Exploring Rich Features for Sentiment Analysis with Various Machine Learning Models

Shuyang Li

Advisor: Xiaoyan Li

Submitted in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Engineering

Department of Operations Research and Financial Engineering

Princeton University

April 2016

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Shuyang Li

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Shuyang Li

Abstract

Understanding sentiment is an important task in natural language processing. In this paper we investigate the use of rich features to extend the bag-of-words model for sentiment analysis using machine learning in the movie review domain. We focus on the areas of subjectivity analysis, negation handling, and aggregate document features, and we investigate three ensemble methods and four singular classifiers. Our experimental results show that AdaBoost performs best among all classifiers on the simple unigram feature set, while the Maximum Entropy classifier provides best performance on our enhanced feature sets. Stochastic Gradient Descent is nearly as accurate as AdaBoost and significantly faster.

We also examine 128 commonly misclassified reviews and identify additional challenges to NLP in the movie review domain. We have been able to increase classifier performance through the addition of aggregate document polarity and purity features and summary sentence features based on manual subjectivity and summary sentence extraction. From this, we see potential to improve classification accuracy through improved automatic subjectivity analysis methods and summarization. Additional gains may be made by using a domain-specific polarity lexicon to generate aggregate features.

We created a manually labeled set of subjective and summary sentences for each review in our corpus. This may serve as a useful benchmark dataset for future work in subjectivity analysis. Using the manually labeled corpus solely to restrict the feature space reduces classifier performance, while using it as a base to generate aggregate features improves accuracy. We also see that using manual subjectivity analysis for both feature restriction and aggregate feature generation further improves classification performance. This suggests that subjectivity analysis is useful for generating rich features as well as for feature space restriction.

Acknowledgements

To my thesis advisor, **Dr. Xiaoyan Li**, who worked with me closely on this thesis every week since September and lent me her wisdom and experience in the field.

To **Eric Huang**, our shared thesis struggles, and making it through ORFE together. Shout out to your masterful cooking in Spelman 16.

To **Xiaonan April Hu** and your infinite patience, who kept me strong at my lowest. You have been a source of joy and inspiration since freshman year.

To **Micah Iticovici**, who has been an overwhelming wellspring of support since we met in middle school. Your chocolate-providing skills are second to none.

To **David Zhao** and **Timothy Seah**, who left the hallowed concrete of Spelman oh so soon. Princeton is but the first chapter in our illustrious friendship.

To **Maggie Zhang**, who kept me motivated and sane through the thesis process.

To **Nancy Xiao** and **Max Kaplan**, who constantly reminded me of the pleasures of good food, good friends, and good conversation.

To **Andy Zhou** and **Darek Johnson** for providing some much-needed distraction when the stress nearly got to me.

To **Hana Ku**, **Tony Jin** and **Alicia Lai** for feeding my snack and bubble tea addiction. I was able to stay healthy and energized thanks to you.

To **Princeton Badminton Club**, the fun of our out-of-town tournaments, and the sweat, soreness, and euphoria of staying (somewhat) in shape.

To **Princeton Tower Club**. It is an honor to call myself True Blue.

Finally, to my **parents**, who have loved and supported me unconditionally for the past two decades and counting.

Table of Contents

Figures

1 Introduction

1.1 The Demand for Opinions

As producers, consumers, and analytical beings, humans value the opinions of others when making decisions of their own. Roman magistrates were elected not only on the basis of their policies, but also on their popularity and reputation among the people. Long before the age of the internet, merchants and nobles wielded influence commensurate to the population that respected and trusted them. Trading and exploratory expeditions chose their destinations based off of the relative popularity of products.

With the advent of the internet, the opinions of reviewers have become all the more important for both manufacturers and other consumers. According to a survey of over 2,000 American adults, 81% of internet users have researched a product online, and over 73% reported reviews significantly influencing their purchasing decisions [1]. Indeed, customers report a willingness to pay over 20% more for a 5-star-rated item than a 4-star-rated item. Additionally, a 2008 Pew Center survey of 2,400 adults indicates that about 30% of online users have commented about or reviewed a product they bought or service they received online [2]. As modern e-commerce evolves, consumers can now rate more than just products—they can review the markets and sellers as well. Studies of large-scale online reputation and review systems, such as the one used by e-commerce site eBay, show that the reputation of individual sellers can predict future sales performance [3]. This reputation system is based on a large network of reviews left by buyers. Ba and Pavlou suggest that consumer-seller trust allows more reputable sellers to charge higher prices, due to the reduced risk of the transaction [4].

While reviews and reputation systems affect businesses through market and consumer impact, corporations also seek to directly leverage consumer opinion. Understanding consumer opinion and reactions allows businesses to make informed decisions about advertising campaigns, product features, and even help them discover untapped markets. In this paper, we investigate sentiment analysis, which is the field of Natural Language Processing (NLP) that seeks to identify positive and negative sentiment within text. Here, we use machine learning methods to automatically classify documents as expressing positive or negative opinion. This allows for large-scale textual analysis, and gives users an ability to quickly and efficiently extract opinion statistics from large corpuses of unlabeled text. We focus on a bag-of-words representation of text documents, which separates the document into word, phrase, and sentence tokens. While this is less understandable from a human standpoint, it allows for fast and varied generation of machine-interpretable features.

At first glance, sentiment analysis on text may seem inefficient, considering the multitude of sites like Gadgetfreaks, TripAdvisor, and Rotten Tomatoes that aggregate user reviews. These review aggregators contain not only reviews, but also numerical ratings and provide an averaged score for each product or service being reviewed. However, we must recognize the importance of review text, as humans frequently express informative opinions without an accompanying numeric rating. Many customers who purchase products refer to these products, their experiences, and their overall opinion in context of normal online conversation. Reddit threads like *"What is the best purchase you have ever made?"* contain tens of thousands of comments revealing customer sentiment, but very little in the way of explicit numerical ratings [5]. Indeed, there is no standardized system on many websites to display numerical rating/polarity data.

We see then that review and text in general is important to inform human decision-making. Chevalier and Mayzlin find that customers looking to buy books from Amazon and Barnes and Noble read review text rather than making their decisions exclusively based on the numerical rating [6]. Correspondingly, there has been an increase of interest in how to mine the sentiment polarity of reviews and texts to reveal consumer sentiments. In this automated sentiment analysis, natural language processing is performed on the text of a review to extract meaning and polarity. This type of textual analysis better mimics human behavior and decision-making patterns than simple summary statistics. Rahmath and Ahmad explore sentiment analysis for e-commerce product reviews [7]. Many influential reviewers also express their unlabeled opinions on blogs and sites like Instagram or Twitter. With hundreds of millions of posts daily, and numerous accounts dedicated to product reviews (@productreviews, @nybooks, and @MovieCriticFeed among them), Twitter is a data trove for companies that want to track reception of their products [8]. Effective sentiment analysis algorithms can yield important insights when applied to this enormous dataset, which in turn can motivate and inform publicity and product campaigns.

We consider a hypothetical situation: a coffee machine is received positively overall by consumers but a majority of them disliked that the water reservoir was not removable. This would most likely be reflected not only in a majority of Amazon reviews on the item but also various online forum threads. The manufacturer could spend time and money to conduct a survey of people who have purchased their product in order to gain this insight. If they were to leverage sentiment analysis, however, the process would be much more streamlined, costing less time, manpower, and money. The company could scrape various review sites, Amazon, and Reddit to grab reviews of their product that contain a reference to a single feature (in this case the water reservoir). Then, they could run sentiment analysis to classify these reviews and quickly learn that,

for example, 85% of reviews of their coffee machine that refer to the water reservoir are negative—even if the product itself has a high average numerical review on Amazon.

The applications of sentiment analysis are not, however, limited to the economic or corporate field. Current research attempts to model stock market behavior and financial trends through opinion mining of forum posts [9]. Similarly, the content and orientation of political tweets are important indicators of public sentiment for campaigns and officials. A study of tweets during the 2009 German parliamentary election indicates that the online sentimental landscape closely reflects the offline landscape [10]. We see thus that all manners of individuals and organizations are part of the demand for automated systems capable of analyzing and reporting sentiment. We hope that our work will aid in the development of more accurate and efficient frameworks for sentiment analysis.

1.2 Challenges in Sentiment Analysis

Liu defines sentiment analysis as such: an opinionated document is written by an opinion holder and consists of a finite set of features consisting of words or phrases. Subsets of these features express direct opinions about a single object or opinions comparing two objects (comparative opinions). Sentiment analysis is thus the task of identifying the tuples of features, opinions, phrases, and opinion holders [11].

Sentiment analysis faces the same problems inherent to natural language process in general. Human readers are particularly good at resolving co-references and anaphora, being able to pinpoint which pronouns refer to existing references or objects [12]. They can also understand when qualities of a parent object can be applied to its constituent parts. Words with multiple meanings are also difficult to resolve – usage in one context can indicate positive opinion (a "strong adhesive") while another can have the opposite effect (a "strong fishy odor").

Sarcasm has also been a persistent problem in natural language processing and naturally proves worrisome for sentiment detection. Current attempts at recognizing sarcasm rely on such methods as detection of a positive sentiment in a negative situation, which points to a circular problem of sarcasm recognition in sentiment analysis requiring robust and accurate sentiment analysis techniques to begin with [13].

A coarse feature space also presents problems for building a deep understanding of the text. Term-based feature spaces can miss out on general concepts within the text. A bag-of-words representation of words or terms as features discards information about order and sentence structure. This in turn makes co-reference and anaphora resolution more difficult. We face the challenge of detecting implicit meaning, which often requires a more detailed feature space.

1.3 Our Proposal

We propose first to investigate the impact of various preprocessing, subjectivity analysis, and feature selection techniques on the accuracy and performance of sentiment classification methods. Pang et al. tested several weak classifiers on movie reviews from the Internet Movie Database (IMDB) and found machine learning methods outperformed simple human-informed rule-based sentiment analysis [14]. We propose to assess several singular classifiers and ensemble methods on each feature set and use stratified 10-fold cross-validation to investigate performance. For singular classifiers we will use Naïve Bayes, Support Vector Machines (SVMs), the Maximum Entropy Classifier (ME), and Stochastic Gradient Descent (SGD/SGDC). We propose to investigate Random Forests, AdaBoost, and Additive Linear Regression (ALR) for ensemble methods.

Our base feature space is the bag-of-words representation of the vocabulary of our entire corpus. We thin this using preprocessing techniques including stopword removal, part-of-speech tagging, and lemmatization. We will also investigate subjectivity detection using several methods: Naïve methods based on presence/frequency of adjectives per sentence, the subjectivity analysis tool OpinionFinder to isolate subjective sentences from each review, and the utility TextBlob for numerical subjectivity and polarity extraction based on the WordNet lexicon [15] [16]. We will also evaluate an altered negation-handling method to understand the effects of limiting negation scope.

We will investigate the usage of semantic/aggregate features, including average word and sentence polarity of the entire review, review purity, and first/last sentence polarity. The purpose of these features is to evaluate ways to extend the bag-of-words model. We will evaluate a number of classes of aggregate features based on polarity and purity for each classifier.

Finally, we seek to understand if the performance obtained by incorporating subjectivity analysis is limited by our specific methods (WordNet via TextBlob, OpinionFinder). In this vein, we will manually label each sentence in our movie reviews as objective or subjective, as well as labeling one summary sentence for each review that is the strongest indicator of that review's overall polarity. Using this theoretically "ideal" subjectivity analysis method, we will then construct our second bag-of-words corpus using only the subjective lines from reviews. On this manually labeled corpus, we will re-evaluate the performance of each classifier. We will additionally construct a new set of aggregate features using the manually labeled corpus as a base, and evaluate their performance when combined with both the full and manually labeled corpus as a base bag-of-words set.

2 Literature Review

The field of analyzing text to ascertain sentiment has been in development for well over a decade under the banners of *Sentiment Analysis* and *Opinion Mining*. Work has mainly focused on supervised learning methods as applied to labeled data, including reviews for a variety of items: Amazon e-commerce items, IMDB movie reviews, restaurant reviews, and various rating blog posts. Here we utilize a dataset of movie reviews that has been well-explored in sentiment analysis research since it was collected by Pang et al. in 2002 [14].

We propose to tackle several of the problems facing sentiment analysis and, as a whole, natural language processing: co-reference resolution, where multiple words refer to the same entity; negation handling; word-sense disambiguation for words with multiple meanings; and anaphora resolution, resolving pronouns that point to previously referenced entities. To accomplish this we use several preprocessing steps taken from prior research in the field, including lemmatization, part-of-speech tagging, and stopword removal.

Cambria et al., in providing a survey of research in the field, note four main approaches: keyword spotting, which looks for the presence of affect words and intensity/negation modifiers; lexical affinity, which assigns arbitrary words a probability of affiliation with an emotion; statistical methods that learn the sentiment polarity and intensity of features within a document; and concept-based approaches which use large knowledge bases to explore language [12]. While the latter two approaches seek to better approximate the structural nature of our language, keyword spotting and lexical affinity remain attractive candidates due to their fundamental simplicity.

Pang, et al. tested several linear classifiers to evaluate statistical methods and feature selection for sentiment analysis. Naïve Bayes was conducted using relative frequency estimation of class (positive/negative) and feature frequency. The Maximum Entropy approach relies on Feature/Class functions and their associated parameter weights, tuned using an iterative scaling algorithm. SVM^{light} was used to perform Support Vector Machine learning on the features. The authors tested different types of features – unigrams, bigrams, part-of-speech labeling, and varying if frequency or presence of features was recorded in the vector representation of the documents. Best performance was obtained by using the presence of unigrams as features [14].

Harb et al. investigated keyword spotting through association rule mining. The authors relied upon the co-occurrence of positive and negative adjectives near respective representative words from a small seed set of adjectives [17]. Church's Mutual Information criteria was then

used to thin the association rule list and create the full set of keywords. The accuracy of association rule mining here varies for the sentiment label, since the frequency of positive and negative adjectives vary within their respective corpora.

Turney and Littman tested two methods of lexical affinity: Pairwise Mutual Information (PMI) and Latent Semantic Analysis (LSA). The semantic orientation of a word is obtained by the difference of the sums of positive and negative word orientations. A base set of positive and negative words is used as the basis of comparison for PMI, and Church's PMI is used as a measure of statistical independence from the words in the base set. LSA uses singular value decomposition on a matrix of words x chunks of text within the document (sentences or phrases) with elements being term frequency-inverse document frequency (tf-idf). Sentiment orientation classification reaches up to 95% accuracy when "mild" words are ignored [18].

Usage of ensemble methods in sentiment analysis is not without precedent; Silva et al. showed that AdaBoost performs well for sentiment analysis on microblogs and Twitter when boosting Naïve Bayes and SVMs [19]. Gokulakrishnan et al. used Random Forests to classify a stream of tweets [20]. Given the limited length of tweets and other microblogs, we aim to examine the performance of these ensemble classifiers on longer, more complex text in the form of movie reviews.

While using the full text of documents can provide a large initial feature space, it can also inflate complexity and runtime if the documents are very long. Pang and Lee noticed that movie reviews tend to contain objective sections describing plot points, and proposed retaining only subjective sections of reviews for sentiment analysis [21]. The proposed method for subjectivity detection relied on minimum cuts on a graph, with sentences as nodes and edge weights determined by physical proximity. A simple linear classifier was used to connect nodes to subjective and objective root nodes, and the minimum cut problem was solved to partition the sentences into subjective and objective subsets. The authors found that subjectivity detection provided increased efficiency and speed, and resulted in equal or better performance for several linear classifiers.

Recent research has focused on domain-independent sentiment classifiers. Such classifiers would drastically improve application efficiency and enable users to use a single method of training rather than create a unique training methodology for each domain in question. Harb et al. discuss a method of domain-independent automatic opinion extraction called AMOD [17]. In the AMOD approach, a training corpus is gathered by specifying a domain-independent set of positive and negative seed words combined with a domain identifier.

While the majority of sentiment analysis research has focused on supervised approaches, unsupervised models have also been investigated. Lin et al. compared several domain-independent Bayesian models for unsupervised learning: latent sentiment model (LSM), joint sentiment-topic model (JST), and reverse-JST [22]. The paper proposes that JST is the most appropriate model for jointly detecting sentiment and topic in text documents.

Kim et al. take a different approach to improving classification accuracy: changing the way emotions and sentiments are represented. They propose to replace the standard binary or ternary sentiment state (positive, [neutral], negative) with a continuous mood manifold [23]. One would then model the stochastic relationship between document, emotion label (happy, sleepy), and projection on the mood manifold. While Kim et al.'s work has implications for multidimensional emotion analysis and classification, we will utilize a positive/negative binary sentiment in this research project.

3 Data Processing

3.1 Dataset

For our experiments, we chose to work in the movie review domain. This is a well-studied domain in the field of sentiment analysis, and the reviews tend to be labeled. Thus, the domain is particularly well-suited for supervised learning techniques such as the ones we investigate. We note that movie reviews seem to be a particularly challenging domain for sentiment analysis, even among other review types [24].

Here we use the second version of the Cornell Movie Review corpus [21]. This contains 2000 randomly selected reviews from IMDB. Half of the reviews are positive and half of the reviews are negative. While the original reviews did not contain a consistent rating system, Pang and Lee labeled the documents as positive and negative based off of numerical ratings present throughout the reviews. For example, a 3.5/5.0 star review is labeled as positive. After labeling the documents, the numerical ratings were stripped from the document. The dataset is divided into ten equal-sized folds with balanced class distributions. All results reported are the average ten-fold cross-validation results on this data.

In its raw text form, each review consists of a number of word and punctuation tokens, with each token separated by a space. Each sentence is separated by a line break.

Figure 1 shows general statistics for our corpus, including the average number of words, unique words, and sentences per review, as well as the shortest and longest reviews by word and sentence-count. We see that negative reviews tend to be shorter than positive reviews, and that there also happens to be a single-line negative review of a movie.

General Statistics		Positive (1000)	Negative (1000)	All Reviews (2000)
Words	Average	707.18	634.35	670.76
	Avg. Unique	348.22	323.15	335.69
	Shortest	119	17	17
	Longest	2471	1903	2471
Sentences	Average	33.94	31.78	32.36
	Shortest	5	1	1
	Longest	112	112	112

Figure 1: Word and sentence statistics in movie review corpus

3.2 Term-Based Feature Extraction

One feature space we are interested in investigating is the term-based (word-level) feature space. In term-based feature selection, we use a bag-of-words model of each document and

represent the feature set as a vector of term n-grams. For several machine learning methods under investigation, we will consider both frequency and presence feature sets. Pang et al. reported using presence of term unigrams instead of frequency results in better performance for the Naïve Bayes, SVM, and Maximum Entropy classifiers [14]. We aim to extend this analysis to ensemble methods and other singular classifiers.

3.2.1 Tokenization

To generate our term-based feature space, we first tokenize each document at the sentence-level and word-level. The text files processed by Pang and Lee were already tokenized in sentence form, with each sentence on a separate line. We collected the sentences from the training corpus by using *readlines()* in Python. We use *word_tokenize* from NLTK to obtain word tokens.

3.2.2 Stop Word Removal

Certain words known as stop words are first filtered out of our term-based vocabulary. These are some of the most common words in a language, as well as containing some common punctuation that tends to be isolated during the tokenization phase. This is a well-established first step in natural language processing, and we use the WordNet Stop Word Corpus to remove stop words. We preserve stop words and punctuation when engineering phrase-level and higher-level feature sets, however.

Our stop word corpus includes pronouns (I, me, myself, you, our, he), as well as short function words (is, be, are, was) and prepositions (through, during, before, after). In general, these words serve to maintain grammatical coherence and lend grammatical structure to a sentence without adding significant meaning. We note that the WordNet corpus does contain several negations ("not", "nor", "no") in the stop word list, and as such we removed these from the list so as not to interfere with negation handling. For bigrams and higher-level feature sets, we thus keep stop words, since they can add context to the bigram: "[this movie] is bad" vs. "[it was] his bad".

3.2.3 Part-of-Speech Tagging

We investigate part-of-speech (POS) tagging in generating our feature set. This is one step in word sense disambiguation—some words, when used as different parts of speech, may convey opposite polarity sentiments. We note that prior research suggests that part-of-speech features may not be useful in sentiment analysis in the microblogging/Twitter domain [25]. However, while microblogging sentiment analysis resembles sentence-level sentiment classification, we propose that part-of-speech tagging can aid in feature disambiguation in our movie review domain, especially as we have found the average review in our corpus to contain 621

words—significantly longer than microblogs and Twitter posts (which are limited to 140 characters). Indeed, part-of-speech tags have been considered a first step in semantic disambiguation [26].

We will use the part-of-speech tagger from the Natural Language Toolkit (NLTK) Python package. The part-of-speech tags are from the Penn Treebank project, and the tagger is applied to a tokenized list: [“and” “now” “for” “something” “completely” “different”] becomes [(“And”, “CC”), (“now”, “RB”), (“for”, “IN”), (“something”, “NN”), (“completely”, “RB”), (“different”, “JJ”)]. “CC”, “RB”, “IN”, “NN”, and “JJ” stand for Coordinating Conjunction, Adverb, Preposition, Noun, and Adjective, respectively.

3.2.4 Stemming

When constructing a term-based feature set, important consideration must be given to the entropy of the set. A series of variations on a word – “great”, “greatest”, “greater”, “greatly” – frequently have the same polarity, but introducing each inflection as a unique feature can result in a feature set with each feature appearing less frequently in the training corpus. This added entropy negatively impacts our training because we may only see the word being used in a select subset of contexts.

As such, we seek to reduce the feature set size. One method that we investigate is stemming. The stemming technique relies on reducing inflected words to a base form—a word stem. In particular, we use the Porter Stemming algorithm, which relies on rules for stripping suffixes [27]. Thus we can map multiple words to each word stem and maintain a single polarity. The set above—[“great” “greatest” “greater” “greatly”] will all be stemmed to “great”.

3.2.5 Lemmatization

We also consider a different approach to feature set reduction: lemmatization. This seeks to improve on stemming by expanding the methods of word stem reduction. Lemmatizers tend to rely on an existing database of inflectional forms of words. While a stemmer only removes suffixes from a word, lemmatization will match words with equivalent meanings—a stemmer will reduce “carts” and “automobiles” to “cart” and “automobile”, while a lemmatizer will assign the same token to the two if it detects (through part-of-speech tagging) that “carts” is being used as a noun rather than an adjective.

A study comparing stemming and lemmatization in document retrieval precision found that lemmatization produced better precision, although the difference between the two was insignificant [28]. We also draw inspiration from a classification study on French language movie reviews, which found that using lemmatization on term unigram feature sets slightly improved

performance of the SVM classifier [29]. In this research project we use the WordNet Lemmatizer from NLTK, based on Stanford WordNet Project.

3.3 Aggregate Document Features

In splitting text into unigrams, we lose contextual information about order, position, and modifiers or modified words. Thus, it is crucial that we find some way of incorporating contextual information into the feature set. To expand our feature set beyond simply the presence of terms, we expand our search first to aggregate document features. These features correspond to various qualities of sentences and words, averaged over the entire document, to give a high-level view of the entire movie review. We started with several relevant features proposed by Gezici et al., relating to the polarity and purity of sentences and terms [30].

Polarity of individual terms and sentences were obtained through the TextBlob framework, described in more detail in section 3.5.5. For analysis on the original full corpus, we also used TextBlob to determine sentence subjectivity. On our manually labeled corpus (see section 3.5.6), we used manual labels to determine if a sentence was subjective or not, and we continued to use TextBlob to determine numerical subjectivity of each subjective sentence. Numerical polarity of terms and sentences is obtained as value between +1.0 (strongly positive) and -1.0 (strongly negative). Numerical subjectivity is obtained as a value between 0.0 (objective) and +1.0 (strongly subjective).

3.3.1 Polarity-Related Features

The simplest features were the average polarity (AP) of words and sentences within the document (AWP, ASP respectively). We also used the average polarity of subjective (P0) words and sentences only (PW0, PS0 respectively). A "subjective" word or sentence in the original corpus has subjectivity greater than 0.0. We seek too to relax the subjectivity restriction from P0, but we also wish to model the increased influence that more subjective words have on document polarity. As a result, we also use the average product of polarity and subjectivity (PS) for words and sentences (PWS, PSS respectively). We also include the standard deviation (STD) of word and sentence polarity (wStd, sStd respectively).

