THESIS

SOCIAL MEDIA SENTIMENT ANALYSIS AND TOPIC DETECTION FOR SINGAPORE ENGLISH

by

Yee Ling Phua

September 2013

Thesis Advisor:	Craig Martell
Co-Advisor:	Pranav Anand

REPORT DOCUMENTATION PAGE				*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis		
4. TITLE AND SUBTITLE SOCIAL MEDIA SENTIMENT ANALYSIS AND TOPIC DETECTION FOR SINGAPORE ENGLISH				**5. FUNDING NUMBERS**
6. AUTHOR(S) Yee Ling, Phua				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A				**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release;distribution is unlimited				**12b. DISTRIBUTION CODE** A
13. ABSTRACT (maximum 200 words)				

Social media has become an increasingly important part of our daily lives in the last few years. With the convenience built into smart devices, many new ways of communicating have been made possible via social-media applications. Sentiment analysis and topic detection are two growing areas in Natural Language Processing, and there are increasing trends of using them in social media analytics. In this thesis, we analyze various standard methods used in supervised sentiment analysis and supervised topic detection on social media for Colloquial Singapore English. For supervised topic detection, we created a naïve Bayes classifier that performed classification on 5000 annotated Facebook posts. We compared the result of our classifier against open source classifiers such as Support Vector Machine (SVM), Maximum Entropy and Labeled Latent Dirichlet Allocation (LDA). For supervised sentiment analysis, we developed a phrasal classifier that analyzed the polarity of 425 argumentative Facebook posts. Our naïve Bayes classifier gave the best accuracy result of 89% for supervised topic detection on two-class classification and 57% accuracy for our six-class classification. For our supervised sentiment analysis, our phrasal sentiment analysis classifier obtained an accuracy of 35.5% with negative polarity class achieving a high precision of 94.3%.

14. SUBJECT TERMS Machine Learning, Topic Detection, Sentiment Analysis, Singapore English, Naive Bayes Classifier			**15. NUMBER OF PAGES** 91
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

SOCIAL MEDIA SENTIMENT ANALYSIS AND TOPIC DETECTION FOR SINGAPORE ENGLISH

Yee Ling, Phua
Civilian, ST Engineering, Singapore
B.CompSci, University of Adelaide, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2013

Author:	Yee Ling, Phua
Approved by:	Craig Martell Thesis Advisor
	Pranav Anand Co-Advisor
	Peter J. Denning Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Social media has become an increasingly important part of our daily lives in the last few years. With the convenience built into smart devices, many new ways of communicating have been made possible via social-media applications. Sentiment analysis and topic detection are two growing areas in Natural Language Processing, and there are increasing trends of using them in social media analytics. In this thesis, we analyze various standard methods used in supervised sentiment analysis and supervised topic detection on social media for Colloquial Singapore English. For supervised topic detection, we created a naïve Bayes classifier that performed classification on 5000 annotated Facebook posts. We compared the result of our classifier against open source classifiers such as Support Vector Machine (SVM), Maximum Entropy and Labeled Latent Dirichlet Allocation (LDA). For supervised sentiment analysis, we developed a phrasal classifier that analyzed the polarity of 425 argumentative Facebook posts. Our naïve Bayes classifier gave the best accuracy result of 89% for supervised topic detection on two-class classification and 57% accuracy for our six-class classification. For our supervised sentiment analysis, our phrasal sentiment analysis classifier obtained an accuracy of 35.5% with negative polarity class achieving a high precision of 94.3%.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ARFF	Attribute-Relation File Format
CSV	Comma-Separated Values
GUI	Graphical Use Interface
JSON	JavaScript Object Notation
LDA	Latent Dirichlet Allocation
MALLET	machine learning for Language Toolkit
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
TMT	Topic Modelling Toolkit
WEKA	Waikato Environment for Knowledge Analysis
XML	Extensible Markup Language

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to many individuals who have provided me with invaluable advice and encouragement during the development of this thesis.

I would like to express my appreciation to Professor Craig Martell and Professor Pranav Anand for their guidance, time and useful feedback through the learning process of this Master’s thesis.

I wish to thank my husband, Calvin, who is back home in Singapore, for his encouragement, patience and immense emotional support that keeps me motivated in my studies at the Naval Postgraduate School.

I would like to thank my mentor, Mr Kum Chee Meng, for his encouragement and invaluable advice to take up this course.

Finally, I would like to thank my sponsor, ST Engineering, for the scholarship to participate in this enriching experience at the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND OF RESEARCH

Social media has become an increasingly important part of our daily lives in the last few years. With the convenience built into smart devices, many new ways of communicating have been made possible via social-media applications.

Sentiment analysis and topic detection are two growing areas in Natural Language Processing (NLP), and there are increasing trends of using them in social media analytics. Many companies use sentiment analysis to mine information about what people think and feel about their products, while political organizations use it to gather information about parties the people support. Topic detection is another emerging trend in social media analytics, and marketing companies use it to find out the current subjects people are talking about and the emerging topics in which people are interested.

In Singapore, many people speak and write in a Colloquial Singapore English, also known as *Singlish*. *Singlish* is a mix of English, Mandarin and many other Chinese and Malay dialects. Because *Singlish* can be used informally and casually, it is commonly used in social media by Singaporeans. Due to the unique blend of multiple languages, features and functions of *Singlish* it has been researched and discussed in the area of Linguistics since the 1960s [1]. However, little research on *Singlish* has been done in Natural Language Processing.

In this research, we want to perform sentiment analysis and topic detection on *Singlish* Facebook posts that discuss a whitepaper on population sustainability issued by the government of Singapore. We want to find out how well standard sentiment analysis and topic detection tools perform on these social media data.

B. MOTIVATION

According to a 2012 report [2] made by ROCKPUBLICITY.COM, there were more than 3.5 million Singaporeans who used social media at least once a week. In 2012, the number of Singapore Facebook, Twitter and YouTube subscribers was 3.2 million, 2.5 million and 3.9 million, respectively.

Many people use social media as their main source of news and social awareness. In recent years, social media has become a popular platform for debates and discussions on elections as well as for opinion polling on political topics.

In this research, we focus on the government-issued document, *A Sustainable Population for a Dynamic Singapore: Population Whitepaper* [3], released in January 2013. The whitepaper discusses the forecast of population growth in Singapore and future actions the government might take to sustain the growth. Many opinions about it have been widely discussed in social media. For our research, we want to discover the topics being discussed and the sentiment of Singaporeans concerning the whitepaper.

C. RESEARCH QUESTION

Our research focuses on using a series of methods that are commonly used in sentiment analysis and topic detection and applying them to our *Singlish* dataset.

D. ORGANIZATION OF THESIS

The thesis is organized into the following chapters:

- Chapter I provides the background and motivation of the research.
- Chapter II discusses the prior and related works in sentiment analysis and topic detection.
- Chapter III discusses the methodologies, the experiment setup and data processing.
- Chapter IV explains experiment results and analysis of the results.
- Chapter V provides a summary and the possible future work.

II. PRIOR AND RELATED WORK

A. PRIOR WORK

Supervised machine learning is a common technique for analyzing social media. Two main areas of growth that are constantly being researched are supervised topic detection and supervised sentiment analysis. The most recent work on both sentiment analysis and topic classification were done respectively by Anta et al. [4] and Batista et al. [5], over Spanish tweets to find out how well the state-of-the-art methods used on English-based tweets work on these tweets. In [6], Narr et al. examined a language-independent sentiment analysis approach of tweets from four different languages (English, German, French and Portuguese) using semi-supervised classification. Results of this analysis showed that independent-language classifiers performed slightly better than the mixed language classifier.

Supervised machine learning involves classification of data using classifiers built from labeled training data. This training data is usually obtained through human intensive annotation. The more training data and accurate annotation is available, the better the performance of the classifier. In [7], Asur et al. used thousands of workers from Amazon Mechanical Turk to annotate the movie *Twitter* dataset of 2.89 million tweets for sentiment analysis. While some researchers created ways, such as heuristic techniques using emoticons to automatically label data in their work [6, 8, 9], others [10] chose to use pre-existing datasets such as the Edinburgh corpus [11] and the Stanford corpus [9] or commercial datasets like SearchMetrics GmbH and iSieve Technologies in their research.

Supervised topic detection is a kind of text classification in which a set of documents is analyzed and classified into topics to which they are related. Common techniques that use text classification for topic detection include naïve Bayes, Support Vector Machine (SVM) and Maximum Entropy. Researchers have developed various toolkits, like WEKA [12], MALLET [13] and NLTK, to facilitate experimentation.

Supervised topic detection has also been achieved through *topic modeling*. Ramage et al. [14] created Labeled LDA in Stanford's Topic Modeling Toolbox, which is a topic model that infers latent *topics* from user labeled data using the latent Dirichlet allocation (LDA) [15] technique. Labeled LDA allows multiple topics to be modeled for each document and constrains LDA by creating a one-to-one mapping between the LDA's latent topics and labels.

As social media has now become a common platform for communication, the topics that social networkers are discussing are ever changing. Topic detection has also been used to identify trending topics on social media. In [16], Lee et al. used both text-based modeling and network-based modeling in their approach towards Twitter trending topic classification. Asur et al. [17] studied the lifetimes of the topics that trended by examining general behavior of Twitter.

Sentiment analysis has often been used to identify attitudes of people towards certain products or political views. Pang and Lee [18] elaborated on a comprehensive literature about the various methods used in opinion mining and sentiment analysis. The most basic approach considers whether a document or a word or phrase within the document contains positive or negative sentiment. Other more complex approaches perform ranking of attitudes into more than two classes (i.e., "star" ratings) and tries to find the sources and targets of these attitudes.

The emoticons[1] dataset was used by Kouliumpis et al. [10] and Pak et al. [8] in their Twitter sentiment analysis. Emoticons provided a semi-supervised approach to labeling the documents in [9], and classifiers trained with these labels are able to achieve an accuracy of above 80%.

[1] Emoticons refer to a pictorial representation of emotions in a textual form e.g. :(, :)

Recent NLP work has revolved around Twitter as compared to Facebook. Twitter provides a more stringent platform due to its limitation of 140 characters. Because of this constraint, tweets are usually one sentence long making them easier to label. On the other hand, Facebook allows much longer posts, and because of the fluctuation in length and number of sentences within a post, it is more challenging to annotate. Some of the NLP work on Facebook includes [19] which used Stanford Classifier, Stanford Tagger and Stanford Topic Modeling Toolbox for sentiment analysis and [20] that performed real time opinion extraction and classification on Facebook posts using SVM.

B. RELATED WORK

1. Naïve Bayes Classifier

The naïve Bayes classifier is one of the simplest and most commonly used machine-learning algorithms for text classification. It uses a probabilistic approach based on Bayes' theorem with strong independence assumptions. It considers each feature that contributes to the probability independently regardless of the presence or absence of any other features.

