Twitter Sentiment Analysis using Fuzzy Integral Classifier Fusion

Patrick Killeen
pkill013@uottawa.ca
University of Ottawa
Ottawa, Ontario

ABSTRACT

Social media websites are becoming increasingly popular. Among these websites is Twitter, an important micro-blogging service for data science, since it offers an API to make its user data easily available, and has more than 500 million messages (tweets) that are created each day. Data mining these tweets can result in interesting findings. By analyzing these messages for their polarity (their sentiment), many applications can benefit from the knowledge discovered. These applications include the following: business intelligence, stock market analysis, and political election statistics. However, it is difficult to analyze all these tweets due to several reasons: their limited character length, the large number of tweets, the speed at which tweets are created, and the unstructured nature of tweets. It is not feasible to manually verify the opinions by analyzing this massive amount of Twitter data.

The solution to address these challenges is to apply sentiment analysis to this Twitter data. Sentiment analysis is an automated solution for discovering the polarity (negative, neutral, or positive) of textual messages. There are several approaches in the literature that propose a method to address Twitter sentiment analysis, and this includes the following approaches: supervised machine learning, semi-supervised machine learning, lexicon-based, and hybrid; among these approaches is a work that uses a data fusion-based approach (they use a fuzzy integral classifier) and also proposes a natural language processing (NLP)-based approach. Their fusion model combines multiple simple models (including their proposed NLP-based method) to create a more robust complex model.

The present work recreates their data fusion approach, NLP-based method, and some of their experiments, which include using a dataset they used in their experiments (the Stanford dataset) and also datasets they did not use; namely, Airline and Dataset3 datasets. The present work uses the following classifiers: Maximum Entropy, Naive Bayes, Support Vector Machine (SVM), NLP-based approach, and the Fusion model. Experimental results reveal that: a) classifier diversity indeed has an effect on the performance of the Fusion model; that is, the Fusion model will not necessarily perform as well as its best performing classifier; b) the presence of emoticons and their effects on performance is inconclusive; c) the presence of symbols and stopwords can have an effect on performance, and stopwords may have some form of sentimental value; d) when the Airline dataset is used to train the classifiers, the average classifier performance is lower compared to the average classifier performance when using the Dataset3 or Stanford dataset; and e) some of the results from the literature were reproduced, specifically the accuracy of the SVM and NLP-based approaches.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning by classification**; **Unsupervised learning**; Cross-validation; **Support vector machines**; **Statistical relational learning**; **Machine learning approaches**.

KEYWORDS

Choquet integral, classifier fusion, data fusion, fuzzy integral, polarity classification, sentiment analysis, text classification, twitter

ACM Reference Format:
Patrick Killeen. 2020. Twitter Sentiment Analysis using Fuzzy Integral Classifier Fusion. In *Proceedings of Ottawa '20: University of Ottawa COMP5118 Course Paper (University of Ottawa '20).* ACM, New York, NY, USA, 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Social media platforms are becoming increasingly popular, and among them the most popular are Google+, Facebook, and Twitter [9][5]. As a result, given the large number of users that use these platforms, a lot of data are being created in the form of posts and messages, which provides data mining opportunities.

1.1 Twitter

One of the important social media sites is Twitter. Twitter is a micro-blogging social media platform [36][32][6][9][43] that allows users to send small messages to each other. By storing all its users' messages, Twitter can offer an API to data analysts and developers to allow them to easily access data by querying recent messages or messages of a target topic [12]. These messages include reviews and other opinions about a variety of subjects (for example, companies, products, political views, and economics [9]). The messages are limited to 280 [42] (or more recently 140 [6][36]) characters in length. Messages in Twitter are referred to as `'tweets'` and often contain emoticons, which are used to indicate the mood of the tweet. An emoticon is usually created using a series of punctuations (for example, `':)'`). Users can also refer to each other using the `'@` symbol in combination with a user name [12]. The `'#'` symbol is used to explicitly specify the topic of the tweet, which allows many users to see it and aggregate that tweet [20].

University of Ottawa '20, April 26, 2020, Ottawa, ON

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.2 Sentiment Analysis

It is important for some companies to examine the general opinion about their product (or service) [9][11][42][32][36], since they can adjust their business plan and/or goals to improve their clients' experience. For example, a company can adjust their business model to cater to individual clients' needs [25]. Another example is a product recommendation system that aggregates product opinions and recommends the most popular products to customers [25], which can help customers' purchasing experience [41]. The naive approach to examining all these opinions would be manually verifying each user's opinion and having a human summarize the general opinion found in the tweets. However, it is not trivial to manually analyze all these tweets for scalability reasons [9]; more than 500 million tweets are created every day [11][32][36].

Therefore, an automated approach is required, namely, **sentiment analysis** [20][9]. The literature has various synonyms for sentiment analysis, which include opinion mining, emotion classification, subjectivity analysis, appraisal extraction, and polarity classification [25][9]. In this paper the term sentiment analysis (SA) will be used. SA is a form of natural language processing [41][9]. Generally speaking, SA involves an automated solution that predicts the polarity (sentiment) of textual data (either positive, negative, and/or neutral) [32][9]. In addition to business intelligence, applications of SA include discovering facts about stock markets and elections [5]. SA systems can even be applied in real-time (or near real-time), which would not be possible if manual approaches were taken [32]. Therefore, designing software tools to automatically determine the sentiment of a tweet is important.

1.3 Contributions

The present work makes the following contributions: a) it recreates some of the work and experiments done in [9] and [12] (note that the work done in [9] is summarized in section 2.3), using the following TSA datasets (see section 5.4 for more details): 1) Airline [3], 2) Dataset3 [2], and 3) Stanford Twitter Sentiment [12]); b) it provides a literature review on TSA; c) it performs experiments to analyze the effects of emoticons, stopwords, and symbols on TSA performance; d) it performs experiments to analyze classifier prediction diversity and the effects on the Fusion model's performance; e) it provides a thorough analysis of the results from the TSA experiments; and f) it provides detailed examples for using the *Stanford CoreNLP* library from the command line, and a list of helpful online resources that enables TSA using the *Stanford CoreNLP* library.

1.4 Paper Overview

This paper is organized as follows: section 2.1 explains the challenges involved when performing Twitter sentiment analysis, section 2.2 goes into detail about the state of the art of the literature on Twitter sentiment analysis, section 2.3 summarizes the work done in [9], section 3 presents the methodology used to perform Twitter sentiment analysis, which includes the steps involved in performing tweet normalization and details on the involved machine learning classifiers, section 4 presents the Twitter sentiment analysis implementation details, which includes the libraries used and the classifiers' implementation details, section 5 provides the results and analytical findings of the Twitter sentiment analysis experiments conducted in the present work, and section 6 concludes the paper.

This paper also includes two appendices: Appendix A, which provides examples on how to use the *Stanford CoreNLP* library's command line API, and Appendix B, which provides useful online resources that helped the present work with the implementation process.

2 BACKGROUND

2.1 Challenges

There are challenges involved when performing Twitter sentiment analysis (TSA). According to [5], one of the challenges is obtaining labeled data [36]. Since the polarity of a tweet is arguably subjective, humans are usually involved with manually labeling Twitter data when a machine learning model needs to be evaluated. This is known as the **label sparsity problem**. This problem can be addressed by querying SA websites to label tweets [34] and by performing crowd sourcing, which involves paying humans to manually label datasets. However, these solutions do not address all challenges; that is, according to [6] and [36], performing TSA is more difficult than standard SA because of the **variety of domains** in Twitter datasets [41]. In other words, the popularity of Twitter is responsible for the variety of topics discussed in tweets (for example, cooking, video games, politics, marketing, etc.), while other social media platforms (such as a product review website, for example) only focus on a specific topic. Not only does Twitter data have a vast array of domains, this large amount of Twitter data also: a) is increasing in size, b) is imprecise, and c) has many different subjects; additionally, the general vagueness of tweets also makes it difficult to determine the overall sentiment and average insight about subjects found in tweets [28]. This means a trained model may under-perform when used on Twitter data from a different domain (for example, sports-related tweets) compared with its training data (for example, movie-related tweets). This is known as the **sentiment drift problem**. This challenge is further complicated by the presence of multiple cultural sublanguages that are found in tweet datasets; that is, the same word may have different sentimental meanings depending on the cultural context/domain. For example, reviews for a specific type of product (a keyboard, for example) may have different terms and slang used to express the same opinion for sports reviews. Some approaches in the literature attempt to address this by gathering tweets from a target event and manually labelling them. The assumption in this approach is that an event will have tweets that share a common sublanguage (a gaming convention, for example). The limitation of this method is that the classifiers may be poorly trained due to lack of training data, and labeled data is generally difficult to obtain. To address the inaccuracies of this method, external data from the same event can be merged with the Twitter data. For example, distant supervision can be used to address the label sparsity and sentiment drift problems, which involves automatically labeling unlabeled tweets [5]. An approach proposed by [5] attempts to address the label sparsity challenge by building an opinion lexicon and creating synthetically labeled tweet data. According to [34], approaches that use lexicons do not need training data; however, it is difficult to use lexicons

(and other types of approaches [19]) for TSA due to the **unstructured grammar and limited length** of tweets. The tweet length limit is responsible for encouraging slang, domain-dependent slang, acronyms, and other unstructured grammar that appear in tweets [19][6]. As a result there is also a general lack of opinion words (words that represent a negative or positive message for a specific domain [5]) or other textual mood indicators/elements found in tweets, which make it challenging to determine a tweet's polarity [18][6]. This type of challenge is unique to TSA, since standard SA approaches typically deal with larger text corpuses that have a better grammatical structure [41]. Another unique challenge to TSA, according to [6], is that tweets tend to mostly have neutral polarity (that is, the **class ratios are skewed**[18]), while other types of SA datasets (such as those from review websites, for example) tend to mostly contain positive and negative messages. Moreover, the unstructured nature of the grammar found in tweets is challenging for feature selection due to the sparsity of the vocabulary used in tweet messages. When tweets are converted into feature vectors, these vectors may be sparse, which hinders the accuracy of SA models [11][36]. That is, **feature selection** is another challenge for TSA [6][9]. Furthermore, some other challenges include the following:

- The **real-time nature** of Twitter data; to analyze tweets in real-time is not a trivial task [32].
- The **sensitivity of the classifiers** to the data. Some classifiers will learn differently than others when using the same training data. A strategy to deal with this problem is to combine classifiers together, which is called a fusion model.
- Dealing with tweets that have **multiple subjects**, since there could be two different polarities for a single tweet (one polarity for each subject).

2.2 Literature Review

According to [9], the first to perform SA on microblogs was [12]. The uniqueness of performing SA on microblogs is typically that the messages are short, have many grammar mistakes, include irony, and also have a vocabulary that depends on the subject/domain of the message (for example, video gamers may have different vocabularies than sports fans). As a result, standard SA techniques may not be directly applied to microblog messages. [4] state that typically the steps involved when performing SA are as follows:

(1) Feature extraction and isolating the text components that are necessary for SA [7][5], where each attribute of the vector represents a feature of the tweet; for example, the frequency of a word or a combination of words (n-grams)[5]
(2) Making a prediction and classifying text (for example, as either positive, negative, or neutral)
(3) Visualizing the aggregated opinion results
(4) Automatic opinion mining.

A literature review of [11], [36], and [41] reveals that strategies used to perform TSA include: supervised machine learning, semi-supervised machine learning, dictionary-based methods, sentiment lexicons, and hybrid approaches. According to [7], common classifiers for TSA include Support Vector Machines (SVM), Naive Bayes (NB), and Maximum Entropy.

[28] and [25] provide a good literature review and survey on the topic of TSA, respectively. [11] propose TSA work that uses five sentiment classes instead of the standard three (positive, neutral, and negative), stating that having middle-positive and middle-negative classes can help brand management practitioners. They also address the sentiment drift problem by applying a TSA model to a variety of domains. [7] propose an Ensemble model approach that combines SVM with *Adaboost* to address the issue of unbalanced labels in datasets. [41] propose a hybrid model that is based on domain-oriented lexicons to address the challenge of multiple domains in Twitter data. Opinion lexicons, which use opinion words, can also be used to perform TSA [5]. [19] propose a supervised learning approach that creates new features by analyzing the relationship between the frequency of positive and negative tweets. [6] investigate the effects on TSA when using bag-of-word feature extraction, feature hashing, and classifier ensembles. [32] do a case study of applying TSA to monitor the opinions of customers who travel to Las Vegas. According to [36], the lack of labeled data can be addressed using semi-supervised machine learning, and they also provide a survey that analyzes various strategies that can be used to perform TSA when labeled data is limited. [43] study the relationship of emoticons and TSA. They find that common emoticons can indeed be beneficial to the accuracy of TSA models; however, the rarer emoticons should be treated with caution, since they may confuse the models. [8] provide a good survey on TSA that covers the details and techniques used in TSA, which includes gathering the Twitter data, machine learning (ML) models used to perform TSA, and the strategies used to evaluate these models. According to [20], the n-gram modeling technique can be used for feature extraction on Twitter data. To deal with slang words and the unstructured nature of tweets, *Handling Polysemy* can be used. It involves replacing slang words with their non-slang synonyms by using a dictionary and expanding acronyms to their full words. They also propose an approach that is based on improving the SVM by using the K-nearest neighbors (KNN) algorithm. [4] propose an Internet of Things (IoT)-based TSA system using a Raspberry Pi, a smart mirror, and a NB classifier. [21] propose an approach using a model based on SVM and KNN, and they only consider including a limited number of features such as: n-gram feature, pattern feature, punctuation feature, keyword-based feature, and word feature. [11] addresses the feature selection challenge by using an approach that uses neural networks and SVM. [42] propose an approach based on concept bagging to perform TSA that focuses on reducing the number of features included in their model. Too many features will lead to inaccurate results. [18] propose an ensemble framework for TSA. [34] propose a TSA system called *SentiCircles*, which is lexicon-based and considers both the entity-level (words) and tweet-level polarity (typically TSA approaches are tweet-level). They analyze the co-occurrences of words while processing polarity, and they update the entity-level polarity of words dynamically.

2.3 Twitter Sentiment Analysis Using Fuzzy Integral Classifier Fusion

This section summarizes the work done in [9]. They perform TSA using the Choquet fuzzy integral (CFI), and their approach contains

both unsupervised and supervised machine learning. They propose a natural language processing (NLP)-based approach and use data fusion to perform TSA. Data fusion involves combining many simple models to create a more robust complex model. Examples of fusion models include using NB and ordered weighted average (OWA) on the classifiers, which can be improved if the weights assigned to each classifier are dynamically determined based on the input sample. That is, for every tweet, a fusion model would adjust the weights to address the efficiency of each classifier for the given tweet. It is at times beneficial for fusion models to have classifiers contradict each other. For example, if a class is more popular than another, when two classifiers predict different classes, then it may be the case that the more frequent class is the correct answer. The CFI is a strong fusion model based on classifier weighting. It considers all possible combinations of classifiers' predictions and assigns each combination with a weight. It has the benefit of requiring little training data compared with other fusion models. They train their fusion model using heuristic least mean square (HLMS) method. Their fusion model includes the following classifiers: Maximum Entropy (MaxEnt), NB, SVM, and their proposed NLP-based model. They normalize tweets by following some of the steps performed in [12]. These steps involve:

(1) Replacing addresses (for example, a URL) with a constant
(2) Removing user referencing
(3) Fixing grammar mistakes (for example, any character that occurs more than twice is replaced with two of that same character)
(4) Replacing certain types of words with constants (emoticons, for example)
(5) Removing stop words

They extract features using bigrams, unigrams, a combination of the two, and part of speech (POS) tags. Their NLP-based model uses lexicons for TSA.