Finally, we notice that sentences in the beginning and end of reviews tend to summarize author opinion. To take advantage of this clustering, we also incorporate the kP feature set, which uses the average polarities of words in the first k sentences (fkR) and last k sentences (lkR), with k from 1 to 5.

3.3.2 Purity-Related Features

Aside from polarity, we also utilize purity, a measure defined for words w_i in a document as:

$$\frac{\sum pol(w_i)}{\sum |pol(w_i)|}$$

If the words in a document are consistent in the sign of their polarity, the document is considered to be more pure; similarly, if a document contains very balanced positive and negative terms, the proportions will be reflected as impure. Purity may take positive or negative sign, and the larger the absolute value of purity, the more "one-sided" a review is.

We include review polarity (pur), review purity with sentences instead of words (SR), as well as first and last k line purity (fkR, lkR). Similar to polarity, we also use a product of polarity and subjectivity to calculate "subjective purity" (subR):

$$\frac{\sum pol(w_i) \times sub(w_i)}{\sum |pol(w_i) \times sub(w_i)|}$$

3.4 Negation Handling

We next delve one step deeper in complexity, establishing context of individual words and phrases. Of particular potential importance is the effect of negation: "witty" and "not witty" give opposite-polarity impressions to the reader. We consider bigrams (and n-grams) to incorporate some level of context in the token, and as such we apply negation handling methods solely to the unigram feature space.

3.4.1 Simple Negation Handling

The first method we investigate for negation handling is a technique described by Das and Chen for analysis of Amazon's message boards. We prepend the tag "NOT_" to every word between a negation word (from the set "not", "nothing", "never", "n't", etc.) and the first following punctuation mark [9]. Thus, the sentence "The performance wasn't the best, but I enjoyed it" would be tokenized to ["the" "performance" "was" "n't" "the" "best" "," "but" "I" "enjoyed" "it"], and then negated to ["the" "performance" "was" "n't" "NOT_the" "NOT_best" "," "but" "I" "enjoyed" "it"].

3.4.2 Limited Negation Scope

We consider now the case of a statement such as "this was not a tremendously exciting movie". Using our prior negation scheme, the sentence would be negated to "this was not NOT_a NOT_tremendously NOT_exciting NOT_movie". The functional section then goes from "not

tremendously exciting movie" to "not-tremendously not-exciting not-movie". We first note the obvious meaningless "not-movie" word, but we also note that the typical English speaker would negate "not tremendously exciting" to "not-tremendously exciting", which implies the movie could fall into a range of excitement from "boring" to "somewhat exciting". However, our prior negation scheme forces the movie to be "somewhat boring".

To fully solve this issue and more accurately model negation, we would presumably need a negation handling method that takes more advantage of sentence structure and subjects for modifiers. In this investigation, we take a step along that route by limiting the scope of our negation method: instead of negating the entire section of text up to the next punctuation, we investigate only negating the next k words (that are not stop-words), with k varying from 1 to 3 [31]. The results are described in section 5.3.

3.5 Subjectivity Analysis

We evaluated several different methods of subjectivity analysis including simple adjective presence, adjective frequency, part-of-speech frequency, sentence position, and Wilson et al.'s OpinionFinder tool. For adjective presence and adjective and POS frequency we trained our subjectivity analysis algorithm on the Subjectivity Dataset from the second version of the Cornell Movie Review corpus [21]. The dataset consists of 5000 subjective sentences and 5000 objective sentences in two files. Each word and punctuation mark is separated by a space, and each sentence is separated by a line break. The sentence position analysis and OpinionFinder methods rely on having individual reviews to analyze, and as such were applied directly to the polarity dataset.

3.5.1 Simple Adjective Presence

Here we use the simplest subjectivity identifier: if a sentence contains adjectives, we determine it to be subjective. Intuitively, we expect there to be higher recall than precision for this subjectivity method since we expect opinions to be expressed mainly through adjectives, but there are also many adjectives that do not express opinion—"red" and "purple" are adjectives, but do not express opinion, while "great" and "funny" indicate polarity.

When applied to our sentiment analysis task, when initially processing a review we discard sentences that do not contain adjectives under the Simple Adjective Presence rule.

3.5.2 Adjective Frequency

To increase precision, we want to better limit the number of objective sentences that we mark as subjective. We noted before that plot summary sentences may contain non-opinionated

adjectives as descriptors. Thus, we alter our hypothesis and propose that subjective sentences tend to use more adjectives than objective sentences. We use three different types of adjective frequency subjectivity classifier. The first is a simple rule-based classifier that discards all sentences with fewer than k adjectives. We found that there was an average of 1.8 adjectives in objective sentences and 2.2 adjectives in subjective sentences in the subjectivity dataset. We investigated values of k between 0 and 4.

The second method was an SVM-based adjective frequency classifier. We trained a linear SVM on the subjectivity dataset with the feature being the number of adjectives in a sentence. We then used it to predict the subjectivity of each sentence and discarded sentences identified as objective.

As an extension of the concept, we also tried to distinguish subjective sentences from objective sentences based on the entire part-of-speech makeup of the sentence. For example, plot-related sentences may have similar numbers of adjectives and nouns, since adjectives are usually used as modifiers in those sentences. Meanwhile, subjective sentences may have more adjectives and fewer nouns, verbs, and adverbs. For each sentence, we record the number of each part of speech present in the sentence as the feature set. Using the POS Subjectivity rule, we trained a linear SVM on the subjectivity dataset and used it to predict the subjectivity of each sentence in each review, discarding sentences identified as objective.

3.5.3 Positional Features

Focusing on adjectives is the natural first step in subjectivity analysis, but to improve performance we next examine structural elements of movie reviews. We reviewed a random sample of movie reviews from IMDB and observed that reviewers tend to summarize their opinions in one final recommendation: e.g. "don't let these [drawbacks] deter you, though; *I Went Down* is a little gem". We have also conducted a quantitative assessment of structural and semantic properties of frequently misclassified reviews—including summarization—in the subjectivity dataset.

These summary sentences are indicative of the overall impression of the reviewer toward the movie. As we want to ultimately predict the overall sentiment, focusing on these sections of the reviews may also avoid confusion due to contrasting sentiments toward specific aspects of the movie (e.g. pacing, set design). We examine a form of subjectivity analysis that keeps only the last k sentences of a movie review.

We have also noticed that a slightly smaller subset of reviews will have indicator sentences near the beginning of the review that give significant clues to the overall sentiment, e.g.

"What the hell has happened to all good American action movies?" Seeing as prior techniques of reducing the movie review text tend to hurt performance, we investigate tagging features based on if they are in the "top" "mid" or "bot" sections of the review, with "top" and "bot" corresponding to the first and last *k* sentences.

3.5.4 OpinionFinder

There has been significant research toward identifying subjectivity in NLP, and a profusion of open-source tools. We extracted and tested one of the more popular programs for subjectivity identification: OpinionFinder, first developed at the Intelligent Systems Program at the University of Pittsburgh [15]. We use the second version of OpinionFinder, updated in 2011.

Specifically, we make use of the subjectivity analysis module of OpinionFinder. By running OpinionFinder on each review, we obtain a list of sentences identified by character number and if they are subjective or objective. OpinionFinder uses a Naïve Bayes classifier that relies on lexical features and context to identify subjectivity. One feature set used by the tool is a set of extraction patterns correlated with objectivity generated using AutoSlog-TS [32]. These patterns take the form of words and syntactic objects: e.g. "<subject> believes" or "supports <np>".

We use OpinionFinder in two ways: first, we exclude all sentences identified as objective and only run our sentiment analysis classifiers on subjective sentences. Secondly, we use term frequency as the feature instead of term presence, and give a *k* frequency bonus to features found in subjective sentences (Subjective Frequency Bonus).

3.5.5 TextBlob

We also utilized another open source tool with both sentiment- and subjectivity analysis functions: the python library TextBlob. Specifically, we use TextBlob 0.11.1, developed by Loria, et al., relying on the NLTK and *pattern* libraries [16]. We use the *sentiment.polarity* and *sentiment.subjectivity* functions in the package to obtain numerical values for polarity (between highly negative -1.0 and highly positive +1.0) and subjectivity (between objective 0.0 and highly subjective +1.0). These can be applied to both sentences and words represented as strings.

The numerical values for polarity and subjectivity are obtained from WordNet3 and recorded in the lexicon for each different sense of the word, and the output is averaged. We note that the English lexicon used by TextBlob incorporates a significant number of words from the Polarity v2.0 movie review dataset, and it has been shown to have 75% classification accuracy when used to train a Naïve Bayes classifier for our dataset [33].

Here we use TextBlob for two separate tasks: to generate aggregate features, as well as to exclude objective sentences (TextBlob subjectivity of 0.0).

3.5.6 Manual Labeling

We recognize several significant drawbacks of automated subjectivity analysis. TextBlob, for instance, generates subjectivity values from a general lexicon. Such a lexicon and word-based approach does not take advantage of sentence structure and context. This is unfortunate, as an important application of subjectivity analysis in the movie review domain centers around removing plot-related sentences. Such sentences may contain opinionated words, but the usage is not meant to convey an opinion: "skeet has a long scar on his chest" contains the word "long", which is identified as a subjective word by TextBlob. However, the sentence is clearly objective, simply identifying a bodily feature on a character.

OpinionFinder is better suited to take advantage of context in this regard, as it uses lexical features such as pattern matching to identifying commonly used sentence forms in subjective sentences. Its subjectivity classification is primarily performed through a Naïve Bayes classifier trained using subjective and objective sentences generated through rule-based classifiers [15]. However, we note that sentence structures may vary widely in English, especially when it comes to the colloquial domain in which many movie reviews fall.

As a result, we aim not only to investigate the effects of specific subjectivity analysis techniques on sentiment analysis, but also to evaluate the theoretical limit of effectiveness of machine learning techniques coupled with bag-of-word features for sentiment analysis. In this vein, we manually labeled each movie review according to the following framework:

1. Read the raw review and determine if it is positive or negative
2. If a line in a movie review expresses no opinion about the movie as a whole (objective) or is not referring to the movie (i.e., a description of other movies directed by the same person), delete it
3. If a line expresses a sentiment opposite to the full review, append a star (*)
4. Append three stars (***) to the line in the review most indicative of the full review sentiment

In short, this will convert a full, raw review into a document consisting of only subjective lines from the review, with sentences opposing the overall sentiment marked with (*) and the sentiment summarizing sentence marked with (***). An example of manual labeling is shown in

Figure 2 on the following page.

We consider this dataset the benchmark for subjectivity analysis, and use it in several ways. First, we construct our feature set using the manually labeled reviews instead of full reviews, investigating the effects of theoretically ideal subjectivity analysis on machine learning techniques for sentiment analysis. Next, we construct our feature set using only the (***) sentiment summary sentences for each review. Finally, we test the full review corpus, manually labeled corpus, and manually labeled summary corpus with aggregate features generated from the manually labeled corpus.

[1] lengthy and lousy are two words to describe the boring drama the english patient .

[2] great acting , music and cinematography were nice , but too many dull sub-plots and characters made the film hard to follow .

[3] ralph fiennes (strange days , schindler's list) gives a gripping performance as count laszlo almasy , a victim of amnesia and horrible burns after world war ii in italy . *

[4] the story revolves around his past , in flashback form , making it even more confusing .

[5] anyway , he is taken in by hana (juliette binoche , the horseman on the roof) , a boring war-torn nurse .

[6] she was never really made into anything , until she met an indian towards the end , developing yet another sub-plot .

~~[7] count almasy begins to remember what happened to him as it is explained by a stranger (willem dafoe , basquiat) .~~

~~[8] his love (kirstin scott thomas , mission impossible) was severely injured in a plane crash , and eventually died in a cave .~~

~~[9] he returned to find her dead and was heart broken .~~

~~[10] so he flew her dead body somewhere , but was shot down from the ground .~~

[11] don't get the wrong idea , it may sound good and the trailer may be tempting , but good is the last thing this film is .

[12] maybe if it were an hour less , it may have been tolerable , but 2 hours and 40 minutes of talking is too much to handle .

[13] the only redeeming qualities about this film are the fine acting of fiennes and dafoe and the beautiful desert cinematography . *

[14] other than these , the english patient is full of worthless scenes of boredom and wastes entirely too much film . ***

Figure 2: Example of manually labeled movie review (cv040_8829.txt, *negative*)

3.6 Feature Sets

Sentiment lexicon learning and Twitter sentiment analysis tasks commonly use multiple *n*-grams as their base feature space. Severyn and Moschitti used unigrams, bigrams, and trigrams for words, as well as trigrams, 4-grams, and 5-grams for character sequences [34]. We propose that these higher order *n*-grams are helpful in the classification task for tweets due to the length constraint (140 characters). However, our movie reviews roughly contain 250-300 unique words each, and their length makes character sequence *n*-grams less useful. Additionally, given the existing sparseness of unigrams and bigrams in the corpus, higher-order word *n*-grams are likely to be too sparsely distributed to positively impact performance—indeed, they are more likely to induce overfitting.

At the most basic level, we used a feature set of only unigrams. We also used bigrams in our feature set in order to capture context and modified words [35]. A simplified approach is represented by the adjectives-only feature set. We also used two combined feature sets. In our unigrams with POS-tagging (Uni+POS) feature set, we tag the unigrams with their parts-of-speech to differentiate between usages of a single word: e.g., "good" *adj.* which indicates positive sentiment, vs. "good" *n.* which describes a commodity and has no clear contribution to document sentiment polarity. Our final feature set combines POS-tagged unigrams with bigrams (Uni & Bi). As noted above, bigrams already serve to differentiate usages of their constituent words, so we do not tag the parts-of-speech for each word in a bigram.

3.7 Feature Selection

When we generate our feature sets, we note that they are perhaps a great deal larger than we want them to be. Our primary concern is over-fitting—our entire corpus consists of 2,000 documents, and yet our feature vocabularies extend to over 40,000 elements for unigrams and over 400,000 for bigrams. As such, we investigated what degree of feature selection to use and plotted the performance against the relevant measure of selection. In addition to over-fitting concerns, we also seek to remove redundant and irrelevant features from the feature set, as well as improve processing time and adhering to memory constraints [36].

We preserve all aggregate features and do not remove them during feature selection.

3.7.1 Frequency Cutoff

One method of feature selection that we explored was simple frequency-based cut-off. We kept only features that appeared in over *k* documents in our training corpus. The general intuition here is that terms that appear only in a single or very few documents in the training set are

unlikely to appear in future documents. In particular, if a term appears only in one context in training but may reasonably have multiple meanings (e.g. "Luke found Darth Vader's *weakness*" vs. "the movie's primary *weakness*"), we run a high risk of over-fitting our models. While a frequency cutoff runs a heavy-handed approach that establishes a single frequency threshold below which features are deemed irrelevant, we note that we already have an extremely large feature space with many features that appear very frequently in documents.

3.7.2 Mutual Information Criterion

We see that our frequency cutoff criterion for feature selection has theoretical merit, but we lack a solid framework for how to establish the cutoff. Certainly, some words that fall below a cutoff may be more relevant than others—terms like "disgusting" or "absolutely mind-blowing" clearly carry strong polarity but may appear infrequently in reviews. Our second method for feature selection, then, relies on an explicit measure of relevance: mutual information, a measure of mutual dependence between two random variables. Here we find the mutual information (dependence) between each feature and a variable representing label. The random variables are discrete, so the mutual information formula is as follows [37]:

$$I(X,C) = \sum_{c \in C} \sum_{x \in X} P(x,c) \log\left(\frac{P(x,c)}{P(x)P(c)}\right)$$

Where X is a feature with values $x \in (1, 0)$, C is the set of labels (+1, -1). Here, $P(x,c)$ is the probability of feature value x appearing with label c, and $P(x),\ P(y)$ are the probabilities of feature value x and label c, respectively. We approximate the probabilities using relative frequency estimation:

$$\hat{I}(X,C) = \sum_{c \in C} \sum_{x \in X} \frac{N_{xc}}{N} \log\left(\frac{\frac{N_{xc}}{N}}{\frac{N_x N_c}{N^2}}\right) = \sum_{c \in C} \sum_{x \in X} \frac{N_{xc}}{N} \log\left(\frac{N N_{xc}}{N_x N_c}\right)$$

Here, N represents the counts, where N_{xc} is the number of documents in the corpus with feature X having value x and label c.

4 Methods

4.1 Singular Classifiers

As with previous research on the Cornell Movie Review corpus, we began by investigating the Naïve Bayes, Maximum Entropy (MaxEnt), and SVM classifiers [14]. Additionally, we include the Stochastic Gradient Descent classifier (SGD/SGDC). We used the *scikit-learn* implementations of these singular classifiers.

4.1.1 Naïve Bayes

The first classifier we will use is the Naïve Bayes classifier. Here we assign a document d the class c^* (positive or negative) that is most probable: $c^* = \arg\max_c P(c|d)$. Using Bayes' rule, our probability equation becomes:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

We decompose the document into its feature vector, and the equation then becomes:

$$P(c|d) = P(c|f_1, f_2, \dots) = \frac{P(c)P(f_1, f_2, \dots|c)}{P(d)}$$

Naïve Bayes relies on the naïve independence assumption, that the conditional probabilities of each feature appearing given a class are independent. We know that this is clearly not the case—certain combinations of words are more likely to appear with others (indeed, that's one of the reasons behind why we consider bigrams for features). However poor the assumptions are, Naïve Bayes nonetheless performs fairly well experimentally [14].

Here then, the equation becomes:

$$P(c|d) = \frac{P(c)\prod_{i=1}^{n} P(f_i|c)}{P(d)} \propto P(c)\prod_{i=1}^{n} P(f_i|c)$$

Specifically, we use Gaussian Naïve Bayes, which uses a Gaussian distribution prior for the conditional feature probabilities.

4.1.2 Maximum Entropy Classifier

Overall, the concept of Maximum Entropy is to choose a model that is consistent with training data while constrained by the fewest assumptions (highest entropy). It is significant here that we have limited and incomplete information, represented by our training data. Given such information, then, making additional unsupported assumptions is likely to move us away from the

correct model. Following Occam's Razor, then, our preferred model is one where all events are considered equally likely given the constraints of the training data [38].

The MaxEnt classifier is a generalization of logistic regression [39]. We solve for the same most probable sentiment polarity for a document: $c^* = \arg\max_c P(c|d)$. Here, the conditional probability is logistic:

$$P_w(c|d) = \frac{1}{1 + e^{-cw^T d}}$$

Where $w \in \mathbb{R}^n$ is the weight vector. To solve for the weight vector, we minimize regularized negative log-likelihood with positive penalty parameter K with our positive and negative sentiment labels $c \in \{-1, +1\}$:

$$P^{LR}(w) = K \sum_i \log\left(1 + e^{-c_i w^T d_i}\right) + \frac{1}{2} w^T w$$

In Maximum Entropy, the conditional probability is instead modeled as:

$$P_w(c|d) = \frac{e^{w^T \boldsymbol{f}(d,c)}}{\sum_i e^{w^T \boldsymbol{f}(d_i,c)}}$$

Here our feature extraction function $f(d, c)$ represents a vector of feature-class functions $F_{i,c}(d, c', f)$. The feature-class function for a feature f and class c will only return 1 if the feature appears in document d and the hypothesized class c' is the same as c. In context of the movie reviews, the feature-class function for "excellent" and positive will only return 1 for a movie review if it contains the word and is hypothesized to be positive.

While Naïve Bayes assumes feature independence, MaxEnt has the advantage of making no such assumptions. As such, even though Naïve Bayes performs well despite the unrealistic assumption we expect MaxEnt to show performance improvements when there is no conditional independence between features.

4.1.3 Support Vector Machines

Another classifier that has seen significant usage in the field of sentiment analysis and opinion mining is the support vector machine (SVM). SVMs have been shown to out-perform Naïve Bayes and other linear classifiers in a variety of textual settings, over different feature sets [14].

Rather than estimating probabilities of classes based on features and features given classes, SVMs seek to find the best hyperplane margin to separate two classes over a space

defined by the features:

$$\vec{w} = \sum_k \alpha_k c_k \overrightarrow{d_k}$$

Here, $c_k \in \{0,1\}$ and represents the label of the training document (vector). $\overrightarrow{d_k}$ is a vector representation of the k-th document, with presence of features as elements. We solve a dual optimization problem to obtain the proper weights α_k, and those documents with $\alpha_k > 0$ are the support vectors:

$$\max_{\alpha_k} \sum_k \alpha_k - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k c_j c_k k(\overrightarrow{d_j}, \overrightarrow{d_k})$$

Where $k(\overrightarrow{d_j}, \overrightarrow{d_k})$ is the kernel of two document vectors, a function that takes in two document (feature) vectors and outputs a scalar. We note that our feature space (~17,500 elements) is significantly larger than our training and testing corpuses. As Gaussian and RGB kernels are more prone to over-fitting the data, we instead elect to use a linear kernel for the SVM classifier [40]. Once we have generated the margin hyperplane, we classify a document by evaluating which side of the margin it falls on.

4.1.4 Stochastic Gradient Descent

Stochastic Gradient Descent is a method to quickly and efficiently compute models for learning algorithms that optimize a cost function. With our objective function as $Q(x, \theta)$ parametrized by a vector of weights θ, standard gradient descent will update, given the learning rate γ:

$$\theta_{i+1} = \theta_i - \gamma \nabla_\theta \mathbb{E}[Q(x, \theta)]$$

However, since calculating $\mathbb{E}[Q(x, \theta)]$ for normal gradient descent methods is $O(n)$ for the size of the training set n, this can get very computation-intensive for large training sets and feature sets. As such, each iteration of SGD limits the expectation over a subset of the training data X_i:

$$\theta_{i+1} = \theta_i - \gamma \nabla_\theta \mathbb{E}_{X_i}[Q(x, \theta)]$$

In each iteration, the subset of the total training data X_i is randomly selected. Here we use a hinge loss function to approximate an SVM with SGD, and our regularization term is the L2-norm.

We primarily examine SGD and the SGD Classifier (SGDC) as a more efficient version of the SVM classifier, since we are using a hinge loss function to approximate an SVM.

Additionally, as SGD uses a small subset of training data in each iteration, we expect it to have reduced over-fitting relative to the SVM. By design, over-fitting on individual iterations of SGD (on small subsets of the training corpus) is less critical than over-fitting on the entire training set, as with the SVM. Indeed, prior research has indicated SGD slightly outperforms SVMs [41].

4.2 Ensemble Methods

For ensemble classifiers, we focused on boosting and bagging algorithms: AdaBoost, Random Forest, and Additive Logistic Regression (ALR). We used the *scikit-learn* implementations of these ensemble classifiers.

4.2.1 AdaBoost

AdaBoost is a boosting-based supervised ensemble method. It aims to reduce bias and variance from aggregated "weak" learners [42]. The algorithm generates weak learners and adjusts the weight of training examples. Here, weak learner refers to a classifier that has reasonably greater than 50% accuracy. We use Decision Trees as our weak learner, with roughly 60-65% base accuracy.

As a boosting algorithm, AdaBoost sequentially generates its weak base learners. Given the classes to be $c_i \in \{-1, +1\}$, we generate n weak learners in iterations $t \in \{1, \dots, n\}$. We have a vector of hypothesis weights λ_t and a vector of training example weights γ_i. For each iteration, the weak learner is generated to minimize weighted error $\epsilon_t = \sum \gamma_i (h_t(d_i) \neq c_i)$. The weight of that hypothesis is set to $\lambda_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ [42]. Then, we update the training example weights, normalizing with the normalization function Z_t:

$$\gamma_i \leftarrow \frac{\gamma_i e^{-\lambda_t c_i h_t(d_i)}}{Z_t}$$

After all weak learners have been created, the hypothesis weights are then normalized. The trained AdaBoost hypothesis consists of a weighted majority vote from all of the weak learners: $H(d) = \text{sgn}(\sum_t \lambda_t h_t(d))$.

We see here that the hypothesis weight increases as its weighted error decreases. Since $\lambda_t \geq 0$ (our weak learners must have $> 50\%$ accuracy), we see that $e^{-\lambda_t c_i h_t(d_i)}$ will be higher if the class and hypothesis are incorrect (if the training example is difficult to classify with the current iteration). Thus, frequently misclassified training examples will increase in weight, and successive weak learners will emphasize correctly classifying difficult training data.

