using no ensemble technique. On the SemEval dataset, Select-Boost is significantly better than Select-Bagging and "None" for all learners. Select-Bagging is significantly better than "None" for all learners excluding LR. The sentiment140 dataset has similar results. Select-Boost has higher performance than Select-Bagging for all learners; however, this difference is only significant for 5-NN. Both Select-Bagging and Select-Boost perform significantly better than using feature selection with no ensemble technique for all learners on this dataset.

Figure 4.7: Tukey's HSD Results: Select-Bagging and Select-Boost on Sentiment140



(a) C4.5 (b) MLP

(c) 5-NN (d) LR

4.6 EMOTICON FEATURES

This section presents the results of our case study comparing the use of emoticon and unigram features with the SemEval datasets, and provides statistical tests verifying the significance of our findings. Additionally, a subsection is included that provides a discussion of which emoticons were useful and how they impacted feature selection. Classification performance results for each of the four learners using the full feature sets of extracted unigrams, emoticons, and the combination of unigrams and emoticons are presented in Table 4.14. The highest AUC value for each learner is highlighted in **boldface**.

Using both emoticons and unigrams results in the highest observed performance for C4.5 and 5-NN, but using emoticons alone performs better for LR and MLP.

Table 4.14: Classification Results Using Emoticons and Unigrams

learner	unigram	emoticons	both
C4.5	0.72120	0.58552	**0.73100**
LR	0.59256	**0.59766**	0.59146
MLP	0.46459	**0.59532**	0.44948
5-NN	0.64365	0.60078	**0.64895**

Table 4.15: Classification Results Using Emoticons and Unigrams with Feature Selection

FS Technique	Learner	Feature Subset Size							
		200		150		100		75	
		unigrams	both	unigrams	both	unigrams	both	unigrams	both
CS	C4.5	0.79051	**0.80275**	0.78915	**0.80133**	0.79002	**0.80588**	0.78515	**0.80022**
	LR	0.82424	**0.83693**	0.82136	**0.83511**	0.81376	**0.82910**	0.80762	**0.82276**
	MLP	0.81734	**0.83494**	0.80754	**0.82627**	0.79547	**0.81427**	0.78945	**0.80453**
	5-NN	0.71732	**0.72843**	0.72389	**0.73621**	0.74123	**0.75856**	0.74290	**0.75957**
ROC	C4.5	0.77558	**0.79030**	0.77083	**0.78532**	0.75395	**0.77154**	0.74308	**0.75696**
	LR	0.82529	**0.84138**	0.81657	**0.83394**	0.80232	**0.81827**	0.79186	**0.80655**
	MLP	0.80413	**0.82328**	0.79516	**0.81223**	0.76714	**0.78478**	0.75915	**0.77554**
	5-NN	0.65437	**0.66439**	0.65847	**0.66482**	0.66254	**0.66902**	0.67081	**0.67441**
PRC	C4.5	0.77584	**0.78561**	0.77070	**0.78116**	0.76801	**0.77757**	0.75741	**0.77278**
	LR	0.81668	**0.82951**	0.81437	**0.82736**	0.80739	**0.82141**	0.79713	**0.81140**
	MLP	0.78973	**0.77833**	0.78399	**0.80433**	0.77685	**0.79356**	0.76600	**0.78207**
	5-NN	0.65702	**0.66439**	0.66773	**0.67102**	0.68166	**0.68604**	0.69082	**0.69833**
MI	C4.5	0.77875	**0.78961**	0.77963	**0.78922**	0.78086	**0.79368**	0.77919	**0.79273**
	LR	0.81829	**0.83202**	0.81739	**0.83150**	0.81356	**0.82628**	0.80457	**0.81815**
	MLP	0.80627	**0.82130**	0.80321	**0.81989**	0.79473	**0.80979**	0.78516	**0.80364**
	5-NN	0.70395	**0.71214**	0.71127	**0.71823**	0.72582	**0.73768**	0.74061	**0.75505**
KS	C4.5	0.77558	**0.79030**	0.77083	**0.78532**	0.75395	**0.77115**	0.74308	**0.75696**
	LR	0.82529	**0.84138**	0.81657	**0.83394**	0.80232	**0.81829**	0.79186	**0.80655**
	MLP	0.80465	**0.82256**	0.79531	**0.81207**	0.76728	**0.78510**	0.75915	**0.77554**
	5-NN	0.65437	**0.66439**	0.65847	**0.66482**	0.66254	**0.66920**	0.67081	**0.67441**

While unigrams never have the highest performance, they have higher performance than using emoticons when using either C4.5 or 5-NN and better than using both types of features for LR and MLP.