In their experiments, they use the following datasets: Stanford Twitter Sentiment, SemEval-2016 Subtask D Two-Point Scale (Positive/Negative), Movie Dataset, and Stanford Twitter Sentiment Corpus. They perform TSA on the SVM, NB, Maximum Entropy, and their proposed NLP-based model. They also include these models in the following fusion models: NB, OWA, and their proposed fusion approach that uses the CFI. Their experimental results reveal that their NLP-based approach does poorly compared with the other machine learning models; however, when combined with their fusion approach, their results are more accurate than those of SVM, NB, and Maximum Entropy. Furthermore, their fuzzy data fusion approach outperforms both NB and OWA fusion models.

3 METHODOLOGY

The present work follows the tweet normalization performed in [12] and [9]. This section includes the tweet normalization process, a detailed discussion that summarizes the work done in [9], and presents the involved classifiers.

3.1 Tweet Normalization

Tweet normalization involves converting raw tweet data from a dataset into an appropriately parsed format that can be used by TSA classifiers. The steps involved in normalizing tweets are as follows: a) replacing URIs with the constant 'URL'; b) replacing any user reference (for example, '@user123') with '(USERNAME)'; c) applying grammar corrections. This simply reduces many consecutive characters; for example, 'goooooood' would become 'good'. However, it is not sophisticated enough to detect spelling mistakes after normalization; for example, the word 'hoooot' would be converted to 'hoot', but the user may have meant 'hot'; d) replacing emoticons with the appropriate constants. Positive emoticons (':)', for example) are are replaced with 'smile' and negative emoticons are replaced with 'frown'; e) stopwords (for example, 'the' is a stopword) are removed (see section 3.1.1 for more details on stopwords); and f) although [9] did not perform this normalization step, the present work also optionally removes symbols. Removing symbols involves removing any non-alphanumeric character from a tweet, with the exception of '(' and ')', since the username template contains parentheses.

3.1.1 Stopword Removal. Stopwords are common words that occur frequently in data and therefore provide less analytical value to classifiers than words that are more rare; for example, the word 'The' may be considered a stopword. According to [33], some research suggests that removing stopwords will hinder sentiment analysis performance, since they argue that stopwords have sentimental value, while other research suggests removing stopwords have beneficial effects on the performance of sentimental analysis. They also find that stopword removal indeed has a negative impact on sentiment analysis results. Therefore, the present work investigated the impact of stopwords on the performance of sentiment analysis experiments.

3.2 Classifiers

3.2.1 Maximum Entropy. According to [12], when using the Maximum Entropy (MaxEnt) classifier, the most uniform models should be preferred. The MaxEnt classifier also uses features like NB, but the difference between the MaxEnt and NB classifier is that MaxEnt does not assume conditional independence between the features in a data instance. This relaxes the constraint on the types of features that can be added (bigrams can be added, for example). The equation below represents the model:

$$P_{ME}(c|d,\lambda) = \frac{\exp[\Sigma_i \lambda_i f_i(c,d)]}{\Sigma_{c'} \exp[\Sigma_i \lambda_i f_i(c,d)]} \tag{1}$$

, where c represents the class (sentiment) of the tweet d, and λ is a weight vector.

3.2.2 Naive Bayes. A literature review of [12] and [24] reveals that Naive Bayes (NB) is a probability-based classifier that relies on the frequency of the features from a dataset, and it assumes conditional independence between the features in a data instance. The sentiment (class) $c*$ of a tweet d can be predicted as follows:

$$\begin{aligned} c* &= \texttt{argmax}_c(P_{NB}(c|d)) \\ P_{NB}(c|d) &= \frac{(P(c) \sum_{i=1}^{m} P(f|c)^{n_i(d)})}{P(d)} \end{aligned} \tag{2}$$

, where f is a feature, $n_i(d)$ represents the number of times feature f_i is contained in tweet d, and m is the number of features.

3.2.3 Support Vector Machine. The Support Vector Machine (SVM) classifier is non-probabilistic and is based on vectors by trying to find a vector that best splits the training data into two different categories to separate the classes from each other [44]. According to [12], it is a popular classifier. [12] use the presence of features (1 if the feature appears, and 0 otherwise) instead of counting the frequency of the features.

3.2.4 Natural Language Processing-based Approach. The NLP-based approach used in [9] is based on the NLP-based approach from the *Stanford CoreNLP* library. A command line example of using this library can be found in section A.2 in the Appendix, and for interested readers section B.1 provides many useful online resources to help users make use of this library.

The sentiment analysis in this approach uses the bag-of-words technique; that is, when the approach encounters a positive or negative word, it counts the word's sentiment score. At the end, it sums all the scores to avoid having the order of the words affect the sentimental results. The underlying technology that drives the sentiment analysis of the NLP-based approach is a Recursive Neural Network that takes advantage of the grammatical structures of sentences. For those that are interested, the full paper that proposed this neural network is found here [37].

Furthermore, before feeding the tweet data into the model for sentiment analysis, tweet normalization and featurization are performed.

3.2.5 Fusion Model. The Fusion model approach, which was proposed by [9], combines the predictions of other classifiers to make its predictions. The Fusion model is based on fuzzy logic involving fuzzy measures and is implemented using the Choquet Fuzzy integral (CFI) and the Heuristic Least Mean Square (HLMS) algorithm.

Choquet Fuzzy Integral

Fuzzy measures attempt to model problems that address decision making based on multiple criteria. When multiple attributes/criteria (the predictions of multiple classifiers, for example) are required to make a decision, it is important to consider the decision-making strength of combining these attributes together. For example, multiple people in a room (the attributes in this example) are trying to come to a final decision on a subject. Each person then makes a decision. If in the past person A and person B were always right when they made the same decision, then this knowledge can be used to deduce that if person A and B make the same decision in this scenario, their decision is probably correct.

In the case of the CFI, the idea is to weight the combination prediction power of all the classifiers used in the Fusion model. For example, if we have a set C of three classifiers: $C = \{A, B, C, \}$ we would need to attribute a fuzzy prediction measure to all possible combinations of these classifiers (that is, the power set of C, which will require 8 (2^3) fuzzy measures). For example, if classifier A and B always make the correct choice together, then the fuzzy measure for {A,B} = 1.0, while if classifier B and C rarely make the correct choice when they make the same decision, let the fuzzy measure for {B,C} = 0.25. The set of measures is defined as μ, where $\mu(Z)$ is the weight importance (the fuzzy measure) of the attributes of the set Z. From our previous example, suppose $Z = \{A, B\}$ and $Y = \{B, C\}$, then $\mu(Z) = 1.0$ and $\mu(Y) = 0.25$. With an appropriate μ, the CFI can be used to make predictions given fusion testing data.

In the scope of the present work, fusion testing data instances take the following form: $\{p_1, p_2, ..., p_n\}$, where p_i is the ith classifier's prediction and n is the number of classifiers. The CFI will output a real number prediction, which means the prediction must be converted to the appropriate class. For example, if we have the following class values: negative = 0, neutral = 2, and positive = 4, suppose the CFI is given the fusion testing data instance {2,0,2} and predicts 1.2. The class predicted is therefore neutral.

Heuristic Least Mean Squares

To make predictions using the CFI, we need to find an appropriate μ using the HLMS algorithm, which was proposed by [13]. For those that are interested, [14] go into detail about fuzzy integrals. Finding μ effectively involves training the Fusion model using fusion training instances. A fusion training instance is similar to a test instance, but it also includes the real tag (actual class) of the fusion instance. For example, fusion training data instances take the following form: $\{p_1, p_2, ..., p_n, L\}$, where p_i is the ith classifier's prediction, n is the number of classifiers, and L is the real-tag/label/class of the data instance. To acquire these fusion training instances, the classifiers included in the Fusion model must first be trained on the training data. Once trained, to evaluate the classifiers, the development data is used as test data. Lastly, the development data (which is a subset of the original training data) predictions of the classifiers are converted into fusion training instances.

4 IMPLEMENTATION

4.1 Libraries

The present work used the *Stanford CoreNLP* library's packages (found in [17]), which is implemented in Java, to implement the MaxEnt, NB, and SVM models. The class used to implement these models is the `edu.stanford.nlp.classify.ColumnDataClassifier` class, and to implement the NLP-based approach, the present work used the `edu.stanford.nlp.pipeline.StanfordCoreNLP` class. Since they used the *Kappalab* package (a R programming library) to implement the CFI and HLMS algorithms, the present work also used this library and the R programming language to implement these algorithms.

4.2 Classifiers

The *Stanford CoreNLP* library was used to implement the MaxEnt, NB, and SVM classifiers. Examples on how the library was used via the command line to run the MaxEnt, NB, SVM, and NLP-based approaches can found in the Appendix in sections A.3, A.4, A.5, and A.2, respectively.

4.3 Natural Language Processing-based Approach

The *Stanford CoreNLP* library parses the sentences of a dataset and predicts the sentiment of each sentence. It uses the part-of-speech (POS) tagger for featurization. A tweet with multiple sentences will therefore have many sentiment classifications attributed to it by the NLP-based approach. Sentiments are represented in the form of a value and a string. As such, since [9] did not specify how they dealt with the above situation, the present work simply took the average

Table 1: *Stanford CoreNLP* sentiment analysis sentiment value map.

CoreNLP Sentiment Value	CoreNLP Sentiment	Mapped
0	very negative	negative
1	negative	negative
2	neutral	neutral
3	positive	positive
4	very positive	positive

sentiment value of all the sentences to decide a tweet's sentiment. Note that the labels used in the present work are negative (0), neutral (2), and positive (4), while the sentiments produced by the *Stanford CoreNLP* library are very negative (0), negative (1), neutral (2), positive (3), and very positive (4). Since the present work is not using five labels, the very negative and very positive labels were simply mapped to negative and positive, respectively (see table 1 for more details). Once the the average sentiment value of all sentences in a tweet was computed, the present work computed a tweet's polarity by rounding the average sentiment to the nearest sentiment value. For example, according to table 1, if the average sentence sentiment value is 0.9, then the tweet would be considered negative, since rounding 0.9 to the nearest integer yields 1. If the average score was 3.6, the nearest integer would be 4, and since the present work's experiments only consider three labels (negative = 0, neutral = 2, and positive = 4), the sentiment value would be converted from 4 to 3 (which is positive).

4.3.1 Neutral Tweets Complication. There is an issue that must be addressed with the implementation method discussed in section 4.3. Since the library predicts neutral sentiments, the present work had to address the situation where a dataset used did not include any neutral tweets (for example, the Dataset3 and Stanford dataset (which are discussed in section 5.4.2 and 5.4.3, respectively) do not contain any neutral tweets); that is, the NLP-based approach should not predict neutral sentiments if the training data has only positive and negative tweets. Therefore, the approach taken was to ignore the sentiment value 2 (the neutral sentiment's value) when rounding to the nearest integer. For example, an average sentiment value of 1.9 would be rounded to 1 (negative), and a value of 2.4 would be rounded to 3 (positive). On the rare chance the average sentiment value was exactly 2 (of equal distance to 1 as to 3), the design approach taken was to select the sentiment randomly from negative and positive. For example, a tweet with an average sentiment value of exactly 2 would be considered positive or negative with equal likelihood (which was achieved using random number generation).

4.3.2 Non-ASCII Characters. The *Stanford CoreNLP* library ignores non-ASCII characters. That is, a word that contains a non-ASCII character (such as 'é', for example) is split into two. As such, a bias exists where a word that is split into two could create a sentimental shift in the sentence.

4.4 Fusion Model

The Fusion model (proposed by [9]) was implemented using the *Kappalab* R programming library, since this library offers the CFI and HLMS algorithms and was used in [9].

For implementation simplicity, the Q statistic used to filter out non-diverse classifiers from the fusion model was not implemented. For readers that are interested in more information on the Q statistic used for measuring classifier diversity, [29] provide many details on this topic.

5 EXPERIMENTS

This section details the experiments run in the present work, which are as follows: a) varying the k in k-fold cross validation (KFCV) when creating development data and analyzing the effects on performance; b) stopword removal and symbol removal, which investigate the effects that stopwords and symbols have on the performance results; c) emoticon set varying, which investigates the effects of a varying of set of emoticons used to replace emoticons with their respective template words; d) comparing the performance of various classifiers, which include the MaxEnt, NB, SVM, NLP-based, and Fusion classifiers; e) investigating the effects of using various datasets as training data and test data on performance results; and f) varying the test data between 1) using the Stanford dataset's test data to evaluate the classifiers' results and 2) using k-fold cross validation to create the test data.

5.1 Experimental Setup

The Java programming language was used to implement most of the experiments, and the R programming language was used to implement the Fusion classifier. The following software versions were used: Java version 1.8.0_171, *Stanford CoreNLP* Java library version 3.9.2, R programming language version 3.6.0, and *Kappalab* R library version 0.4-7.

5.2 K-fold Cross Validation

Some datasets did not have any test data, and in particular, since the Fusion model requires development data, KFCV was used to address this; that is, [9] defined their development data as 20% of their training data, so the present work used KFCV to select a similar development data partition (k=5 achieved a 20% subset of the training data).

According to [24], KFCV is a technique used to evaluate a classifier when only training data is available. The idea is that the training dataset is partitioned into k evenly-sized mutually exclusive partitions (folds). The evaluation process proceeds by iterating through each fold, where the fold i represents the testing data for iteration i. The training dataset at iteration i is created by subtracting the ith fold from the original training dataset. The evaluation process for the classifier at iteration i proceeds by training the classifier using the training data at fold i and evaluating the classifier using the testing data at iteration i. The total number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) is counted (more details on the evaluation techniques is discussed in section 5.3). At the end of the iterations the accuracy is computed using the total of these measures, which averages out the accuracy of each fold (the accuracy is defined in equation 3). Note that an

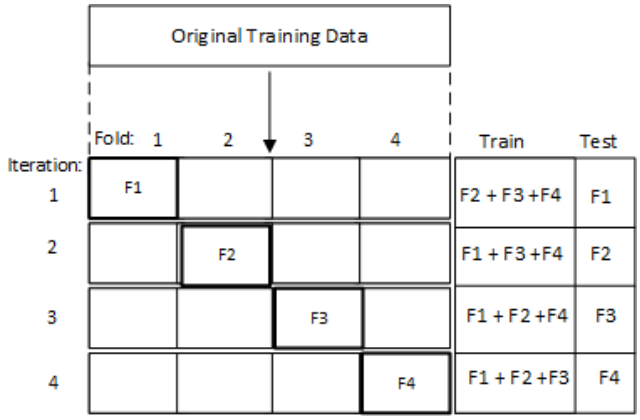

Figure 1: Example of 4-fold cross validation to create test data.