4.2.2 Random Forest

We investigated the Random Forest classifier as first described by Leo Breiman. These classifiers are based on a forest of different decision tree classifiers. In addition to decision trees, we also investigate decision stumps—decision trees with only one level, making a prediction using just a single input feature. We note that decision stumps have been shown empirically to deliver competitive performance when used as base classifiers for boosting algorithms—most notably AdaBoost [43].

When aggregating classifiers in an ensemble framework, we use weak learners for base classifiers. Prior literature indicates that boosting with C4.5 decision trees results in significantly improved performance, and boosting with decision stumps can approximate a well-tuned decision tree [43]. As such, we expect our random forests with decision trees to perform better than with decision stumps.

Random Forest is a bagging-based ensemble learner, as compared to the boost-based AdaBoost. While AdaBoost sequentially generates weak learners and changes training weights to compensate for difficulty, Random Forest independently generates weak learners. Decision tree algorithms split nodes using the best split out of all variables—in a random forest, nodes are split using the best split among a randomly chosen subset of variables. Because these randomly chosen subsets can overlap among trees, we create a multi-set with the same cardinality as the original training data, and thus classifier variance is reduced.

It can be shown that Random Forests do not over-fit with increasing numbers of trees generated, due to convergence [44]. This property allows us to increase the number of estimators to our computational limit. We observe this property when tuning the number of trees (see section 5.1.4).

4.2.3 Additive Logistic Regression

Additive Logistic Regression (ALR) is an ensemble learner based on a generalization of boosting algorithms [45]. In general, it creates M singular classifiers $f_m(d)$ and solves for the log-probability:

$$F(d) = \log\frac{P(c = +1|d)}{P(c = -1|d)} = \sum_{m=1}^{M} f_m(d)$$

From this, we obtain the probabilities:

$$P(c = +1|d) = \frac{e^{F(d)}}{1 + e^{F(d)}}$$

While the individual hypotheses of AdaBoost are decision trees, ALR boosts the MaxEnt (Logistic Regression) classifier by minimizing logistic loss $\sum_m \left(1 + e^{-c_i F(d_i)}\right)$. We see that while AdaBoost emphasizes learning on difficult training examples, ALR does not. As such, if there is significant noise in the labels, we expect ALR to outperform AdaBoost.

5 Experiments

5.1 Design

5.1.1 Accuracy Measures

To measure accuracy of our classifiers, we use stratified 10-fold cross validation. The raw data set documents have already been divided into 10 sets of 100 documents each, where each set has roughly the same characteristics. For each "fold" of our validation, we hold out 1 set of positive and negative documents as the test set, and use the remaining 9 sets of positive and negative documents totaling 1800 documents to train our classifier. We then report the mean of the accuracy proportions reported in each fold (% of correctly classified results).

5.1.2 Choosing Preprocessing Methods

We need primarily to evaluate the usefulness of negation handling, and to choose whether to use stemming or lemmatization. To do so, we designed a test harness to run the Naïve Bayes classifier for unigram feature sets. To test negation handling, we used the paired 10-fold cross-validated *t*-test for unigram feature sets with and without negation handling. Thus, we evaluated for difference feature sets (presence vs. frequency, part-of-speech tagging, stemming and lemmatization) whether adding negation handling made a significant difference in accuracy. We used the same methodology to compare stemming and lemmatization.

We tested negation handling with various sized feature sets, limiting the feature space to features encountered n or more times. We report here the statistics for $n = 9$ in Figure 3. We can see that, although statistically insignificant, the addition of negation handling uniformly improves performance for the unigram feature space.

Negation Handling Investigation		No Negation	Negation
Presence	POS + Stemming	0.7350	0.7395
	POS + Lemmatization	0.7330	0.7385
	Stemming	0.7110	0.7170
	Lemmatization	0.7140	0.7170
Frequency	POS + Stemming	0.7235	0.7365
	POS + Lemmatization	0.7250	0.7355
	Stemming	0.7065	0.7175
	Lemmatization	0.7100	0.7195

Figure 3: Naive Bayes classification accuracy with and without negation handling

We applied the same methodology to test our methods of inflection reduction. As seen in Figure 4, performance was roughly even between the two types of inflection reduction, with no

significant statistical difference at $n=9$ cutoff for Naïve Bayes. As such, we decided to use lemmatization with our text, as it can handle (roughly) part-of-speech usages and synonyms.

Inflection Reduction Methods		Stemming	Lemmatization
Presence	POS	0.7395	0.7385
	Raw	0.7365	0.7355
Frequency	POS	0.7170	0.7170
	Raw	0.7175	0.7195

Figure 4: Naive Bayes classification accuracy with stemming and lemmatization

5.1.3 Feature Selection

When testing simple frequency cutoff for feature selection, we ran our singular classifiers on the polarity dataset for various cutoff values k. The results are summarized in Figure 5. We see here that Naïve Bayes performance tends to increase with cutoff, trending in an opposite direction from that of SVM, SGDC, and Maximum Entropy. It appears to be more sensitive to over-fitting and outliers, since vocabulary size decreases with increasing cutoff. We note that because cutoff k means keeping only features that appear in k or more documents and there are a great many feature that appear in very few documents, vocabulary size decreases very quickly initially and decreases much more slowly as we increase cutoff beyond 8 or 10.

With the exception of Naïve Bayes, our classifiers tend to do better at smaller cutoffs (larger vocabulary sizes). We see that the classifiers perform best on the feature set consisting of unigrams with part-of-speech tagging and bigrams, with part-of-speech tagged unigrams coming a close second. It does appear that for the two feature sets containing bigrams, Naïve Bayes follows the same performance trend as the other classifiers. This may be due to the relatively large initial feature space (~400,000 bigrams) that is already significantly constrained at the first cutoff tested (8 for bigrams, 11 for unigrams and bigrams). At that point, we have already removed many bigrams that appear very sparsely among the training corpus. The goal of feature selection is to keep the most relevant features, and with simple selection, we assume relevance is correlated heavily with term frequency across the corpus.

We performed mutual information feature selection by keeping the top n features by mutual information, where n ranged from 2500 to 17500 in increments of 500. The results are summarized in Figure 6 for singular classifiers on each feature set.

It is interesting to observe that while classifier performance does appear in increase with vocabulary size, the magnitude of improvement seems very small. At the same time, performance of non-Naïve Bayes classifiers is generally higher using mutual information than simple frequency cutoff. This indicates that mutual information is a much better criterion for feature selection than

simple frequency within the corpus. This is expected, as our criterion here explicitly measures dependency between document polarity and each feature. As with simple feature selection, we observe that performance is generally best on the combined unigram-bigram feature set and the unigram set with part-of-speech tagging. Similarly, we see that the trend for Naïve Bayes is closest to the trend for the other classifiers on the feature sets containing bigrams.

We note that the direction of the x-axis is reversed in these graphs (Figure 6) as compared to our graphs for simple feature selection (Figure 5). In the earlier graphs, our independent variable was cutoff frequency—here, our independent variable is the number of features in the vocabulary. Looking here, we note roughly the same trend across the two feature selection methods: performance increases with vocabulary size.

We note that mutual information is a much better criterion for feature selection performance-wise. We also see that keeping a relatively large number of features after sorting by feature selection seems to work particularly well. From this investigation, we decided to use the mutual information criterion for feature selection, limiting our feature sets to 17500 elements each.

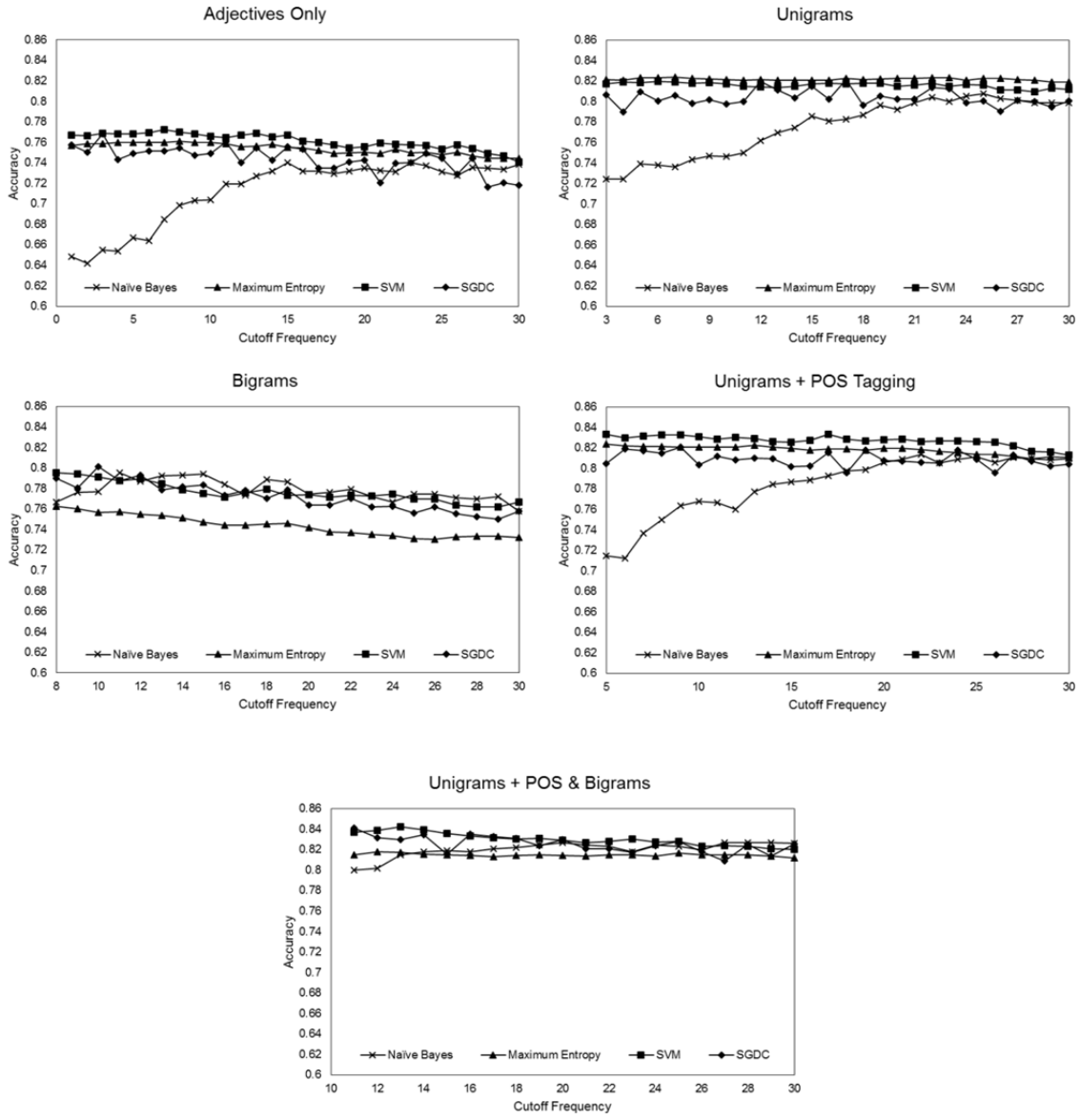

Figure 5: Classification Accuracy vs. Cutoff Frequency using simple feature selection

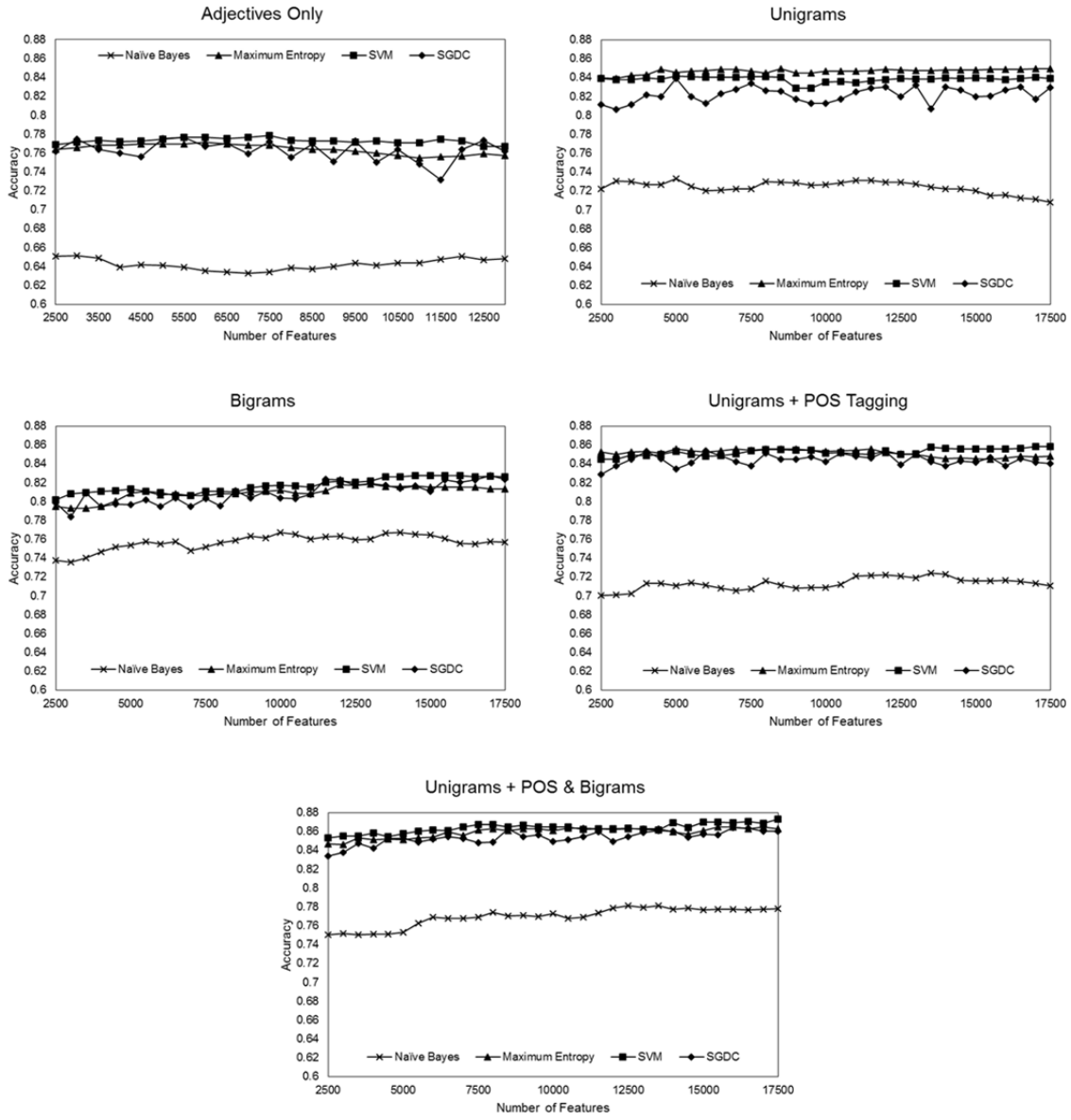

Figure 6: Classification Accuracy vs. Feature Set Size using Mutual Information

5.1.3 Tuning Ensemble Parameters

The most important parameter that we tuned for ensemble methods was the number of estimators/weak classifiers. For Random Forest, we varied the number of estimators from 50 to 1000, in increments of 50, and plotted classification accuracy against the number of estimators for each of our base feature sets. The results are shown in Figure 7. For decision tree-based random forests, we used Gini impurity as the split quality criterion, the square root of the number of features as the max features, no max depth for the tree, and a minimum of 1 sample per leaf. For decision stump-based random forests, we used the same parameters, but our max depth was 1.

We see a general trend that increasing the number of trees increases the performance of random forests. Indeed, the classifier seems to follow the same performance trend regardless of which of the five feature sets we use. We also see a much larger improvement in accuracy when using decision stumps. This may be due to the weaker nature of decision stumps on their own when compared to decision trees. However, the decision stumps' avoidance of over-fitting—or perhaps the propensity for large decision trees to over fit—can be seen as random forest performance with decision stumps approaches that of decision trees (~85% classification accuracy) as we increase the number of estimators. We will not investigate numbers of estimators greater than 1000, as we are already approaching the number of observations in our corpus. While we see that the adjectives-only feature set performs rather poorly compared to the unigram and combined or tagged unigram feature sets, the pure bigram feature set also sees poor performance. This result is interesting, given that we expect bigrams to predict word sense well [46]. We consider perhaps that adjectives and bigrams have significant contributions to certain aspects of sentiment analysis, but individually they lack the predictive power of unigrams.

We observed roughly the same trends when we tuned the number of estimators for ALR and AdaBoost with Trees and Stumps. For ALR, we also varied the number of estimators from 50 to 1000 in increments of 50. The graph of 10-fold cross-validated classification accuracy against the number of estimators is shown in Figure 8. We notice again a noticeable increase in accuracy when moving from 50 estimators to 200, and roughly constant if slightly increasing accuracy for more than 200 estimators.

Due to the computational requirements of AdaBoost on our machines, we varied the number of estimators instead from 50 to 500 in increments of 50 and plotted classification accuracy against the number of estimators for each feature set. The results are shown in Figure 9. We note that the AdaBoost with decision trees classification accuracy is very low and approaches un-boosted decision tree performance.

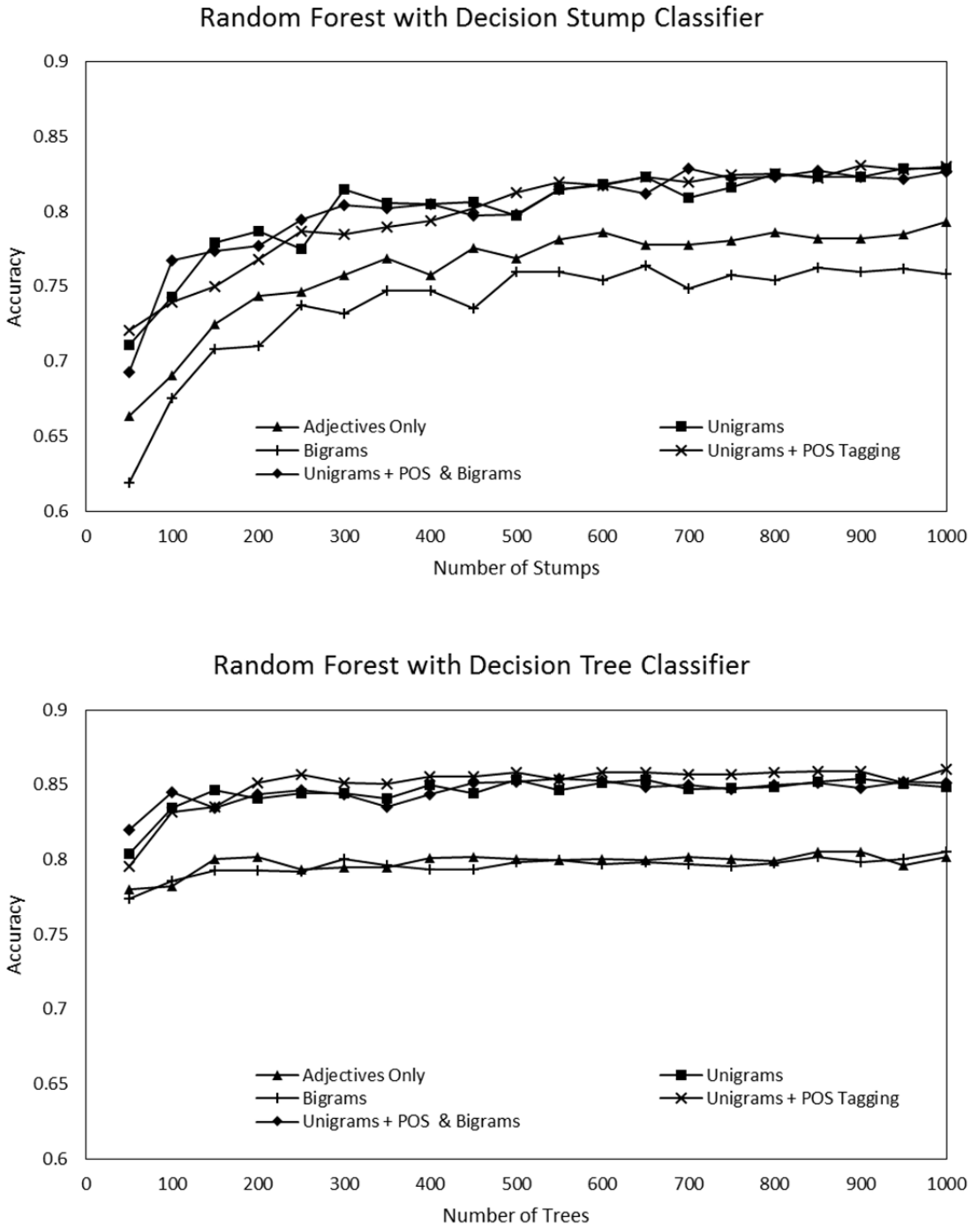

Figure 7: Tuning number of trees for Random Forest estimator

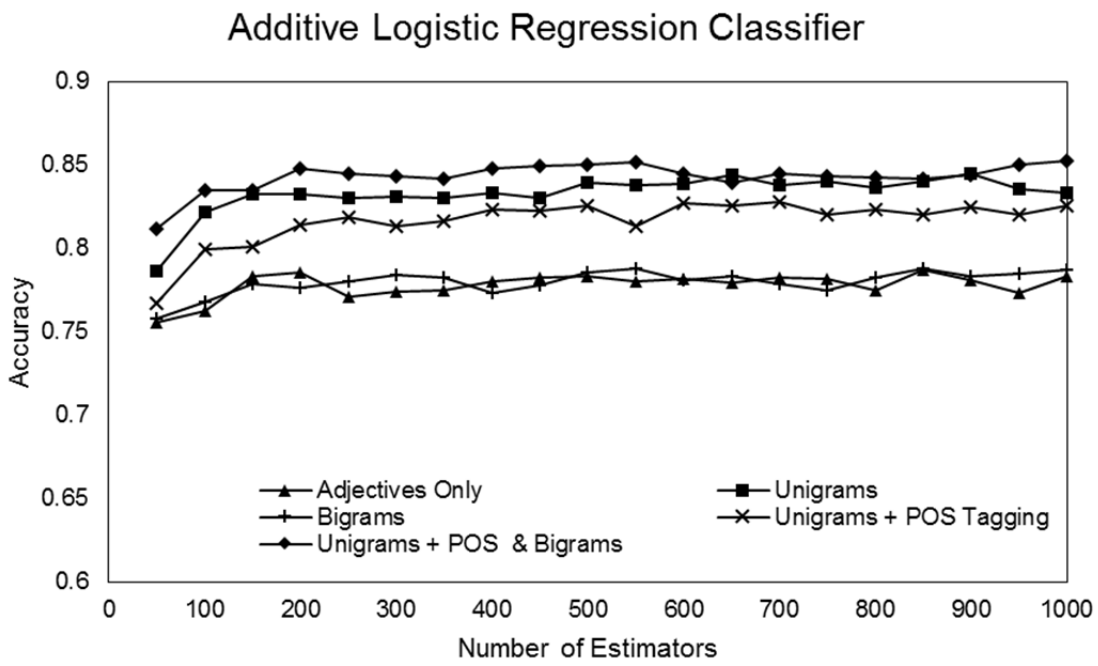

Figure 8: Tuning the number of estimators for Additive Logistic Regression classifier

Given that Random Forest and AdaBoost have been shown to converge and see no accuracy decrease when increasing the number of estimators, we elected to use 1000 estimators for each of our ensemble methods. However, we also tuned two additional parameters to fix the performance of AdaBoost with decision trees.

We hypothesized that AdaBoost was creating trees that were too deep, thus removing the advantages of boosting a weak learner. We approached this in two ways. First, we tuned the maximum number of features that AdaBoost and Random Forest would consider to split a node by. This produced the performance curve seen in Figure 10. Here we see that Random Forest performance peaks between 100 and 150 features for both stumps and trees. As a result, we kept the default value of $\sqrt{n}$ features, where n is the total number of features (17500). The curve for AdaBoost with trees is increasing but very noisy, so we look to the curve for AdaBoost with stumps instead. Here, we see a rough maximum at 450 features, which is the value we will use for both tree- and stump-based AdaBoost. However, this clearly has not solved the issue of tree-based AdaBoost performance lagging behind the other ensemble methods.

Here we try to manually limit the size of trees by changing the minimum number of training examples required to form a leaf. If there are not enough examples to form a new leaf from a node, then that node becomes an end node (leaf) itself. We see this should clearly limit over-fitting, since depending on the threshold, the ensemble learner cannot split on rare features.