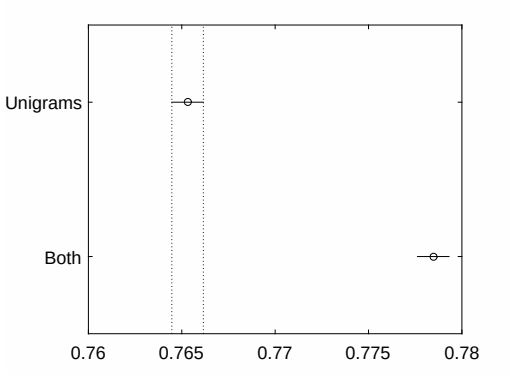
As seen in Table 4.15, when using Feature Selection (FS), the addition of emoticon features is beneficial for all rankers and feature subset sizes. Additionally, comparison of classifier performance in Table 4.14 and Table 4.15 show that using feature selection results in higher classification performance than was observed when using no feature selection. This matches our previous observations on the impact of feature selection on classification performance for these learners [38]. The highest performing classifier was trained using LR with 200 features and either KS or ROC as a ranker; however, CS and MI perform better than KS overall. While the highest performing classifier was trained using 200 features and using 200 features often offers the best performance, using 150 features offers comparable results. The difference between using unigrams and using both unigrams and emoticons is fairly small, generally resulting in a difference of AUC of between 1-2%.

4.6.1 Statistical Analysis

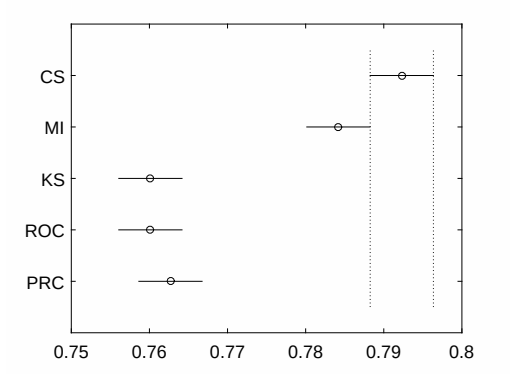
The significance of our results was tested using ANOVA to verify our observations about the impact of using emoticon features. The results for experiments with no feature selection and with feature selection are presented in Table 4.16 and show that there is no significant difference in classification performance between using any of the three feature sets (emoticons, unigrams or their combination). ANOVA results for feature selection and show that the choice of feature set (unigrams or both unigrams and emoticons), ranker, and feature subset size all significantly impact classification performance. To determine which factors offer the best performance, a Tukey's HSD test was conducted for each factor using MATLAB, presented in Figure 4.8.

Looking at the results of the HSD test in Figure , we observe that the increase in

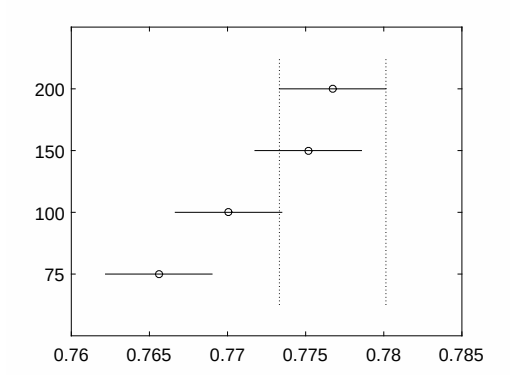
Figure 4.8: Tukey's HSD Results: Comparisons of Feature Sets with Feature Selection



(a) Feature Types



(b) Choice of Ranker



(c) Subset size

Table 4.16: ANOVA: Emoticons and Unigrams

Experiment	Source	Sum Sq.	d.f.	Mean Sq.	F	Prob¿F
No FS	Feature Sets	0.00593	2	0.00296	0.27	0.7619
	Error	2.58004	237	0.01089		
	Total	2.58597	239			
With FS	Feature Sets	0.13845	1	0.13845	49.15	2.88E-12
	FS	0.59492	4	0.14873	52.8	3.91E-43
	subset	0.06163	3	0.02054	7.29	7.15E-05
	Error	8.98858	3191	0.00282		
	Total	9.78359	3199			

performance due to using both emoticons and unigrams compared to using unigrams alone is significant. Figure 2 shows that CS is the best ranker, followed by MI and the remaining three are not significantly different. Finally, Figure 3 shows that while using 200 features yields the highest performance, it is not significantly better than using 150, though both are significantly better than using 75 features.