Many projects [4, 6, 8, 9, 16] have used naïve Bayes as the first approach to text classification due to its simplicity. Tools like WEKA, MALLET and NLTK incorporate naïve Bayes as one of their machine learning classifiers for research evaluation.

In text classification, a naïve Bayes classifier first learns from a list of training documents for each class. Each document is treated as a *bag* of features. The frequency of each feature for each class is then calculated. The probability of each feature is the frequency of the feature over the total number of occurrences. When a test dataset is input to naïve Bayes, the probability of each feature in each test document is matched against that of trained models. The probability of each class is then calculated based on these models.

Each class has a prior probability. The class prior is a known probability of the class based on the previously observed features. It is defined as the count of the number of items in the class divided by the total number of items in the training set.

$$P(Class) = \frac{Count(W \mid Class)}{Count(W)} \tag{2.1}$$

For each document, the probability of the document coming from a class is calculated based on the all features in that document. The probability of each class given a set of features is defined as the multiplication of the class prior times the product of probabilities of features given a class over the product of probabilities of all features in the classifier.

$$P\left(\text{Class}|\text{W}_{1,}\text{W}_{2,}\text{W}_{3,\ldots}\text{W}_{n}\right) = \frac{P\left(\text{Class}\right)\prod_{i=1}^{n} P(\text{W}_i \mid \text{Class})}{\prod_{i=1}^{n} P(\text{W}_i)} \tag{2.2}$$

The probabilities of the classes for each document are then compared to provide the most likely class for that document. The *argmax* function is used to determine which class contains the highest probability. The product of probabilities of all features in the classifier is dropped from the denominator in the *argmax* function because it is constant across all classes, thus there is no impact in the calculation.

$$\begin{aligned} &\arg\max\text{C}\left[\text{P(Class} \mid \text{W}_{1,}\text{W}_2, \text{W}_3, \ldots \text{W}_n)\right] \\ &= \arg\max\text{C}\left[\frac{P(Class)\prod_{i=1}^{n} P(W_i \mid Class)}{\prod_{i=1}^{n} P(W_i)}\right] \\ &= \arg\max\text{C}\left[P(Class)\prod_{i=1}^{n} P(W_i \mid Class)\right] \end{aligned} \tag{2.3}$$

For a feature that is not observed in the training data for a particular class, the probability of its occurrence is zero. Hence the probability of the class will end up being zero if such a feature occurs. This would cause the classifier to ignore all other features because of this rarely occurring feature. Smoothing techniques are used to help mitigate such problems. A popular smoothing technique that is commonly used [9] is the Laplace or Add-one smoothing. This technique simply adds one or α value to the probability of each feature such that the each probability will not end up with zero. In the following equation, α is defined as 0 < α ≤ 1 and V is the total number of vocabulary in the corpus.

$$\mathrm{P}_{laplace}\left(\mathrm{w}_i | \mathrm{w}_{i-1}\right) = \frac{\alpha + \mathrm{c}\left(w_{i-1} w_i\right)}{\alpha |V| + \sum_{wi} \mathrm{c}\left(w_{i-1} w_i\right)} \tag{2.4}$$

Another important smoothing technique is called Witten Bell. In Witten-Bell smoothing, two equations are used.

- If the count of a feature in the training data is 0,

$$P_{wittenbell}\left(W_i\right) = \frac{T}{Z\left(N+T\right)} \tag{2.5}$$

- If the count of the feature in the training data is greater than 0,

$$P_{wittenbell}\left(W_i\right) = \frac{C\left(W_i\right)}{N+T} \tag{2.6}$$

where

- T is the number of different feature types that are observed.
- N is the total number of occurrences of all features.
- Z is the estimate of the number of words in the evaluation dataset that are not observed in the training data

In our experiments for topic detection, we developed a multi-class naïve Bayes classifier to predict the topics on *Singlish* Facebook posts pertaining to the Singapore whitepaper.

2. Features

In machine learning, we need to determine the types of attributes that can best describe the data. Feature engineering is the process of deducing the best set of features that can be used to maximize prediction. There are many different types of features that can represent document in a text classification. In [10], n-gram, part of speech (POS) and lexicon were used as features for Twitter sentiment analysis. In [18], Pang et al. provided a comprehensive description of various types of features, including syntax and negation.

N-gram models are commonly used in text classification for the prediction of the next item in a continuous sequence of text. N-grams that are most commonly used in text classification are unigrams, bigrams and trigrams. Unigrams represents each individual character or word in a given text. Bigrams represents a two character or word slice within the given text. Given a text string of “The brown fox jumped over the lazy dog,” the character bigrams are “Th,” “he,” “e ,” “ b,” “br,” and so on, while the word bigrams are “The brown,” “brown fox,” “fox jumped,” and so on. Likewise, a trigram is a three character or word slice and, in general, n-grams are n-characters or word slices. One use of n-gram models is that we can measure the similarity between two strings by counting the number of n-grams that are common to them.

We use the phrase *lexicon features* to mean words that have polarity sentiments. Lexicon features are commonly used in sentiment analysis where there exist lists of positive, neutral or negative lexicon words that are used in classification. There are many sentiment-lexicon resources available, such as MPQA Subjectivity Lexicon[2] and Opinion Lexicon[3].

3. Support Vector Machine and Maximum Entropy

Support Vector Machine (SVM) is a supervised machine learning algorithm that is commonly used in classification and regression analysis. It works on the concept of finding an optimal hyper plane which separates all data points of one class from those of the other class.

[2] A list of positive and negative words differentiated by strong and weak subjectivity created by Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Please refer to [27].

[3] A list of positive and negative opinion words for English created by M. Hu and B. Liu. Please refer to [28].

Maximum Entropy is another supervised machine learning technique that learns probability distribution from the training data set. As opposed to naïve Bayes classification, it does not assume independent features and probability distribution other than the features that are observed. It will select the best probability distribution based on the observed features.

For comparison, we used Support Vector Machine from WEKA [12] and Maximum Entropy from MALLET [13]. Please refer to [21] and [22] for further discussion on these common techniques.

4. Labeled LDA

Labeled LDA is a topic model algorithm that was created by Ramage et al. 14] as part of the Stanford's Topic Modeling Toolbox. It is a supervised variant of latent Dirichlet allocation, which was created by Blei et al. [15], to infer topics from labeled data. Labeled LDA introduces supervision by constraining the model only to topics that are observed in the labeled dataset. A one-to-one mapping is created between the LDA's latent topics and labels, so that Labeled LDA can learn directly from these sets of words that go with the particular topic. Labeled LDA also allows multiple topics to be modeled for each document.

The graphical representation of the Labeled LDA model is presented in Figure 1.

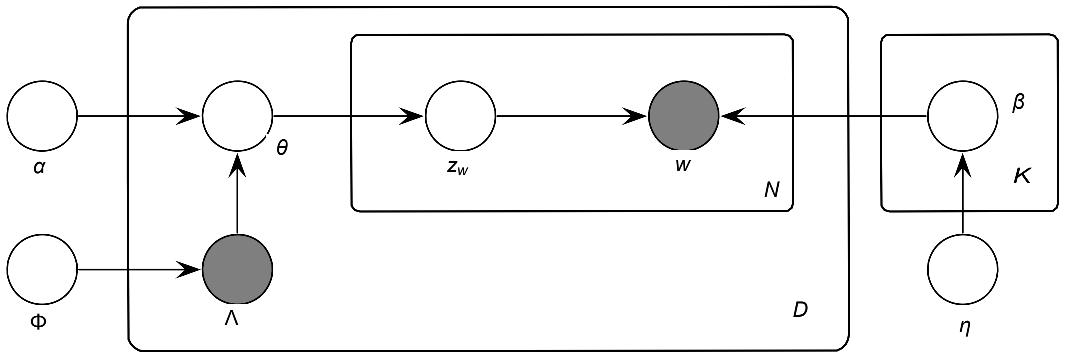


Figure 1. Graphical model of Labeled LDA.

In the Labeled LDA model in Figure 1,

- D refers to each document.
- w refers to each word in the document.
- N refers to number of words in the document.
- K refers to the number of topics.
- β refers to per-word multinomial distribution over the vocabulary in the corpus.
- Λ refers to the labeled dataset.
- η is the symmetric Dirichlet word prior.
- α is a symmetric Dirichlet topic prior.
- θ refers to the per-document multinomial distribution over only the topics in Λ.
- Φ is the label prior for each topic.
- z_w refers to the word-topic assignment of each document over θ and β.

In [6a] where the experiment was performed using del.icio.us corpus of tagged web pages, Labeled LDA outperforms SVM by more than three times.

5. Synthetic Minority Oversampling Technique (SMOTE)

In many real world applications, data that are mined tend to be skewed or imbalanced. Research [23, 24] has shown that imbalanced training data has a greater effect on the classifier. Data in the minority class may contain an important feature or event but because of its infrequency, the classifier is not able to learn the concept related to it. The classifier created will be biased and produce skewed results of low accuracy for the minority class but high accuracy for the majority classes.

There are many studies [23, 24] that discuss the various methods to balance the data by boosting the minority class. One of the methods is to collect and annotate more training data for the minority class. This method is the most effective, but it is also the most costly. Other methods include creating data by oversampling or undersampling the existing training dataset. In [25], Chawla et al. introduced a method called Synthetic Minority Over-sampling Technique that synthetically creates extra training data by oversampling the real data of the minority class.

In the SMOTE algorithm, a synthetic example $s=(s_1, s_2, \ldots, s_n)$ is created from an original data point, $d=(d_1, d_2, \ldots, d_n)$, where $(x_1, x_2, \ldots, x_n)$ indicates the representation of a data point in an n-dimensional feature space. For each synthetic feature, s_i, one of *d*'s k nearest neighbors, *nn*, is chosen, and $s_i=a*(d_i-nn_i)$, where *a* is a random number between 0 and 1.

A C# version of the SMOTE algorithm is implemented in our experiment to boost our minority class.

6. Phrasal Contextual Classifier

In [26], Harihara et al. developed a dual contextual sentiment analysis classifier that looked into identifying sentiments of a word or phrase in Twitter posts instead. Two classifiers, one for words and the other for phrases, were built to evaluate the polarity of text surrounding these targets. Different window sizes that contained the contextual words were evaluated for the different n-grams. A lexicon of positive and negative words and a list of emoticons were also used to classify the tweets.

In our research, we looked into developing a similar phrasal contextual classifier using window size and lexicon list to evaluate the sentiments of our *Singlish* Facebook posts.