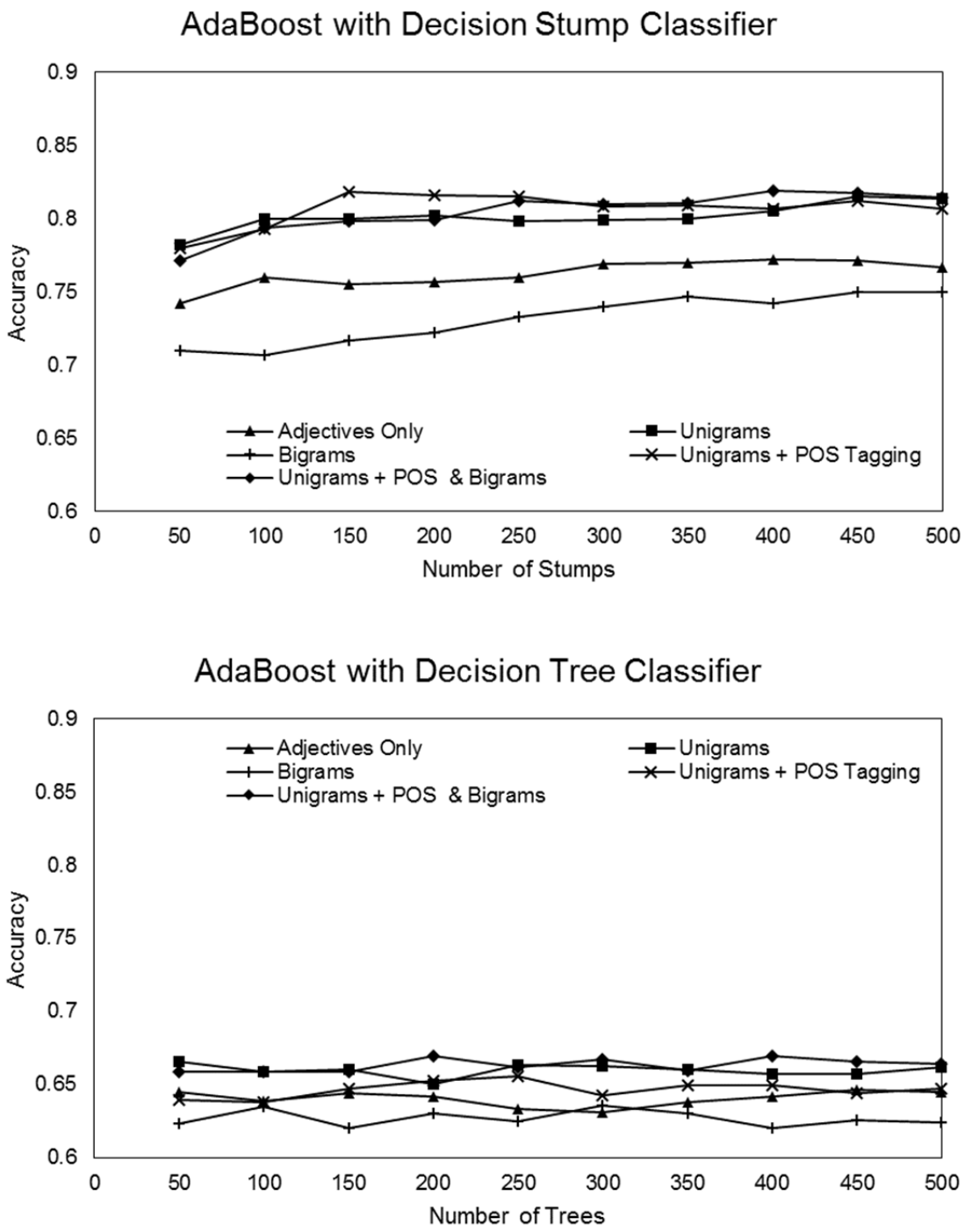

Figure 9: Tuning the number of estimators for AdaBoost

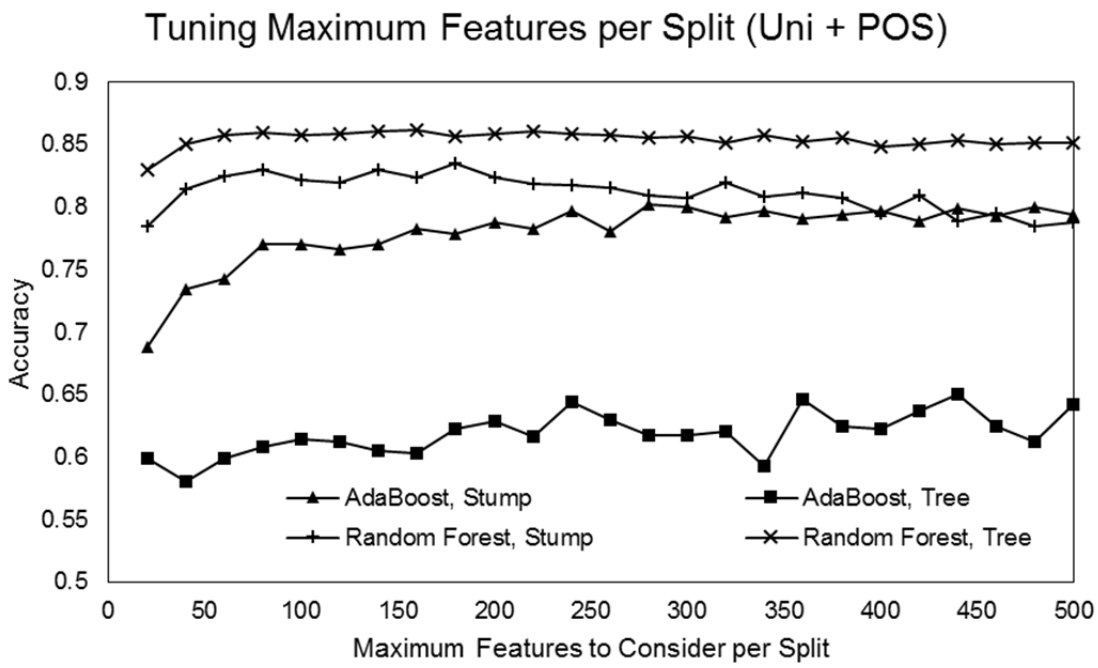

Figure 10: Tuning the maximum number of features to consider when splitting at a node

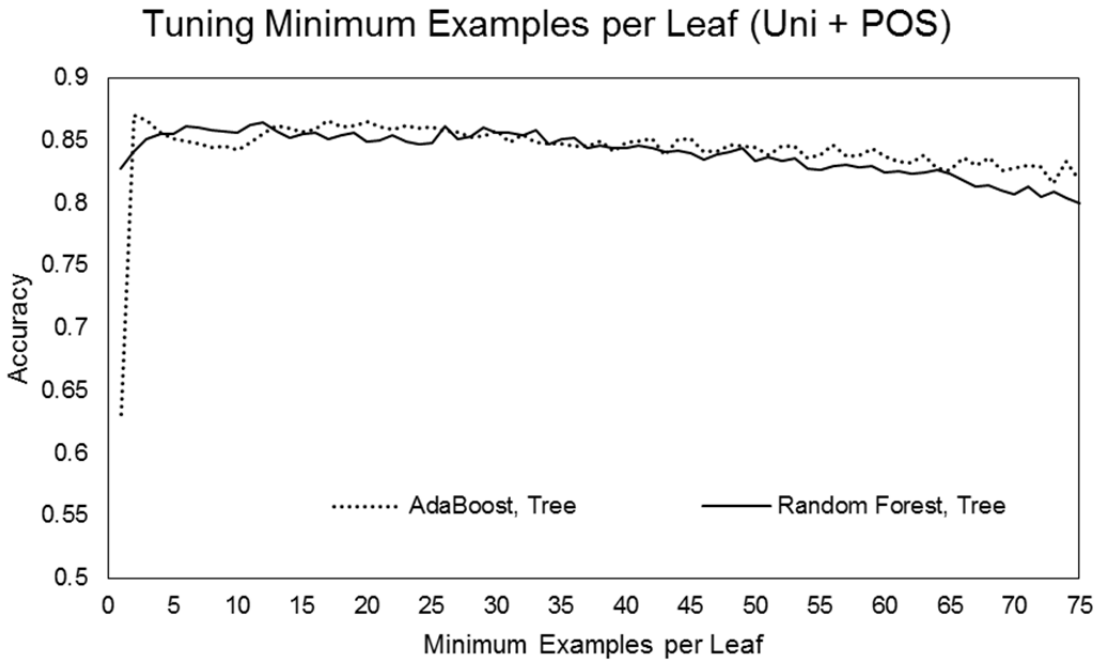

Figure 11: Tuning the minimum examples required to form a leaf

The exercise is pointless for stump-based Random Forest and AdaBoost, since they are single-level trees. Thus, we varied the minimum examples required per leaf from 1 to 75 in increments of 1 for AdaBoost and Random Forest using decision trees. The results, with accuracy derived from average performance in 10-fold cross-validation, are graphed against the parameter value in Figure 11.

The spike in performance from 1 to 2 minimum examples explains the sub-70% accuracy that we saw in previous experiments with tree-based AdaBoost. There is a near-immediate drop in performance after that point for AdaBoost, while the peak performance lies between 5 and 10 for Random Forest. We propose that as we increase the threshold for leaf formation, we very quickly shrink the possible tree sizes. This would indicate that many of the features appear only in few documents ($<k$).

Ultimately, we use AdaBoost with decision trees and Random Forest with decision trees instead of their stump-based variants, as the ensemble methods with trees uniformly outperform their stump-based algorithms.

5.2 Full Corpus Results

Feature Set	Naïve Bayes	MaxEnt	SVM	SGDC	AdaBoost	Random Forest	ALR
Unigrams	0.6915	0.8530	0.8315	0.8210	**0.8635**	0.8515	0.8440
Bigrams	0.7495	0.8340	0.7760	0.8165	**0.8365**	0.7985	0.7875
Adjectives	0.6150	0.7845	0.7420	0.7645	**0.8015**	0.7980	0.7830
Uni+POS	0.6940	**0.8690**	0.8230	0.8450	0.8680	0.8640	0.8380
Uni & Bi	0.7595	**0.8840**	0.8220	0.8740	0.8745	0.8585	0.8550

Figure 12: Average 10-fold cross-validation accuracies. Boldface: best performance for feature set.

5.2.1 Classifier Performance

Our results for each classifier on each feature set are summarized in Figure 12. As consistent with prior literature, we observed that the addition of POS tagging gave better performance than feature sets consisting solely of unigrams, bigrams, or adjectives. Bigrams alone fared poorly, worse than unigrams for most classifiers, as consistent with results from Pang, et al. on the first version of the corpus [14]. However, while that study found that unigrams also outperformed a combined unigram-bigram feature set, we find that the combined unigrams and bigrams feature set is the best-performing feature set overall.

Prior research found that SVMs performed best among singular classifiers [14]. Our experiment suggests that the Maximum Entropy classifier consistently outperforms SVMs, often by a significant margin. We also noted, like Bottou, that Stochastic Gradient Descent

outperformed SVMs by around 2-3% on most feature sets [41]. We achieved accuracy results comparable to the best existing classification results on the same data set [21].

On the simple unigram feature set, AdaBoost outperformed all singular classifiers as well as other ensemble methods. MaxEnt performed best on our more complex feature sets. However, the addition of POS and bigram features did not significantly improve accuracy. This indicates that future improvements should explore features beyond simple bag-of-words and syntactic patterns.

We also broke down performance of each classifier on our combined feature set into false positives and false negatives, as seen in Figure 13. While MaxEnt, Naïve Bayes, AdaBoost and ALR had balanced false positives and negatives, SGD, SVMs, and Random Forest were skewed. SGD performs best in correctly classifying negative reviews, with the lowest number of false positives. However, it also exhibits the highest number of false negatives. Random Forest exhibits the same trend, but to a lesser degree. Naïve Bayes, while exhibiting poor overall performance, correctly classifies positive reviews with the same accuracy as MaxEnt and AdaBoost. However, this is tempered by its poor classification accuracy on negative reviews.

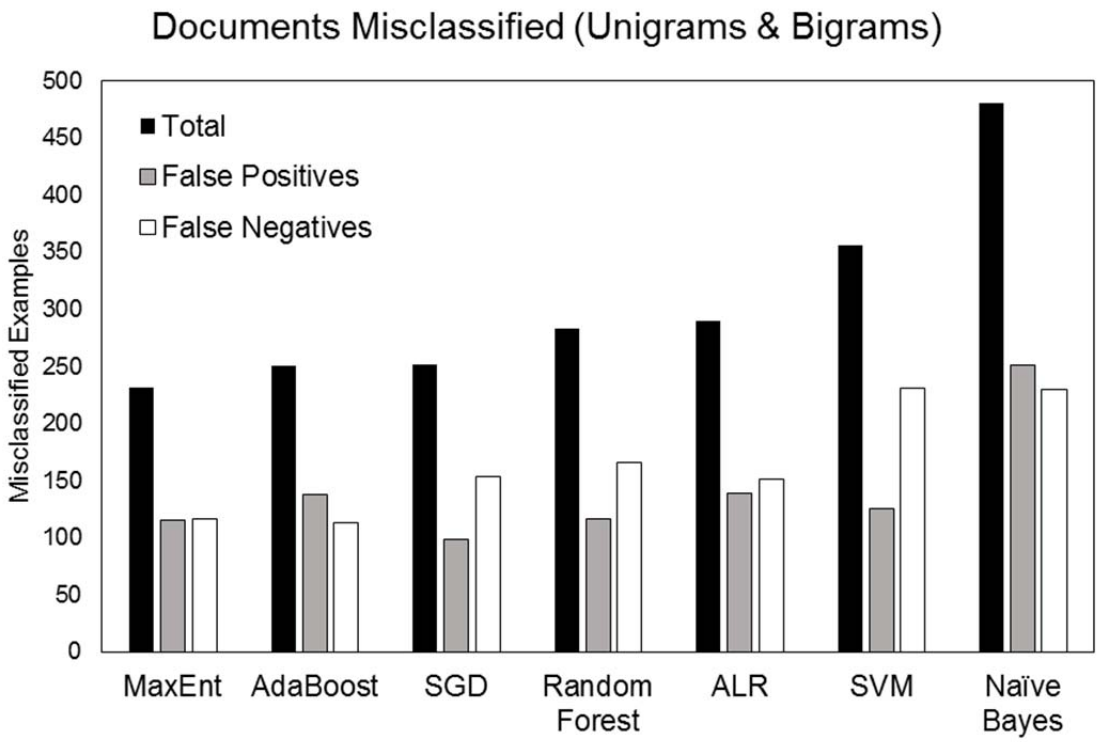

Figure 13: Misclassified reviews over 2000 documents: total, false positives, and false negatives.

5.2.2 Analysis of Misclassified Reviews

To better identify challenges facing sentiment analysis of movie reviews, we analyzed the 128 reviews that were found to be misclassified by all of our classifiers. We observed that 72 (56%) movie reviewers summarized their overall opinion in the last few sentences of their reviews. Lines such as "It's touching, it's sincere, and it's what ultimately make this movie work" and "Don't let these deter you, though; *I Went Down* is a little gem" often come at the end of reviews that lambast the movie for its failing attributes, such as soundtrack, that are not significantly determinant of a reviewer's final opinion. We also notice that these opinion summaries use more complex sentence structures and subtle negation to convey their overall thoughts. This includes direct negations of weak polarity words such as "not a disappointment", as well as doubly-negated sentences like "this is not a great motion picture but, considering how bad most January releases are…" This may be addressed with a combination of effective subjectivity analysis to strip out plot summaries and more detailed linguistic analysis.

Among negative reviews, 13 (21%) contain another type of complex linguistic construct—sarcasm: sentences like "What a *great* idea!" and "Last year, the benevolent studio gods gave us *Digimon*, and this year, they bestow *Max Keeble's Big Move* on delighted moviegoers across the country" appear to convey positive sentiment out of context, but in fact are expressions of the reviewer's negative opinion. Even some human readers have trouble interpreting sarcasm, and NLP has an especially difficult time detecting sarcasm in the "bag"-type feature sets that we use, since the feature extraction strips out most context. This is a significant problem facing sentiment analysis, as current attempts to recognize sarcasm rely on accurate sentiment analysis techniques to begin with [13]. One potential approach would be to use sentence-level sentiment analysis to identify sarcasm through polarity contrast in neighboring sentences and use the results in our final document-level analysis.

Movie reviewers also have a tendency to qualify positive statements with negative follow-ups, and vice-versa, such as "though good-looking, its lavish sets...can do little to compensate for the emotional wasteland". 59 (46%) reviews do so among the misclassified. This qualifying construction balances the number of phrases or sentences marked as positive and negative in a review, and may stymie efforts to extract document-level sentiment polarity using sentence-level or term-based counts. More than half of the current misclassified reviews may be classified correctly with improved negation handling.

	Positive	Negative	Total
Avg. # Sentences Containing Negation	7.23	7.978	7.604
Avg. % Sentences Containing Negation	21.76%	25.40%	23.58%
# of Reviews with Negation	979	992	1971

Figure 14: Statistics on sentences and reviews containing negation

5.3 Negation Handling Results

We investigated the effects of limiting the scope of our negation handling method to only the k words following a negation word, with k from 1 to 3. We compared classifier performance for these three scopes with the unlimited negation scope case, which corresponds to our base negation handling methodology. Accuracy results, including total misclassified documents, false positives, and false negatives for each scope are displayed in Figure 15.

We see that the effect of limiting scope on overall classification accuracy is rather minor, without an obvious trend. For our best-performing singular and ensemble classifiers, MaxEnt and AdaBoost respectively, we see that accuracy increases with enlarged scope, with unlimited negation scope performing the best. However, this trend is absent in SGDC, Random Forest, and Naïve Bayes.

Of greater interest perhaps are the trends exhibited in the false positives and false negatives. In particular, we see that as negation scope increases, the number of false positives increases and the number of false negatives decreases with SGDC. For MaxEnt, AdaBoost, ALR, and SVM, we see the same trend of false negatives decreasing with increased negation scope. This suggests that in positive movie reviews, sentences containing negations are structured so that the entire clause is negated. This structure may be important to intuit, and as such we explore the structure corresponding to limiting negation scope.

We recall that limiting negation scope to $k = 1$ should theoretically improve negation handling or sentiment extraction from some phrases of the form <negation> <adjective modifier> <adjective>, e.g. "not very exciting", where the interpretation "not-very exciting" (somewhat exciting) is more accurate than "not-very not-exciting" (somewhat boring). $k = 2$ covers other instances of the same form, such as "isn't distractingly bad", whose $k = 2$ interpretation is "not-distractingly not-bad" (tolerably okay), as compared to the $k = 1$ interpretation of "not-distractingly bad" (tolerably bad). We note that while "tolerably bad" may more accurately express the author's full view, we would classify this phrase as indicative of positive sentiment more than negative sentiment (minus context). As such, the machine learning classifiers may perform better with the "tolerably okay" interpretation.

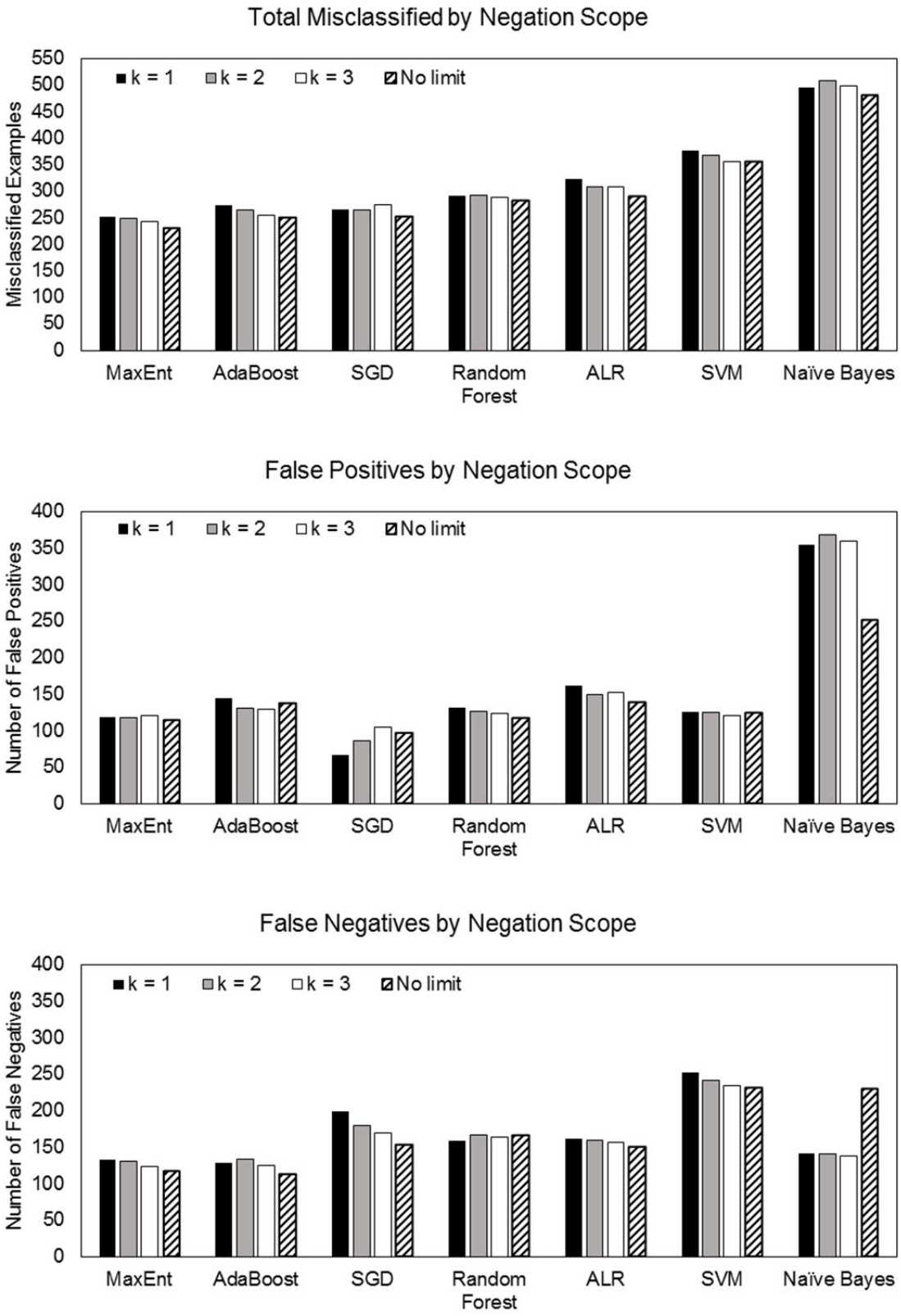

Figure 15: Misclassified reviews, limiting negation to the *k* words after the negation word

We provide general statistics regarding sentences and reviews containing negation in Figure 14. From here, we note that more than 98% of the reviews contain at least one instance of negation, and 23.58% of the sentences in an average review contain a negation. We note the disparity between positive and negative documents: just over one fifth of the sentences in an average positive review contain negations, while over one fourth of the sentences in an average negative review contain negations. We could reasonably expect from this that negation handling would affect the number of false positives (mislabeled negative reviews) more than the number of false negatives (mislabeled positive reviews). However, we cannot make a conclusive statement to that regard, since our investigation is primarily centered around subjectivity analysis methods, and we have not investigated more advanced negation handling methods.

5.4 Subjectivity Analysis Results

Prior to implementing the subjectivity analysis techniques on the polarity dataset, we first evaluated their performance on the subjectivity sentence dataset. We recorded precision of subjectivity classifiers as the percentage of predicted subjective/objective sentences that were classified correctly. Recall was the percentage of total subjective/objective sentences that were predicted to be the correct label. We used the balanced F-score (F_1) to score the various methods:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

The results are shown below in Figure 16. We tested the Simple Adjective Presence (SAP), Simple Adjective Frequency (SAF), Adjective Frequency with SVM (AF), and Part-of-Speech Composition with SVM (POS Comp) classifiers.