4.6.2 Emoticon Discussion

As seen in the results of our experiments, the inclusion of emoticons in combination with unigrams outperforms using unigrams alone when performing feature selection. Without feature selection, the addition of emoticons offers no significant improvement to classifier performance. As previously mentioned, using unigrams as features results in a very large number of features being extracted, and in our recent study, we established that high dimensionality negatively impacts classifier performance [38]. Our current experiments show that high dimensionality has the additional effect of hiding or diminishing the impact of including emoticons. Despite emoticons clearly containing valuable information on the sentiment of a tweet, their addition results in no significant change in classifier performance if feature selection is not used. It appears that high dimensionality must be addressed when adding additional types of features to unigrams to ensure the benefit of their inclusion can be observed.

Table 4.17: Emoticons selected using the Unigrams+Emoticons dataset

FS	subset size			
	75	100	150	200
CS	4	4	6	7
AUC	4	5	7	7
PRC	3	3	5	5
MI	4	4	6	7
KS	4	5	7	7

As discussed in the section on feature extraction, 45 emoticons were extracted as features; however, most of these emoticons were not included among the top features when conducting feature selection. Table 4.17 presents how many different emoticon features were selected for each subset size and ranker combination. It can be observed that the number of emoticons increases with feature subset size and that no more than seven were ever selected as part of the top 200 features. Using PRC results in the least amount of emoticons being selected.

Table 4.18 presents the emoticon features that were selected along with the frequency of their selection, and the emotion each emoticon is intended to portray. No more than seven of these emoticons were ever selected by a single ranker. Of the eight emoticons, “8D” and “XP” were ranked in the top 75 features by all five rankers and three more emoticons, “:3”, “=]” and “:-(”, were present in the majority of feature subsets. Interestingly, when examining these five emoticons, four are positive in sentiment while the remaining is negative. Additionally, when considering all eight emoticons, six are clearly positive while only one is clearly negative. This may suggests that positive emoticons were found to be more useful in predicting a tweet sentiment. Alternatively, many negative emoticons may not have sufficient frequency to be of good use, since there are roughly three times as many positive instances as negative in the SemEval dataset.

Table 4.18: Emoticon Count and Meaning

Emoticon	Count	Emotion
8-0	20	Surprise/Shock
XP	20	Cheeky/Playfull
:3	18	Happy
=]	16	Happy
:-(	12	Sad
XD	7	Laughing/Grin
8D	6	Laughing/Grin
:*	1	Kiss

CHAPTER 5
CONCLUSIONS AND FUTURE WORK

Current research has primarily focused on using feature engineering to achieve this; however the results show the type of features that yields the best performance is dependent on the dataset being used and most methods of feature engineering result in datasets suffering from high dimensionality. Additionally, current tweet sentiment dataset collection and labeling techniques may lead to quality of data concerns. Employing semi-supervised methods of collecting and labeling tweets leads to noisy data sets with mislabeled instances. Training datasets may also be class imbalanced, having a minority class and majority class(es). Feature selection, data sampling and ensemble techniques offer potential solutions to these concerns. Feature selection techniques select an optimal subset of features, reducing dimensionality by eliminating redundant or irrelevant features, and improving performance by reducing overfitting. Data sampling addresses class imbalance by creating a new, more balanced training set to reduce classifier bias towards the majority class. Ensemble techniques combine multiple classifiers into a single classifier that is more robust to class imbalance and noise, and offered better performance than any of the individual classifiers comprising it. Limited study of these techniques has been conducted in the domain of tweet sentiment classification; however, the current body of research lacks in depth studies and comparisons of these techniques. Additionally, very few techniques within each of these families have been implemented in this domain. This thesis seeks to provide a more thorough investigation of these techniques. For this purpose we conducted case studies evaluating the impact of various implementations of feature selection, data sampling and ensemble techniques.