7. Performance Measurement

We used the following performance metrics to evaluate our experiment results and our classifiers.

a. Confusion Matrix

A confusion matrix is used as a form of visualizing the performance of a classifier. It is displayed in a table format in which the columns represent the actual values (true and false) and the rows represent the predicted values (positive and negative). It can easily be generalized for multi-class classifiers. The table reports the results of a classifier in terms of the number of true positives (tp), false positives (fp), false negatives (fn) and true negatives (tn).

	Truth	
Labeled	*tp*	*fp*
	fn	*tn*

Table 1. Confusion Matrix.

b. Accuracy, Precision, Recall, F-Score

The four types of measures that are commonly used in machine learning as a result of confusion matrix are *accuracy*, *precision, recall* and *F-score*. Accuracy is defined as the percentage of correct predictions over the total sample size. Precision is defined as the percentage of positive predictions that are correct. Recall is defined as the percentage of actual positives that are labeled as positive.

$$Accuracy = \frac{tp + tn}{tp + fp + tn + fn} \tag{2.7}$$

$$Precision = \frac{tp}{tp + fp} \tag{2.8}$$

$$Recall = \frac{tp}{tp + fn} \tag{2.9}$$

F-score is the harmonic mean of precision and recall.

$$F\text{-}score = 2 \times \frac{Precision \text{ x } Recall}{Precision + Recall} \tag{2.10}$$

III. EXPERIMENT SETUP

Our experiment covered two main areas of machine learning, and it was broken into two parts, supervised topic detection and supervised sentiment analysis of Facebook posts. We wanted to know if there were signals in our *Singlish* dataset using various methods of topic detection and sentiment analysis and how well these methods perform in the dataset.

We first looked into supervised topic detection where we created a naïve Bayes classifier to perform topic classification. Other classifiers such as SVM from WEKA, Maximum Entropy from MALLET and Labeled LDA from Stanford TMT were also used to check against the performance of our naive Bayes classifier.

The Singapore population white paper was prepared using feedback from public discussions and dialogue sessions. Through these public discussions and dialogue sessions, a total of seven topics where categorized, and they include

- Marriage and Parenthood,
- Singaporeans Abroad,
- Integration and Identity,
- Immigration,
- Cost of Living, Social Support,
- Economy and Workforce and
- Livability, Environment, Land Planning

In order to prepare for our dataset and supervised topic detection, we identified six topics from the above topics. Singapore Aboard was removed from our experiment because it only constituted 1% of the feedback received from public discussions and dialogue sessions. Hence, we concluded that it would not be widely discussed in social media.

Additionally, we segregated out those posts that our annotators thought were not argumentative, but simply expressions of sentiment. We labeled this class *Pure Polarity*. The remaining posts from the six topics were categorized as *Argumentative*. The posts from the argumentative category were then used for sentiment analysis and run through a lexical classifier to determine if the author of each sentiment post was giving a positive or negative comment. Sections B.1 and C.1 further elaborate on the topics that were identified for our experiment.

Figure 2. shows a simple flow of how we performed our experiments.

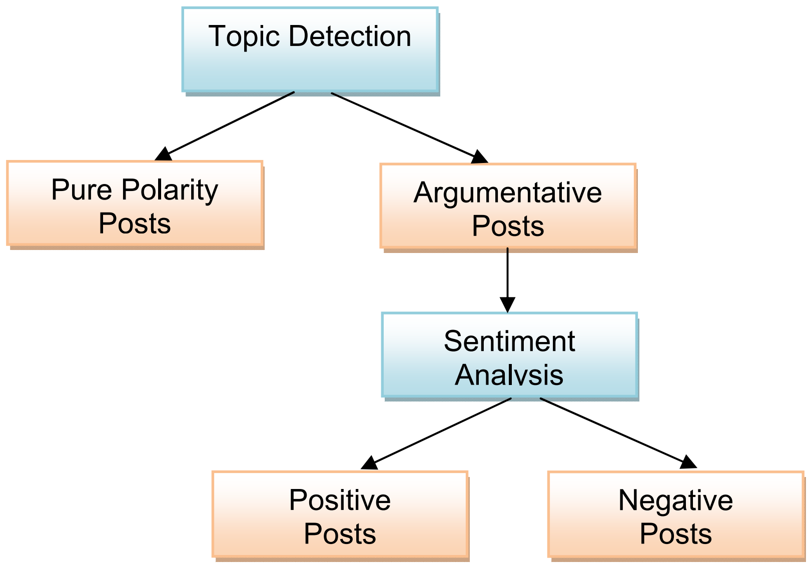


Figure 2. Experiment Flow.

A. DATA COLLECTION

In order to have a diversified mix of posts, we collected the posts over seven Facebook pages, out of which three were from the news and media, two belonged to a political party and two were from community pages. A total of 2237 posts were gathered. Note that all of the posts were from publically available pages.

B. TOPIC DETECTION

1. Pre-processing of Data

We first annotated the Facebook messages into these categories:

- Marriage and Parenthood,
- Integration and Identity,
- Immigrant,
- Cost of Living and Social Support,
- Economy and Workforce,
- Livability, Environment & Land,
- Pure polarity,

Examples of our annotated Facebook posts in their respective categories are shown in the Table 2. Table 3. shows the number of posts for each topic.

Topics	Facebook posts
Marriage and Parenthood	“More profamily bosses will be go. Then women can stay continue to work after hvg children. My ex boss will give me black face when I take leave to look after my kiddo when he was sick. It's always difficult coz you will get torn between home n work.”
Integration and Identity	“Singapore is like rojak to me now.”
Immigrant	“Foreigners are working at all levels now lah. No longer just jobs we don't wanna do. Even aunties get their jobs taken.”
Cost of Living and Social Support	“I earn less than 2K a month after deducting my CPF and I am the only person working in my family. I guess I better off dead than getting old and convert my status from citizen to slave.”
Economy and Workforce	“Long working hours low pay is not healthy & productivity.”
Livability Environment & Land	“I work in town and move around a fair bit during off peak hours as well. I feel the the trains and buses during off peak hours are not packed at all unlike during peak hours.”
Pure Polarity	“standing against the white paper...”

Table 2. Examples of Annotated Facebook Posts in Their Respective Categories.

Topics	Number of Posts
Marriage and Parenthood	61
Integration and Identity	136
Immigrant	233
Cost of Living and Social Support	378
Economy and Workforce	267
Livability, Environment & Land	255
Pure Polarity	907

Table 3. Number of Facebook Posts for Each Topic.

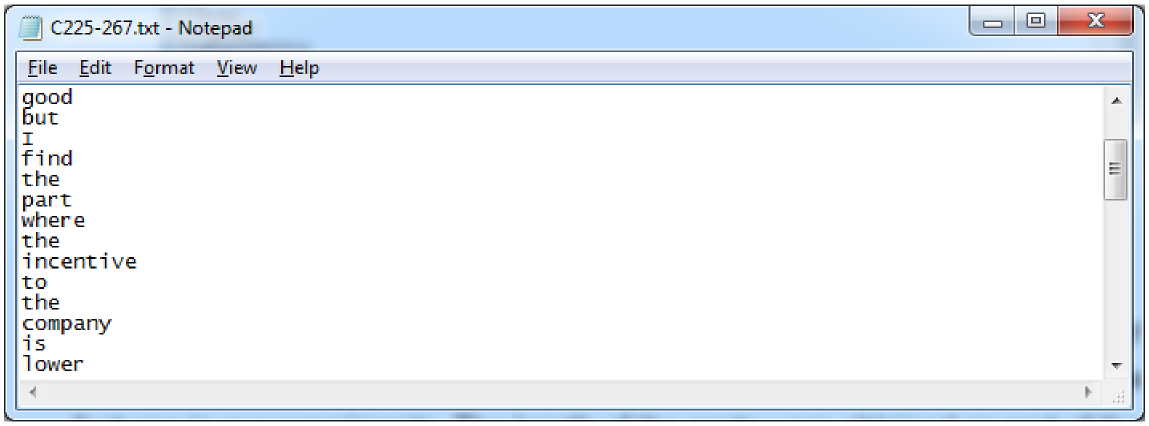
2. SMOTE

Due to our small annotated dataset, there were some topics that contained fewer data as compared to others. This caused the result of our classifier to be skewed towards the majority class. Hence, we implemented the SMOTE technique, as described in Chapter II.B.5, in our pre-processing to generate synthetic data for our minority class so as to determine how much better the classifier could perform if the data were balanced. The SMOTE technique increased the number of posts by using the real data in the minority class. This increased the number of token occurrences for that class yet retained the number of observed features in it.

A C# version of the SMOTE algorithm [25] was developed using k-Nearest Neighbors algorithm from the Accord.NET API [29] and Fisher–Yates Shuffle [30] techniques.

3. Tokenization

To test if different n-grams had impact on our classifier, the 2237 posts were tokenized into four different types of datasets, namely the word-unigrams, word-bigrams, word-trigrams and character-trigrams. A file generator application was written to split the characters in the posts into the desired n-gram type. The input file containing the posts was put into CSV format and passed through the application to generate one text file per post. Punctuation was removed from posts. The tokens for each post were separated into lines in each text file. Figure 3. shows an example of a tokenized post in text file format.



```
good
but
I
find
the
part
where
the
incentive
to
the
company
is
lower
```

Figure 3. Example of Tokenized Post.

4. Entropy Analysis

In our experiment, we used entropy to determine the usefulness of words in our classification. Some words appeared to be "noise" and they did not help in describing the contents of the posts. These noisy words might be too rarely or frequently occurring, or they had the same number of occurrences in the topics, hence cancelling out the effect on the classification. As a result, we used entropy as a measurement of information content of the words in our dataset to determine the words that had same effects or the same number of occurrences in each topic. We calculated entropy using the following equation that was defined in the information theory. For every distinct word in our training dataset, we determined probability of its occurrence in each topic. We used this equation to determine the entropy of the word using the probability generated for each topic.

$$H(x) = -\sum_{i=0}^{n} P(x_i)\log_2 P(x_i) \tag{3.1}$$

For words that occurred the same number of times in each topic, the probability of the word in each topic would be the same. Hence H(x) would be summed up to log_2 of the number of classes. Since we were looking at just two classes for this experiment, we created a list that contained all these words with H(x) = 1 and excluded them in our classification.

5. Naïve Bayes Classifier

We wrote a naïve Bayes classifier using C# for our experiment. A Graphical Use Interface (GUI), shown Figure 4. was created to facilitate our different test setups.