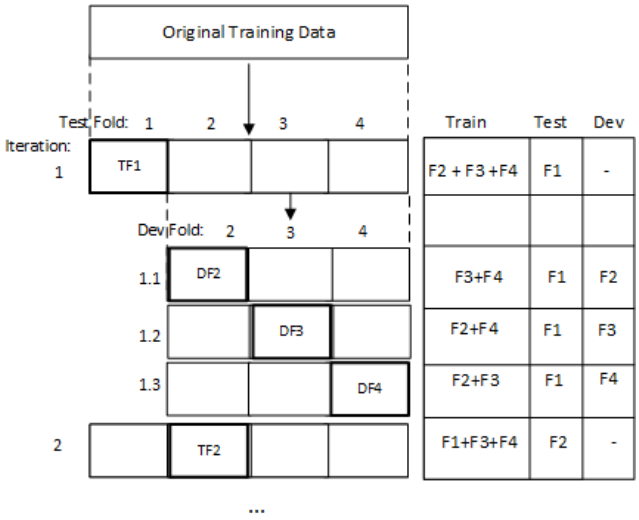

Figure 2: Example of 4-fold cross validation to create test data and 3-fold cross validation to create development data.

example of using 4-fold cross validation to create test data can be seen in figure 1.

5.2.1 Creating Testing and Development Data. This section includes a more detailed approach on using KFCV to create test and development data. Note that in the experiments, when development data was required, KFCV was always used to create the development data; on the other hand, depending on the experiment, the test data was either created using KFCV or was the Stanford dataset's test data.

The methodology used to create the development data (and if applicable, the test data) was done as follows: Let A denote the training dataset. If the test data is being created using KFCV, A is partitioned accordingly into k folds. Let a_i denote the ith fold (where $a_i \in A$ and $|a_i| = \frac{|A|}{k}$, and the last fold a_k's size may be slightly larger to accommodate overflow remainder from A divided by k) and let $\overline{A_i}$ denote the remaining training data after removing the testing data from it (that is, $\overline{A_i} = A - a_i$). Once the test data is created, a similar process is applied to create the development data. Instead of applying this method to A, for k' development data folds, the method is instead applied to A_i; that is, A_i is partitioned accordingly into k' folds. Let b_j denote the ith fold (where $b_j \in A_i$ and $|b_j| = \frac{|A_i|}{k'}$) and let $\overline{B_i}$ denote the remaining training data after removing the testing and development data (that is, $\overline{B_i} = A - a_i - b_i = A_i - b_i$). In the case that the test data (denoted as T) is provided, the KFCV is not required to create the test data, and therefore, the KFCV method applied to create the development data is applied using the training set A instead of A_i. An example of this methodology is illustrated in figure 2.

5.2.2 Dataset Sampling. In the experiments, to ensure the datasets using KFCV were sampled properly without bias, the lines of the target dataset were randomly sorted. The datasets were only randomly sorted once to guarantee deterministic results; that is, when changing the experimental parameters, the dataset and KFCV folds would remain static throughout the experiments, and therefore, any change to the performance results could be attributed only to parameter changes. Otherwise, if the datasets were sorted randomly for each experiment, it would be difficult to attribute a result change to a parameter change, since maybe the dataset's permutation was responsible for the change in results. An extreme example of this bias is the case where the dataset is originally sorted by label/sentiment. In this example the folds would contain only tweets of a certain sentiment (all positive, for example), which would produce useless/insignificant performance results.

5.3 Experimental Evaluation

To evaluate the models, confusion matrices were built in order to compute the evaluation measures, which included TP count, TN count, FP count, FN count, recall, precision, and f1-score for each class/label. Additionally, for the overall measures of a model's performance, I used the accuracy measure, which is defined in equation 3 according to [27]:

$$\texttt{accuracy} = \frac{\text{TP+TN}}{\text{FP+FN+TN+TP}} \tag{3}$$

, where TP + TN is the total number of data instances properly labeled, and FP + FN + TN + TP is the total number of data instances.

Note that a web tutorial on computing such measures can be found here [35] and a more formal definition of some of these measures (and confusion matrices) can be found here [24].

5.3.1 Combining the Accuracy When Using Development and Test Data. Usually when running an experiment, the accuracy metric was used to summarize the general performance of a classifier. The methodology used to run the Fusion model approach is an interesting case, since there is training data, development data, and testing data. Furthermore, since the Fusion model requires fusion training and fusion testing data to evaluate, this means that the predictions of the classifiers (trained using the training data) require using the development data and testing data as a means to evaluate the classifiers to create the fusion training data. This

Table 2: Example confusion matrix of 25 prediction results of a 3-class (sentiment) classifier. The bold text results are correct predictions.

Sentiment Actual / Prediction	Positive	Neutral	Negative
Positive	**4**	6	3
Neutral	1	**2**	0
Negative	1	2	**6**

means the Fusion model has a single accuracy result and there can be two accuracy results for each other classifier (one accuracy when evaluating a classifier using development data and a second accuracy when using the testing data to evaluate). Therefore, care must be taken to compute a single accuracy measure from both result sets, since it involves adding the results when using the development data and testing data into a single confusion matrix to compute the single accuracy result, for each classifier.

Unfortunately, due to an implementation oversight, the present work did not consider this. As a result, in the following sections, when results that involve the Fusion model are presented, there are duplicate result sets; that is, there are usually accuracy results for when the development data and test data were used. Although it is not ideal nor incorrect to present both accuracy results, it takes up more space in the document and makes the present work less complete since the single accuracy measure for a classifier is missing.

It is worth mentioning that although the Fusion model is trained and tested/evaluated using the fusion training data and fusion testing data (instead of technically being trained using the training data and tested/evaluated using the development and testing data), for presentation purposes, I included the Fusion model's single accuracy in figures that are described to contain the accuracy of classifiers trained using the training data and evaluated using the development or testing data.

5.3.2 Confusion Matrix Example. An example of confusion matrix, which was inspired by [35], is illustrated in table 2. Note that when counting the TP, TN, FP, and FN rates, they must be counted for each class. For example, for the positive predictions, the TP rate is 4, the TN rate is 8 (2+6), the FP rate is 9 (6+3), and the FN rate is 2 (1+1). To compute the accuracy, the total number of TP, TN, FP, and FN rates for each class are used when applying equation 3.

5.4 Datasets

This section discusses the details of the dataset used in the present work. The following TSA datasets were used: a) Airline [10][3], b) Dataset3 [2], and c) Stanford Twitter Sentiment [12] (obtained from [1]). The Stanford Twitter Sentiment dataset was used in [9], while the datasets a) and b) were not used in their experiments. A summary of each of the above dataset's specifications can be found in table 3. Note that the reason table 3 includes the number of tweets with non-ASCII characters is that the NLP-based approach ignores non-ASCII tokens when processing the words in tweets. Furthermore, some non-ASCII characters made the *Stanford CoreNLP* library hang (the API calls would not return), so they were removed. This information could help the analysis of the experimental results and is presented for completeness. This section also includes details on the stopword list used as well.

5.4.1 Airline. This Twitter dataset contains various opinions about six different US airlines. Note that there were a few tweets that spanned multiple lines. For simplicity reasons, these tweets were removed from the dataset. This dataset only contains labeled training data (positive, neutral, and negative tweets). This dataset was not used in [9].

5.4.2 Dataset3. This dataset did not have much information about when it was gathered or how the tweets were labeled. This dataset only contains labeled training data (positive and negative tweets). This dataset was not used in [9]. A quick glance at the dataset reveals the tweets do not seem to share a common topic and are therefore from a variety of domains.

5.4.3 Stanford. The Stanford dataset was originally gathered and labeled by [12]. It includes labeled training data and testing data, where the training data contains positive and negative tweets, and the testing data contains positive, neutral, and negative tweets. The training data was labeled using a semi-supervised machine learning approach. [12] queried the Twitter API using a set of emoticons as queries. The tweets returned from positive emoticon queries were labeled as positive, and the tweets that were returned from negative emoticon queries were labeled as negative. The assumption was that tweets resulting from a positive emoticon query will be positive and the tweets resulting from negative emoticon queries will be negative. They removed any emoticon (those used to query the API) from the resulting training dataset. To create the testing data, they queried the Twitter API using a set of key words from a variety of domains. The resulting tweets were manually labeled.

Note that for experiments in the present work that use the Stanford dataset's test data, the neutral tweets were removed if the training data contained no neutral tweets. Otherwise, when the training data contained neutral tweets, the testing data was unchanged.

What is interesting is that [9] follows the same sentiment analysis approach as [12]; however, when [9] refer to the Stanford dataset (created by [12]), [9] state that the training data included neutral tweets, which is a contradiction to the approach described by [12]. Because of this contradiction, it was difficult to understand how [9] labeled neutral tweets in the [12] dataset, so the present work does not perform any experiments with neutral tweets in the Stanford dataset training data, since the Stanford dataset is publicly available here [1] and the training data does not have any neutral tweets.

5.4.4 Stopwords. The present work used a different set of stopwords than those used in [9], since they did not explicitly mention which set of stopwords they used. The stopword list used in the present work was originally retrieved from [22], since this set of stopwords was used in the past in a TSA project for an undergraduate information retrieval course, and can now be found here [26] in an open source code repository.

The same stopword list was used in all the experiments. The stopword list file size is 6 KB and there are 779 stopwords. A few sample stopwords from the dataset follow: 'the', 'maybe', 'have', 'until', and 'who'.

Table 3: Dataset specifications

Airline	**Training Dataset** [10][3]:	
	Date obtained	February 2015
	Dataset size	3342 KB
	# of tweets	14,873
	# of positive tweets	2,368
	# of negative tweets	9,178
	# of neutral tweets	3,099
	# of tweets with non-ASCII characters	1,911
	# of tweets spanning multiple lines	233
Dataset3	**Training Dataset** [2]:	
	Date obtained	Not clear
	Dataset size	8461 KB
	# of tweets	100,000
	# of positive tweets	56,462
	# of negative tweets	43,538
	# of neutral tweets	0
	# of tweets with non-ASCII characters	2,317
Stanford	**Training Dataset** [12][1]:	
	Date obtained	From April 6th 2009 to June 25th 2009
	Dataset size	233MB
	# of tweets	1,571,678
	# of positive tweets	789,605
	# of negative tweets	782,073
	# of neutral tweets	0
	# of tweets with non-ASCII characters	29,962
Stanford	**Testing Dataset** [12][1]:	
	Date obtained	From May 11th 2009 to June 14th 2009
	Dataset size	73 KB
	# of tweets	498
	# of positive tweets	182
	# of negative tweets	139
	# of neutral tweets	177
	# of tweets with non-ASCII characters	0

5.5 Classifier Configuration

This section explains the chosen parameter values used to configure the classifiers in the experiments and why these values were chosen. These values may be a source of error in the performance results, since the present work did not investigate the effects of varying these classifier parameters. Most of the classifiers' parameters were set to default values. That being said, future work could involve investigating the effects of varying theses parameters on the accuracy of the results.

5.5.1 Chosen Features. Features (numeric values) are used by classifiers to train their mathematical models and are used to make predictions, given test data. Since Twitter data is textual, it must be converted to numeric values (featurized) to be able to be represented in a classifier's model. As such, standard featurization techniques offered by the *Stanford CoreNLP* library were chosen. For example, the bigram and unigram features were used to parse tweets word-by-word. This was accomplished using the following parameter configuration: 'useNGrams=true', 'maxNGramLeng=2', and 'minNGramLeng=1'. These features were chosen because: a) these features were used in [12], and b) a simple experiment revealed that combining bigrams and unigrams produced the best results for the NB classifier; that is, the simple experiment involved running the NB classifier on one of the *Stanford CoreNLP* library's example datasets, and by observing the accuracy of each feature configuration, the unigram and bigram features yielded the best results. Furthermore, the other included featurization technique was used to create unigrams and bigrams from the prefix and suffix of the parsed tokens (words) found in tweets. This was accomplished using the following parameter configuration: 'useSplitPrefixSuffixNGrams=true'.

5.5.2 Support Vector Machine. To configure the SVM classifier, all the default parameters provided by the *Stanford CoreNLP* library were used. Note that according to [16], the SVM functionality of the library is implemented using *SVMLight* (see [23] for more information).

5.5.3 Maximum Entropy and Naive Bayes. Both these classifiers used the default parameters (if any) offered by the *Stanford CoreNLP* library.

5.5.4 Natural Language Processing-based Approach. Similar to [9], the following annotator parameters were used for implementing the NLP-based approach using the *Stanford CoreNLP* library: 'tokenize', 'ssplit', 'pos', 'lemma', 'parse', and 'sentiment'. Note that the 'dcoref' parameter was omitted, since this parameter was not recognized by the library, and the 'ner' parameter should have been added as well, since [9] used this parameter, but unfortunately it was only noticed at the end of the present work that this parameter was missing, and as such, due to time constraints, this parameter could not be added.

5.5.5 Fusion Model. In [9], they implemented an algorithm to find the optimal parameters for the HLMS algorithm. The default configuration was used for the HLMS algorithm in the present work, where it was assumed that all classifier combinations had equally likely prediction power (their fuzzy measure were all identical). However, this is probably a source of bias, since some algorithms perform better than others (they are not identical), which means they most likely have different combination prediction power. However, this bias proved to not be significant, since the Fusion model generally produced significantly accurate results (greater than 99% accuracy).

The following parameter configuration was used for the HLMS algorithm: 'alpha = 0.05', 'epsilon = 1e-6', and 'maxiter = 500'. Note that the 'epsilon' and 'maxiter' parameters are set to their default values. [31] provide a good example on using the *Kappalab* R programming library and use the above parameter configuration to run the HLMS algorithm, which is the reason this configuration was chosen in the present work.

Table 4: Accuracy of each method found in various works when using the Stanford dataset. U: Unigram, B: Bigram, and POS: part of speech tags features used.

Work	MaxEnt	NB	SVM	NLP	Fusion + NLP	Fusion - NLP
[9]	79.9%	74.3%	80.5%	55.1%	81.5%	79.9%
[12]:						
U	80.5%	81.3%	82.2%	-	-	-
B	79.1%	81.6%	78.8%	-	-	-
U+B	83.0%	82.7%	81.6%	-	-	-
U +POS	79.9%	79.9%	81.9%	-	-	-

In [9], their parameters were configured as follows, ''alpha = 0.25', 'epsilon = 1e−10', and 'maxiter = 500', which is slightly different than the configuration used in the present work. This could be a source of error in the results, since the present work's Fusion model is implemented in a slightly different way than in [9]. Unfortunately, this was only realized at the end of the present work, and therefore, due to time constraints, these parameters were not adjusted.

5.6 Experimental Reproduction

Some of the experiments in the present work attempted to reproduce the results detailed in [9] and [12]. The only dataset that the present work and these other two papers have in common is the Stanford dataset. Therefore, one of the goals of the present work was to reproduce some of the classifiers' accuracy results (when trained and tested using the Stanford dataset) found in table 4, which lists a summary of the results found in [9] and [12].

5.6.1 Development Data K-Fold Cross Validation Experiment. In this experiment set, the goal was to analyze the effects on performance results when varying the k used to perform KFCV to create the development data.The test data was created using KFCV with $k' = 10$, and the development data was created similarly but with k varied from 2 to 50.