Technique	Subjective			Objective		
	Precision	Recall	F_1	Precision	Recall	F_1
SAP	52.76%	89.32%	66.33%	65.21%	20.02%	30.64%
SAF						
$f = 0$	50.00%	100.00%	66.67%	0%	0%	0%
$f = 1$	52.76%	89.32%	66.33%	65.21%	20.02%	30.64%
$f = 2$	56.51%	66.36%	61.04%	59.25%	48.92%	53.59%
$f = 3$	59.13%	38.74%	46.81%	54.45%	73.22%	62.45%
$f = 4$	60.21%	18.64%	28.47%	51.87%	87.68%	65.18%
AF	52.91%	89.76%	66.57%	66.24%	20.08%	30.82%
POS Comp	59.63%	4.00%	7.50%	50.34%	97.32%	66.36%

Figure 16: Evaluating subjectivity analysis techniques on subjectivity dataset.

We note first that SAF with $f = 0$ is equivalent to classifying everything as subjective. SAF with $f = 1$ is equivalent to SAP. As expected, the subjective recall drops precariously as we increase the SAF threshold. For the non-redundant SAF thresholds, we chose $f = 2$ as the best-

performing which we can apply to sentiment analysis. This also makes intuitive sense, as the average number of adjectives in a subjective sentence in the subjectivity corpus is 2.2, and there are an average of 1.8 adjectives in an objective sentence in the subjectivity corpus. Since the sentences in the subjectivity corpus are taken from random IMDB reviews (non-overlapping with the polarity dataset), we expect a similar distribution of adjectives in the polarity dataset. We expect that the addition of these subjectivity analysis techniques is likely to significantly reduce the feature space, since recall for subjective sentences varies from 89.32% to 4%.

5.4.1 Part-of-Speech-based Rules

We first investigate the effect of SAP, SAF, AF, and POS Comp on sentiment analysis classifier performance in our polarity dataset. In these set of experiments, we compared AdaBoost, Random Forest, MaxEnt, and SGDC as the top two ensemble and singular classifiers. We used each subjectivity analysis method to identify and only keep subjective sentences in our corpus, and then used those sentences instead of full reviews for sentiment analysis. A table of results for 10-fold cross-validated accuracy is shown in Figure 17.

We note that with SAP subjectivity analysis, our ensemble classifiers uniformly outperform the singular classifiers. While AdaBoost was the best performing ensemble classifier in the full corpus, with the application of SAF and POS Comp, Random Forest handily outperforms AdaBoost, with AdaBoost performance falling to un-boosted decision tree and Naïve Bayes levels. This may be correlated with the feature space—both SAP and AF have close to 90% recall, as seen in Figure 16. Meanwhile, SAF has 66% recall and POS Comp has an extremely low 4% recall.

We expect that with smaller feature spaces past a certain threshold, precision tends to matter more than recall. POS Comp has significantly lower recall than SAF, but approximately 3% greater precision. However, we see that each classifier performs significantly better on all feature sets under POS Comp as compared to SAF, with improvements ranging from 4% to 7%. As expected, overall best performance is reached on the unigram-based feature sets regardless of subjectivity detection method.

We propose that the effect of feature space reduction (reduced recall on both subjective and objective sentences) is significantly greater than the effect of increased precision in subjectivity classifiers when working with high-recall classifiers. Indeed, SAP has slightly lower recall for subjective and objective sentences than AF, as well as lower precision. However, we see that classifiers under SAP outperform the same classifiers under AF by 2-5%. Further investigation is necessary to verify and explore the effects of these forms of subjectivity analysis.

Full Corpus				
Feature Set	**MaxEnt**	**SVM**	**AdaBoost**	**Random Forest**
Unigrams	0.8530	0.8315	**0.8635**	0.8515
Bigrams	0.8340	0.7760	**0.8365**	0.7985
Adjectives	0.7845	0.7420	**0.8015**	0.7980
Uni + POS	**0.8690**	0.8230	0.8680	0.8640
Uni & Bi	**0.8840**	0.8220	0.8745	0.8585

Simple Adjective Presence				
Feature Set	**MaxEnt**	**SGDC**	**AdaBoost**	**Random Forest**
Unigrams	0.8400	0.8320	**0.8460**	0.8450
Bigrams	0.8125	0.80600	**0.8140**	0.7890
Adjectives	0.7845	0.7630	0.7950	**0.7990**
Uni + POS	0.8480	0.8310	**0.8555**	0.8530
Uni & Bi	0.8545	0.8470	**0.8555**	0.8420

Simple Adjective Frequency, f=2				
Feature Set	**MaxEnt**	**SGDC**	**AdaBoost**	**Random Forest**
Unigrams	0.7485	0.7500	0.6425	**0.7500**
Bigrams	0.7185	**0.7210**	0.6005	0.6860
Adjectives	0.7160	0.6890	0.6395	**0.7170**
Uni + POS	**0.7585**	0.7370	0.6400	0.7480
Uni & Bi	**0.7720**	0.7710	0.6395	0.7400

Adjective Frequency with SVM				
Feature Set	**MaxEnt**	**SGDC**	**AdaBoost**	**Random Forest**
Unigrams	**0.8210**	0.7690	0.8090	0.8010
Bigrams	**0.7800**	0.7745	0.7575	0.7415
Adjectives	0.7645	0.7215	0.7495	**0.7710**
Uni + POS	**0.8155**	0.7910	0.8050	0.8030
Uni & Bi	**0.8310**	0.8295	0.8210	0.8060

Part-of-Speech Composition with SVM				
Feature Set	**MaxEnt**	**SGDC**	**AdaBoost**	**Random Forest**
Unigrams	0.8095	0.7985	0.7160	**0.8135**
Bigrams	**0.7850**	0.7715	0.6665	0.7380
Adjectives	0.75550	0.7420	0.6820	**0.7615**
Uni + POS	**0.8295**	0.8005	0.6985	0.8050
Uni & Bi	**0.8340**	0.8285	0.7035	0.8165

Figure 17: Average 10-fold cross-validation accuracies for different subjectivity analysis techniques. Boldface: best performance for feature set.

5.4.2 Sentence Position

For sentence position subjectivity classifiers, we tested the MaxEnt and SGDC classifiers. First, we tested keeping only the last *k* sentences and using the Unigrams & Bigrams feature set. Cross-validated performance results for each value of *k* from 1 to 7 are shown in Figure 18.

We expect that in general with fewer features, we will see a drop in performance with a corresponding increase in the number of misclassified examples. However, this effect does not seem to be mitigated by taking relevant summary lines (last *k* lines of the review). We do consider the possibility that reviewers may tend toward "grand summarization" or memorable sound bites toward the end of their reviews. As noted in the analysis of frequently misclassified reviews, these memorable moments can make use of more complicated language and syntactic structures that challenge NLP algorithms. By only targeting those regions, perhaps we ignore the simpler indicators, which detracts from the performance of our classifiers.

The graphs in Figure 18 also reveal some interesting insights about the skew of the MaxEnt and SGDC classifiers. For Maximum Entropy, we see roughly equal numbers of false positives and false negatives on the full corpus. As we shrink the feature space by reducing the number of sentences preserved, we see the number of false negatives increase much faster than the number of false positives.

We see a curious reverse in skew for SGDC—on the full corpus, SGDC classifies negative documents with significantly greater precision than positive documents (many more false negatives than false positives), but when we reduce the feature space the trend reverses. This merits further analysis of the SGDC classifier and future examination of whether the linear SVM (which SGDC approximates) shows the same skew reversal. Excepting the full corpus results, we see that the trend of increasing false negatives is also present for SGDC, with false positive numbers exhibiting no directional change as we reduce the number of sentences preserved, and the number of false negatives increasing. This trend points to more complex negation structures in the final sentences. Double negations such as "Don't get me wrong, you don't want to miss this movie" are difficult to manage using our simple negation scheme, and our classifiers may identify this positive recommendation as negative sentiment. This also speaks to the difficulty of negation handling when using term-based feature sets, as even bigrams cannot perfectly capture context required to understand complex sentences.

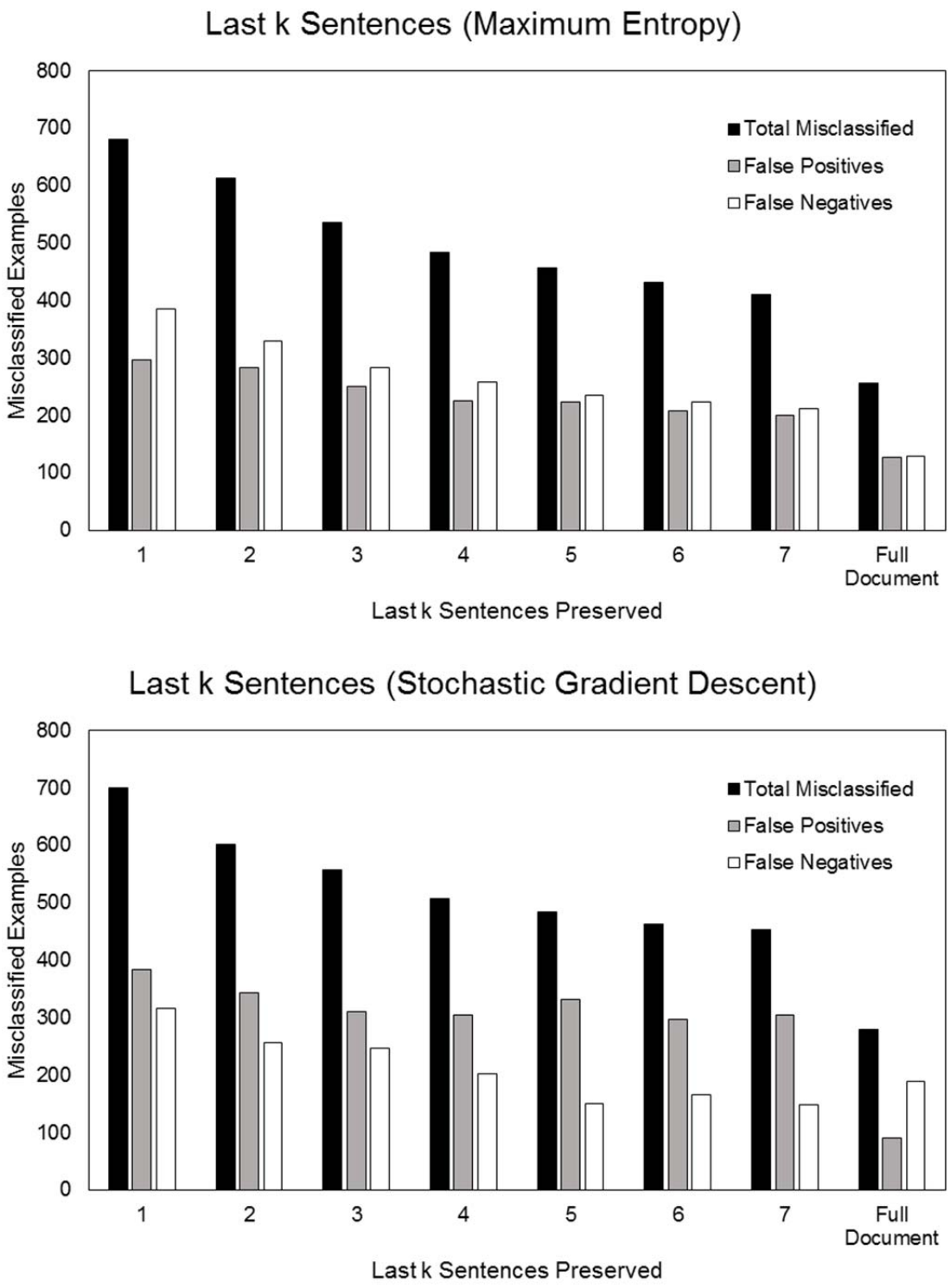

Figure 18: Misclassified, False Positives, and False Negatives, preserving last *k* sentences

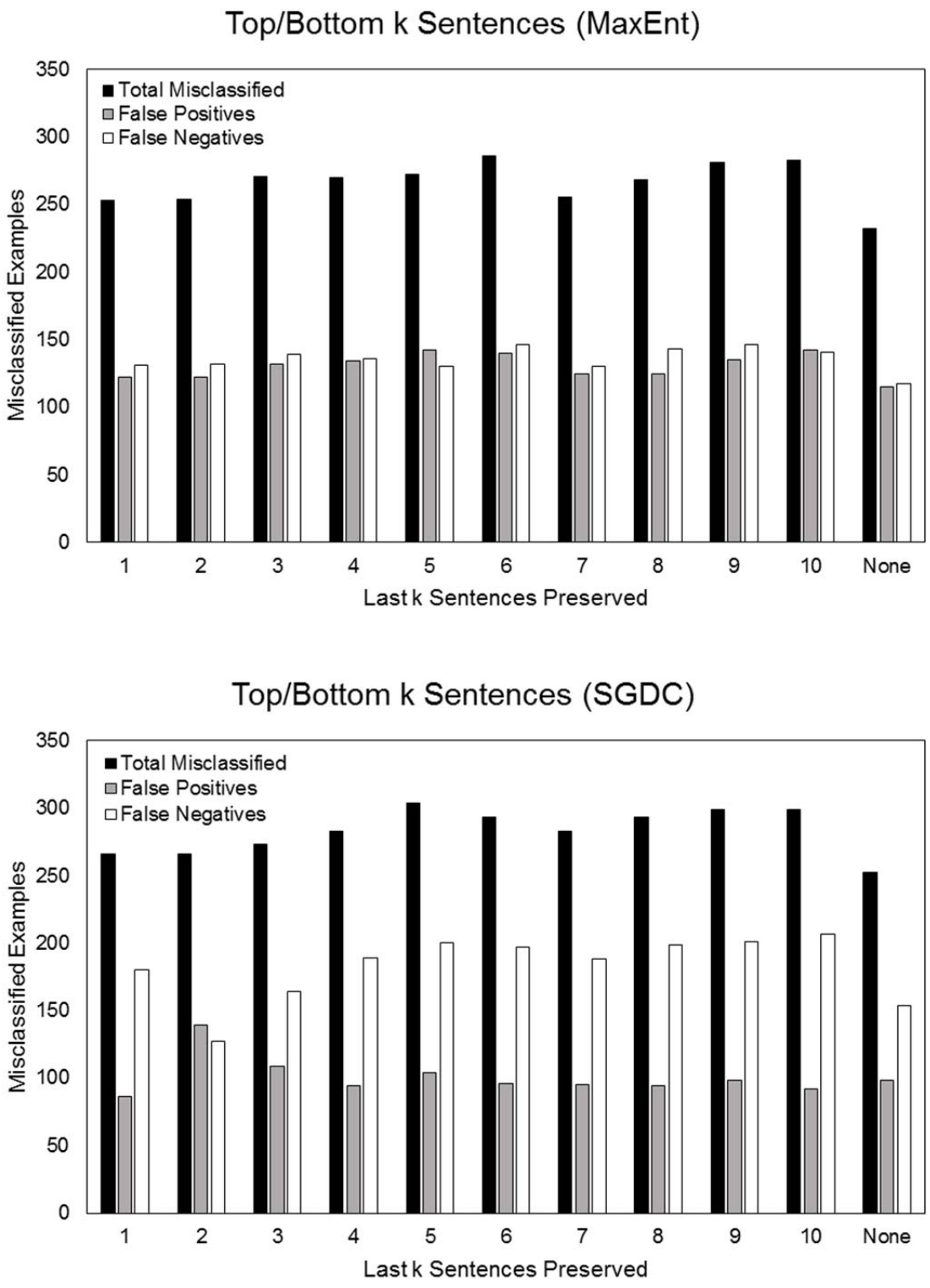

Figure 19: Misclassified, False Positives, and False Negatives in Uni & Bi feature set, preserving first and last *k* sentences. *None* indicates full document sentiment analysis.

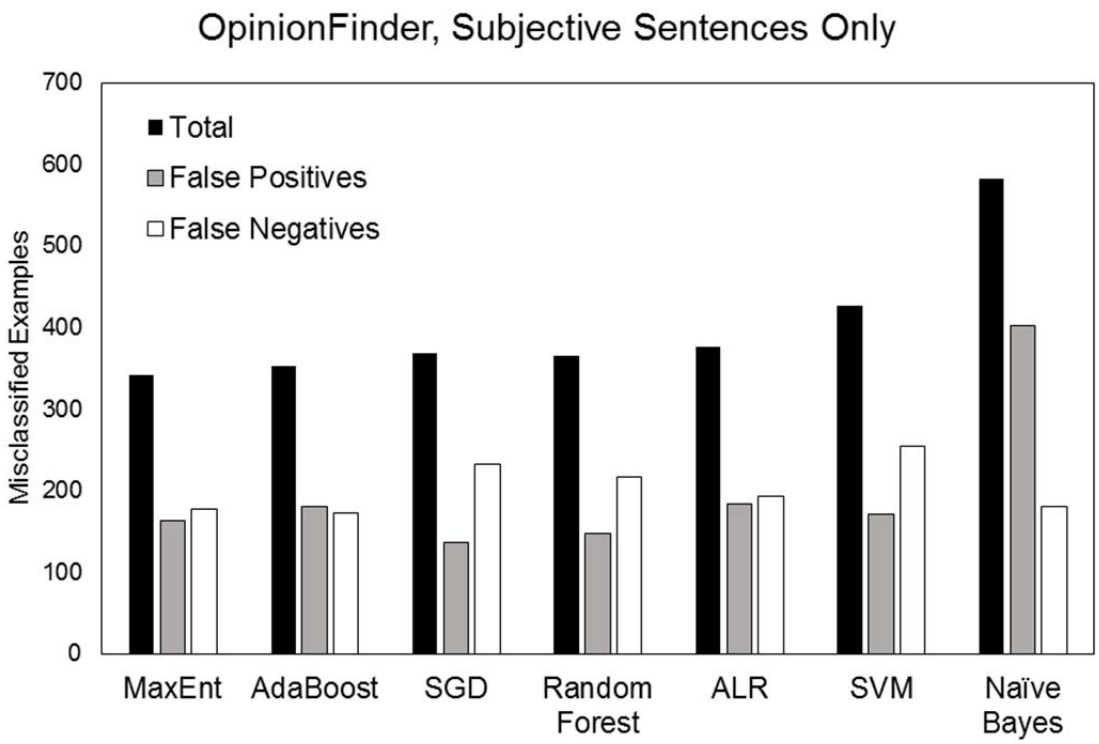

Figure 20: Uni & Bi feature set, using only sentences identified as subjective by OpinionFinder

Our experiments with keeping both the top and bottom *k* sentences yielded fewer actionable insights. Cross-validated results for *k* from 1 to 10 on the Unigrams & Bigrams feature set are shown in Figure 19 on the following page. We see no real trend in classification accuracy as we decrease *k*. This in itself suggests that the beginning of movie reviews syntactically counteract the complexity of final summaries. This may be due to complex negation schemes that do not rely on classic negation words or other structures. This would help explain the consistent classification accuracy seen in our graphs. Of particular interest is the case of $k=2$ for SGDC, where false positives and negatives suddenly even out, though the overall accuracy does not seem to be impacted. This could be caused by a variety of different factors, and merits further exploration. We hypothesize that the first two sentences of a review are significant from a subjectivity standpoint, and that increasing *k* beyond that brings in more plot summary that ultimately clouds sentiment analysis.

For SGDC in particular, with the exception of $k=2$, the skew of the false positives and negatives is consistent with the full corpus. This supports the conjecture that sentence structure and syntactic complexity increases as we approach the end of movie reviews, and that the beginning sentences of movie reviews also contain important clues to overall sentiment.

5.4.3 OpinionFinder

Our first experiment with OpinionFinder kept only sentences marked as subjective by the tool. We tested this with all of our classifiers, and the cross-validated misclassified example statistics are shown in Figure 20. We see that the accuracy trend roughly resembles that of the full corpus numbers in Figure 13. Naïve Bayes sees a significant increase in skew toward false positives, and SVM sees a reduction in skew. When we look at the raw accuracy percentages, we observe that sentiment classifiers after applying OpinionFinder perform similarly to POS Comp and AF, and give inferior performance to SAP. OpinionFinder relies on some context-based cues that make it difficult to analyze the subjectivity dataset, but we ran OpinionFinder on this set to provide precision and recall comparisons to SAP.

On the subjectivity dataset, OpinionFinder exhibited subjective precision of 59.69%, subjective recall of 59.27%, and a subjective F_1 score of 59.48%. It exhibited objective precision of 55.19%, objective recall of 55.63%, and an objective F_1 score of 55.41%. OpinionFinder's subjective precision is higher than that of SAP, with a significantly reduced recall. This supports the proposition that excessive feature space reduction hurts classifier performance. We also propose that movie reviews tend to contain more complex sentence structures in order to link opinions about various topics within context of the movie itself.

Our second experiment with OpinionFinder sought to avoid the excessive feature space reduction issue by keeping predicted objective sentences, but giving a boost to features in subjective sentences. As a result, instead of using term presence as features, we used term frequency, with each term's appearance in subjective sentences counting for k appearances instead of 1, with k varying from 1 to 10. The results are shown in Figure 21.

Immediately, we notice several things. Using term frequency without bonuses (k=1) performs worse than using term presence, consistent with prior literature in this domain [14]. For the MaxEnt classifier, granting any form of bonus decreases performance, although the proportion of false positives to false negatives remains constant. For the SGD classifier, however, we see an immediate reversal of skew as compared to the term presence feature set. The overall trend for SGDC does not seem to tend toward a single direction, but the variance may be due to the stochastic nature of the algorithm itself. Since we see the skew for all values of k, including no bonus at all, this may be due to using frequency instead of presence. We propose further exploration into the usage of various classifiers with frequency instead of presence, as well as a more detailed semantic analysis of misclassified reviews to determine the effects of term frequency on positive and negative documents.