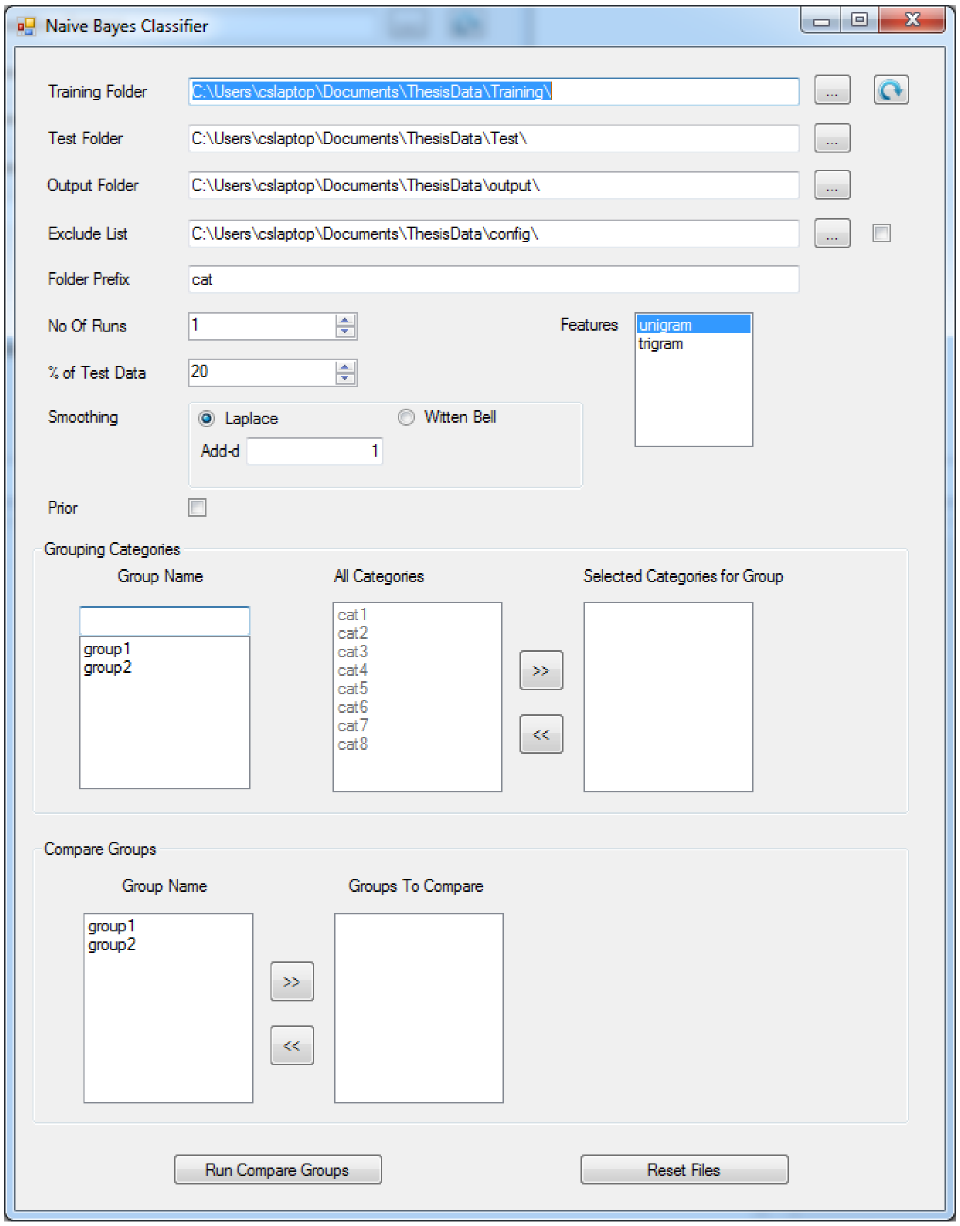
In our experiment, we used the hold-out method where a portion of the data annotated would be set aside as test data while the rest is used as our training data. Using annotated test data helps us determine the accuracy of our classifier. Our GUI allowed us to specify the percentage of data used for the testing. The GUI was also created with the options to specify the number of repeated hold-out runs for each experiment, the choice of smoothing technique (Laplace and Witten Bell), the α value for Laplace smoothing technique and the choice to include class prior.

In our setup, our data are placed in folder under the Training Folder directory. The following are the mappings of topics to folder names.

Topics	**Folder Name Mapping**
Marriage and Parenthood	cat1
Integration and Identity	cat2
Immigrant	cat3
Cost of Living and Social Support	cat4
Economy and Workforce	cat5
Livability, Environment & Land	cat6
Pure Polarity	cat7

Table 4. Topics to Folder Name Mapping.

The GUI allowed us to combine the data from two or more topics into one group, hence allowing us to perform different configurations of experiments. An example of our experiments was to test the prediction of pure polarity against argumentative posts. Group1 was selected for pure polarity, and it contained cat7 posts, while Group2 was used for argumentative, and it contained cat1 to cat6 posts.



Training Folder: C:\Users\cslaptop\Documents\ThesisData\Training\
Test Folder: C:\Users\cslaptop\Documents\ThesisData\Test\
Output Folder: C:\Users\cslaptop\Documents\ThesisData\output\
Exclude List: C:\Users\cslaptop\Documents\ThesisData\config\
Folder Prefix: cat
No Of Runs: 1
% of Test Data: 20
Features: unigram, trigram
Smoothing: [✓ Laplace] Witten Bell
Add-d: 1
Prior

Group Name: group1, group2
All Categories: cat1, cat2, cat3, cat4, cat5, cat6, cat7, cat8
Selected Categories for Group

Group Name: group1, group2
Groups To Compare

Figure 4. GUI for Naïve Bayes Classifier.

For each experiment setup, we perform the following steps:

a. We create the groups by selecting the topics to compare.

b. We select smoothing technique, Laplace or Witten Bell. We select Add-α value if Laplace smoothing is chosen.

c. We determine if class priors should be included.

d. We determine if stop-word or entropy list should be excluded.

e. We determine the number of repeated hold-out runs. Each run would randomly draw test data from the training set.

f. We determine the percentage of test data.

The classifier would then perform the following steps:

a. It determines vocabulary size based on n-gram type.

b. For each run of the experiment, it will randomly choose test data from training folder and put the data into the test folder based on the selected topics and percentage provided.

c. It trains the system by reading the tokens in each training file of each group.

d. It checks if the token exists in the group dictionary and increments the count of the token.

e. It populates the prior probability by using number of files in the group over the total number of files in training data.

f. It also populates the total count of features and token occurrences.

g. It performs testing by reading the tokens in each test file of each group.

h. It checks if any words are to be excluded and skips the token if it matches those words. No probability will be populated for that token.

i. It determines the smoothing technique selected.

j. If Laplace smoothing is used, it uses Equation 2.4.

k. If Witten Bell smoothing is used, it uses Equations 2.5 and 2.6.

l. It populates the probability for each token and product of the token's probabilities for each test file.

m. It then determines highest probability of each test file using the argmax function.

n. Finally, it generates the result of each test file for analysis.

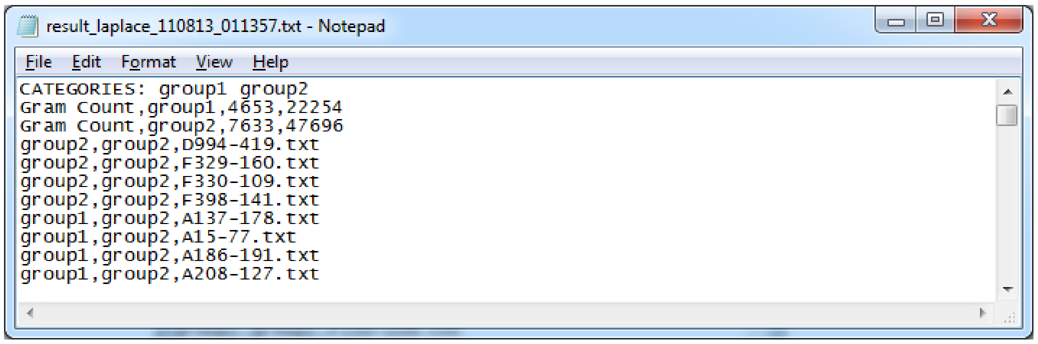
The probability of each post tends to get smaller after multiplying the probabilities of tokens together. Hence we used logarithmic probability (log-prob) in our algorithm to deal with the small probability issue.

The experiments that we would conduct for our naïve Bayes classifier included:

- Determining the baseline for pure polarity posts versus argumentative posts
- Determining the best α for Laplace smoothing
- Evaluating the results between Laplace and Witten Bell smoothing
- Determining the best n-gram to use for our experiment
- Determining the performance of SMOTE technique

6. Confusion Matrix

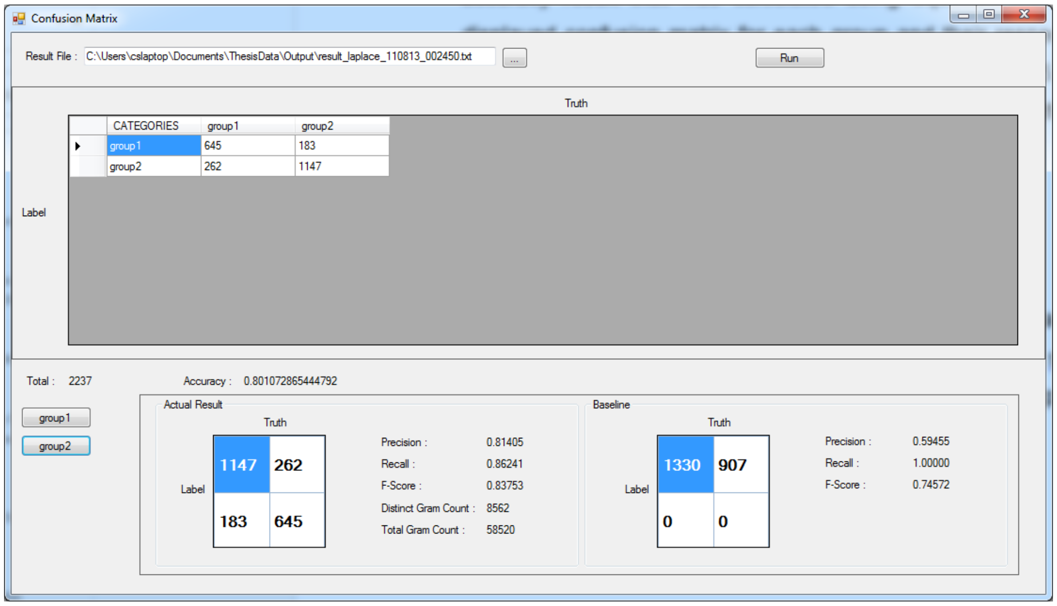
The output of our naïve Bayes classifier is generated in the format shown in Figure 5.


result_laplace_110813_011357.txt - Notepad
File Edit Format View Help

```
CATEGORIES: group1 group2
Gram Count,group1,4653,22254
Gram Count,group2,7633,47696
group2,group2,D994-419.txt
group2,group2,F329-160.txt
group2,group2,F330-109.txt
group2,group2,F398-141.txt
group1,group2,A137-178.txt
group1,group2,A15-77.txt
group1,group2,A186-191.txt
group1,group2,A208-127.txt
```

Figure 5. Output File of Naïve Bayes Classifier.