5.1 CONCLUSIONS

Our first case study examined ten filter-based feature selection techniques and compared them against using no feature selection across four diverse learners. These techniques are used to select ten different feature subsets from the“ 50:50 sentiment140 corpus” dataset. Our experiments show that feature selection can significantly improve classifier performance for all learners. We observed no statistical difference between using 100, 150 and 200 features. Using feature selection to select 50 or fewer features generally results in poor performance, inferior to using no feature selection. Additionally it was found that only CS, KS, MI, PRC and ROC resulted in statistically significant performance improvements compared to using no feature selection. Additionally the difference between the top performing rankers, CS and MI, is not statistically significant; however, the gap between these and the other rankers is statistically significant.

Our second study implemented random undersampling in an effort to improve performance on class imbalanced data. Two post-sampling class distribution ratios were tested, 35:65 and 50:50. We found that RUS improves classifier performance on all datasets, and has a greater impact on severely imbalanced data. RUS routinely improved performance for five of the eight classifiers, and improved performance for seven of the eight learners on the 5% minority dataset. Unfortunately results for the 1% minority dataset are inconclusive, as this dataset contains an insufficient number of minority instances with which to train a classifier. For all imbalanced datasets (excluding the 1% minority dataset), RUS combined with NB learner yielded the best performance. The statistical significance of our results was tested and confirmed using ANOVA analysis and Tukey’s HSD test. These tests showed that RUS significantly improves performance compared to using no data sampling, however there is no significant difference between RUS50 and RUS35 for our data at any level of class imbalance.

In our case study on ensemble techniques we tested two popular techniques, bagging and boosting. We found that for the large, automatically labeled dataset, using bagging resulted in the greatest improvement in classifier performance for all base learners excluding 5NN. AdaBoost-r also improved performance for the majority of learners, but this was smaller than the increase observed when using bagging. On the smaller dataset, our results were less uniform; however, the performance of all base learners was improved by at least one of the ensemble techniques and bagging improved performance for four of the seven base learners (LR, MLP, RBF Network, and SVM). Testing the statistical significance of our results found that for four of the seven learners using an ensemble learner is significantly better than using no ensemble technique, and that there was no significant difference between ensemble techniques for any learner.

We also compared the impact of all three of these types of techniques against each other on the "50:50 Sentiment140", "20:80 Sentiment140" and "5:95 Sentiment140" datasets. We found, for all three datasets, that most learners achieved highest AUC values when using feature selection. Bagging offered similar increases in performance to many of the learners, but is less desirable as bagging increases computational costs associated with training a classifier, while feature selection reduces computational complexity. RUS and boosting were found to be less valuable than feature selection techniques or bagging, but still significantly improved performance in some scenarios. Additionally, this case study showed that the threshold based feature selection technique ROC performs better than CS on imbalanced data, despite CS being significantly better on balanced data.

An additional study was conducted combining the two most beneficial types of techniques, feature selection and ensemble techniques. These were compared against using only feature selection. On the manually labeled "SemEval unigram" dataset Select-Boost performed significantly better than Select-Bagging for all four learners.

Additionally, Select-Bagging performed significantly better than using no ensemble technique for three of the four base learners. While Select-Boost always performed better than Select-Bagging on the "50:50 Sentiment140" dataset, this difference was only significant for one learner; however, both ensemble techniques performed significantly better than feature selection with no ensemble technique. While Select-Boost being better for the manually labeled SemEval dataset is not entirely unexpected, it is surprising that Select-Boost is better than Select-Bagging on both datasets. Previously, we observed bagging to be better than boosting for this dataset. These results indicate that the introduction of a threshold based feature selection technique and reduction of dimensionality changes the behavior of classifiers trained on tweet sentiment datasets.

Finally, in an effort to determine if the inclusion of emoticon features significantly improves the performance of tweet sentiment classifiers, we compared the performance of classifiers trained using emoticons, unigrams and the combination of emoticons and unigrams. We also trained classifiers using feature selection to address the potential concern of high dimensionality hiding the benefit of including emoticon features.

When using no feature selection, we found no significant difference between using the three feature sets; however, experiments using feature selection determined that using emoticons significantly improved classification performance. Our experiments showed that for all tested combinations of learners, feature selection techniques and subset sizes, using both emoticons and unigrams results in higher performance than using only unigrams. Additionally, the difference between these feature spaces is significant. Further examination of adding emoticons when conducting feature selection showed that only eight of the 45 emoticons in the dataset were responsible for the observed increase in performance, suggesting that the others are either too infrequent in occurrence, or unclear in their use to be helpful when determining the sentiment of a tweet.