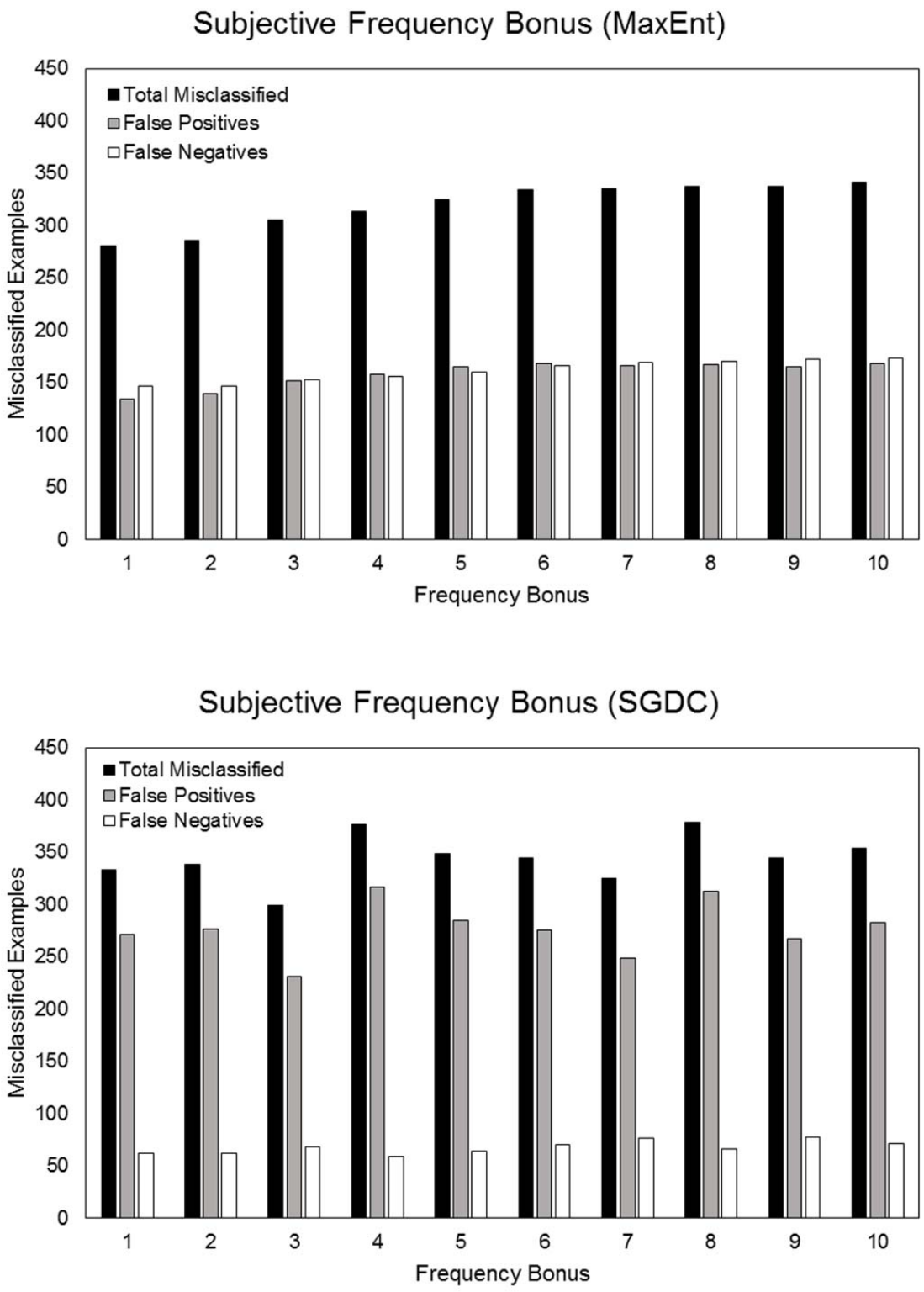

Figure 21: Uni & Bi features, frequency bonus to terms in subjective sentences (OpinionFinder)

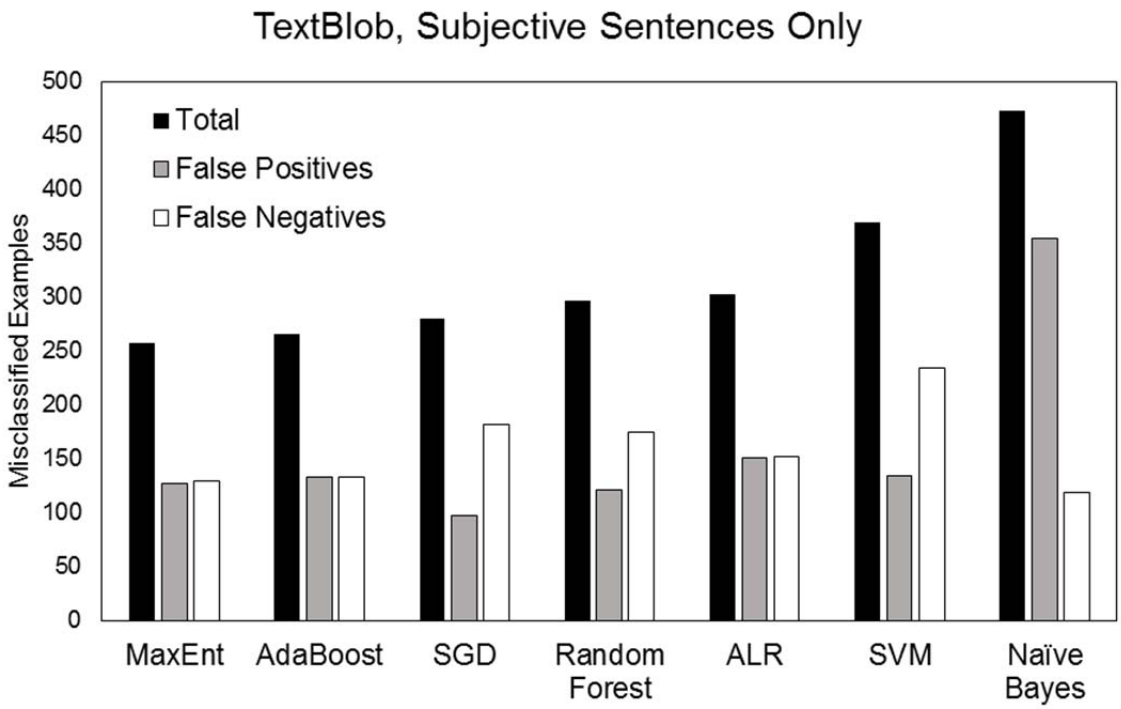

Figure 22: Uni & Bi feature set, using only sentences identified as subjective by TextBlob

Analysis Method	MaxEnt	SGDC	AdaBoost	Random Forest
Full Corpus	0.8840	0.8740	0.8745	0.8585
SAP	0.8545	0.8470	0.8555	0.8420
SAF, f = 2	0.7720	0.7710	0.6395	0.7400
AF	0.8310	0.8295	0.8210	0.8060
POS SVM	0.8340	0.8285	0.7035	0.8165
OpinionFinder	0.8290	0.8155	0.8235	0.8170
TextBlob	0.8710	0.8600	0.8670	0.8515

Figure 23: Classification accuracy for each subjectivity analysis method

5.4.4 TextBlob

We initially utilized TextBlob to limit our corpus to only subjective sentences. Here, we considered any sentence whose numerical subjectivity according to TextBlob was greater than 0.0 to be subjective. We tested this with all of our classifiers, and the cross-validated misclassified example statistics are shown in Figure 22. We see the same trend here as when using OpinionFinder to limit our corpus (Figure 20). Classification accuracy results for the two best singular and ensemble classifiers (MaxEnt, SGDC, AdaBoost, Random Forest) across the full corpus and each form of subjectivity analysis are displayed in Figure 23.

On the subjectivity dataset, TextBlob exhibited subjective precision of 52.14%, subjective recall of 100%, and a subjective F_1 score of 68.54%. It exhibited objective precision of 100%,

objective recall of 8.22%, and an objective F_1 score of 15.19%. We note here that out of all of our subjectivity analysis methods, TextBlob is the only method to achieve 100% subjective recall with greater than 50% subjective precision. Simultaneously, it has the lowest objective recall by far, and 100% objective precision. This suggests that all of the sentences TextBlob classifies as objective are truly objective. The subjectivity corpus test results also suggest that TextBlob always classifies truly subjective sentences as subjective. Essentially, the method discards only the subset of objective sentences that it is absolutely confident of, and preserves all subjective sentences, along with some objective sentences.

This trend places TextBlob as the best subjectivity method we have tested, including favorably comparing to the arguably more sophisticated pattern-matching OpinionFinder utility. Indeed, TextBlob beats out the next best method, SAP, by nearly 1.00% across the board for the four best singular and ensemble classifiers. We do note, however, that the accuracy of a subjectivity-limited corpus here still falls beneath the full corpus classification accuracy. On one hand, TextBlob's performance indicates that with better subjectivity analysis methods, we may reach and perhaps surpass the accuracy on a full corpus. On the other hand, there may be a natural limit to accuracy when we attempt to reduce the feature space (reduce corpus size). If that turns out to be true, we ought also find ways of using subjectivity analysis that do not involve feature space or corpus reduction. Indeed, we investigate such methods in our experiments with manually labeling subjective sentences.

While TextBlob performs better than other subjectivity analysis methods with regards to limiting the corpus, there are still significant weaknesses to the utility. As expected, sentences such as "in 1984, he break-danced during the closing ceremonies of the Olympic games in los angeles", given a 0.0 numerical subjectivity value by TextBlob, are indeed objective. However, as noted above TextBlob tends to rate weakly objective sentences or objective sentences containing some opinionated words as subjective. We see such sentences as "they are soon engaged , and max , because of his own raging libido , grows suspicious of samantha's fidelity" and "the film begins in new york , where we see children dying from a mysterious disease , which is being carried by cockroaches" rated by TextBlob as strongly subjective with subjectivity scores 1.0 and 0.73, respectively.

TextBlob can be considered a Naïve subjectivity classifier, since it returns a weighted average of word subjectivities within a sentence, given an "intensity" modifier determined by preceding words/tokens [16]. For the polarity and subjectivity of each single word, TextBlob averages the scores from all senses of the word listed in its lexicon [33]. As a result, the utility does not take handle part-of-speech or word sense. In the prior examples, we see that "raging"

and "suspicious" contribute to a subjective rating of their parent sentence, although their usages in the sentence do not convey any opinion about the subject of the *review*. Certainly, at the word level, we cannot hope for a subjectivity classifier to consider document-level context such as subjects and topics. As a result, perhaps the subjectivity analysis could be enhanced by being paired with a summarization or topic analysis tool.

5.5 Aggregate Feature Results

Our aggregate features are sorted into classes. The polarity-related features are as follows: *AP* refers to average polarity. *PS* is the average product of polarity and subjectivity. *P0* is average polarity of subjective terms (TextBlob subjectivity score greater than 0.0). *STD* is the standard deviation of polarity. Each polarity feature set contains the aggregate features applied at both the word-level and the sentence-level in a review.

Purity-based aggregate features contain a single feature per set: *pur* is the word purity of a review. *SR* is the sentence purity of a review. *SOR* is the purity of subjective words only, and *subR* is the subjective purity, defined as:

$$\frac{\sum pol(w_i) \times sub(w_i)}{\sum| pol(w_i) \times sub(w_i) |}$$

Finally, we included a collection of aggregate features for the first and last k lines of the review: average polarity (*fkP*, *lkP*) and purity (*fkR*, *lkR*).

Before we applied the aggregate features to machine learning methods, we conducted an initial investigation of classification accuracy via a simple threshold method: if the aggregate feature value was greater than 0.0, we classified the document as positive. Otherwise, we classified the document as negative. We tested this with average word polarity (*AWP*), average sentence polarity (*ASP*), average product of word polarity and subjectivity (*PSW*), average product of sentence polarity and subjectivity (*PSS*), average polarity of subjective words (*PW0*), average polarity of subjective sentences (*PS0*), and average polarity of first or last k sentences (*fkP, lkP*, with k from 1 to 5). Classification results are shown in Figure 24.

We see that as a group, product of polarity and subjectivity tends to do best, while average polarity and average polarity only of subjective words perform comparably with a simple threshold classifier. We note that the full-document aggregate features (*AWP, ASP, PSW, PSS, PW0, PS0*) each are highly skewed toward false positives, which means that many negative reviews had overall positive average polarity as measured by TextBlob.

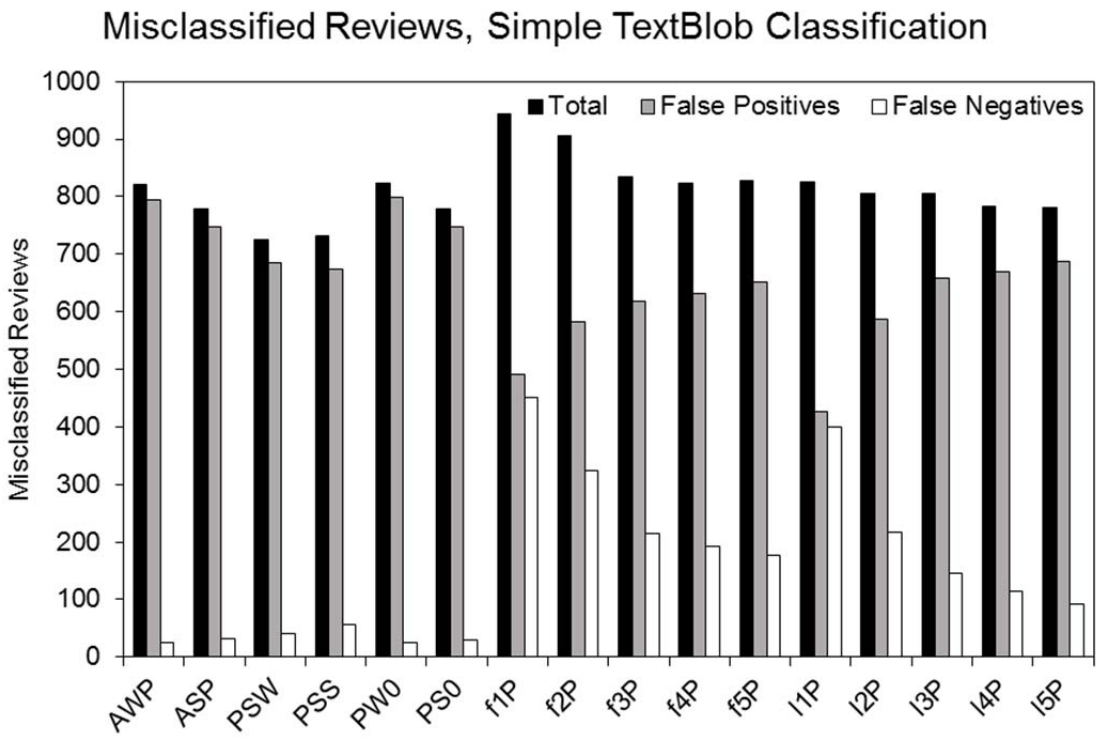

Figure 24: Misclassified reviews, using simple threshold classification with aggregate features

This is not unexpected, as TextBlob's method to return numeric polarities results in sentences like "there are better ways to waste two hours of your life" to be given a positive polarity (+0.15). Here, for example, the contextual usage of "better" here is ignored, and instead TextBlob simply returns the polarity of the average use case of "better", which is positive. In the case of sarcastic or comparative sentences (e.g., "at least it only cost 8 dollars" or "almost any other movie would be better than watching this"), we see TextBlob tends to incorrectly classify the polarity, which would affect our aggregate features. This also supports the observation we made in section 5.2.2, that qualifying sentences interfere with lexicon-based and Naïve polarity extraction methods. In fact, our results suggest that qualifying statements occur more frequently in negative reviews, and thus TextBlob is likely to infer a false positive from the qualifying phrase.

We also observe a pronounced trend in the first- and last-*k* sentence average polarity features. While taking the average polarity of only the first or last sentence results in a fairly even number of false positives and false negatives, increasing the number of sentences (increasing the scope of the aggregate feature) drastically increases the number of false positives and decreases the number of false negatives. This suggests that there is indeed some form of summarization present in most reviews, where the first and last sentences are more straightforwardly opinionated.

5.5.1 Numerical Aggregate Features

We first investigate using aggregate features as numerical features. Our first test used only these aggregate features. Our classification results are displayed in Figure 25.

Aggregate Features	AdaBoost	Random Forest	ALR	MaxEnt	SGDC	SVM
AP	0.6925	0.7105	0.7075	0.7370	0.5285	0.5000
PS	0.6785	0.7020	0.6930	0.7205	0.5300	0.5000
P0	0.6835	0.6945	0.7000	0.7470	0.5275	0.5000
STD	0.5205	0.5170	0.5330	0.5320	0.4995	0.5000
pur	0.6625	0.6620	0.6910	0.7400	0.6230	0.5900
kP	0.6735	0.6845	0.6410	0.6730	0.5000	0.5000
kR	0.6195	0.660	0.6350	0.6895	0.6140	0.6550
AP+PS+P0	0.7135	0.7205	0.7065	0.7435	0.5885	0.5095
Full Combo	**0.7505**	**0.7470**	**0.7220**	**0.7460**	**0.6965**	**0.6640**

Figure 25: Classification accuracy, numerical aggregate features only. Boldface: best performing aggregate feature set for each classifier.

Gezici, et al. found that using a similar set of aggregate features gave 79-81% classification accuracy on the TripAdvisor dataset using SVMs and Logistic Regression [30]. We see here that our accuracy values are significantly lower. However, they are not directly comparable, since we are using a different dataset. Additionally, Gezici et al. utilize tf-idf features, while we use a similar strategy—quantifying how "important" a term is to a document in a corpus—via Mutual Information to limit our term-based feature set. This feature is thus absent when we utilize only numerical aggregate features and ignore term-based features entirely.

We do note that among the aggregate feature classes, the average polarity-based features seem to prove most significant, with classifier performance almost uniformly better on *AP*, *PS*, and *P0* when compared to *STD*, *pur*, *kP*, and *kR*. Nonetheless, using all features (Full Combo) ultimately performs best, consistent with the general trend observed by Gezici, et al. on the TripAdvisor dataset. Full Combo also results in a significantly increased accuracy over just combining the polarity-based aggregate features (AP+PS+P0).

Also of interest is the relatively poor performance of SVMs applied to these numerical features, with most aggregate features leading to performance just as bad as random guessing (50%).

We see somewhat different trends when we included our term-based Unigrams & Bigrams with POS-tagging feature set along with the aggregate features. The results are shown in Figure 26.

Aggregate Features	AdaBoost	Random Forest	ALR	MaxEnt	SGDC	SVM
No Aggregate	0.8745	0.8585	0.8550	0.8840	0.8740	0.8220
AP	**0.8680**	0.8580	0.8470	0.8735	0.8620	0.8215
PS	0.8645	0.8600	0.8405	0.8735	0.8685	0.8210
P0	0.8655	0.8615	0.8455	0.8740	0.8695	0.8225
STD	0.8535	0.8550	0.8415	0.8725	0.8690	0.8180
pur	0.8570	**0.8620**	**0.8550**	0.8785	0.8710	**0.8275**
kP	0.8525	0.8355	0.8430	0.8730	0.8740	0.8210
kR	0.8500	0.8390	0.8480	0.8755	0.8645	0.7905
AP+PS+P0	0.8465	0.8225	0.8385	0.8755	**0.8745**	0.8220
Full Combo	0.8355	0.8115	0.8335	**0.8790**	0.8645	0.7860

Figure 26: Classification accuracy, Uni & Bi + POS with numerical aggregate features. Boldface: best performing aggregate feature for each classifier.

With the addition of term-based features, including all of the aggregate features is no longer the best strategy across-the-board. While MaxEnt still performs best with a combination of all aggregate features, the ensemble classifiers each perform best with a single class of aggregate feature: AdaBoost prefers average polarity, while Random Forest and ALR prefer review purity. Interestingly, SVM no longer exhibits accuracy equivalent to random guessing; in fact, the inclusion of several different classes of aggregate features improves SVM accuracy. While SGDC, SVM, and Random Forest see minor improvements in accuracy with the addition of numerical aggregate features, our two best-performing classifiers on the term-based feature set see decreased performance. While remaining the best-performing classifier, MaxEnt falls from 88.4% to 87.9% classification accuracy. AdaBoost, on the other hand, stays the best ensemble classifier but falls to third place for all classifiers, behind MaxEnt and SGDC.

5.5.2 Binarized Aggregate Features

Perhaps the structure of the features negatively impacted our classifier performance. Our aggregate features so far were numeric features, returning values between -1.0 and +1.0. Meanwhile, our term-based features are all binary features (term presence). As a result, we created binarized aggregate features, where we transformed a numerical aggregate feature F_0 into binary aggregate feature F_1, with $F_1 = TRUE$ if $F_0 \leq m$ and FALSE otherwise, with m being a margin value.

First, we tuned the margin m for each aggregate feature class. We used AdaBoost, MaxEnt, and SGDC classifiers to ensure the tuning wasn't specific to any single classifier. The resulting tuning curves are shown in Figure 27, Figure 28, and Figure 29.