We developed a confusion matrix GUI that reads the output file from naïve Bayes classifier and displays the result in a table form. The table shows the confusion matrix table as described in Chapter II.B.7.a, where the truth is the column and label is the row. The GUI showed the number of files used for testing and accuracy result that was calculated using Equation 2.7. The GUI also displayed a confusion matrix for each group and their respective precision, recall and F-score using Equations 2.8, 2.9 and 2.10. The baseline for each group was also calculated to see how well the classifier performed for that set of data.



Result File : C:\Users\cslaptop\Documents\ThesisData\Output\result_laplace_110813_002450.txt

CATEGORIES	group1	group2
group1	645	183
group2	262	1147

Total : 2237

Accuracy : 0.801072865444792

Actual Result

1147	262
183	645

Precision :	0.81405
Recall :	0.86241
F-Score :	0.83753
Distinct Gram Count :	8562
Total Gram Count :	58520

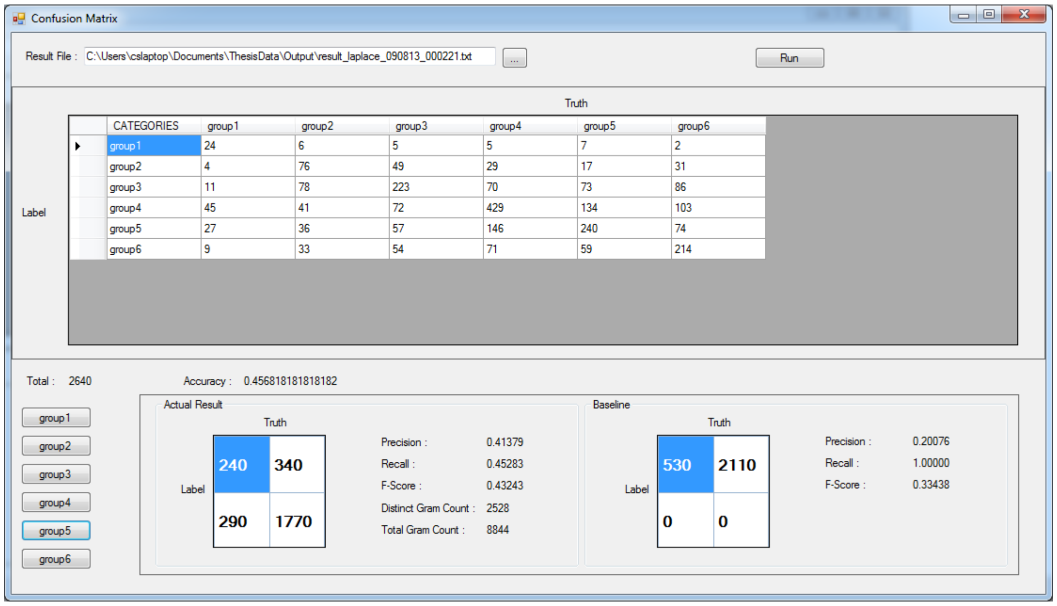
Baseline

1330	907
0	0

Precision :	0.59455
Recall :	1.00000
F-Score :	0.74572

Figure 6. Confusion Matrix GUI.

The confusion matrix was developed to allow classification results of multiple groups to be displayed dynamically. Figure 7. shows the confusion matrix GUI displaying six groups of topics based on the output file from the naïve Bayes classifier.



Result File : C:\Users\cslaptop\Documents\ThesisData\Output\result_laplace_090813_000221.txt

Truth / Label

CATEGORIES	group1	group2	group3	group4	group5	group6
group1	24	6	5	5	7	2
group2	4	76	49	29	17	31
group3	11	78	223	70	73	86
group4	45	41	72	429	134	103
group5	27	36	57	146	240	74
group6	9	33	54	71	59	214

Total : 2640   Accuracy : 0.456818181818182

Actual Result

Truth / Label

240	340
290	1770

Precision : 0.41379
Recall : 0.45283
F-Score : 0.43243
Distinct Gram Count : 2528
Total Gram Count : 8844

Baseline

Truth / Label

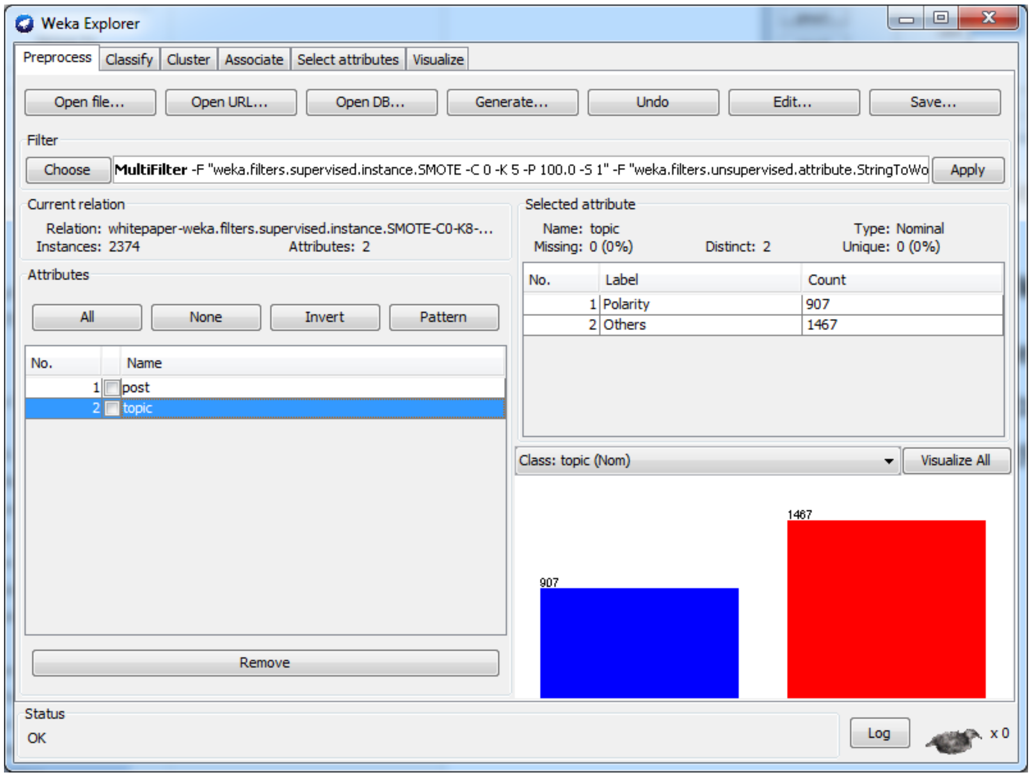
530	2110
0	0

Precision : 0.20076
Recall : 1.00000
F-Score : 0.33438

Figure 7. Confusion Matrix GUI Displayed with Six Groups of Topics.

7. SVM Using WEKA

We use WEKA to test Support Vector Machine (SVM) on our dataset to see if the SVM classifier could perform better than the naïve Bayes classifier. We first put our annotated dataset into the ARFF format that is accepted by WEKA. Using WEKA Explorer, shown on Figure 8. , we selected our input file and filters for pre-processing. The filters helped to convert the input file into the format that was accepted by the classifiers. We then performed the classification using LibSVM. WEKA also had the option to perform SMOTE on the pre-processed data; hence, we applied SMOTE in our experiments to determine the effects of imbalanced and balanced data had on the SVM classifier.



Filter

MultiFilter -F "weka.filters.supervised.instance.SMOTE -C 0 -K 5 -P 100.0 -S 1" -F "weka.filters.unsupervised.attribute.StringToWo

Current relation

Relation: whitepaper-weka.filters.supervised.instance.SMOTE-C0-K8-...
Instances: 2374   Attributes: 2

Selected attribute

Name: topic   Type: Nominal
Missing: 0 (0%)   Distinct: 2   Unique: 0 (0%)

No.	Label	Count
1	Polarity	907
2	Others	1467

Attributes

No.	Name
1	post
2	topic

Class: topic (Nom)

Status
OK

Figure 8. WEKA Explorer.

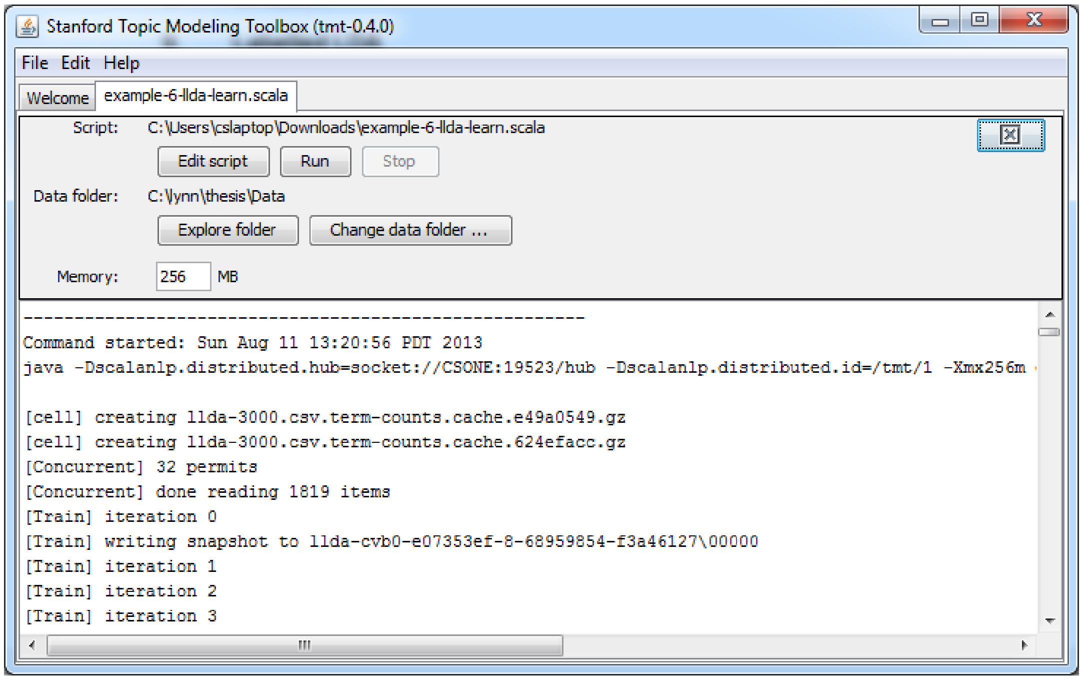
8. Maximum Entropy using MALLET

We used MALLET to perform Maximum Entropy Classification on our dataset to see how well it performed against our naïve Bayes classifier. MALLET took in the input files that we had prepared in our pre-processing and generated the files into the MALLET processing format. We first trained MALLET with the processed data, and then we chose the Maximum Entropy algorithm to perform the evaluation. MALLET had the option to split the processed data into training and test datasets and allowed the removal of stop-words.