From these results, we conclude that feature selection techniques are effective in helping to alleviate problems associated with high dimensional datasets within this domain and should be used in all future experiments. Feature selection significantly improved performance on every dataset tested while reducing the computational resources required for training of classifiers. We recommend using feature selection in order to gain the full benefit of including additional types of features. Both choice of ranker and choice of subset size have significant impact on classifier performance. We recommend five of the rankers, CS, KS, MI, PRC and ROC as they have all been shown to significantly improve performance. Choice of ranker depends on the dataset. Chi squared is best for balanced data while threshold based feature selection techniques perform better on imbalanced data. Ensemble techniques can also significantly improve classifier performance. Experiments using the sentiment140 datasets show that bagging routinely outperformed boosting, supporting previous findings that it is better than boosting for large, potentially noisy, data; however, when combining bagging and boosting with feature selection we found that Select-Boost was better than Select-Bagging. As feature selection should be considered mandatory, we recommend Select-Boost over Select-Bagging. While RUS can be used to significantly improve classifier performance, it is not as beneficial as using feature selection or ensemble techniques.

5.2 FUTURE WORK

The case studies presented in this thesis provide a basis for future experimentation in the domain of tweet sentiment classification, or related application domains. Additionally, our experiments were limited by availability of labeled training data, and primarily relied on unigram features. Potential avenues for future work building off of, or complimenting the research presented in this thesis are included below.

- In this thesis we have investigated feature selection, data sampling, ensemble

techniques and the combination of feature selection and ensemble techniques. It is also important to investigate other combinations of these techniques including data sampling and feature selection, data sampling and ensemble techniques, and the combination of all three types of techniques.

- We conducted a study on combining unigram and emoticon features, and found that it is important to use feature selection when combining sets of features. Additional methods of feature engineering and combining the many types of feature that can be extracted from tweets should also be investigated in tandem with feature selection in order to improve performance and determine what types of features are most important.

- Our studies were conducted on datasets created from the sentiment140 corpus and SemEval 2015 tweet sentiment data. Additional research should seek to find or create additional tweet sentiment datasets that can be used in future studies. Additionally, datasets from similar application domains could be used with the methodology proposed in this thesis to see if our results generalize to related domains.

BIBLIOGRAPHY

[1] Ahmed Abbasi, Hsinchun Chen, and Arab Salem. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems (TOIS)*, 26(3):12, 2008.

[2] Hervé Abdi and Lynne J Williams. Tukeys honestly significant difference (hsd) test. *Encyclopedia of Research Design. Thousand Oaks, CA: Sage*, pages 1–5, 2010.

[3] Ahmad Abu Shanab, Taghi M. Khoshgoftaar, Randall Wald, and Amri Napolitano. Impact of noise and data sampling on stability of feature ranking techniques for biological datasets. In *2012 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 415–422, August 2012.

[4] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, pages 30–38. Association for Computational Linguistics, 2011.

[5] Ameeta Agrawal and Aijun An. Kea: Sentiment analysis of phrases within short texts. *SemEval 2014*, page 380, 2014.

[6] Amir Asiaee T., Mariano Tepper, Arindam Banerjee, and Guillermo Sapiro. If you are happy and you know it... tweet. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1602–1606, 2012.

[7] Sitaram Asur, Bernardo Huberman, et al. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE, 2010.

[8] Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 36–44. Association for Computational Linguistics, 2010.

[9] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1):20–29, 2004.

[10] M. L. Berenson, M. Goldstein, and D. Levine. *Intermediate Statistical Methods and Applications: A Computer Package Approach 2nd Edition*. Prentice Hall, 1983.

[11] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[12] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011.

[13] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[14] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[15] Wilas Chamlertwat, Pattarasinee Bhattarakosol, Tippakorn Rungkasiri, and Choochart Haruechaiyasak. Discovering consumer insight from twitter via sentiment analysis. *J. UCS*, 18(8):973–992, 2012.

[16] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics, 2010.

[17] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[18] D. J. Dittman, T. M. Khoshgoftaar, R. Wald, and J. Van Hulse. Comparative analysis of dna microarray data through the use of feature selection techniques. In *Proceedings of the Ninth IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 147–152. ICMLA, 2010.

[19] Lucie Flekova, Oliver Ferschke, and Iryna Gurevych. Ukpdipf: A lexical semantic approach to sentiment polarity prediction in twitter data. *SemEval 2014*, page 704, 2014.