Since the test data folds were created before the development data folds (see section 5.2 for more details) and the k' remains constant throughout the experiment set, the test dataset size always remains as approximately 10% of the size of the Airline dataset, while the size of the training data and development data varied as k changes for the development data KFCV. As k was increased, the amount of training data increased. The results are displayed in tables 5 and 6, and by analyzing the results some findings were made and are detailed later in this section.

Experimental Configuration

The Airline dataset was used. The classifiers involved were as follows: Fusion, MaxEnt, NB, SVM, and NLP-based. The emoticon set used was the small set, which is shown in table 10. Stopwords were removed. Symbols were not removed.

Findings

- **Finding a)**: It can be seen that the SVM and NB are the classifiers that are most affected by the amount of training data.
- **Finding b)**: We can observe that as k increases the accuracy of NB increases. It can also be seen that the increase rate of the accuracy of NB is the most significant for small k, but as k becomes large, the increase rate of NB's accuracy decreases (figure 3 illustrates this).
- **Finding c)**: The NB classifier's best accuracy (90%) is for the largest k (k=50). This trend is not found in the other classifiers' results. In fact, the Fusion model's accuracy has the opposite behavior: the accuracy decreases (very slightly) as k increases. Although the decrease rate of the Fusion model's accuracy is almost negligible, it, nevertheless, is still present.
  - This is most likely the case because with larger k, the development data fold size is smaller, which means the Fusion model has less fusion training data, while NB has more training data.
- **Finding d)**: The SVM model appears to be slightly more sensitive to the amount of training data it was trained with compared with the other classifiers; that is, its accuracy fluctuates more than the others, without demonstrating any clear trend. In the first three columns (for k = 2, 3, and 4) of table 5, we can see the SVM's accuracy fluctuates between ~38% to ~41%.

Table 5: Accuracy (in percentage) of the classifiers using development data as the test data when varying the number of development data folds, k.

Model/K	2	3	4	5	7	15	25	50
Fusion	100	99.9	99.9	99.3	99.8	99.7	99.5	99.1
MaxEnt	100	100	100	100	100	100	100	100
NB	83.4	86.8	87.8	88.4	89.0	89.6	89.9	90.0
NLP	41.3	41.3	41.3	41.3	41.4	41.3	41.3	41.3
SVM	38.2	41.3	38.6	41.3	41.9	41.9	41.0	41.3

Table 6: Accuracy (in percentage) of the classifiers simply using the test data as the test data when varying the number of development data folds, k. Training data: Airline, Testing data: 10-fold cross validation, Development data: (2 to 50)-fold cross validation.

Model/K	2	3	4	5	7	15	25	50
Fusion	100	99.9	99.9	99.3	99.8	99.7	99.5	99.1
MaxEnt	100	100	100	100	100	100	100	100
NB	83.4	86.8	88.0	88.5	89.0	89.6	89.9	90.0
NLP	41.3	41.3	41.3	41.3	41.3	41.3	41.3	41.3
SVM	43.1	43.9	44.8	43.9	44.6	43.2	43.5	43.2

5.7 Stopword and Symbol Removal Experiment

The goal of this experiment set was to investigate the effects that symbols and stopwords have on the accuracy of the classifiers. The effects of symbols were analyzed because [12] state that emoticons have a negative impact on the performance of the MaxEnt and SVM models, and emoticons are generally a set of symbols. The reason the effects of stopwords were analyzed is because [33] state that some research suggests stopwords contain sentimental information,

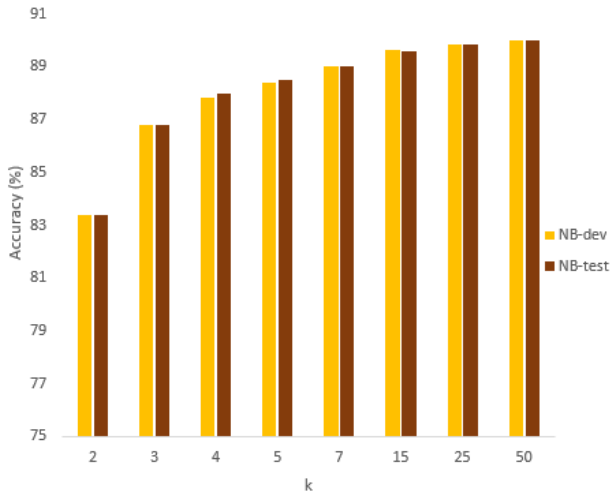

Figure 3: Accuracy of the NB classifier when varying the number of development data folds, k. Training data: Airline, Testing data: 10-fold cross validation, Development data: (2 to 50)-fold cross validation, NB-dev: NB tested on development data, and NB-test: NB tested on testing data.

thus, removing stopwords may negatively impact classifier performance results, while other research suggests removing stopwords has a positive impact on performance. Note that similar experiments were conducted on the Stanford and Dataset3 datasets in section 5.7.1.

In this experiment set the stopword removal and symbol removal were varied; that is, four types of experiments were conducted: a) stopwords were removed and symbols were removed (RSW & RSYM), b) stopwords were removed and symbols were kept (RSW & KSYM), c) stopwords were kept and symbols were removed (KSW & RSYM), and d) stopwords were kept and symbols were kept (KSW & KSYM).

The experimental results are illustrated in figures 4 and 5. To gain insight into the performance effects of symbols and stopwords, both figure's results must be considered since they present the accuracy when using KFCF (using $k = 2$) to create both the testing data and development data.

Limitations

Due to time constraints, a source of error in this experiment set exists, since 5-fold cross validation (5FCV) was not used for the development data (the size of the development data was not 20% of the size of the Airline dataset) nor was 10FCV used to create the test data. For time constraint reasons, 2FCV was used to create both the development data and test data, since a greater number of folds would have taken considerably more execution time.

Experimental Configuration

The Airline dataset was used. The classifiers involved were as follows: Fusion, MaxEnt, NB, SVM, and NLP-based. The emoticon set used was the small set, which is shown in table 10. Stopword removal and symbol removal were varied.

Findings

- **Finding a)**: It is difficult to conclude anything significant regarding the SVM, since: a) for the KSW & KSYM experiment, the SVM's accuracy (50.1%) is quite different (~8%) when using the 2FCV testing data compared with the accuracy (41.8%) when using the 2FCV development data to evaluate the classifier, and b) from the results detailed in section 5.6.1, it is not clear how significant the SVM's accuracy fluctuations can be attributed to the difference in training data as a result of the use of a small k.
  - In general, the accuracy of a model when using these two different datasets used to evaluate the models (development and test data) should be similar (which was confirmed in a few experiments that were not discussed in this paper, when $k = 10$ for the test data and $k = 5$ for the development data), but it seems the low value of k is responsible for these inconclusive results regarding the SVM classifier.
- **Finding b)**: For the other classifiers, since the accuracy results are consistent between both figures for each of the four experiments, it appears the choice of fold for the test data was not responsible for the change in accuracy, which means it can be deduced that:
  - The NB model does better when stopwords are kept and symbols are removed, which supports the idea that stopwords have sentimental value for classifiers.
  - The NLP-based model performs more poorly when stopwords are removed and symbols are kept. A proposed hypothesis as to why this behavior occurred is that the NLP-based approach uses a bag-of-words technique [9], which means it analyzes the sentiment of each word found in a sentence. It attributes symbols with neutral sentimental value, which means the polarity of a sentence could be shifted to neutral by the presence of symbols, and since the Airline dataset consists primarily of negative tweets (see section 5.4.1), negative tweets are probably predicted as neutral tweets because of the presence of symbols. Interesting future work could involve reproducing this experiment with a dataset that has a large ratio of neutral tweets. In such an experiment the NLP-based approach's performance would most likely benefit from keeping symbols in the dataset.
  - The MaxEnt and Fusion model are virtually unaffected by the presence of stopwords and symbols, and since this is the case, it would be interesting to reproduce this experiment while varying the classifier sets used in the fusion model (like in the experiment detailed in section 5.8) to see what effects symbols and stopwords have on the Fusion model when the Fusion model's accuracy is not ~100%.

5.7.1 Varying Datasets. This experiment set is similar to the one discussed in section 5.7. The goal of this experiment set is to see what effects stopwords and symbols have on performance for a variety of datasets. The results of the experiments can be found in figures 6, 7, and 8, when the classifiers are trained using the Airline, Dataset3, and Stanford dataset, respectively, and the findings are discussed later in this section.

Limitations

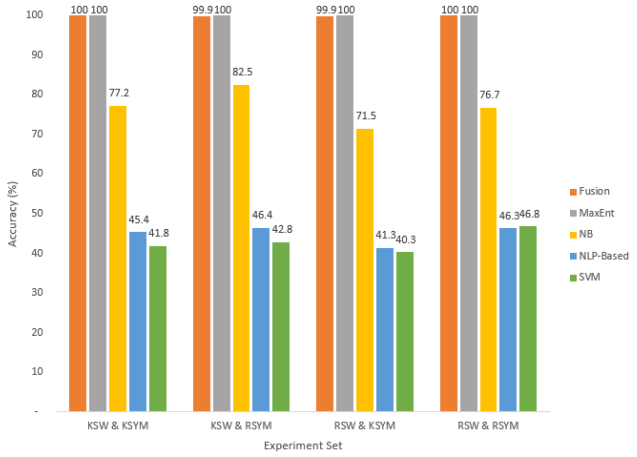

Figure 4: Stopword and symbol removal analysis when data used to test the classifiers is the development data. Training data: Airline, Testing data: 2FCV, Development data: 2FCV, RSW: removed stopwords, KSW: kept stopwords, RSYM: removed symbols, and KSYM: kept symbols.

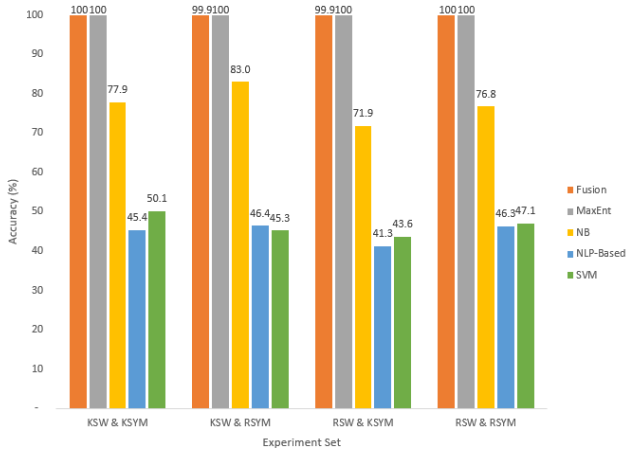

Figure 5: Stopword and symbol removal analysis when data used to test the classifiers is the test data. Training data: Airline, Testing data: 2FCV, Development data: 2FCV, RSW: removed stopwords, KSW: kept stopwords, RSYM: removed symbols, and KSYM: kept symbols.

For time constraint reasons, only the following types of experiments were performed: a) stopwords are kept and symbols are removed (KSW & RSYM) and b) stopwords are removed and symbols are kept (RSW & KSYM). Furthermore, the Fusion model and NLP-based model were not included in this experiment set for similar reasons.

Experimental Configuration

The Airline, Dataset3, and Stanford datasets were used. The testing data used was from the Stanford dataset and created using 10-fold cross validation. There was no development data set created in this experiment set. The classifiers involved were as follows:

Figure 6: Stopword and symbol removal analysis without Fusion and NLP-based models. Training data: Airline, Testing data: Stanford and 10FCV, RSW: removed stopwords, KSW: kept stopwords, RSYM: removed symbols, and KSYM: kept symbols.

Figure 7: Stopword and symbol removal analysis without Fusion and NLP-based models. Training data: Dataset3, Testing data: Stanford and 10FCV, RSW: removed stopwords, KSW: kept stopwords, RSYM: removed symbols, and KSYM: kept symbols.

Figure 8: Stopword and symbol removal analysis without Fusion and NLP-based models. Training data: Stanford, Testing data: Stanford and 10FCV, RSW: removed stopwords, KSW: kept stopwords, RSYM: removed symbols, and KSYM: kept symbols.

MaxEnt, NB, and SVM. The emoticon set used was the small set, which is shown in table 10. Stopword removal was varied and symbol removal was varied.

Findings

- **Finding a)**: Although the following analysis is not related to stopword or symbol removal, by analyzing the results, the following results suggest that in general the performance of

the classifiers underperform when trained using the Airline dataset compared with when trained using other datasets:
  - The NB classifier is the only classifier for which its performance is not perfect when using 10FCV to create the test data (see figure 6).
  - The accuracy of NB, when trained on the Dataset3 and Stanford datasets, is at least 99.7%.
  - The accuracy of each classifier is generally lower when using the Airline dataset as training, than when using other datasets as training data (most of the accuracies are lower in the Airline dataset).

  A hypothesis as to why the above is the case could be due to either of the following (which have supporting evidence from the results discussed in later sections):
  - The size of the Airline dataset (it is the dataset with the fewest tweets)
  - The limited domain of the Airline dataset (the dataset contains only airline-related tweets)
  - The additional class/label in the Airline dataset that is not present in the other datasets; that is, the Dataset3 and Stanford dataset have only positive and negative tweets, while the Airline dataset includes neutral tweets.
  - A combination of the above.
- **Finding b)**: the performance of the classifiers are significantly higher (especially for the SVM) when 10FCV is used to evaluate the models.
  - This is likely the case because the test data is of the same domain as the domain of the training data when using 10-fold cross validation; that is, the Stanford test data contains tweets from a variety of domains, so if the training data did not contain many tweets of the Stanford test dataset's domain, then the classifier's model (built from the training data) did not properly capture the domain's particularities. As a result, the Stanford test data will lead to decreased performance compared with the performance when using 10FCV to create the test data.
- **Finding c)**: when using the Airline dataset as training data, the average performance results are the worst; when using Dataset3, the average performance is higher than Airline; and when using the Stanford dataset the average performance results are the best in this experiment set.
  - It seems increasing the amount of training data increases the performance of the classifiers, since the smallest dataset is Airline (which has the worst average performance), the next smallest is Dataset3 (which has better average performance than the Airline dataset), and the biggest dataset is the Stanford dataset (which has the best average performance in this experiment set).
- **Finding d)**: for the SVM classifier, when the training data used is Stanford and Dataset3, the SVM's performance benefits the most when stopwords are included and symbols are removed; that is, when comparing the KSW & RSYM and RSW & KSYM experiments, there is a performance increase of 8.1% and 7.8% when using the Stanford and Dataset3 dataset, respectively.
  - The above is in agreement with the remarks made in [12] and [33], that symbols can hinder the SVM's performance and stopwords hold sentimental value for classifiers, respectively.
- **Finding e)**: What is interesting is that for the Airline dataset the opposite of *Finding d)* is the case: The SVM classifier's performance benefits most when stopwords are removed and symbols are kept. This difference may be attributed to any of the following points:
  - The limited size of the Airline dataset. A hypothesis is that a large dataset may benefit a classifier's performance when keeping stopwords, since due to the size of the dataset, the classifier can take advantage of the many relationships among stopwords and non-stopwords. However, in the case of a smaller dataset, stopwords may be so rare that such relationships cannot be found, and therefore, cannot be exploited by the classifier effectively. As a result, in this case a classifier is better to ignore stopword relationships (by removing stopwords) and focus on taking advantage of the limited number of more relevant/stronger relationships found in the small dataset.
  - The additional class/label in the Airline dataset that is not present in the other datasets; that is, the Dataset3 and Stanford dataset only have positive and negative tweets, while the Airline dataset includes neutral tweets.
  - My hypothesis is that the stopword removal had the greatest impact on the performance compared with the effects of symbols. In other words, removing symbols and stopwords would yield better accuracy than removing stopwords and keeping symbols, but due to the limitations mentioned at the start of this section, I leave this investigation as future work.