9. Labeled LDA

Labeled LDA is a supervised version of the LDA that was created as part of the Stanford Topic Modeling Toolbox (TMT). We used Labeled LDA in our experiment to determine how different Topic Modeling is from the conventional term frequency classification of Naïve Bayes, SVM and Maximum Entropy methods.

We first divided our annotated dataset into training and test datasets and put them into individual CSV files. We then learnt from the training dataset by running the Labeled LDA script in the Stanford TMT. After the training dataset was populated, we ran the test dataset against the training dataset using an infer script. Figure 9. shows the GUI of the Stanford TMT.



Script: C:\Users\cslaptop\Downloads\example-6-llda-learn.scala
Data folder: C:\lynn\thesis\Data
Memory: 256 MB

```
-----------------------------------------------------
Command started: Sun Aug 11 13:20:56 PDT 2013
java -Dscalanlp.distributed.hub=socket://CSONE:19523/hub -Dscalanlp.distributed.id=/tmt/1 -Xmx256m

[cell] creating llda-3000.csv.term-counts.cache.e49a0549.gz
[cell] creating llda-3000.csv.term-counts.cache.624efacc.gz
[Concurrent] 32 permits
[Concurrent] done reading 1819 items
[Train] iteration 0
[Train] writing snapshot to llda-cvb0-e07353ef-8-68959854-f3a46127\00000
[Train] iteration 1
[Train] iteration 2
[Train] iteration 3
```

Figure 9. Stanford Topic Modeling Toolbox.

C. SENTIMENT ANALYSIS

We performed the sentiment analysis after we determined the performance of our topic detection classification. Our sentiment analysis focused on argumentative posts of our annotated data. We used the following six topics in our analysis to determine the polarity of the posts:

- Marriage and Parenthood,
- Integration and Identity,
- Immigrant,
- Cost of Living and Social Support,
- Economy and Workforce, and
- Livability, Environment & Land

1. Pre-processing of Data

Out of the 1330 argumentative posts from the six topics, we annotated 425 posts. As Facebook posts tend to be longer, there could be a mixture of positive and negative sentiments within them. Hence instead of the traditional way of finding polarity of a post as a whole, we looked for target phrases within them. In each post, we determined target phrases and the polarity of these target phrases based on the contextual sentiments around these target phrases.

In our annotation, we marked the target phrases using brackets and giving each target phrase a positive sign (+) or a negative sign (-) based on its sentiment. The following are some examples of the annotated posts:

> What everybody here wants is [-super congested roads].

> it's time to [+attract better entrepreneurs to reshape SME]. [+more employment opportunities n wages reform] will benefit more locals to be employed..

The annotated posts were stored in a CSV file and used as input into our classifier.

2. Lexicons

In order to determine the contextual sentiments surrounding our target phrases, we needed to have a list of positive and negative words. We used the list of opinion lexicons from Bing Liu in our classifier to determine the polarity of the words within and surrounding the target phrases.

3. Sentiment Analysis Classifier

We used unigrams in our sentiment analysis, and we determined the window size at which we would take the surrounding unigrams of the target phrases into account. We performed experiments using two window sizes to test how the surrounding unigrams affect the classification. For the first window size, we took all unigrams surrounding the target phrases into account while for the second window size, we used the mean of the count of unigrams between the target phrases.

An example of the annotated post was shown as followed.

> How does [+minimum wage] even equate to job loss? If anything it would encourage [+more jobs and more productivity] within it because people in those jobs will feel better [+being paid more] than before.

For the first window size, we first determined the target phrase to be “minimum wage.” We then took into account the words from the start to end of the post including all other target phrases.

For the second experiment, we used the following equation to determine our window size for each post. In the above example, we defined our window size to be six using the equation:

$$\bar{x} = \frac{\begin{array}{l}\text{count of x from start of post to first target phrase +}\\ \text{count of x from last target phrase to end of post +}\\ \sum \text{x between target phrases}\end{array}}{\text{total number of interval between target phrases and start and end of post}} \qquad (3.2)$$

We created a phrasal sentiment analysis classifier using C#. The classifier first read and tokenized each post into target phrases and unigrams. Each target phrase had a set of positive and negative bins for counting the unigrams. Based on the type of window size chosen, we read the unigrams within and surrounding the target phrase. For each unigram, we compared it against the positive and negative lexicons. For a match found in the positive lexicon, we incremented a count in the positive bin. Likewise, if a match was found in negative lexicon, the negative bin count would be incremented. For unigrams with no match in either of the lexicon lists, we termed them as neutral, and they were ignored in our experiment.

After classification, each target phrase would generate a set of counts in the positive and negative bins. If the numbers of unigrams that fell under the positive bin was higher than that of the negative bin, then the target phrase would be classified as positive. Otherwise, it would be classified as negative. If there were equal numbers of positive and negative matches, then the target phrase would be classified as neutral.

The count result of the target phrases was put into a CSV output file, and we put the classified results against our annotated data to determine if there was a match in the polarity of the target phrases.

4. Confusion Matrix

We used confusion matrices to analyze our results from the sentiment analysis classifier. We calculated the accuracy, precision, recall and f-score from the confusion matrices based on Equations 2.8, 2.9 and 2.10, respectively. Since neutral polarity did not exist in our annotated data, for target phrases that were reported as neutral from our classifier, we would need to take them into account. We incorporated the neutral results into our false negative and true negative counts such that the count of target phrases for our experiments was correct.

5. Contextual Lexicon Tests

A word may have multiple meanings, and when it is used in different contexts it may have different meanings. A word may appear to be positive in a dictionary but when it is applied to a certain domain, it may turn out to be negative or neutral.

In our experiments, some words that appeared in the positive lexicons seemed to have effects if they were moved to the negative lexicons or taken out of the lexicons. We wanted to know what words had impact in our result for our Singapore white paper context. If these words were shifted from positive lexicons to negative lexicons, how much impact would that shift have on our results?

We modified our sentiment analysis classifier in such a way that every lexicon in the positive and negative lists was either shifted to the other list or deleted to test for neutral polarity. We shifted one lexicon at a time from the positive list to the negative list. We then performed the sentiment analysis classification and generated the accuracy and precision results for that lexicon. Our results are discussed in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS AND ANALYSIS

In our experiments, we wanted to determine if our *Singlish* dataset had signals good enough for classification. Using the various experiment setups as described in Chapter III, we examined our classification results for topic detection and sentiment analysis.

A. TOPIC DETECTION RESULTS

1. Naïve Bayes Classifier Results

We wanted to know the performance of our classifier on the two classes for 907 pure polarity posts versus 1330 argumentative posts. We ran the posts using unigrams with 10 repeated runs of the hold-out method, 80% for training data and 20% for test data. For these first tests, we were concerned with how well the features discerned the classes. Accordingly, we did not use class priors. The baseline for pure polarity versus argumentative posts is shown in Table 5. This baseline is important in determining how well our classifier performs for topic detection in these two categories.

Baseline	Pure Polarity Posts	Argumentative Posts
Accuracy	0.595	
Precision	0.407	0.593
Recall	1.000	1.000
F-score	0.578	0.745
Number of posts	907	1330

Table 5. Baseline for Topic Detection Experiments using 2374 Facebook Posts.

In order to determine the best α value for Laplace smoothing, we conducted experiments using five different α values on unigrams. Our result in Table 6. showed that α value of 0.001 gave the best accuracy, 74.2%, and relatively better F-scores for both pure polarity and argumentative posts as compared to the baseline.

		Laplace with no prior (α=1)	Laplace with no prior (α=0.1)	Laplace with no prior (α =0.01)	Laplace with no prior (α =0.001)	Laplace with no prior (α =0.0001)
Confusion Matrix		See Appendix A.A				
Accuracy		0.610	0.642	0.736	**0.742**	0.737
pure polarity Posts	Precision	**0.836**	0.812	0.759	0.719	0.709
	Recall	0.051	0.155	0.512	**0.601**	**0.601**
	F-score	0.096	0.260	0.612	**0.655**	0.650
argumentative Posts	Precision	0.604	0.627	0.727	**0.754**	0.752
	Recall	**0.993**	0.975	0.889	0.839	0.831
	F-score	0.751	0.764	**0.799**	0.794	0.790

Table 6. Experiments on Naïve Bayes Classifier using Different α on Laplace Smoothing. (Bolded entries are the best for each row.)

Using the Laplace smoothing α value of 0.001, we performed a classification against the Witten Bell smoothing. The result in Table 7. showed that Witten Bell smoothing performed better with an accuracy of 75.7% and improved F-scores for both categories.

We then applied class priors to classifications on both Laplace and Witten Bell smoothing, and it was noted that the accuracy results worsened by 0.5% and 1.4%, respectively.

An entropy exclusion list was created based on pure polarity and argumentative posts to remove words that did not have an impact on both categories. The list was then applied to both smoothing techniques. It showed a significant improvement in both smoothing techniques, with Laplace performing better than Witten Bell. Laplace smoothing had an increase of 4.9% in accuracy, as well as improved precision of 78.4% and 79.4% for pure polarity and argumentative posts, respectively. Witten Bell smoothing achieved an accuracy increase of 4% and precision of 81.3% and 77% for pure polarity and argumentative posts, respectively.

		Laplace (no prior)	Witten Bell(no prior)	Laplace (prior)	Witten Bell (prior)	Laplace (Entropy Exclude List	Witten Bell (Entropy Exclude List)
Confusion Matrix		See Appendix A.B					
Accuracy		0.742	0.757	0.737	0.743	**0.791**	0.783
Pure Polarity Posts	Precision	0.719	0.771	0.727	0.763	0.784	**0.813**
	Recall	0.601	0.573	0.567	0.536	**0.670**	0.607
	F-score	0.655	0.658	0.637	0.629	**0.723**	0.695
Argumentative Posts	Precision	0.754	0.751	0.742	0.736	**0.794**	0.770
	Recall	0.839	0.883	0.854	0.886	0.873	**0.904**
	F-score	0.794	0.812	0.794	0.804	**0.832**	**0.832**

Table 7. Experiments on Naïve Bayes Classifier with Laplace versus Witten Bell Smoothing. (Bolded entries are the best for each row.)