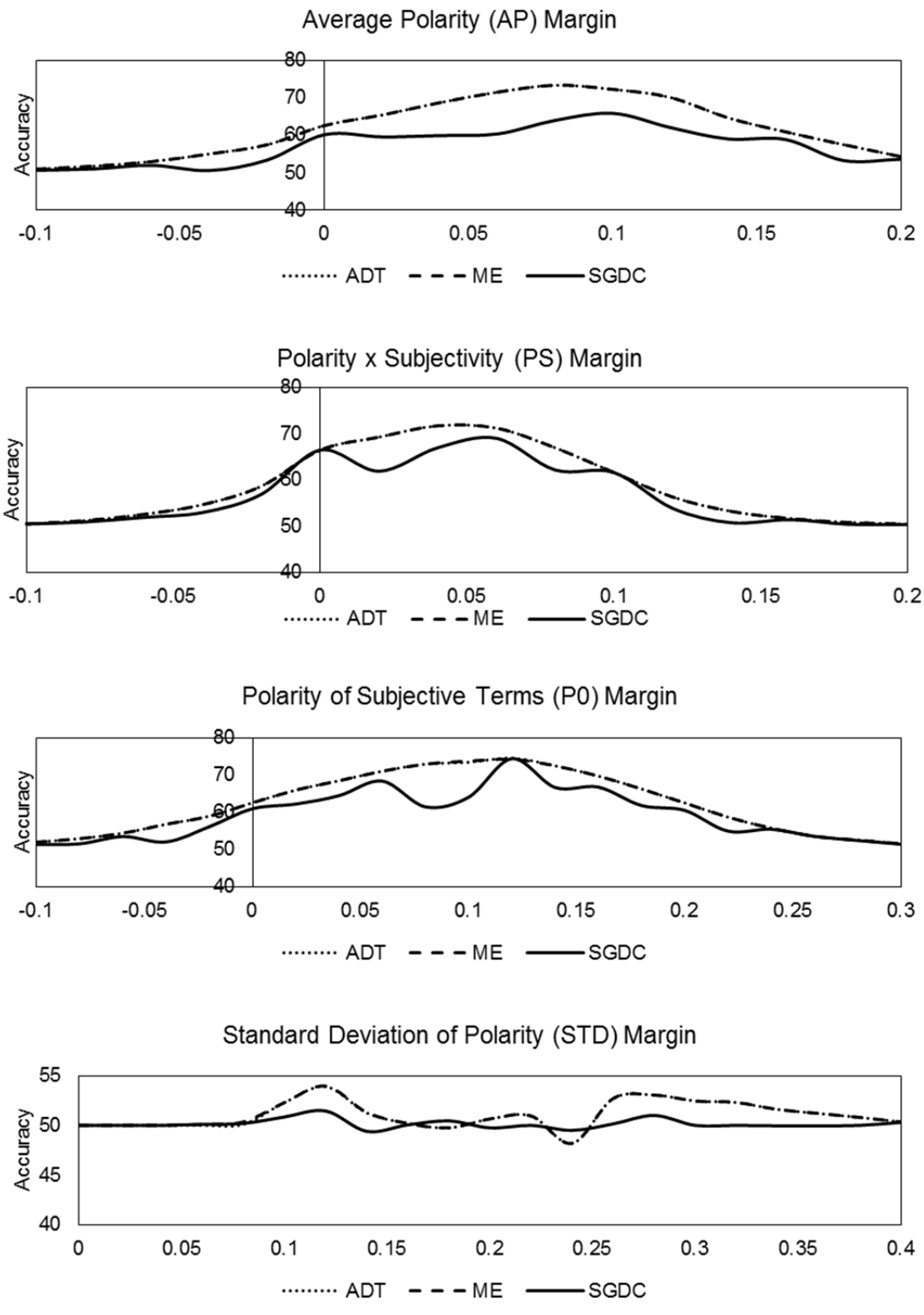

Figure 27: Binary margin tuning for polarity-based aggregate features

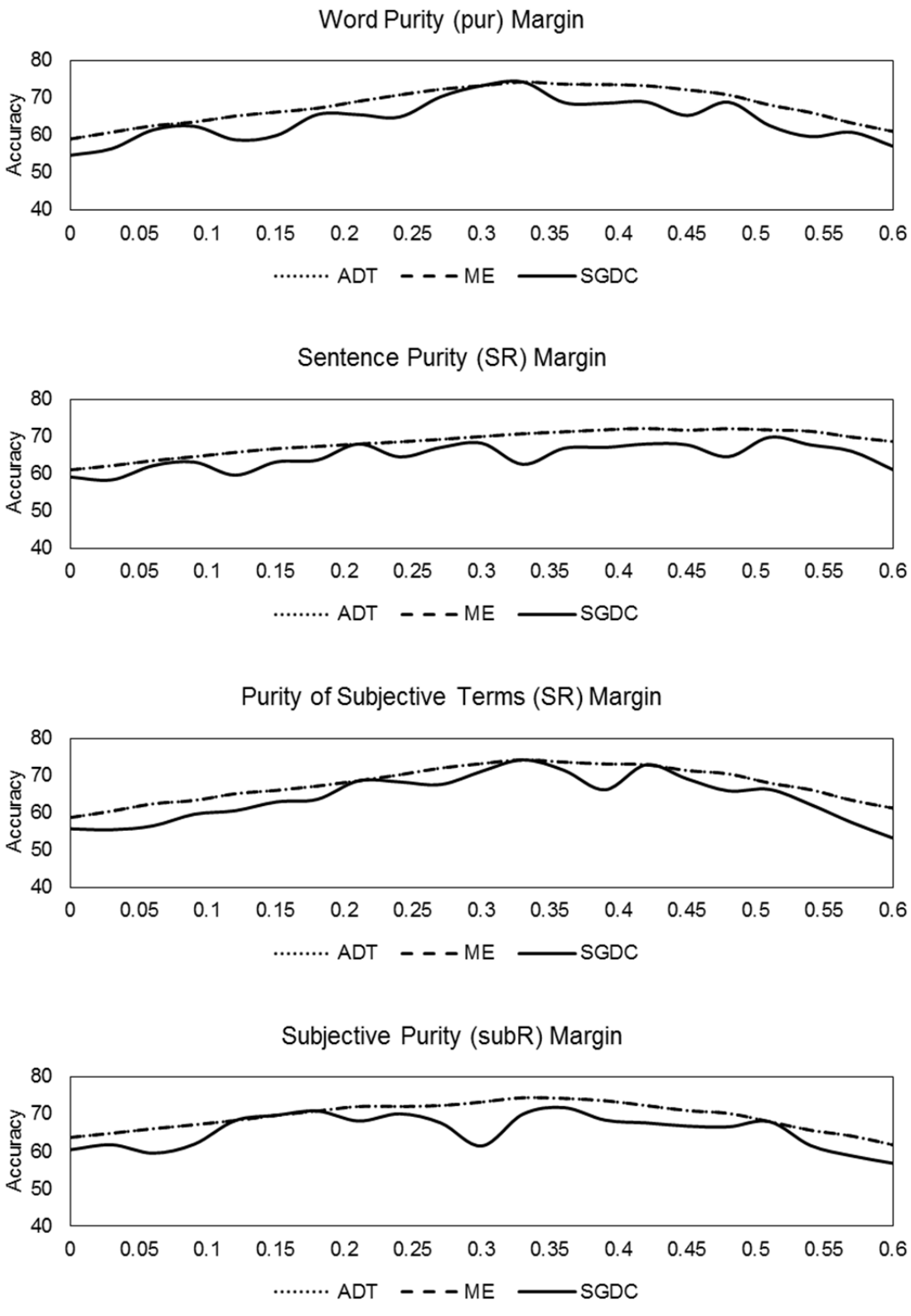

Figure 28: Binary margin tuning for purity-based aggregate features

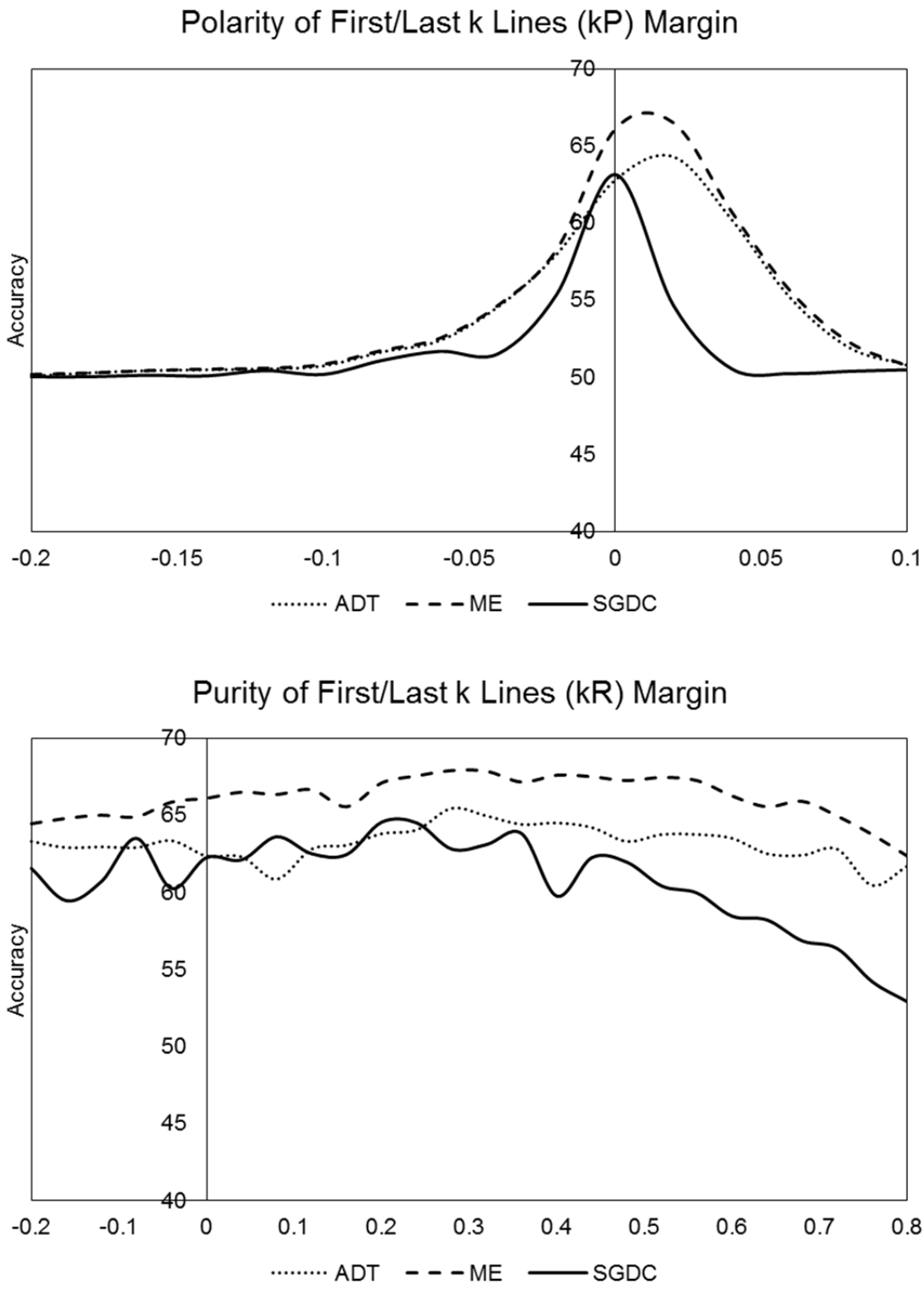

Figure 29: Binary margin tuning for positional aggregate features

Aggregate Features	MaxEnt	AdaBoost	SGDC
No Aggregate	0.8840	0.8745	0.8740
AP	0.8800	0.8595	0.8125
SR	0.8795	0.8625	**0.8615**
SOR	0.8810	0.8565	0.8455
subR	0.8785	0.8670	0.8530
PS	0.8770	0.8560	0.8255
P0	**0.8820**	**0.8695**	0.8425
pur	0.8790	0.8625	0.8510
STD	0.8730	0.8660	0.8430
kP	0.8750	0.8555	0.7335
kR	0.8755	0.8645	0.8300
Combined	0.8720	0.8595	0.6155

Figure 30: Classification accuracy, Uni & Bi + POS with binary aggregate features. Boldface: best performing aggregate feature for each classifier.

The best margins for each aggregate feature class are positive. While this is obviously explained in the case of standard deviation of polarity (STD), it is more interesting when applied to the polarity-based features. Given our binarization convention, a polarity-based feature is binarized to TRUE if the value is less than or equal to the margin. A positive margin thus suggests that there are relatively more negative reviews with average positive term/sentence polarity than vice versa. This is further evidence for our conjecture that negative reviews tend to use more qualifying phrases and sentences, leading to TextBlob/WordNet extracting positive polarities. We are careful to note, however, that this is not the only possible explanation, and other factors may lead to the same trend. For example, it may be that TextBlob/WordNet is more likely to assign an average positive polarity to a "negative" word in its lexicon, or that there are relatively more words that indicate negativity in context of a movie review that have benign or positive meanings in non-movie-related conversation (e.g., "fluffy" and "uneven").

We proceeded to test AdaBoost, MaxEnt, and SGDC with the binarized aggregate features using our tuned margins. Results are shown in Figure 30. We see that while for numeric aggregate features, average word purity and polarity proved the best aggregate features, subjective word polarity dominated binary aggregate features for our two best-performing classifiers (MaxEnt and AdaBoost). SGDC underwent some strange behavior with the inclusion of binary aggregate features, dropping to 61.55% accuracy when combining all features, and dropping to 73.35% when using the first/last k sentence polarity features. However, the aggregate features in general seem to hold promise, as MaxEnt classification accuracy with P0 (average

polarity of words in subjective sentences) is 88.20%, only 0.20% below its accuracy on the full corpus.

We directly compare each classifier's performance on the Uni & Bi + POS feature set with numeric and binary aggregate features in Figure 31. We see that for MaxEnt, binarizing aggregate features improved performance for every aggregate feature on its own, but a combination of numerical aggregate features outperformed the combination of binarized aggregate features. For AdaBoost, on the other hand, combining binary aggregate features outperformed the combination of numerical features; however, the product of polarity and subjectivity (PS) aggregate feature showed better performance when left as numeric. SGDC displayed significantly decreased performance in all cases when binarizing aggregate features, including a particularly egregious drop from 87.45% to 61.55% when combining aggregate features.

Aggregate Features	MaxEnt		AdaBoost		SGDC	
	Numeric	Binary	Numeric	Binary	Numeric	Binary
AP	0.8735	0.8800	0.8680	0.8595	0.8620	0.8125
PS	0.8735	0.8770	0.8645	0.8560	0.8685	0.8255
P0	0.8740	0.8820	0.8655	0.8695	0.8695	0.8425
STD	0.8725	0.8730	0.8535	0.8660	0.8690	0.8430
pur	0.8785	0.8790	0.8570	0.8625	0.8710	0.8510
kP	0.8730	0.8750	0.8525	0.8555	0.8740	0.7335
kR	0.8755	0.8755	0.8500	0.8645	0.8645	0.8300
Combo	0.8790	0.8720	0.8355	0.8595	0.8745	0.6155

Figure 31: Comparing classification accuracy between using numeric and binary aggregate features. Shaded: classification accuracy with binary aggregate features.

5.6 Manual Labeling Results

Our investigation thus far raise new questions regarding the effectiveness of the general term-based ("bag-of-words") model: is there a limit to the accuracy of machine learning classifiers applied to a term-based feature set, regardless of what types of aggregate features or subjectivity analysis is applied? Do aggregate features only improve relatively poor-performing classifiers? Is it possible to surpass the 88.4% accuracy obtained by MaxEnt on the full corpus without aggregate features or subjectivity analysis? To help answer these questions, we manually labeled our move reviews to create a benchmark for "ideal" subjectivity analysis, using the method shown in Figure 2. This generated two new corpuses in addition to the full review corpus: a manually labeled corpus containing only subjective sentences from reviews, and a manually labeled corpus consisting of only the single most significant sentence (opinion-summarizing sentence) from each review. We first applied the three best classifiers (MaxEnt, AdaBoost, SGDC) to each of our

three corpuses. We then investigated adding aggregate features drawn from the manually labeled subjective corpus instead of the full text corpus. Finally, we tested incorporating the summary sentence into our full text and subjective corpuses as a separate feature.

We note here that although we attempted to create a benchmark with an "ideal" version of subjectivity analysis, our numeric polarities and subjectivities for the aggregate features were drawn again from TextBlob. Given our time constraints, we were unable to manually label each word and sentence with a numeric polarity or subjectivity. Thus, we must consider the possibility that a better word- or sentence-level polarity/subjectivity extraction algorithm could further improve aggregate features and thus lead to better classifier accuracy.

Statistic	Original			Manual		
	Positive	Negative	Total	Positive	Negative	Total
Avg. Words	802.93	721.56	762.25	292.47	210.43	251.45
Avg. Sentences	32.94	31.78	32.36	12.25	9.45	10.85
# Contrasting				1.558	1.236	1.397
% Contrasting				12.52%	12.28%	12.40%

	Positive	Negative	Total
Avg. % Subjective Sentences	38.90%	32.77%	35.84%

Figure 32: Subjectivity, summary, and contrasting sentence statistics for manually labeled corpus

5.6.1 Manually Labeled Corpus

Statistics for our manually labeled corpus and comparisons to the full corpus are shown in Figure 32. We immediately notice that after manually labeling the reviews and removing objective sentences, our reviews are much shorter. We see that on average, positive reviews tend to contain significantly more relevant subjective sentences than negative reviews, both proportionally and in absolute numbers of sentences. This may be explained by our methodology for manual labeling: we removed sentences that expressed opinions about movies or performances other than the movie being reviewed. We observed that negative reviews are more likely than positive reviews to mention other movies and then compare those movies to the one being reviewed. As a result, a greater proportion of the negative reviews is likely to have been removed, on average. Interestingly, positive and negative reviews tend to have roughly the same percentage of relevant subjective sentences that express an opinion opposing the overall review polarity. This suggests that qualifying sentences, which seem to be the most frequent form of contrasting polarity sentence, play an important role regardless of the review polarity and are a hallmark of the structure of movie reviews as a whole.

We have also provided histograms for the location of the sentence in each movie review that best summarizes/expresses the opinion of the review as a whole. The position ("normalized

position") was obtained by dividing the sentence position (indexed from 0 for the beginning of the review) by the number of sentences in the document. Normalized position ranges from 0, beginning of the document, to 1 representing the terminal line. These histograms are shown in Figure 33.

We see that the summary sentences for movie reviews tend to be concentrated in the beginning or the end of the review, with the likelihood of summarization decreasing as we approach the center (body) of the review. For negative reviews, reviewers are slightly more likely to summarize their opinions toward the end of the review, but it seems fairly equally likely for summarization to occur at the beginning. For positive reviews, however, reviewers are much more likely to summarize their reviews and recommendations at the end of the review, and structure their review so that there is a gradual "build-up" to the final opinion. However, we do note that sentences expressing general opinion to tend to appear at both ends of the review—we have only labeled, however, the strongest of such sentences. We observe that sentences toward the end of reviews tend to be more of a recommendation-format (e.g. "as a horror film , 'hollow man' is unsophisticated and disturbing (in its intent , not its achievements) and not worth your time or your hard-earned dollars"), while opinionated sentences in the beginning tend toward more straightforward summary of the reviewer's opinions on the various aspects of the movie (e.g. "virtually every aspect of dead-bang is inept and ineffective"). This suggests that the beginning of movie reviews may aid in providing structure for topic analysis to infer opinion about a movie's music, cinematography, acting, or other specific aspects.

Here, we have three distinct corpuses for use with bag-of-words feature extraction: the original corpus consisting of the full movie reviews, here referred to as "original" or "full"; the manually labeled corpus consisting of only the subjective sentences from each movie review, referred to in this section as "manually labeled" or "subjective"; and the corpus consisting of the single summary sentence of each movie review (tagged in our manual labeling with ***), referred to here as the "summary" corpus. We performed classification on each base corpus with MaxEnt, AdaBoost, and SGDC. The misclassified reviews charts are shown in Figure 34. The results in context of our other subjectivity analysis algorithms are shown in Figure 35.

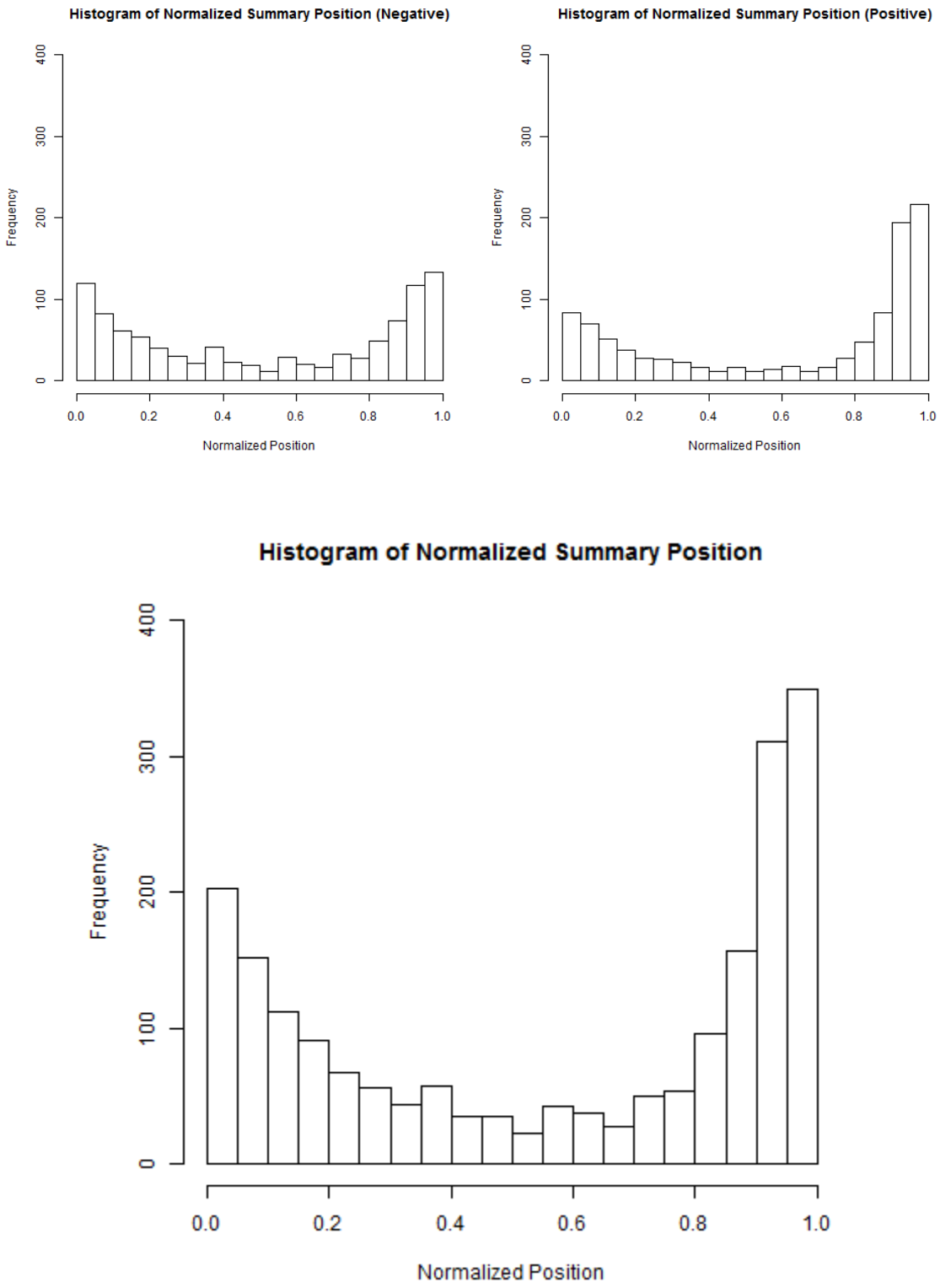

Figure 33: Normalized positions of summary sentences for negative, positive, and all reviews

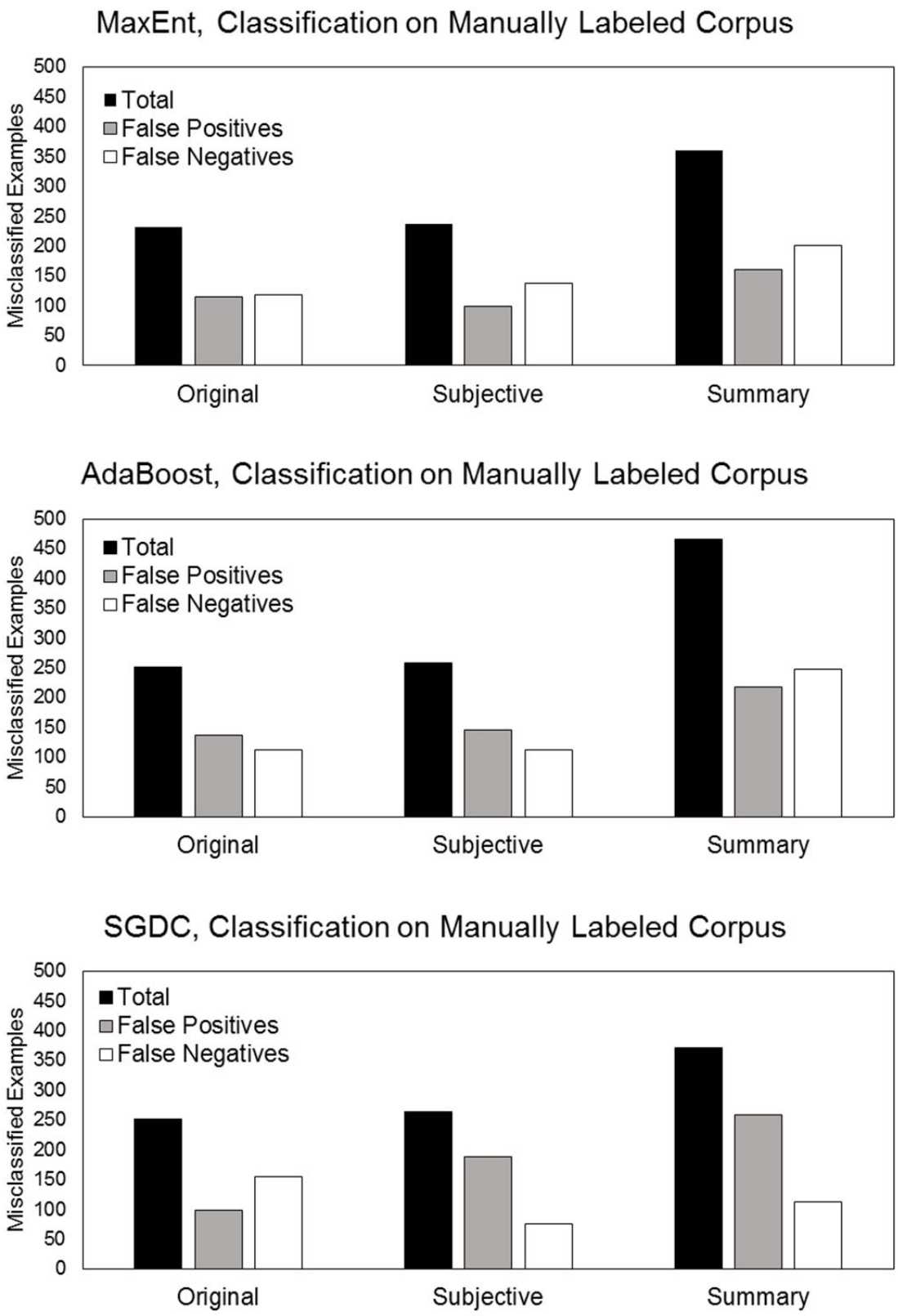

Figure 34: Misclassified reviews, using manually labeled subjective and summary sentences

Analysis Method	MaxEnt	AdaBoost	SGDC
Full Corpus	0.8840	0.8745	0.8740
SAP	0.8545	0.8555	0.8470
SAF, f = 2	0.7720	0.6395	0.7710
AF	0.8310	0.8210	0.8295
POS SVM	0.8340	0.7035	0.8285
OpinionFinder	0.8290	0.8235	0.8155
TextBlob	0.8710	0.8670	0.8600
Manual Labeling	0.8820	0.8705	0.8680
Manual Summary	0.8200	0.7670	0.8145

Figure 35: Classification accuracy for each subjectivity analysis method, including manual

As expected, here we see that manual labeling of subjective sentences outperforms all other forms of subjectivity analysis. Keeping only the summary sentences, however, performs rather poorly, outperforming only simple adjective frequency (and POS SVM for AdaBoost). This is most likely due to the severely restricted feature space caused by only keeping a single sentence from each review. However, our theoretically ideal subjectivity analysis method, manual labeling, still underperforms classification on the full corpus. This is further evidence that classification performance on a bag-of-words representation of text is reliant on the size of the full feature space, and that to improve performance, we should focus on methods to add features rather than feature space restriction.

5.6.2 Aggregate Features from Manually Labeled Corpus

Thus, we look to adding aggregate features, with the manually labeled reviews as base documents. Aggregate features are drawn from known subjective sentences. We note that this makes average polarity of subjective sentences (*PS0*) equivalent to the average sentence polarity of the document (*ASP*). We performed classification using a Unigrams & Bigrams + POS representation of each base corpus (original, subjective, summary) for each class of aggregate feature, using MaxEnt, AdaBoost, and SGDC. The results are shown in Figure 36, with classifier names compressed as ME, ADT, and SGDC respectively.

Our results are highly encouraging—we see an improvement over using only the full corpus for many of the feature sets. More importantly, we see that combining the base subjective corpus with aggregate features drawn from the subjective corpus can also result in better classification accuracy than only using the full corpus model. We note that the best performance noted thus far come from the MaxEnt classifier on the *P0* feature set, including average polarity of subjective words and sentences. While Gezici et al. found they achieved best performance when combining all classes of aggregate features, we note that individual aggregate feature classes provide better performance than simply adding every aggregate feature ("Combined"). Combined

performance does approach best-in-class performance for the Maximum Entropy classifier, however (89.10% vs. best-performing 89.20% for *P0*).

The high performance of the *P0* aggregate feature class suggests that further gains may be made by improving our word polarity lexicon. As WordNet and SentiWordNet are general lexicons, it may be helpful to utilize domain-specific lexicons for creating aggregate features.