5.8 Fusion Model and Classifier Diversity Experiment

The goal of this experiment set was to see the the effects of classifier diversity on the performance of the Fusion model. By leaving out some classifiers and including others in the Fusion model, analyzing the performance results will provide insight into the effects of classifier diversity. Recall that there was no diversity measure implemented in this experiment set to filter non-diverse classifiers, such as the Q statistic implemented in [9]. Note that this set of experiments is divided in two subsection: section 5.8.1 and section 5.8.2 discuss the results when using the Airline and Stanford dataset as training data, respectively.

Limitations

Due to time constraints, I did not run this experiment set for the Dataset3 dataset, and I only ran a subset of this experiment set (only for a couple of classifier sets) for the Stanford dataset. It would have been interesting to see the results, since they may have given more insight than available by only observing the results shown in figure 9 and table 7. For example, it is not clear if the accuracy of the Fusion model is limited by the maximum accuracy of a classifier in its classifier set, since for each classifier set, the Fusion model's accuracy was at most the best accuracy of its individual classifiers. Although the results found in [9] suggest the Fusion model can outperform the accuracy of its individual classifiers, there are not enough results in the present work to reproduce [9]'s

findings. Furthermore, future work could involve reproducing these experiments while comparing the diversity of the classifiers.

Experimental Configuration

The Airline and Stanford dataset were used. The testing data used was different for both subsets of experiments. The classifiers included were also different for both subsets of experiments. Otherwise, the following experimental configuration was shared between both experiment sets: a) 5-fold cross validation was used to create the development dataset; b) stopwords were removed; c) symbols were not removed; and d) the emoticon set used was the small set, which is shown in table 10.

5.8.1 Airline Dataset. This experiment set investigates classifier diversity and its effects on the Fusion model's performance when using the Airline dataset as training data.

The experimental results can be seen in figure 9, where each bar represents the accuracy of the Fusion model for a different subset of classifiers. The minimum classifier set size was 2, since the Fusion model combines classifier predictions (it would not make sense to have a single classifier in a classifier set, because there would be no data fusion at all). To analyze these results, the performance of the other classifiers in these experiments was also considered. Table 7 contains the accuracies of the classifiers used in this experiment, including their accuracy when testing data and when development data was used to evaluate the classifier. The findings that were found are discussed later in this section.

Experimental Configuration

In this experiment set, the Airline dataset was used as training data and 10-fold cross validation was used to create the test dataset. The classifiers that were included in this experiment are as follows: Fusion, MaxEnt, NB, SVM, and NLP-based.

Findings

- **Finding a)**: At first glance, it would appear that the Fusion model only performs as well as its best performing classifier; however, the accuracy of the Fusion model when using the MaxEnt, SVM, and NLP-based classifiers would suggest otherwise. MaxEnt has 100% accuracy, yet the Fusion model only obtained 85.5% accuracy.
  - The remark above supports the claim made by [9]: that diversity of the classifier's predictions is important; that is, having classifiers that make predictions that contradict each other is good for the Fusion model.
- **Finding b)**: Another interesting result is comparing the accuracy of the Fusion model for the {MaxEnt, SVM} and {NB,SVM} classifier sets, which will be referred to as classifier sets a) and b), respectively, in this section, for sake of discussion simplicity. Both sets a) and b) contain the SVM model, which does relatively poorly on its own (~42% accuracy), and contain a model that does relatively well. Observing the accuracy of the Fusion model for the classifier set a) reveals the accuracy is nearly identical to the accuracy of the single MaxEnt model (100% v.s 99.5%), while the accuracy of the fusion model for set b) is significantly lower than the accuracy of the single NB model (~88% vs. 58.8%).
  - This is another finding that supports the idea that classifier diversity has an effect on the Fusion model's accuracy.

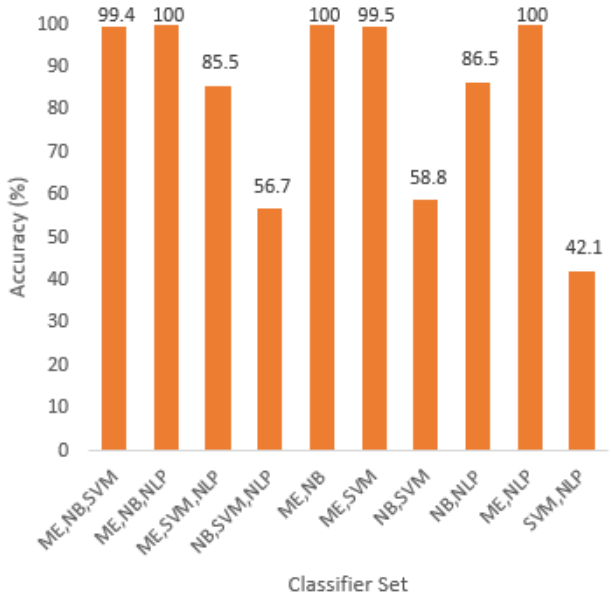

Figure 9: Fusion model accuracy for all possible classifier subsets of size two among the MaxEnt (ME), NB, SVM, and NLP-based (NLP). Training data: Airline, Testing data: 10-fold cross validation, Development data: 5-fold cross validation.

Table 7: Accuracy of the various classifiers used in the fusion model diversity experiment. Training data: Airline, Testing data: 10FCV, and Development data: 10FCV.

Classifier	Testing data	Accuracy (%)
MaxEnt	dev	100
MaxEnt	test	100
NB	dev	88.4
NB	test	88.5
NLP	dev	41.3
NLP	test	41.3
SVM	dev	41.3
SVM	test	43.9

5.8.2 Stanford Dataset. This experiment set investigates classifier diversity and its effects on the Fusion model's performance when using the Stanford dataset as training data. The results of this experiment set can be seen in tables 8 and 9. The findings that were found are discussed later in this section.

Limitations

Only two classifier sets were used in this experiment set due to time constraints, namely: the sets {MaxEnt, NB} and {MaxEnt, NB, NLP-based}. The SVM was excluded, since the implementation was not efficient enough to run quickly on a large dataset (future work could involve improving the efficiency of my Java project's API to the *Stanford CoreNLP* library's SVM).

Experimental Configuration

Table 8: Accuracy of the classifiers for both classifier sets using development data to test the classifiers. Training data: Stanford, Testing data: Stanford, and Development data: 5FCV.

Classifier/Classifier Set	{MaxEnt, NB}	{MaxEnt, NB, NLP}
Fusion	100%	100%
MaxEnt	100%	100%
NB	99.9%	99.9%
NLP-based	-	55.4%

Table 9: Accuracy of the classifiers for both classifier sets using test data to test the classifiers. Training data: Stanford, Testing data: Stanford, and Development data: 5FCV.

Classifier/Classifier Set	{MaxEnt, NB}	{MaxEnt, NB, NLP}
Fusion	100%	100%
MaxEnt	100%	100%
NB	100%	100%
NLP-based	-	58.8%

In this experiment set, the Stanford dataset was used as training data and as test dataset. The classifiers that were included in this experiment are as follows: Fusion, MaxEnt, NB, and NLP-based.

Findings

- **Finding a)**: The Fusion classifier obtains 100% accuracy for both classifier sets.
  - It would have been interesting to see the effects on performance when using other classifier sets, since when using the Airline dataset and the classifier sets {MaxEnt, NB} and {MaxEnt, NB, NLP-based}, the Fusion model also had 100% accuracy.
- **Finding b)**: It can also be seen that the NLP-based method does relatively poorly, although the results are close to accuracy results (using the Stanford dataset) found by [9], which are shown in table 4. This is good, since one of the goals of the experiments of the present work was to reproduce some of the experimental results found [9] and [12].

5.9 Emoticon Experiment

The goal of this experiment set was to see the effects emoticons have on performance. The following lists of emoticons were used during the tweet normalization phase, which is discussed in section 3.1, to investigate their effects on performance: an empty list of emoticons, a small list of emoticons (see table 10), and an extended list of emoticons (see table 13). The experiments found in this section can be summarized as follows: section 5.9.1 analyzes the frequency of emoticons in the datasets, section 5.9.2 analyzes the effects of emoticons on the NLP-based approach, section 5.9.3 analyzes the effects of emoticons for each dataset while using the Stanford dataset as test data, analyzes the effects of emoticons for each dataset while using 10FCV to create test data 5.9.4, and section 5.9.5 analyzes the effects of emoticons for each classifier using 10FCV for creating test data.

Motivation

Table 10: The small emoticon set used in some of the present work's experiments, which are also used in the experiments performed in [12].

Positive	Negative
:-)	:-(
:)	:(
=)	=(
:D	

The motivation for conducting these experiments is as follows: [9] used some of the emoticons (some of which are illustrated in table 10) used in the experiments from [12]. It can be argued that this list is limited in the context of applying it to tweet normalization; that is, many emoticons ('=D', for example) will not be converted to their appropriate template words. As a result, since emoticons consist primarily of symbols and [12] state symbols negatively affect performance, an extended list of emoticons would most likely benefit performance. Furthermore, another limiting factor is that [12] include emoticons such as ':)'; however, they do not include its mirror-equivalent ('(:') emoticon. Both ':)' and '(:' are arguably positive emoticons. In the present work, the term mirror-equivalent emoticon is used to denote an emoticon that keeps its sentiment when the order of its characters is reversed, and non-symmetric characters (such as '(', for example) are replaced with their mirrored character (for example, ')'). Symmetric characters such as ':' can simply be found in the mirror-equivalent emoticon without being replaced. Some emoticons may not have a mirror-equivalent emotion; for example, the positive emoticon ':D' does not have a mirror-equivalent emoticon, since 'D:' would be considered a negative emoticon.

Extended Emoticon Set

As such, the present work proposed an extended list of emoticons, which can be found in table 13. The extended emoticon list was compiled by simply browsing the web. Each emoticon was manually inspected to make sure the sentiment of the emoticon was accurate. The extended set also included each emoticons' mirror when possible.

Limitations

- A source of error is present regarding the emoticons used in [12]; that is, the emoticons they used to query the Twitter API for creating the Stanford dataset included two emoticons that each contained a white space (':)' and ': (‘). Since the implementation of the present work parsed tweets word-by-word (that is, by delimiting the tweet by white spaces), the emoticons that include white spaces would never have been detected in the present work's implementation. As such, these emoticons were removed from the present work.

5.9.1 Emoticon Frequency Analysis. This experiment analyzes the presence of emoticons in the datasets. To get an idea of the presence of emoticons in the three datasets used in the experiments of the present work, the number of emoticons in each dataset were counted. Since this experiment set used two sets of emoticons (the small emoticon set shown in table 10 and the extended emoticon set found in table 13 in the Appendix), these sets were used to count the

number of emoticons found in each dataset. Furthermore, since the tweet normalization (for more information on emoticons and their role in tweet normalization see section 3.1) converts emoticons into their template words (positive and negative emoticons are replaced with the word 'smile' and 'frown', respectively), it may be important to count the frequency of these template words in the datasets before normalization. Since some of the datasets are relatively large (the Stanford dataset, for example), some automation was added to help count the emoticon frequencies. The method used is presented as follows:

(1) The frequency of the whole words 'smile' and 'frown' found in the datasets were counted. Words that contained these words, for example, 'smiling', were not counted.
   - This enabled the ability to find the frequency of these words that are actually found in the datasets. Without this step, after replacing emoticons with their template words ('smile' or 'frown'), there would have been no way of knowing how many emoticons were replaced with their template words vs. how many template words were already in the dataset.
(2) The frequency of the emoticon template words in the normalized datasets (the datasets that had their emoticons replaced with the emoticon's respective template word) were counted.
(3) The template word frequencies of the normalized datasets were subtracted from the template word frequencies of the original datasets, which resulted in the emoticon frequency.

Table 11 summarizes the emoticon frequency analysis results obtained by applying the steps above, and by observing these results, the following remarks can be made:

Findings Part 1

- **Finding a)**: The emoticons present in the Airline dataset primarily consists of the emoticons from the small emoticon set, since there are only 15 and 10 additional positive and negative emoticons, respectively, recognized using the extended set.
- **Finding b)**: Both the Dataset3 and Stanford datasets have a significant number of additional emoticons from the extended emoticon set compared with the small emoticon set.

Furthermore, to gain a better understanding when comparing the emoticon frequencies between datasets, the emoticons' frequency were normalized with respect to the number of tweets in their respective dataset, which can be seen in table 12. This way, the normalized frequency table can illustrate the fraction of tweets that have emoticons. The following can be deduced when using the extended emoticon set vs. the small emoticon set:

Findings Part 2

- **Finding a)**: For the Airline dataset, there are approximately 1.2 and 1.3 times more normalized positive and negative emoticons, respectively.
- **Finding b)**: For the Dataset3 dataset, there are approximately 1529 and 18.1 times more normalized positive and negative emoticons, respectively.
- **Finding c)**: For the Stanford dataset, there are approximately 51 and 17.9 times more normalized positive and negative emoticons, respectively.

Table 11: Number of emoticons found in datasets. PE-S: positive emoticon from small set, NE-S: negative emoticon from small set, PE-E: positive emoticon from extended set, NE-E: negative emoticon from extended set, and ∄Neu and ∃Neu: without and with neutral tweets, respectively. The 'smile' and 'frown' column represent the frequency of these words in the datasets before normalization.

Dataset	'smile'	'frown'	PE-S	NE-S	PE-E	NE-E
Airline	3	1	63	31	78	41
Dataset3	142	4	1	40	1529	722
Stanford:						
Train	1952	78	341	529	17402	9478
Test: ∄Neu	0	0	26	13	33	17
Test: ∃Neu	0	0	26	13	33	18

Table 12: Fraction (in percentage) of tweets that contain emoticons. PE-S: positive emoticon from small set, NE-S: negative emoticon from small set, PE-E: positive emoticon from extended set, NE-E: negative emoticon from extended set, and ∄Neu and ∃Neu: without and with neutral tweets, respectively. The 'smile' and 'frown' column represent the fraction of tweets that contain these words in the datasets before normalization.

Dataset	'smile'	'frown'	PE-S	NE-S	PE-E	NE-E
Airline	0.020	0.007	0.424	0.208	0.524	0.276
Dataset3	0.142	0.004	0.001	0.040	1.529	0.722
Stanford:						
Train	0.124	0.005	0.022	0.0337	1.107	0.603
Test: ∄Neu	0	0	5.221	2.610	6.627	3.414
Test: ∃Neu	0	0	7.242	3.621	9.192	5.014

Using these remarks, the following hypotheses were made: a) the Dataset3 should be the dataset that benefits the most from using the extended emoticon set, b) this should also be the case for the Stanford dataset to a lesser degree, and c) for the Airline dataset, the performance increase will be negligible compared with the other two datasets.