In order to find out which n-gram features can produce the best results, we performed classification on unigrams, word bigrams, word trigrams and character trigrams using both Laplace and Witten Bell smoothing. The entropy exclusion list was used, and the Laplace smoothing α value was set to 0.001. It was noted in Table 8. and Table 9. that the unigrams worked best for both Laplace and Witten Bell smoothing, while word trigrams produced the worst results in both cases. Character trigrams worked better than word bigrams in Laplace smoothing with a slight improvement of 0.7% in accuracy. An interesting observation was the high precision of 85.8% achieved by argumentative posts in word bigrams. In the case of Witten Bell smoothing, word bigrams performed better than character trigrams with an accuracy difference of 1.1%.

		word unigrams (no prior)	word bigrams (no prior)	word trigrams (no prior)	char trigrams (no prior)
Confusion Matrix		See Appendix A.C			
Accuracy		**0.791**	0.755	0.474	0.762
Pure Polarity Posts	Precision	**0.784**	0.657	0.433	0.769
	Recall	0.670	0.830	**0.948**	0.592
	F-score	0.723	**0.734**	0.594	0.669
Argumentative Posts	Precision	0.794	**0.858**	0.806	0.758
	Recall	0.873	0.703	0.150	**0.878**
	F-score	**0.832**	0.773	0.252	0.814

Table 8. Experiments on Naïve Bayes Classifier with n-grams using Laplace Smoothing. (Bolded entries are the best for each row.)

		word unigrams (no prior)	word bigrams (no prior)	word trigrams (no prior)	char trigrams (no prior)
Confusion Matrix		See Appendix A.C			
Accuracy		**0.783**	0.780	0.659	0.769
Pure polarity Posts	Precision	**0.813**	0.764	0.671	0.807
	Recall	0.607	**0.662**	0.317	0.567
	F-score	0.695	**0.710**	0.431	0.666
Argumentative Posts	Precision	0.770	**0.788**	0.656	0.754
	Recall	0.904	0.860	0.893	**0.968**
	F-score	**0.832**	0.822	0.756	0.823

Table 9. Experiments on Naïve Bayes Classifier with n-grams using Witten Bell Smoothing. (Bolded entries are the best for each row.)

As it was noted, the precision of the argumentative posts tended to be greater than that of the pure polarity posts because there were more Facebook posts collected for the argumentative category. The results for such an imbalanced dataset were usually skewed to the majority class, causing inaccuracy in the performance of the classifier. We hence performed the SMOTE technique to balance the dataset, and we wanted to verify if boosting the feature space on the minority class helped in improving our classifier.

We tested on different percentages of synthetic samples created by SMOTE for the minority class to find out their impacts on our classifier. By creating 100% synthetic samples would mean that the number of documents in pure polarity category would be doubled from 907 to 1814. Table 10. shows the results on original dataset, 45%, 65% and 100% increase of synthetic samples for minority class created by SMOTE, respectively.

		0% SMOTE		45% SMOTE		65% SMOTE		100% SMOTE	
		Laplace (no prior)	Witten Bell (no prior)	Laplace (no prior)	Witten Bell (no prior)	Laplace (no prior)	Witten Bell (no prior)	Laplace (no prior)	Witten Bell (no prior)
Confusion Matrix		See Appendix A.D							
Accuracy		0.791	0.783	0.851	0.852	0.869	0.875	0.884	**0.890**
Pure Polarity Posts	Precision	0.784	0.813	0.878	0.886	0.898	0.907	**0.915**	0.913
	Recall	0.670	0.607	0.814	0.806	0.850	0.851	0.881	**0.896**
	F-score	0.723	0.695	0.845	0.844	0.873	0.878	0.898	**0.904**
	Total distinct count	4732	4650	5135	5092	5356	5287	5460	5558
	Total gram count	23187	22092	39007	38506	48179	46358	52329	54331
Argu-mentative Posts	Precision	0.794	0.770	0.827	0.823	0.839	0.843	0.845	**0.861**
	Recall	0.873	**0.904**	0.887	0.897	0.891	0.901	0.887	0.883
	F-score	0.832	0.832	0.856	0.858	0.865	0.871	0.865	**0.872**
	Total distinct count	7647	7650	7151	7306	7715	7582	7664	7498
	Total gram count	47524	47588	43790	44563	48569	46844	47745	46650

Table 10. Experiments on Naïve Bayes Classifier using SMOTE Technique to Boost Minority Class. (Bolded entries are the best results for Accuracy, Precision, Recall and F-score.)

Our results showed an increasing accuracy as more synthetic samples were created for the minority class. Our classifier created with 100% synthetic samples for minority class using Witten Bell smoothing and applying the entropy excluding list achieved a high accuracy of 89%. The precision for both pure polarity and argumentative posts also gave high scores of 91.3% and 86.1%, respectively. With 45% synthetic samples applied on SMOTE, the number of documents in both categories was balanced, creating a 6.9% increase from the original dataset. Using 65% synthetic samples, we were able to balance the number of occurrences, hence creating 1 to 2% increase in accuracy. It was also observed that the precision in both categories increased as the number of synthetic samples increased.

We also performed classification on the six argumentative topics using our naïve Bayes classifier to find out how well it fared. The baseline for the six argumentative topics is shown in Table 11.

Baseline		With no prior
Accuracy		0.284
Marriage and Parenthood	Precision	0.045
	Recall	1
	F-score	0.087
Integration and Identity	Precision	0.102
	Recall	1
	F-score	0.186
Immigrant	Precision	0.174
	Recall	1
	F-score	0.298
Cost of Living and Social Support	Precision	0.284
	Recall	1
	F-score	0.442
Economy and Workforce	Precision	0.200
	Recall	1
	F-score	0334
Livability, Environment & Land	Precision	0.193
	Recall	1
	F-score	0.324

Table 11. Baseline of Naïve Bayes Classifier on the Six Argumentative Topics.

		Laplace (no prior)	Witten Bell (no prior)
Confusion Matrix		See Appendix A.E	
Accuracy		0.571	**0.578**
Marriage and Parenthood	Precision	0.168	0.176
	Recall	**0.833**	0.783
	F-score	0.280	0.288
Integration and Identity	Precision	0.579	0.537
	Recall	0.519	0.452
	F-score	0.547	0.496
Immigrant	Precision	0.723	0.702
	Recall	0.511	0.476
	F-score	0.599	0.567
Cost of Living and Social Support	Precision	**0.816**	0.803
	Recall	0.545	0.581
	F-score	**0.654**	0.674
Economy and Workforce	Precision	0.776	0.766
	Recall	0.557	0.574
	F-score	0.648	0.656
Livability, Environment & Land	Precision	0.548	0.557
	Recall	0.643	0.686
	F-score	0.592	0.615

Table 12. Results of Naïve Bayes Classifier on the Six Argumentative Topics. (Bolded entries are the best results for Accuracy, Precision, Recall and F-score.)

We applied the entropy exclusion list on both Laplace and Witten Bell smoothing, and the result of the classification is shown in Table 12. We were able to achieve an accuracy of approximately 57% for both smoothing techniques. It was noted that precisions across the six topics had a great range, from 16.8% to 81.6%, with the Marriage and Parenthood topic having the lowest precision.

2. SVM Results using WEKA

LibSVM was used as a plugin in WEKA to allow us to perform SVM classification. We performed the SVM classification experiment on our pure polarity posts versus argumentative posts to see how well it performed against our naïve Bayes classifier.

In this experiment, we achieved an accuracy of 70.8% with the precision, recall and F-score results as shown in Table 13.

	Confusion Matrix	TP Rate	FP Rate	Precision	Recall	F-Score
Pure Polarity Posts	See Appendix B	0.668	0.264	0.651	0.668	0.66
Argumentative Posts		0.736	0.332	**0.751**	**0.738**	**0.744**

Table 13. SVM Results on Pure Polarity and Argumentative Posts using WEKA. (Bolded entries are the best results for Precision, Recall and F-score.)

LibSVM allows multi-class classification by performing one-to-one classification in each iteration. Hence, we were able to perform classification on our six argumentative topics shown on Table 14. with an accuracy of 42.86%. This performance is approximately 14% poorer than our naïve Bayes classifier.

	Confusion Matrix	TP Rate	FP Rate	Precision	Recall	F-Score
Marriage	See Appendix B	0.286	0.032	0.333	0.286	0.308
Identity		0.296	0.021	0.615	0.296	0.4
Immigrant		0.293	0.142	0.273	0.293	0.282
Economy		0.263	0.014	**0.833**	0.263	0.4
Cost		0.785	0.481	0.408	**0.785**	**0.537**
Livability		0.271	0.064	0.481	0.271	0.347

Table 14. SVM Results on Six Argumentative Posts using WEKA. (Bolded entries are the best results for Precision, Recall and F-score.)

We also performed SMOTE on our pure polarity posts using WEKA to test if SVM has any effect on imbalanced dataset. Our results in Table 15. showed that there was a 2% increase in accuracy, to 72.8%, after SMOTE was run on the minority class.

Compare these results to our best naïve Bayes results, applying the SMOTE technique on SVM is 16.2% poorer in accuracy.

	Confusion Matrix	TP Rate	FP Rate	Precision	Recall	F-Score
Pure Polarity Posts	See Appendix B	0.513	0.000	**1.000**	0.513	0.678
Argumentative Posts		1.000	0.487	0.832	**0.728**	**0.716**

Table 15. SVM Results Using SMOTE in WEKA. (Bolded entries are the best results for Precision, Recall and F-score.)

3. Maximum Entropy Results Using Mallet

We used Maximum Entropy from MALLET to compare its results with those of our naïve Bayes classifier. In our experiments we preserved the case of the words as of the unigrams in the naïve Bayes classifier. We ran 10 rounds of trials with the average result tabulated in Table 16.

	Pure Polarity versus Argumentative	6 Topics Comparison
Confusion Matrix	See Appendix C	
Accuracy	0.771	0.515
Standard Deviation	0.012	0.027

Table 16. Maximum Entropy Results using MALLET.

Comparing these results to those of our naïve Bayes classifier in Table 7 and Table 12, our classifier fared better by 2% in accuracy for two-class classification and 6% better for the six topics within argumentative posts.

4. Results Using Labeled LDA

Using the Stanford TMT, we performed Topic Modeling using Labeled LDA. We first learnt the system by running the training dataset against it. This produced a dataset of word-topic distributions as described in Chapter II.B.3. We then ran the test dataset against the learning system through an inferring script.

The pure polarity versus argumentative posts experiment produced an accuracy of 70.7%, and its precision, recall and F-score results are shown in Table 17.

	Confusion Matrix	Precision	Recall	F-score
Polarity Posts	See Appendix D	**0.717**	**0.838**	**0.773**
Argumentative Posts		0.684	0.514	0.587

Table 17. Results on Pure Polarity and Argumentative Posts using Labeled LDA. (Bolded entries are the best results for each column.)