Aggregate Feature	Original			Subjective			Summary		
	ME	ADT	SGDC	ME	ADT	SGDC	ME	ADT	SGDC
AP	0.8815	0.8760	0.8740	0.8880	0.8725	0.8690	**0.8705**	0.8285	**0.8320**
SR	**0.8870**	0.8790	0.8640	0.8915	**0.8780**	0.8570	0.8580	0.8075	0.8170
SOR	0.8830	0.8770	0.8725	0.8875	0.8745	0.8515	0.8575	0.8160	0.8030
subR	0.8855	**0.8805**	0.8745	0.8900	0.8765	0.8540	0.8535	0.8015	0.8155
PS	0.8775	0.8785	0.8760	0.8870	0.8670	0.8785	0.8605	**0.8325**	0.8305
P0	0.8840	0.8800	**0.8845**	**0.8920**	0.8670	0.8620	0.8640	0.8225	0.8305
pur	0.8830	0.8770	0.8725	0.8900	0.8775	0.8510	0.8550	0.8040	0.8085
STD	0.8730	0.8720	0.8655	0.8810	0.8705	**0.8805**	0.8215	0.7650	0.8160
kP	0.8750	0.8675	0.8720	0.8830	0.8550	0.8700	0.8390	0.8215	0.8200
kR	0.8855	0.8620	0.8420	0.8895	0.8595	0.8415	0.8485	0.8155	0.7970
Combined	0.8850	0.8510	0.8300	0.8910	0.8455	0.8425	0.8560	0.8015	0.8255

Figure 36: Classification accuracy for aggregate features based on manually labeled subjective documents. Boldface: best performing aggregate feature class for each base corpus and classifier.

5.6.3 Using Summary Sentences

We have thus far observed relatively low classification accuracy for using summary sentences as a base corpus. We turn now to using these summary sentences to improve our other base corpuses, rather than as a corpus by itself. We thus create a feature set consisting of the average word polarity, sentence polarity, and purity of the summary sentence. The numerical values required for calculating averages and purity are once again acquired using TextBlob. We ran classification on the original and summary base corpuses with the addition of only summary sentence features, as well as the addition of all aggregate features and summary sentence features. The classification accuracies are reported in Figure 37 and misclassified examples in Figure 38.

Base Corpus	Additions	MaxEnt	AdaBoost	SGDC
Original	None	0.8840	**0.8745**	0.8740
Subjective		0.8820	0.8705	0.8680
Summary		0.8200	0.7670	0.8145
Original	Summary	0.8895	0.8690	**0.8920**
Subjective		0.8960	0.8725	0.8570
Original	Summary + Aggregate	0.8970	0.8640	0.8550
Subjective		**0.8980**	0.8515	0.8570

Figure 37: Classification accuracy for various base corpuses, including summary features and aggregate features. Boldface: best-performing corpus-addition combination for each classifier.

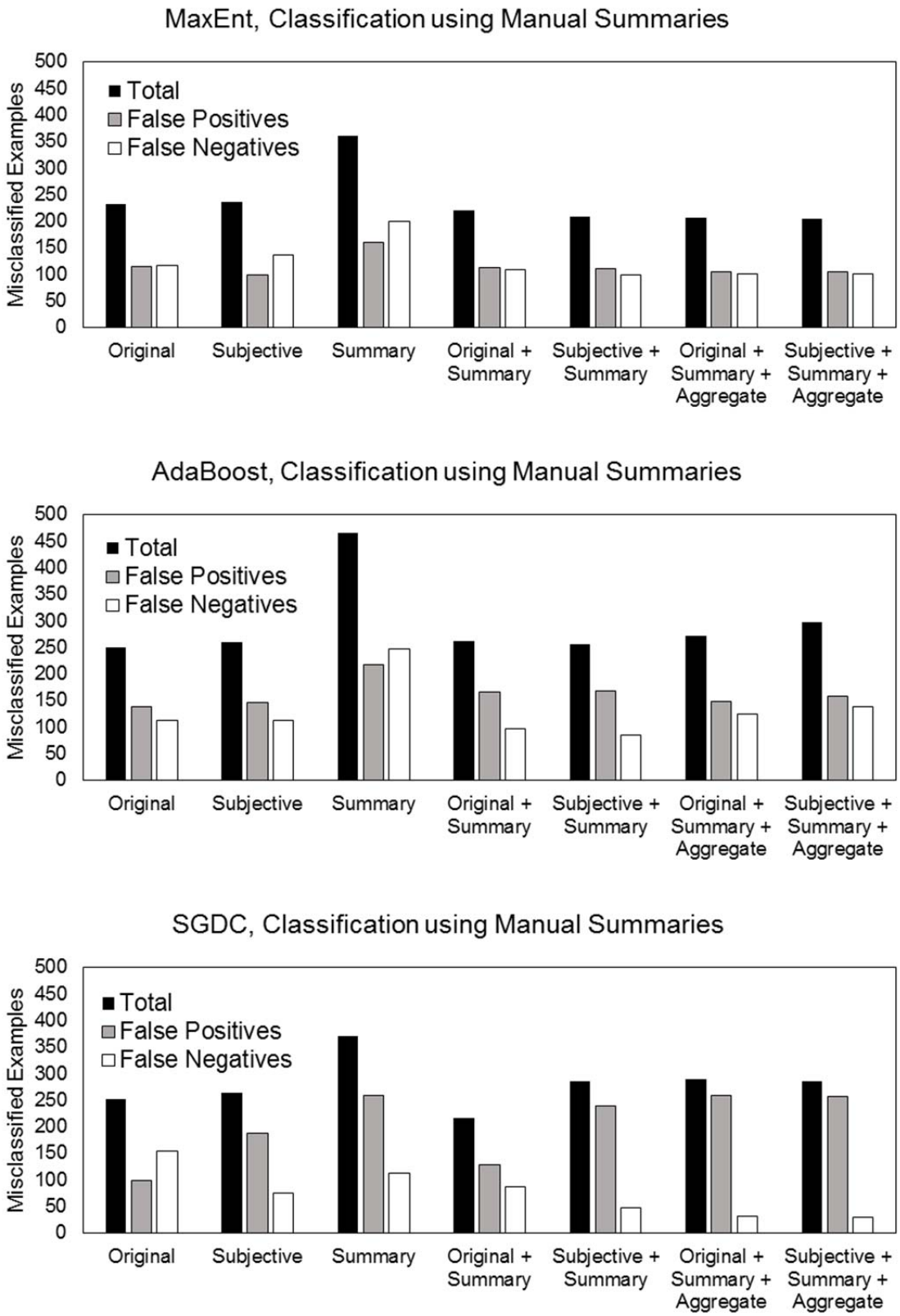

Figure 38: Misclassified reviews, incorporating summary features and aggregate features

We see that our best performance across all experiments is obtained using the MaxEnt classifier on the feature set generated by adding our summary features and all aggregate feature classes (drawn from the manually labeled subjective corpus) to the subjective base corpus. The addition of only summary features to the original feature set provides the best performance for SGDC. For AdaBoost, however, we see that summary features do not improve the original base corpus. While adding summary features only to the subjective feature set does boost performance, it does not reach or surpass AdaBoost performance on the original corpus. This is particularly interesting in light of the improvements to AdaBoost performance made with certain aggregate feature classes added to the full and subjective corpuses. Further research is necessary to determine why MaxEnt outstrips ensemble methods (and other singular methods) when aggregate features and subjectivity analysis come into play.

We also note that the general skew of false positives to false negatives is preserved for MaxEnt and AdaBoost when adding summary and/or aggregate features to the base corpus. For SGDC, however, the skew is reversed, and indeed we see much better performance from SGDC on positive reviews (significantly higher false positives than false negatives) with the addition of summary and aggregate features. We propose further trials to understand whether this is inherent to the classifier or a property of the review text.

6 Conclusion and Future Work

6.1 Conclusions

We have conducted experiments on three ensemble methods and four singular classifiers, demonstrating that AdaBoost outperforms all other classifiers with simple unigrams in movie reviews. We have also found that Maximum Entropy and AdaBoost perform best out of all classifiers when we utilize methods to improve and add detail to the feature set. Overall, the Maximum Entropy classifier provides best performance, especially as aggregate features and subjectivity analysis are applied to the base corpus. SGDC results in similar performance to AdaBoost, but is much faster to train, and as such may be preferred to ensemble methods in real-world usage with tight time constraints. In further analyzing misclassified reviews, we have also identified literary tendencies in movie reviews toward sarcasm, summarization and qualifying phrases that obscure sentiment and add additional challenges to sentiment analysis tasks in this domain. We propose future work to combine semantic and syntactic features with simple bag-of-word features.

To give more focus to reviewer opinion summaries, we look to subjectivity analysis. Pang and Lee showed that taking only subjective sentences using a supervised minimum-cut-based approach improves training speed but does not significantly impact accuracy [21]. Preliminary tests with other, simpler supervised subjectivity analysis approaches based on adjective presence and frequency resulted in reduced accuracy. We have demonstrated that even an ideal form of subjectivity analysis represented by manual labeling cannot improve classifier performance with just a feature set consisting of unigrams, bigrams, and part-of-speech labeling. However, our manual subjectivity analysis combined with aggregate features and summary sentence features have shown to outperform the base corpus. This suggests that subjectivity analysis methods can still contribute to the furtherance of feature extraction methods in the field of sentiment analysis. In future work we hope to evaluate an unsupervised subjectivity classification algorithm based on calibrated expectation minimization [47]. We do note as well that even imperfect subjectivity analysis can reduce the feature space enough to decrease training time and feature generation time. As such, imperfect methods of subjectivity analysis, when combined with aggregate features, may achieve similar classification performance to full corpus sentiment analysis while decreasing time needed.

As discussed previously, aggregate features play an important role in improving sentiment analysis. With improved subjectivity analysis, we hope to see better performance with aggregate features drawing from the subjective corpus. At the word-level, we see potential in

utilizing automated generation of domain-specific polarity lexicons to improve aggregate features [18] [48]. In the same vein, we saw improvements to accuracy with the inclusion of summary sentence features, and thus we aim to use automated methods for identifying summary sentences or generating summaries of each movie review [49] [50].

Based on further analysis of misclassified reviews, we see potential to improve performance with better negation handling. We have evaluated a simple method to limit the scope of our negation handling, but we have measured no improvement over the base method. We consider starting with a more sophisticated negation handling method and incorporating it into the AdaBoost and Maximum Entropy frameworks: a rule-based approach relying on the polarity of the first and second words in a bigram [51].

For the movie review domain, we have identified several topics within the realm of movies that can trigger opinions, including characterization, acting, costuming, and dialogue. We see potential to improve performance by using topic analysis to break up the review into discrete aspects. This may also be used to provide services to consumers to better tailor movie recommendations or evaluations for moviegoers with varying preferences. The approach would be similar to Hu and Liu's approach to mining and summarizing product reviews [52].

6.2 Limitations

Sentiment analysis of movie reviews may benefit from a better raw corpus. The existing corpus is entirely in lower case. As a result, it is hard for machines to detect movie names. Some sentences in movie reviews talk about other movies as setup to compare the two movies, and these "background" sentences should be removed prior to analysis, since they may unduly influence machine learning methods, especially Naïve methods. We also consider topic analysis to be a potentially effective way of controlling for multiple subjects in a document.

Regarding our studies of the manually labeled corpus, we noted that single aggregate feature classes outperformed a combination of all features. For our experiments with the summary sentences (section 5.6.3), one of the feature sets we used was a combination of a base corpus Unigrams & Bigrams + POS set with the summary features and a combination of all aggregate features. Due to time constraints, we were unable to run classification using the base corpus, summary statistics, and best-performing single aggregate feature class for each different classifier.

As of now, the author is the sole contributor to the manually labeled corpus. We may achieve better results through multiple human viewpoints contributing to manual labeling. The most efficient way to accomplish this would be to submit each review to multiple humans for manual labeling, and taking an average or weighted vote (or even random selection) to determine

which sentences should be removed as objective or irrelevant. We suggest the usage of utilities to leverage large-scale human input and manpower, such as Amazon's Mechanical Turk system, to generate a more robust benchmark data set.

6.3 Future Directions

Our research, though exploring new avenues in sentiment analysis, also leaves many questions in natural language processing and sentiment analysis unanswered. In the future, we hope to conduct research on more robust methods of sarcasm detection within reviews, as well as concept-based feature selection. We also propose to conduct future research on pairing these ensemble and data processing methods with the AMOD framework, testing on various domains.

These algorithms can also be implemented as part of an application or website for review aggregation and querying. We envision future research centering on the creation of a responsive chat-based system for product and service recommendations and reviews. Such a system may intelligently develop new questions and topics of discussion based on a user's responses. We also envision a multi-step tool for NLP research, where one may submit a document and observe each processing step. This can allow for on-the-fly rule adjustment, as well as online learning for our classifiers.

We have noticed different trends in the skew of misclassified reviews toward false positives or negatives. Future work may investigate why certain classifiers tend to skew in different directions, and why the addition of aggregate features or summary features in some cases reverses the skew.

References

[1] comScore and The Kelsey Group, "Online Consumer-Generated Reviews Have Significant Impact on Offline Purchase Behavior," comScore, 2007.

[2] J. B. Horrigan, "Online Shopping," Pew Internet & American Life Project, 2008.

[3] P. Resnick and R. Zeckhauser, "Trust Among Strangers in Internet Transactions: Empirical Analysis of eBay's Reputation System," *Advances in Applied Microeconomics,* vol. 11, pp. 127-157, 2002.

[4] S. Ba and P. A. Pavlou, "Evidence of the Effect of Trust Building Technology in Electronic Markets: Price Premiums and Buyer Behavior," *MIS quarterly,* pp. 243-268, 2002.

[5] Gxlnz, "What is the best purchase you have ever made?," Reddit, 27 October 2015. [Online]. Available: https://www.reddit.com/r/AskReddit/comments/3qep18/what_is_the_best_purchase_you_have_ever_made/.

[6] J. A. Chevalier and D. Mayzlin, "The Effect of Word of Mouth on Sales: Online Book Reviews," *Journal of Marketing Research,* vol. 43, pp. 345-354, 2006.

[7] H. Rahmath and T. Ahmad, "Fuzzy based Sentiment Analysis of Online Product Reviews using Machine Learning Techniques," *International Journal of Computer Applications,* vol. 66, no. 17, pp. 9-16, 2013.

[8] "Internet Live Stats," Real Time Statistics Project, [Online]. Available: http://www.internetlivestats.com/twitter-statistics/. [Accessed 5 October 2015].

[9] S. R. Das and M. Y. Chen, "Yahoo! for Amazon: Sentiment Parsing from Small Talk on the Web," in *EFA 2001 Barcelona Meetings*, 2001.

[10] A. Tumasjan, T. O. Sprenger, P. G. Sandner and I. M. Welpe, "Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment," in *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[11] B. Liu, "Sentiment Analysis: A Multi-Faceted Problem," *IEEE Intelligent Systems,* vol. 25, no. 3, pp. 76-80, 2010.

[12] E. Cambria, B. Schuller, Y. Xia and C. Havasi, "New Avenues in Opinion Mining and Sentiment Analysis," *IEEE Intelligent Systems,* vol. 28, no. 2, pp. 15-21, 2013.

[13] E. Riloff, A. Qadir, P. Surve, L. De silva, N. Gilbert and R. Huang, "Sarcasm as Contrast between a Positive Sentiment and Negative Situation," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013.

[14] B. Pang, L. Lee and V. Shivakumar, "Thumbs Up? Sentiment classification using machine learning techniques," in *Proceedings of the Conference on Empirical Methods in Natural*

Language Processing, 2002.

[15] T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff and S. Patwardhan, "OpinionFinder: A system for subjectivity analysis," in *Proceedings of hlt/emnlp on interactive demonstrations*, 2005.

[16] S. Loria, P. Keen, M. Honnibal, R. Yankovsky, D. Karesh, E. Dempsey, W. Childs, J. Schnurr, A. Qalieh, L. Ragnarsson, J. Coe and A. L. Calvo, "TextBlob: Simplified Text Processing," TextBlob, 17 2 2016. [Online]. Available: https://textblob.readthedocs.org/en/dev/.

[17] A. Harb, M. Plantie, G. Dray, M. Roche and F. Trousset, "Web Opinion Mining: How to extract opinions from blogs?," in *International Conference on Soft Computing as Transdisciplinary Science and Technology*, 2008.

[18] P. Turney and M. Littman, "Measuring Praise and Criticism: Inference of Semantic Orientation from Association," *ACM Transactions on Information Systems,* vol. 21, no. 4, pp. 315-346, 2003.

[19] N. Silva, E. Hruschka and E. R. Hruschka, "Biocom Usp: Tweet Sentiment Analysis with Adaptive Boosting Ensemble," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, Dublin, 2014.

[20] B. Gokulakrishnan, P. Priyanthan, T. Ragavan, N. Prasath and A. Perera, "Opinion Mining and Sentiment Analysis on a Twitter Data Stream," in *The International Conference on Advances in ICT for Emerging Regions - ICTer 2012*, 2012.

[21] B. Pang and L. Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts," in *Proceedings of the 42nd ACL*, 2004.

[22] C. Lin, Y. He and R. Everson, "A Comparative Study of Bayesian Models for Unsupervised Sentiment Detection," in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, 2010.

[23] S. Kim, F. Li, G. Lebanon and I. Essa, "Beyond Sentiment: The Manifold of Human Emotions," *arXiv:1202.1568,* pp. 1-15, 2013.

[24] P. Turney, "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, 2002.

[25] E. Kouloumpis, T. Wilson and J. Moore, "Twitter Sentiment Analysis: The Good the Bad and the OMG!," in *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, 2011.

[26] Y. Wilks and M. Stevenson, "The grammar of sense: Using part-of-speech tags as a first step in semantic disambiguation," *Natural Language Engineering,* vol. 4, no. 2, pp. 135-143, 1998.

[27] M. F. Porter, "An algorithm for suffix stripping," *Program,* vol. 14, no. 3, pp. 130-137,

1980.

[28] V. Balakrishnan and E. Lloyd-Yemoh, "Stemming and Lemmatization: A Comparison of Retrieval Performances," *Lecture Notes on Software Engineering,* vol. 2, no. 3, pp. 262-267, 2014.

[29] H. Ghorbel and D. Jacot, "Sentiment analysis of French movie reviews," in *Advances in Distributed Agent-Based Retrieval Tools*, Berlin, 2011.

[30] G. Gezici, B. Yanikoglu, D. Tapucu and Y. Saygın, "New features for sentiment analysis: Do sentences matter," in *SDAD 2012 The 1st International Workshop on Sentiment Discovery from Affective Data*, Bristol, 2012.

[31] A. Hogenboom, P. van Iterson, B. Heerschop, F. Frasincar and U. Kaymak, "Determining Negation Scope and Strength in Sentiment Analysis," in *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, 2011.

[32] J. Wiebe and E. Riloff, "Creating Subjective and Objective Sentence Classifiers from Unannotated Texts," in *Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2005)*, 2005.

[33] T. De Smedt and W. Daelemans, "TextBlob/en-sentiment.xml," 5 7 2014. [Online]. Available: https://github.com/sloria/TextBlob/blob/eb08c120d364e908646731d60b4e4c6c1712ff63/textblob/en/en-sentiment.xml.

[34] A. Severyn and A. Moschitti, "On the Automatic Learning of Sentiment Lexicons," in *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, 2015.

[35] S. Wang and C. D. Manning, "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Jeju, 2012.

[36] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research,* vol. 3, pp. 1157-1182, 2003.

[37] V. Narayanan, I. Arora and A. Bhatia, "Fast and accurate sentiment classification using an enhanced Naive Bayes model," *Intelligent Data Engineering and Automated Learning Lecture Notes in Computer Science,* vol. 8206, pp. 194-201, 2013.

[38] A. L. Berger, S. A. Della Pietra and V. J. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics,* vol. 22, no. 1, pp. 39-71, 1996.

[39] H.-F. Yu, F.-L. Huang and C.-J. Lin, "Dual coordinate descent methods for logistic regression and maximum entropy models," *Machine Learning,* vol. 85, no. 1-2, pp. 41-75, 2011.

[40] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," in *Data mining techniques for the life sciences*, Humana Press, 2010, pp. 223-239.

[41] L. Bottou, "Stochastic Gradient Descent Tricks," in *Neural Networks: Tricks of the Trade*, Berlin, Springer Berlin Heidelberg, 2012, pp. 421-436.

[42] Y. Freund, R. E. Schapire and N. Abe, "A Short Introduction to Boosting," *Journal of Japanese Society for Artificial Intelligence,* vol. 14, no. 5, pp. 771-780, 1999.

[43] R. E. Schapire, "A Brief Introduction to Boosting," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[44] L. Breiman, "Random forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[45] J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)," *The Annals of Statistics,* vol. 28, no. 2, pp. 337-407, 2000.

[46] T. Pedersen, "A decision tree of bigrams is an accurate predictor of word sense," in *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, 2001.

[47] D. Wang and Y. Liu, "A Cross-corpus Study of Unsupervised Subjectivity Identification based on Calibrated EM," in *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, 2011.

[48] G. Qiu, B. Liu, J. Bu and C. Chen, "Expanding Domain Sentiment Lexicon through Double Propagation," *IJCAI,* vol. 9, pp. 1199-1204, 2009.

[49] A. Nenkova and K. McKeown, "A Survey of Text Summarization Techniques," in *Mining Text Data*, New York, Springer, 2012, pp. 43-76.

[50] M. A. Fattah and F. Ren, "Automatic Text Summarization," *World Academy of Science, Engineering and Technology,* vol. 37, p. 2008, 2008.

[51] A. Asmi and T. Ishaya, "Negation Identification and Calculation in Sentiment Analysis," in *IMMM 2012 : The Second International Conference on Advances in Information Mining and Management*, 2012.

[52] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *PRoceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004.

Appendix

Manually Labeled Dataset

The manually labeled dataset may be downloaded at: https://www.dropbox.com/s/aoy5qj2j2vxw71l/reviews_Manual.zip?dl=0

The file/folder structure follows that of the original Subjectivity Dataset, separated into positive reviews (folder "pos") and negative reviews (folder "neg"). File names for each manually labeled review correspond to the full review name from the Cornell Polarity Dataset v2.

Experimental Code

The code may be downloaded at: https://www.dropbox.com/sh/ii9s3ppzhuongqt/AABTCe6rdNa-jvo6nc280Lfwa?dl=0 In various scripts, directory and file names have been partially hard-coded.

lexiprocess.py—this python script classifies movie reviews according to a simple threshold for different aggregate features using TextBlob to retrieve polarities and subjectivities. Note that this is not a machine learning based classifier.

manualstats.py—this python script prints out summary and comparison statistics for the manually labeled and full corpuses.

misclassified.py—this python script prints out tables for classification accuracy. There are two different portions of code within. For investigations on a single feature set with different classifiers, the first block of code prints, for each classifier, accuracy, number of misclassified reviews, false positives, and false negatives in a structured format. For investigations with multiple feature sets, the second block of code prints four tables (one for accuracy, one for misclassified reviews, one for false positives, and the last for false negatives) in which the row corresponds to feature set and the column corresponds to classifier. The second block of code is meant to be run and piped into a CSV file for display purposes.

negstats.py—this python script prints out summary statistics on negation (average number of sentences containing negation, etc.).

opinionfinder.py—this python script will generate a list of documents that can be used to direct OpinionFinder to batch process movie reviews. Additionally, it provides subjectivity statistics (recall, precision, F1) on OpinionFinder.

OpinionFinderLists.zip—lists of documents generated by opinionfinder.py, passed into OpinionFinder. The corpus has been split into 20 parts to allow for batch processing with memory constraints.

processdoc.py—this python script is called to generate rich and base feature sets, and to perform machine learning classification on these feature sets. Parameters for each function are explained within the code body. This script will write the predicted label, true label, and prediction confidence (for ensemble classifiers) for each movie review in tabular format in a CSV file.

subjectivityTest.py—this python script prints out subjectivity statistics (recall, precision, F1) for simple subjectivity analysis techniques (adjective presence, simple adjective frequency, adjective frequency with SVMs, etc.). This does not generate statistics for OpinionFinder or TextBlob.

test_textblobsubstats.py—this python script prints out subjectivity statistics (recall, precision, F1) for TextBlob.