5.9.2 Effects of Emoticons using All Datasets on the NLP-based Approach. In this experiment set, the emoticon list used to replace emoticons with their template words were varied, and the effects on the NLP-based approach were examined. The results can been seen in figure 10, and the findings that were found are discussed later in this section.

Limitations

- Due to time constraints, not all combinations of datasets and emoticon sets were included in this experiment set.
- Although development data was created, it is worth noting that the NLP-based approach does not require development data to be evaluated, but due to implementation limitations, all models were run at once.

Experimental Configuration

In this experiment set, the Airline, Dataset3, and Stanford datasets were used as training data. The Stanford dataset was used as the

test dataset. 5FCV was used to create the development dataset. Only the NLP-based approach was included in this experiment set. Stopwords were removed. Symbols were not removed. The varied emoticon lists are as follows: an empty list (no emoticons are replaced), the small emoticon list, and the extended emoticon list.

Findings

- **Finding a)**: The NLP-based approach performs relatively similar for the following: Dataset3 using an empty emoticon set, Dataset3 using the extended emoticon set, and Stanford using the small emoticon set; that is, each accuracy is within 1.1% of each other.
- **Finding b)**: The model's accuracy is ~14% lower when performed on the Airline dataset using the small emoticon set. This behavior may be attributed to the following reasons:
  - The limited domain of the Airline dataset. Recall that the test data is from the Stanford datasets (also note that the Stanford dataset contains neutral tweets that are removed when used to test the other datasets that do not have neutral tweets), which include tweets of a variety of domains according to [12]. All tweets in the Airline dataset are airline-related, while the Stanford and Dataset3 datasets have tweets from a variety of domains.
  - The additional class/label in the Airline dataset that is not present in the other datasets; that is, the Dataset3 and Stanford dataset only have positive and negative tweets while the Airline dataset includes neutral tweets. With an extra class to predict, the NLP-based method may struggle compared with predicting only two sentiment classes.
  - A combination of the two points above may also be the reason for the observed behavior.
- **Finding c)**: Although the increase is small (0.5%), it can be seen that when using the Dataset3, the extended emoticon set increases the NLP-based approach's accuracy more than the the empty emoticon set.

5.9.3 Effects of Emoticons using All Datasets and Stanford Test Data. In this experiment set, the emoticon list used to replace emoticons with their template words was varied for each dataset, and the effects on the MaxEnt, NB, and SVM classifier were examined. The results when trained on the Airline, Dataset3, and Stanford dataset is shown in figures 11, 12, and 13, respectively. The findings that were found are discussed later in this section.

Experimental Configuration

In this experiment set, the Airline, Dataset3, and Stanford datasets were used as training data. The Stanford dataset was used as the test dataset. The classifiers included in this experiment set follow: MaxEnt, NB, and SVM. Stopwords were removed. Symbols were not removed. The varied emoticon lists are as follows: an empty list (no emoticons are replaced), the small emoticon list, and the extended emoticon list.

Findings

- **Finding a)**: When the Airline dataset is used as the training data, the accuracy is generally lower than when the other datasets are used as training data.
- **Finding b)**: Emoticons appear to affect the SVM classifier more significantly than the other classifiers. The greatest

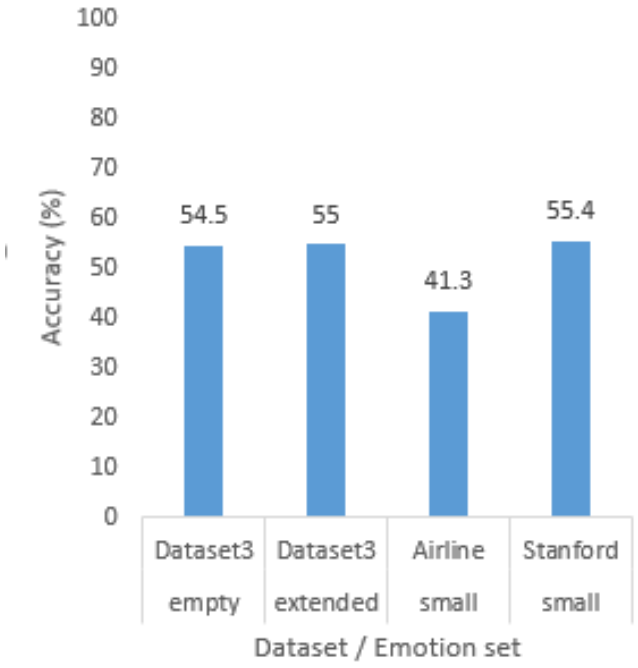

Figure 10: Emoticon analysis on NLP-based classifier. Testing data: Stanford, and Development data: 5FCV.

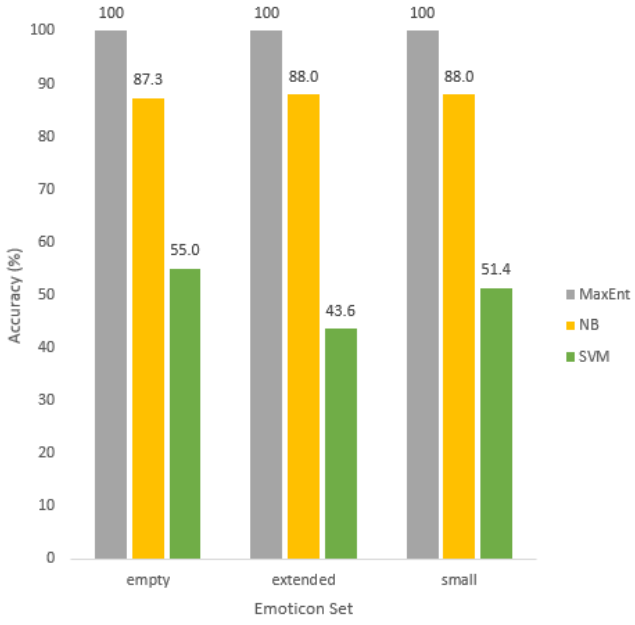

Figure 11: Emoticon analysis on MaxEnt, NB, and SVM classifiers. Training data: Airline, and Test data: Stanford.

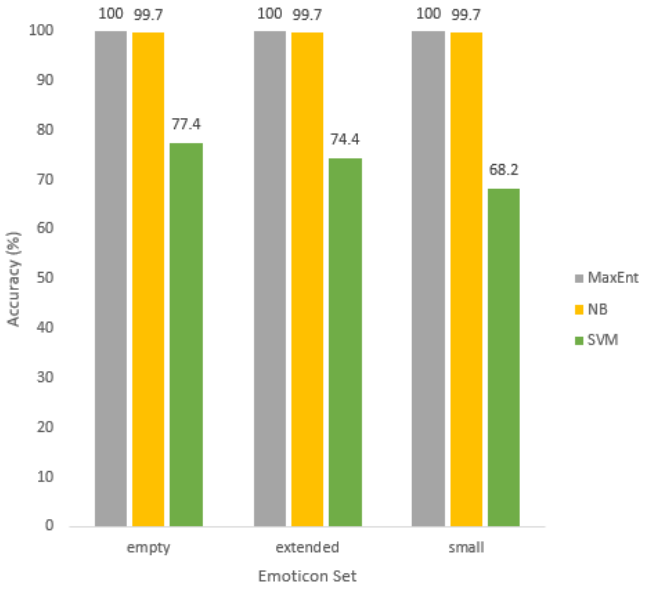

Figure 12: Emoticon analysis on MaxEnt, NB, and SVM classifiers. Training data: Dataset3, and Test data: Stanford.

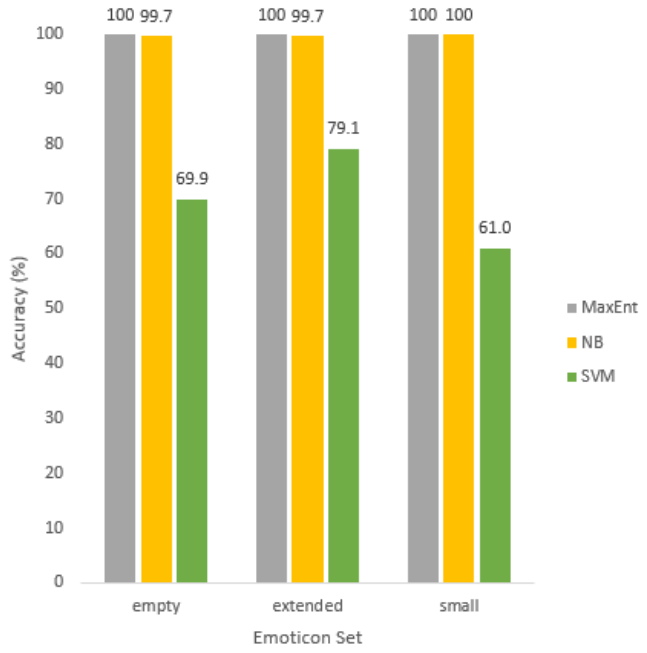

Figure 13: Emoticon analysis on MaxEnt, NB, and SVM classifiers. Training data: Stanford, and Test data: Stanford.

accuracy difference the SVM classifier faces as a result of the emoticon set difference can been seen in figure 13; that is, when the Stanford dataset is used as training data, there is a ~18% between the accuracy of the SVM classifier when using the extended emoticon list vs. the small emoticon list.

- **Finding c)**: Compared with the findings in the later section 5.9.5, the emoticon sets in this experiment set affect the accuracy of the classifiers more significantly. This may be because the Stanford dataset's test data is used to evaluate the accuracy instead of 10FCV.
  - There is no obvious explanation as to why this is the case. Upon examining tables 11 and 12, there was no clear link between the emoticon statistics of the datasets and the performance results.
- **Finding d)**: What also cannot be explained is why the SVM's performance is best when using the extended emoticon set and trained on the Stanford dataset, while when trained on the Dataset3 and Airline datasets, the SVM performs best without any emoticon replacements (the empty set).

5.9.4 Effects of Emoticons using All Datasets and 10FCV Test Data. This experiment is identical to the experiment detailed in 5.9.3, but instead of using the Stanford dataset's test data, 10FCV is used to create the test data.

Findings

- **Finding a)**: Results of this experiment set revealed that the MaxEnt, NB, and SVM classifiers each obtained at least 99.9% accuracy on all three datasets for each set of emoticons. There is, however, one case where the NB classifier obtained less than 99.9% accuracy; that is, when trained using the Airline dataset, the NB classifier obtained 92.0%, 91.8%, and 92.0% when using the empty, extended, and small emoticon lists, respectively. This accuracy decrease for the NB classifier may be due to the following reasons:
  - Since 10FCV validation was used to create the test data, the test data shares the same domain as the training data, which means the domain of the Airline dataset cannot be attributed to the observed behavior.
  - The small size of the Airline dataset.
  - The additional class/label in the Airline dataset that is not present in the other datasets; that is, the Dataset3 and Stanford dataset only have positive and negative tweets, while the Airline dataset includes neutral tweets.
  - NB may also be more sensitive to smaller training datasets compared to the MaxEnt and SVM classifiers.
  - More experimentation is required to confirm these hypotheses.
- **Finding b)**: similarly to the results detailed in section 5.9.3, the emoticon list appears to have little effect on the accuracy of the classifiers.
  - The only thing that is different from this experiment set and the experiment set detailed in section 5.9.3, which shows that the emoticon set can affect the accuracy, is the use of 10FCV validation to create the test data. More research is required to find out why this is the case.

5.9.5 Effects of Emoticons on All Classifiers. In this experiment set, the emoticon list used to replace emoticons with their template words were varied for the Airline and Dataset datasets, and the effects on each classifier were examined. The results when trained on the Airline dataset are shown in figures 14 and 15, and those

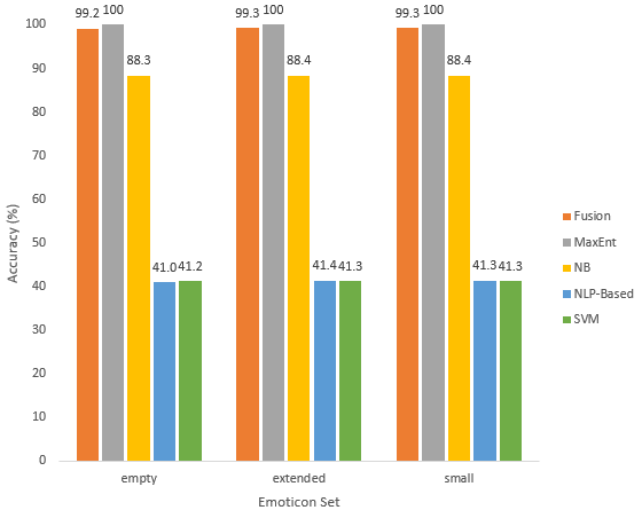

Figure 14: Emoticon analysis on each classifier (development data is used to test the classifiers for these results). Training data: Airline, Test data: 10FCV, and Development data: 5FCV.

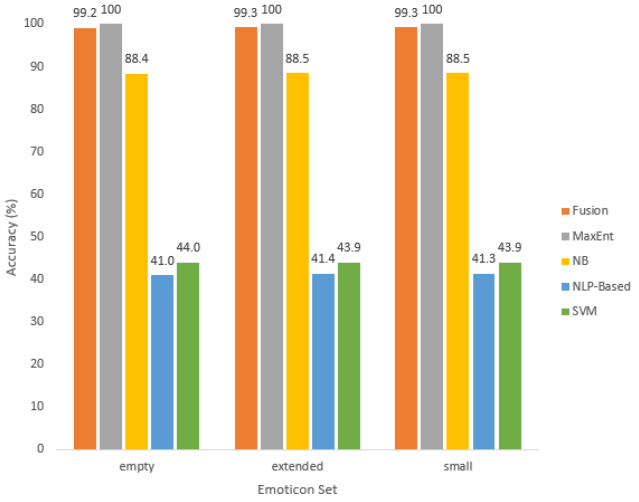

Figure 15: Emoticon analysis on each classifier (test data is used to test the classifiers for these results). Training data: Airline, Test data: 10FCV, and Development data: 5FCV.

trained on the Dataset3 dataset are shown in figures 16 and 17. The findings are discussed later in this section.

Experimental Configuration

In this experiment set, the Airline, and Dataset3 were used as training data. 10FCV was used to create the test dataset. 5FCV was used to create the development data. The classifiers included in this experiment set follow: Fusion, MaxEnt, NB, SVM, and NLP-based. Stopwords were removed. Symbols were not removed. The varied emoticon lists are as follows: an empty list (no emoticons are replaced), the small emoticon list, and the extended emoticon list.

The hypothesis was that the extended emoticon dataset would significantly improve the performance of classifiers using the Dataset3 dataset (compared with only using the small emoticon dataset), since when using the extended emoticon dataset 2251 (1529 + 722) emoticons were detected instead of 41 (1 + 40) emoticons when using the small emoticon dataset (see section 5.9.1 and table 11 for more details).

Findings

- **Finding a)**: The emoticon set has an insignificant effect on the accuracy of the classifiers; that is, the greatest accuracy difference is only 0.4% for all classifiers on both the Airline and Dataset3.
  - It seemed strange at first, but these results do make sense. Observing the relative frequency of the emoticons (shown in table 12) reveals that at most there are only ~2.2% of tweets with emoticons for Dataset3 using the extended emoticon list. In other words, since all other cases result in fewer than 2.2% of tweets with emoticons, it is not surprising that the accuracy did not change much from varying the emoticon sets.