[20] George Forman. An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, pages 3:1289–1305, 2003.

[21] Rudolf Jakob Freund and Ramon C Littell. *SAS for linear models: a guide to the ANOVA and GLM procedures*, volume 1. Sas Institute, 1981.

[22] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics, 2011.

[23] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, pages 1–12, 2009.

[24] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, March 2003.

[25] Asif Hassan, Ali Abbasi, and Deze Zeng. Twitter sentiment analysis: A bootstrap ensemble framework. In *Social Computing (SocialCom), 2013 International Conference on*, pages 357–364. IEEE, 2013.

[26] T. M. Khoshgoftaar, D. J. Dittman, R. Wald, and W. Awada. A review of ensemble classification for dna microarrays data. In *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, pages 381–389, Nov 2013.

[27] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! *ICWSM*, 11:538–541, 2011.

[28] Shoushan Li, Zhongqing Wang, Guodong Zhou, and Sophia Yat Mei Lee. Semi-supervised learning for imbalanced sentiment classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1826, 2011.

[29] Huan Liu and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.

[30] Yang Liu, Xiangji Huang, Aijun An, and Xiaohui Yu. Arsa: a sentiment-aware model for predicting sales performance using blogs. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 607–614. ACM, 2007.

[31] Catie Meador and Jonathan Gluck. Analyzing the relationship between tweets, box-office performance and stocks. *Methods*, 2009.

[32] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. Fast and accurate sentiment classification using an enhanced naive bayes model. In *Intelligent Data Engineering and Automated Learning–IDEAL 2013*, pages 194–201. Springer, 2013.

[33] Nuno Oliveira, Paulo Cortez, and Nelson Areal. On the predictability of stock market behavior using stocktwits sentiment and posting volume. In *Progress in Artificial Intelligence*, pages 355–365. Springer, 2013.

[34] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[35] Robin L Plackett. Karl pearson and the chi-squared test. *International Statistical Review/Revue Internationale de Statistique*, pages 59–72, 1983.

[36] Foster J Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML*, volume 98, pages 445–453, 1998.

[37] Joseph D Prusa, Taghi M Khoshgoftaar, and David J Dittman. Using ensemble learners to improve classifier performance on tweet sentiment data. In *Proceedings of the16th IEEE International Conference on Information Reuse and Integration*, August 2015.

[38] Joseph D Prusa, Taghi M Khoshgoftaar, and David J Dittman. Impact of feature selection techniques for tweet sentiment classification. In *Proceedings of the 28th International FLAIRS conference*, pages 299–304, May 2015.

[39] Joseph D Prusa, Taghi M Khoshgoftaar, David J Dittman, and Amri Napolitano. Using random undersampling to alleviate class imbalance on tweet sentiment data. In *Proceedings of the16th IEEE International Conference on Information Reuse and Integration*, August 2015.

[40] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In *Encyclopedia of database systems*, pages 532–538. Springer, 2009.

[41] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval*, 2015.

[42] Yvan Saeys, Iaki Inza, and Pedro Larraaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[43] Hassan Saif, Yulan He, and Harith Alani. Alleviating data sparsity for twitter sentiment analysis. CEUR Workshop Proceedings (CEUR-WS. org), 2012.

[44] Hassan Saif, Yulan He, and Harith Alani. Semantic sentiment analysis of twitter. In *The Semantic Web–ISWC 2012*, pages 508–524. Springer, 2012.

[45] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Andres Folleco. An empirical study of the classification performance of learners on imbalanced and noisy software quality data. *Information Sciences*, 259:571–595, 2014.

[46] Nádia FF Silva, Eduardo R Hruschka, and Estevam Rafael Hruschka Jr. Biocom usp: Tweet sentiment analysis with adaptive boosting ensemble. *SemEval 2014*, page 123, 2014.

[47] Jason Van Hulse, Taghi M. Khoshgoftaar, and Amri Napolitano. Experimental perspectives on learning from imbalanced data. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 935–942, New York, NY, USA, 2007. ACM.

[48] Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 115–120, 2012.

[49] Gary M Weiss and Foster J Provost. Learning when training data are costly: the effect of class distribution on tree induction. *J. Artif. Intell. Res.(JAIR)*, 19:315–354, 2003.

[50] I. H Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd edition, 2011.