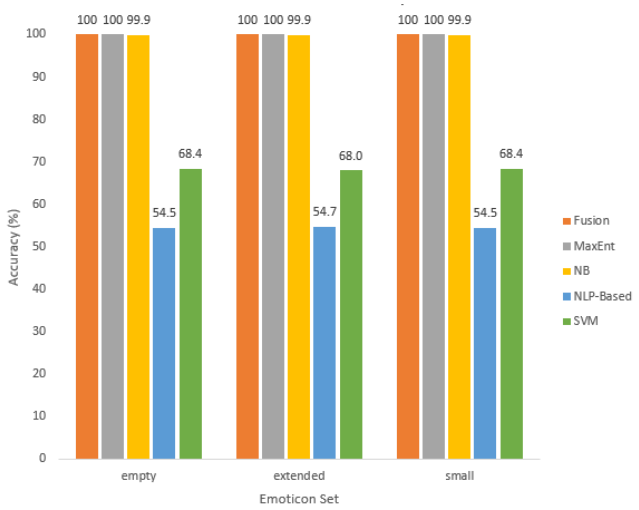

Figure 16: Emoticon analysis on each classifier (development data is used to test the classifiers for these results). Training data: Dataset3, Test data: 10FCV, and Development data: 5FCV.

  - However, the above hypothesis does not really apply to the results discussed in section 5.9.3; therefore, nothing significant can be concluded about the emoticon sets and their effects on accuracy.

5.10 Sources of Error

This section summarizes sources of error that may have had an effect on performance results:

- **NLP-based Implementation**: [9] included the 'ner' parameter in the command line call to the *Stanford CoreNLP* library

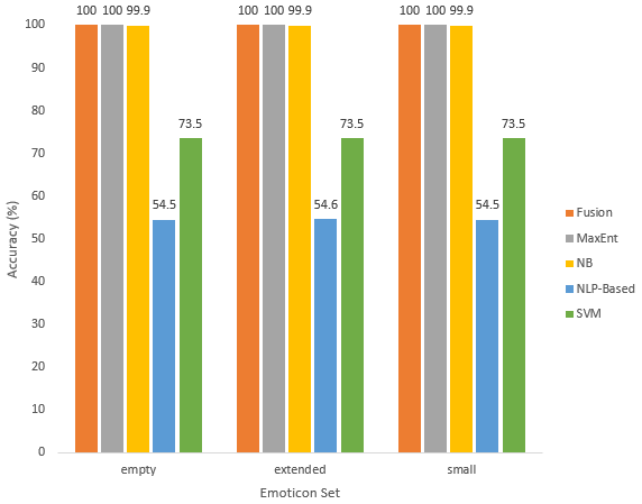

Figure 17: Emoticon analysis on each classifier (test data is used to test the classifiers for these results). Training data: Dataset3, Test data: 10FCV, and Development data: 5FCV.

API, while this parameter was mistakenly not included in the implementation on the present work. As a result, there may be differences between the implementation of the NLP-based approach of the present work and [9].

- **NB Features**: NB assumes that the features are conditionally independent, but this is not true for the bigram features, which were used with the NB classifier in the present work. This may have hindered the results.
- **Neutral Tweets and the Stanford Dataset**: Note that for Stanford dataset, since the training data contained no neutral tweets, neutral tweets were removed from the Stanford test data when evaluating classifiers that were training using the Dataset3 or Stanford dataset. However, in the details provided by [9], they in fact had neutral tweets in their Stanford training data. This inexplicable difference may lead to different results. It is unclear how [9] obtained neutral tweets, since they claim they were using the Stanford dataset that was created in [12], and [12] explicitly stated their training data did not contain any neutral tweets.
- **Lack of the Q Diversity Measure**: There may have been a source of error when reproducing the Fusion model results, since the present work did not use the Q measure to diversify classifiers (for implementation simplicity reasons, the Q statistic was not implemented).
- **MaxEnt's Perfect Performance**: Although it was exciting to see that the MaxEnt classifier predicted every tweet's polarity perfectly, it was worrying, since classifiers rarely perform this well. Furthermore, the accuracy results of the MaxEnt classifier found in [9] and [12] were not 100%. This suggested the perfect accuracy of the MaxEnt classifier may have been due to an implementation bug. It was not obvious that such a bug existed; however, since even when the classifier was run in the command line using the *Stanford CoreNLP* library's command line API, the accuracy was also perfect. To determine the cause of the suspected error, invalid test data instances were purposefully added into the test data to see if the MaxEnt classifier would produce incorrect predictions. In the Stanford dataset's test data, labels of the positive tweets were replaced with negative labels. As suspected, the MaxEnt classifier still obtained 100% accuracy, which means either the command used to run the classifier was incorrect, the library's configuration was incorrect, or the library had a bug in its API. Furthermore, this bug also suggests that the results found in this paper may be inaccurate, since if the MaxEnt classifier had a bug, then NB and SVM may also have had a bug.
- **Non-ASCII Characters**: The *Stanford CoreNLP* library fails to deal with tweets that are exclusively non-ASCII characters and empty strings. Since this had not been realized sooner, some of the experimental results included this bias, but it is not clear which experiment may have suffered (non-ASCII characters were later removed from the dataset in the experiments later performed). When computing the average sentiment of each sentence of a tweet, if a tweet exclusively contained non-ASCII characters, then the tweet would be treated as if it had no sentence. This was also the case for empty tweets, the *Stanford CoreNLP* library would not provide any sentiment, but in the implementation of the present work, the implementation ignored tweets that were empty after normalization. As a result, the implementation would find an average sentiment score of 0 and would interpret this as negative (see table 1). To examine the impact of this bias, the number of tweets of each dataset that contained only non-ASCII characters (white-spaces includes) were counted. The Airline and Dataset3 datasets had no such tweets, and in the Stanford dataset, there was only a single tweet (a completely non-English tweet). The non-ASCII tweet is positive. Since this bias is insignificant, the effects a purely non-ASCII tweet had on the other classifiers used in the experiments was not investigated. The bias was therefore not significant, but for completeness it was discussed in this section.
- **NLP-based Emoticon Analysis Bias**: In the experiment detailed in section 5.9.2, each normalized dataset should have had no symbols removed; however, the Stanford dataset's test data that was used had the character '.' removed from each tweet accidentally when both the Stanford and Dataset3 datasets were used as training data. When using the Airline dataset as training data, the '.' was not removed (as desired) from the Stanford test data.
- **Stopwords and Words with Sentimental Value**: Although the point of the stopword list was to remove words that occur frequently and do not bring much analytical meaning to a sentence, there are words that can have sentimental value depending on the context; for example, the word 'like' is among the stopwords used in the experiments. When the word is used in the following context: 'This movie is *like* the next movie.', it arguably holds less sentimental value than when used in this context: 'I *like* this movie.'. Therefore, removing stopwords that contain sentimental value is a source error in the experiments conducted in the present work.

5.11 Summary

This section summarizes the important results and conclusive remarks that can be made from analyzing the experimental results of the present work.

Airline Dataset

- When the Airline dataset was used to train the classifiers, generally the performance of these classifiers was lower than when the classifiers were trained using the Dataset3 and Stanford datasets. This was most likely the case because of one of more of the following:
  - The Airline dataset was smaller than the other datasets, which means the classifiers had less training data.
  - The Airline dataset only had one domain (airline-focused tweets) instead of a variety of domains (the Dataset3 and Stanford datasets include more domains than the Airline dataset).
  - The additional class/label in the Airline dataset that was not present in the other datasets; that is, the Dataset3 and Stanford dataset had only positive and negative tweets, while the Airline dataset included neutral tweets.
- The performance of classifiers was generally better when the testing data was created using 10FCV compared with when using the Stanford dataset's test data. This was most likely due to the test data being from the same domain as the training data's domain when using 10FCV.

Classifier Performance

- The MaxEnt classifier performed perfectly in all experiments; this should be taken with a grain of salt (see section 5.10).
- The Fusion classifier performed exceptionally well in all experiments, as it has at least 99% accuracy in all experiments, although since it included the MaxEnt classifier most of the time, its results should also be taken with a grain of salt.
- On average, the performance of the classifiers can be summarized from best to worst in the following order: MaxEnt, Fusion, NB, SVM, and NLP-based.
- The SVM classifier was the classifier that had the most variety in its accuracy. It can be reasoned that the SVM was more sensitive to changes in data distributions (especially training data) than the other classifiers.
- The NB classifier performed well using the Dataset3 and Stanford datasets as training data, and performed slightly worse when using the Airline dataset as training data.

Effects of Stopwords and Symbols

- It is difficult to conclude anything significant about the effects of the presence of stopwords and symbols, due to the limited number of experiments and since the effects vary across datasets. Remarks that can be made are as follows:
  - The effects on the NB model are not clear for each dataset.
  - The SVM model, when using the Airline dataset, performs better when stopwords are removed and symbols are kept than when stopwords are kept and symbols are removed. When using the Dataset3 or Stanford dataset as training data, the opposite was the case.
  - Stopwords and symbols have little effect on performance when using 10FCV to create the test data for each dataset.
  - Stopwords can in fact have some form sentimental value when used by a classifier.
  - Symbols can affect the performance results.

Effects of Emoticons

- Emoticons and their effects on performance are generally not clear.
  - Using the Stanford dataset's test data and varying the emoticon set produced accuracy differences for the SVM classifier only. The other classifiers are not affected.
  - In a similar experimental configuration, if instead the test data was created using 10FCV, then no effects on accuracy can be observed for any classifier when varying the emoticon set.
  - One of the experimental results suggested that the NLP-based approach benefits from using the extended emoticon set over the empty set, although the accuracy increase was only 0.5%.

Result Reproduction

In terms of reproducing the results of previous works found in table 4, below are the findings:

- MaxEnt: reproducing the MaxEnt results failed due to an unknown bug.
- NB: In the same experiment as the one detailed below in the SVM bullet, 100% accuracy was achieved with the NB classifier. However, there are doubts about the correctness of this accuracy result, since the MaxEnt had a bug.
- SVM: the best accuracy obtained, when the training data and testing data were from the Stanford dataset, was 82.2% accuracy when stopwords were kept and symbols were removed (see figure 8). In fact, 82.2% accuracy was also the best accuracy found for the SVM classifier in the previous works.
- NLP-based: the best accuracy obtained was 58.8% when using the Stanford dataset (see table 9). This was an improvement on the accuracy that [9] obtained (55.14%), although, their best accuracy on a dataset (Movie), which was not used in the present work, was 59.66%.
- Fusion: Due to time constraints, the exact Fusion model experiments performed in [9] could not be reproduced; that is, the experiments of the present work did not include all classifiers into the Fusion model using the Stanford dataset. Also, since the MaxEnt classifier had a bug (and potentially NB as well), most of the Fusion model results found in this paper cannot be trusted. However, what can be concluded is that the Fusion model does depend on the diversity of its classifiers' predictions; that is, just because a classifier performs well does not mean when it is included in the Fusion model the Fusion will perform as accurately.

6 CONCLUSION

The present work investigates performing sentiment analysis on Twitter data. The challenges that present themselves when performing Twitter sentiment analysis include the label sparsity problem, the variety of domains in the Twitter data, the sentiment drift problem, the unstructured grammar and limited length of tweets, the class ratio skew, and feature selection.

The present work recreated some of the experiments performed in [12] and [9]. The goal was to reproduce some of their results using the Stanford dataset. The classifiers used in the experiments of the present work include Maximum Entropy, Naive Bayes, Support Vector Machine, a natural language processing (NLP)-based approach used in [9], and a Fusion model proposed by [9]. The datasets used are the Airline, Dataset3, and Stanford datasets.

The experiments conducted in the present work investigate classifier diversity and its effects on the Fusion model's performance, a classifier performance analysis and k-fold cross valiation, and the effects that emoticons, stopwords, and symbols have on the performance of each classifier.

Results

- The diversity of the classifiers' predictions had an impact on the Fusion model's performance; that is, the Fusion model's performance was not necessarily as good as its best performing classifier.
- The effect of emoticons on performance is not clear, although some experiments did reveal some classifiers can be affected by emoticons.
- The presence of symbols and stopwords had a small effect on some of the performance results, but the results are not significant enough to draw any strong conclusions. Nevertheless, it can be concluded that stopwords do in fact have some form of sentimental value, and that the presence of symbols can affect performance results.
- The average performance of classifiers is negatively affected when using the Airline dataset as training data instead of the Dataset3 or Stanford datasets. This was most likely due to the small size of the Airline datasets, its narrow domain, and the fact it had three sentiment classes (positive, neutral, and negative) instead of two classes.
- The accuracy results found in [12] and [9] for the SVM and NLP-based classifiers using the Stanford dataset were able to be reproduced (within a small margin of error) in the present work's experiments.

Future Work

- Varying the use of the bigram and unigram features to see the effects on the performance. Specifically, analyzing the effects on NB would be of particular interest, since the unigram feature satisfies the feature conditional independence assumption of NB (bigrams do not respect this constraint).
- Investigating what went wrong with the MaxEnt classifier (it had 100% accuracy in each experiment).
- Conducting experiments with additional test datasets (instead of only using 10-fold cross validation or the Stanford dataset).
- Running the SVM using the *SVMLight* library instead of the *Stanford CoreNLP* library or using feature presence instead of frequency, as suggested by [12]. It was not efficiently implemented in the present work.
- Analyzing the sentiment of the stopword list used in the experiments of the present work would be interesting, since it may help to further analyze and understand the results found in the present work. For example, there may be many neutral stopwords, which could be the reason of the relatively poor performance when using the Airline dataset compared with using the other datasets (the other datasets do not have neutral tweets).
- Performing experiments that investigate the performance effects of varying the classifiers' parameters (since these parameters remained static throughout the experiments).
- Reproducing the experiments that included the NLP-based approach and adding the 'ner' parameter to the command that calls the *Stanford CoreNLP* library's API.

REFERENCES

[1] [n.d.]. Stanford Twitter Sentiment Dataset. https://www.kaggle.com/kazanova/sentiment140/data. Accessed: 2020-02-17.

[2] [n.d.]. Twitter Sentiment Dataset. https://www.kaggle.com/imrandude/twitter-sentiment-analysis. Accessed: 2020-02-17.

[3] [n.d.]. Twitter US Airline Sentiment Dataset. https://www.kaggle.com/crowdflower/twitter-airline-sentiment. Accessed: 2020-02-17.

[4] Salha M Alzahrani. 2018. Development of IoT mining machine for Twitter sentiment analysis: mining in the cloud and results on the mirror. In *2018 15th Learning and Technology Conference (L&T)*. IEEE, 86–95.

[5] Felipe Bravo-Marquez. 2017. *Acquiring and Exploiting Lexical Knowledge for Twitter Sentiment Analysis.* Ph.D. Dissertation. University of Waikato.

[6] Nadia FF Da Silva, Eduardo R Hruschka, and Estevam R Hruschka Jr. 2014. Tweet sentiment analysis with classifier ensembles. *Decision Support Systems* 66 (2014), 170–179.

[7] Chintan Dedhia and Jyoti Ramteke. 2017. Ensemble model for Twitter sentiment analysis. In *2017 International Conference on Inventive Systems and Control (ICISC)*. IEEE, 1–5.

[8] Mitali Desai and Mayuri A Mehta. 2016. Techniques for sentiment analysis of Twitter data: A comprehensive survey. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 149–154.

[9] Mehdi Emadi and Maseud Rahgozar. 2019. Twitter sentiment analysis using fuzzy integral classifier fusion. *Journal of Information Science* (2019), 0165551519828627.

[10] Crowd Flower. [n.d.]. *Data for everyone.* https://www.figure-eight.com/data-for-everyone/

[11] Manoochehr Ghiassi and S Lee. 2018. A domain transferable lexicon set for Twitter sentiment analysis using a supervised machine learning approach. *Expert Systems with Applications* 106 (2018), 197–216.

[12] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford* 1, 12 (2009), 2009.

[13] Michel Grabisch. 1995. A new algorithm for identifying fuzzy measures and its application to pattern recognition. In *Proceedings of 1995 IEEE International Conference on Fuzzy Systems.*, Vol. 1. IEEE, 145–150.

[14] Michel Grabisch. 2000. Fuzzy integral for classification and feature extraction. *Fuzzy Measures and Integrals: Theory and Applications* 1 (2000), 415–434.

[15] Michel Grabisch, Ivan Kojadinovic, and Patrick Meyer. [n.d.]. *Kappalab R library documentation.* https://cran.r-project.org/web/packages/kappalab/kappalab.pdf

[16] Stanford NLP Group. [n.d.]. *Class ColumnDataClassifier Java Docs.* https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/classify/ColumnDataClassifier.html

[17] Stanford Natural Processing Group. [n.d.]. *Stanford Classifier.* https://stanfordnlp.github.io/CoreNLP/

[18] Ammar Hassan, Ahmed Abbasi, and Daniel Zeng. 2013. Twitter sentiment analysis: A bootstrap ensemble framework. In *2013 International Conference on Social Computing*. IEEE, 357–364.

[19] Shinnosuke Himeno and Masaki Aono. 2017. Tweet polarity classification focused on positive and negative term frequency ratio. In *2017 International Conference on Advanced Informatics, Concepts, Theory, and Applications (ICAICTA)*. IEEE, 1–5.

[20] Soudamini Hota and Sudhir Phatak. 2018. KNN classifier-based approach for multi-class sentiment analysis of twitter data. *International Journal of Engineering and Technology* (2018).

[21] Mohammad Rezwanul Huq, Ahmad Ali, and Anika Rahman. 2017. Sentiment analysis on Twitter data using KNN and SVM. *IJACSA) International Journal of Advanced Computer Science and Applications* 8, 6 (2017), 19–25.

[22] Diana Inkpen. [n.d.]. *Information Retrieval Stop Words.* http://www.site.uottawa.ca/~diana/csi5180/StopWords

[23] Thorsten Joachims. [n.d.]. *SVM-Light Support Vector Machine.* http://svmlight.joachims.org/

[24] John D Kelleher, Brian Mac Namee, and Aoife D'Arcy. 2015. Fundamentals of Machine Learning for Predictive Analytics. , 267 and 400–410 pages.

[25] Vishal Kharde, Prof Sonawane, et al. 2016. Sentiment analysis of twitter data: a survey of techniques. *arXiv preprint arXiv:1601.06971* (2016).

[26] Patrick Killeen. [n.d.]. *Twitter sentiment analysis project source code.* https://github.com/patkilleen/twitter_sentiment_analysis

[27] Patrick Killeen. 2020. *Knowledge-Based Predictive Maintenance for Fleet Management.* Master's thesis. Université d'Ottawa/University of Ottawa.
[28] Akshi Kumar and Arunima Jaiswal. 2020. Systematic literature review of sentiment analysis on Twitter using soft computing techniques. *Concurrency and Computation: Practice and Experience* 32, 1 (2020), e5107.
[29] Ludmila I Kuncheva, Christopher J Whitaker, Catherine A Shipp, and Robert PW Duin. 2003. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications* 6, 1 (2003), 22–31.
[30] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. 55–60.
[31] Patrick Meyer. 2007. *Progressive Methods in Multiple Criteria Decision Analysis.* Ph.D. Dissertation. Springer.
[32] Kahlil Philander, Y Zhong, et al. 2016. Twitter sentiment analysis: Capturing sentiment from integrated resort tweets. *International Journal of Hospitality Management* 55, 2016 (2016), 16–24.
[33] Hassan Saif. 2015. *Semantic Sentiment Analysis of Microblogs.* Ph.D. Dissertation. The Open University.
[34] Hassan Saif, Yulan He, Miriam Fernandez, and Harith Alani. 2016. Contextual semantics for sentiment analysis of Twitter. *Information Processing & Management* 52, 1 (2016), 5–19.
[35] Boaz Shmueli. [n.d.]. *Multi-Class Metrics Made Simple, Part I: Precision and Recall.* https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2
[36] Nadia Felix F Da Silva, Luiz FS Coletta, and Eduardo R Hruschka. 2016. A survey and comparative study of tweet sentiment analysis via semi-supervised learning. *ACM Computing Surveys (CSUR)* 49, 1 (2016), 1–26.
[37] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.
[38] Stanford. [n.d.]. *Deeply Moving: Deep Learning for Sentiment Analysis.* https://nlp.stanford.edu/sentiment/
[39] Stanford. [n.d.]. *StanfordCoreNLP pipeline class Java documentation.* https://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/pipeline/StanfordCoreNLP.html
[40] Stanford. [n.d.]. *Using StanfordCoreNLP from the command line.* https://stanfordnlp.github.io/CoreNLP/cmdline.html
[41] Manju Venugopalan and Deepa Gupta. 2015. Exploring sentiment analysis on twitter data. In *2015 eighth international conference on contemporary computing (IC3)*. IEEE, 241–247.
[42] Jorge Villegas, Carlos Cobos, Martha Mendoza, and Enrique Herrera-Viedma. 2018. Feature selection using sampling with replacement, covering arrays and rule-induction techniques to aid polarity detection in Twitter sentiment analysis. In *Ibero-American Conference on Artificial Intelligence*. Springer, 467–480.
[43] Hao Wang and Jorge A Castanon. 2015. Sentiment expression via emoticons on social media. In *2015 ieee international conference on big data (big data)*. IEEE, 2404–2408.
[44] Wikipedia. [n.d.]. *Support Vector Machine.* https://en.wikipedia.org/wiki/Support-vector_machinel

A STANFORD CORENLP LIBRARY

This section contains a few examples on using the *Stanford CoreNLP* library via the command line.

A.1 Dataset Parsing and Formats

In order to run the classifiers offered by the *Stanford CoreNLP* library's command line API, the input datasets must be of the appropriate format expected by the library. The *Stanford CoreNLP* format is presented as follows:

- Each line in the file represents a data sample (a tweet in the present work's case)
- The label/class of the data sample is found at the beginning of the line
- The sample's data is found to the right of the label separated by a tab.

Table 13: The extended emoticon set used in some of the present work's experiments.

Positive	Negative	Positive	Negative
<3	:-(	0:-3	</3
:-)	)-:	O:-)	>.<
:)	:(	d:	S:
=)	:(	:b	:S
:D	=(	:-b	=L
:o)	)=	=p	:L
:]	>:[	:p	\=
:3	]:<	:-p	=\
:c)	:-c	:P	/=
:>	:c	:-P	=/
=]	:-<	>:P	\:
8)	>-:	;D	/:
=)	:<	;]	:/
:}	>:	;-]	/-:
:^)	:-[	;)	:-/
;^)	]-:	;-)	/:<
:-D	:[	:')	>:/
8-D	]:	:'-)	\:<
8D	:{	:-))	>:\
x-D	}:	:-J	D-':
xD	;(	3:)	v.v
X-D	);	3:-)	DX
XD	:-||	}:)	D=
=-D	||-:	|;-)	D;
=D	>:(	}:-)	D8
=-3	):<	>:-)	D:
=3	:'-(	>;)	D:<
B^D	)-':	>:)	)':
:-))	:'(	0:3	
0:-)		0:)	
0;^)			

The following is an example of a positive tweet that follows the above format, where '4' is the label that represents positive:

```
"4  I am happy. Hello World!"
```

Note that the double quotes are not part of data sample, they are simply included for presentation purposes.

A.2 Running the NLP-based Approach Over the Command Line

The command below will run the NLP-based approach using the *Stanford CoreNLP* library's command line interface:

```
  java -cp ./lib/stanford-core-nlp/*
edu.stanford.nlp.pipeline.StanfordCoreNLP
-annotators tokenize,ssplit,pos,lemma,parse,sentiment
-file ./input/data.txt
-outputDirectory ./output/
```

, where each sentence in the file `./input/data.txt` will be parsed and their sentiment will be predicted. The output will be

written to the file `./output/inputFilePath.txt.xml` in XML format, and `./lib/stanford-core-nlp/*` is the class path to all the *Stanford CoreNLP* library's JAR files.

Note that instead of using an input file, the command line's standard input can be used instead; that is, the user can interact with the library via the command line by typing sentences (delimited by '.', '!', and '?', for example). In this case the sentences' sentimental information (in XML format) will be output to the standard output stream of the library (which will be displayed on the command line in this example) instead of being output to a file. This can be done using the command below:

```
java -cp ./lib/stanford-core-nlp/*
edu.stanford.nlp.pipeline.StanfordCoreNLP
-annotators tokenize,ssplit,pos,lemma,parse,sentiment
-stdIn
```

, where the `-file` and `-outputDirectory` arguments are omitted, and the `-stdIn` argument specifies the input will come from the standard input stream.

A.3 Running MaxEnt Over the Command Line

To run the MaxEnt classifier using the *Stanford CoreNLP* library's command line interface, the command below can be run:

```
java -cp ./lib/stanford-core-nlp/*
edu.stanford.nlp.classify.ColumnDataClassifier
-prop ./input/config.prop
-trainFile ./input/traindata.txt
-testFile ./input/testdata.txt
```

, where `./lib/stanford-core-nlp/*` is the class path to all the *Stanford CoreNLP* library's JAR files, `./input/config.prop` is the properties file used to specify additional command line arguments (for example, such as what feature to use), `./input/testdata.txt` is the test dataset, which is appropriately formatted (see section A.1 in the Appendix), and `./input/traindata.txt` is the training dataset file, which is also appropriately formatted. The output of this command is sent to the library's standard output stream. The prediction output for each tweet in the test dataset is displayed in the command line.

A.4 Running NB Over the Command Line

To run NB using the *Stanford CoreNLP* library's command line interface is very similar to running the MaxEnt classifier (which is detailed in section A.3 of the Appendix). The only difference is the additional `-useNB` argument. An example command is found below:

```
java -cp ./lib/stanford-core-nlp/*
edu.stanford.nlp.classify.ColumnDataClassifier
|-prop ./input/config.prop-trainFile ./input/traindata.txt
-testFile ./input/testdata.txt
-useNB
```

A.5 Running SVM Over the Command Line

To run the SVM classifier using the *Stanford CoreNLP* library's command line interface, the training (and testing, if 10-fold cross validation is not being used) datasets must be first converted from the library's appropriate format (see section A.1) into *SVMLight* format.

A.5.1 Converting the Training Data to SVMLight. To convert the format of the training data into *SVMLight*, the following command can be run:

```
java -cp ./lib/stanford-core-nlp/*
edu.stanford.nlp.classify.ColumnDataClassifier
-prop ./input/config.prop
-trainFile ./input/traindata.txt
-testFile  ./input/testdata.txt
-printSVMLightFormatTo ./output/train-data.svml
```

, where `./lib/stanford-core-nlp/*` is the class path to all the *Stanford CoreNLP* library's JAR files, `./input/config.prop` is the properties file used to specify additional command line arguments, `./input/testdata.txt` is the test dataset, which is appropriately formatted (see section A.1 in the Appendix), and `./input/traindata.txt` is the training dataset file that will be converted to *SVMLight* format, which is also appropriately formatted, and `./output/train-data.svml` is the output file that contains the training data parsed into *SVMLight* format. Note that this command also unnecessarily runs the MaxEnt classifier using the training and testing data, which is why the test dataset is required in the command. Although the test data is not necessary for the goal of converting the training data to *SVMLight*, it is required for the command to work.

A.5.2 Converting the Testing Data to SVMLight. Converting the format of the testing data into *SVMLight* is similar to the command (which is shown above) for converting the training data to *SVMLight*. The following command converts the test data into *SVMLight* format:

```
java -cp ./lib/stanford-core-nlp/*
edu.stanford.nlp.classify.ColumnDataClassifier
-prop ./input/config.prop
-trainFile ./input/testdata.txt
-testFile  ./input/testdata.txt
-printSVMLightFormatTo ./output/test-data.svml
```

, where the only two differences are that: a) the command argument `-trainFile ./input/testdata.txt` is specified, to indicate that we want to convert the test data into *SVMLight* format, and the command line argument `-printSVMLightFormatTo ./output/test-data.svml` is used to indicate that the output file that will hold the *SVMLight*-formatted test data is in the file `./output/test-data.svml`.

A.5.3 Running the SVM. Lastly, once the input datasets are converted into *SVMLight*, they can then be used to run the SVM via the command line by using the following command:

```
  java -cp ./lib/stanford-core-nlp/*
edu.stanford.nlp.classify.ColumnDataClassifier
-prop ./input/config.prop
-trainFile ./output/train-data.svml
-testFile  ./output/test-data.svml
-trainFromSVMLight true
-testFromSVMLight true
```

, where in this case the different arguments are: a) `-trainFile ./output/train-data.svml`, which specifies the *SVMLight*-formatted training data input file path, b) `-testFile ./output/test-data.svml`, which specifies the *SVMLight*-formatted testing data input file path, c) `-trainFromSVMLight true`, which specifies we are training from *SVMLight*-formatted training data, and d) `-testFromSVMLight true`, which specifies we are training from *SVMLight*-formatted testing data. The prediction output is sent to the library's standard output stream, and in this case is, it is therefore displayed on the command line.

B ONLINE RESOURCES

This section summarizes a few useful online resources that were helpful in the present work's implementation.

B.1 Stanford CoreNLP

- The *Stanford CoreNLP* library can be download from here [17].
- The web page [38] provides a demo on how to use the *Stanford CoreNLP* library's sentiment analysis module.
- The Java documentation of the command line interface of the `StanfordCoreNLP` class, which provides the API to sentiment analysis and is used to implement the NLP-based approach, can be found here [39].
- The Java documentation of the command line interface `ColumnDataClassif` class, which provides the API to run machine learning classifiers, can be found here [16].
- The website [40] provides a good tutorial on how to use the *Stanford CoreNLP* library via the command line.
- [30] provide a detailed guide on using the *Stanford CoreNLP* library.

B.2 Fusion Model and R Programming

- The *Kappalab* R programming library's documentation can be found here [15].
- [31] provides a good example on how to use the HLMS and CFI algorithm using the *Kappalab* package from the R programming library.

B.3 Other

- A tutorial on computing evaluation metrics for multi-class classifiers can be found here [35]. It provides many examples on how to compute recall, precision, f1-score, and other metrics using a confusion matrix.
- The source code used in the present work, which is open source, can be found here [26].