We applied Labeled LDA on the six argumentative topics and obtained an accuracy of 45.2%. The precision, recall and F-score of the six topics are presented in Table 18.

		Precision	Recall	F-score
Cost	See Appendix D	**0.524**	**0.595**	**0.557**
Economy		0.5	0.528	0.514
Identity		0.429	0.444	0.436
Immigrant		0.409	0.391	0.4
Livability		0.375	0.294	0.33
Marriage		0.167	0.182	0.174

Table 18. Results on Pure Polarity and Argumentative Posts using Labeled LDA.

Comparing both results with our naïve Bayes classifier, our classifier fared better by 8.4% and 12.6% in accuracy for pure polarity versus argumentative classification and six argumentative topics classification, respectively.

B. ANALYSIS OF OUR TOPIC DETECTION RESULTS

From the results of the various topic detection experiments, we deduced that there were signals in our *Singlish* dataset that we could use to perform classification. In our two-class classification, all our results fared significantly better than that of the baseline score of 59.5%. Our summary results for pure polarity versus argumentative posts in Figure 10. shows that our naïve Bayes classifier gave the best accuracy with the use of the entropy exclusion list and the SMOTE technique with 100% increase.

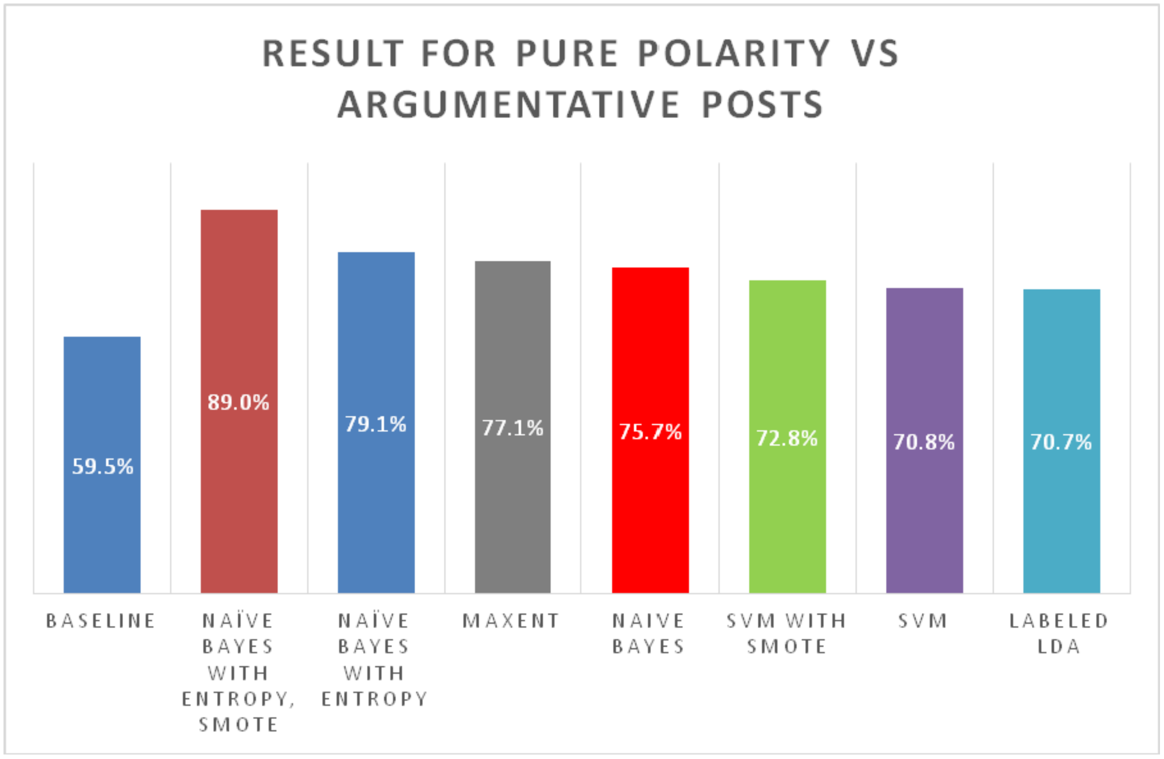


Figure 10. Summary Results of Topic Detection on Various Techniques for Pure Polarity versus Argumentative Posts.

The use of the SMOTE technique introduced more synthetic examples into the minority class. This boost provided a more balanced dataset, which in turn created an increase in occurrences for features that rarely appear. We noted in Table 10. that as the number of occurrences increases, the performance of the system improved. By increasing the occurrence count, it increases the probability of that feature, and hence, increases the probability of that class. This boost also improved our F-scores for both classes where they become more balanced.

In SVM, we also saw improvement in accuracy when the minority class was boosted with SMOTE. Hence having a balanced dataset does help in improving the overall accuracy and having good precision for both classes.

In our experiments for pure polarity posts versus argumentative posts, we could see that our results on the different classifiers with no entropy analysis or SMOTE application range from 70.7% to 77.1% in accuracy. “Noise” in the system affects the performance of the classifier. Entropy analysis was only introduced for the naïve Bayes classifier to remove words that had no effects in the system. With entropy analysis, we could see an increase in the accuracy, which shows that entropy analysis can help in reducing “noise” in the system.

In our experiments, adding the class prior made things worse because it caused the system to skew towards the majority class. We did not try experimenting with the prior after augmenting the minority class via SMOTE. We believe that the prior will have less of an affect in this case.

In our experiments, we also determined the α value of the Laplace smoothing to achieve the best accuracy result. As our vocabulary size V is as large as 1,013,913 words for unigrams, giving α value of 1 would give the system too much mass for unseen words. This produced a probability that is negligible to the class. We found an optimal value of 0.001 such that it allowed the accounting of the unobserved word while not having adverse effects on the unseen mass.

In our experiments, we also made comparisons between Laplace smoothing and Witten Bell smoothing. The two techniques differed by 1% in accuracy in most cases. Hence, there was no clear indication of which smoothing technique was better.

For our topic detection results on the six argumentative topics shown in Figure 11. , the accuracies for the various techniques range from 42% to 58% compared to a MLE baseline of 28.4%. In most cases, we could see that Marriage and Parenthood topic gave the worst F-score due to the fact that it had only 61 posts. On the other hand, the Cost of Living topic produced the best F-score using 378 posts.

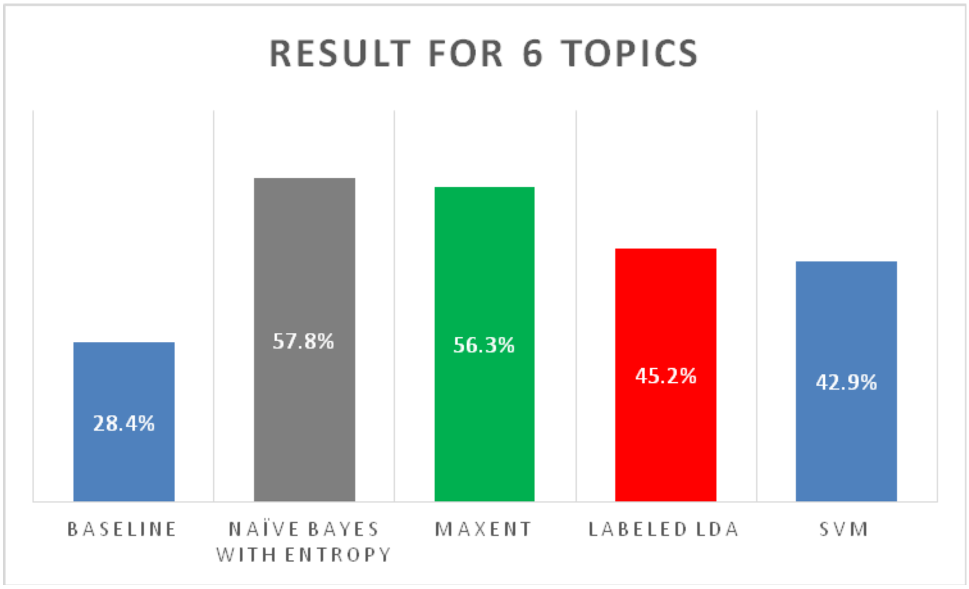


Figure 11. Summary Results of Topic Detection on Various Techniques for Six Topics.

C. SENTIMENT ANALYSIS RESULTS

After determining that there was signal for our topic detection, we moved on to our sentiment analysis where we performed classification on the argumentative posts to find out what people felt about the Singapore white paper. Out of the 425 *Singlish* posts that we annotated, we obtained 128 positive and 770 negative target phrases. We determined the baseline of our phrasal sentiment analysis classifier in Table 19.

	Accuracy	Precision	Recall	F-score
Positive	0.143	0.143	1	0.250
Negative		0.857	1	0.923

Table 19. Baseline for Sentiment Analysis using 898 Target Phrases.

In this experiment, we performed analysis using the first window size, where we took all the unigrams surrounding the target phrases into account. The confusion matrix shown in Table 20. was our result after running the phrasal sentiment analysis classifier. For target phrases that had the same number of positive and negative polarity results, we classified these phrases as having neutral sentiments. Hence in our confusion matrix, it can be seen that although our truth only had positive and negative polarity, our labeled data contained positive, neutral and negative sentiments. It is also noted that the neutral target phrases constituted one third of our overall target phrases.

	Positive	Negative	Total
Positive	72	271	343
Neutral	41	252	293
Negative	15	247	262
Total	128	770	898

Table 20. Confusion Matrix for Sentiment Analysis.

Our accuracy as a result from the confusion matrix was 35.5%. This result is 21.2% better than our baseline accuracy. We also noted that our precision result for negative target phrases was remarkably high at 94.3%, and our classifier did better than its baseline precision.

	Precision	Recall	F-score
Positive	0.210	**0.563**	0.306
Negative	**0.943**	0.321	**0.479**

Table 21. Results for Sentiment Analysis using 898 Target Phrases. (Bolded entries are the best results for each row.)

We also performed the analysis using the mean of unigrams between the target phrases as our window size. Our result showed a drop in accuracy to 28.1% with neutral polarity constituting to almost 50% of result. Our positive precision declined slightly to 20.55%, while the negative precision increased by

1% to 95.2%. The overall result did not show a positive improvement over our first experiment.

	Negative	Positive	Total
Negative	200	10	210
Neutral	369	66	435
Positive	201	52	253
Total	770	128	898

Table 22. Confusion Matrix for Sentiment Analysis using Mean of Unigrams between Target Phrases.

	Precision	Recall	F-score
Positive	0.205534	0.40625	0.272966
Negative	0.952381	0.25974	0.408163

Table 23. Results for Sentiment Analysis using Mean of Unigrams between Target Phrases.

In our experiments, we observed that some words, like "rich," 'like," and "talents" that exist in our positive lexicons, were not actually positive in our context. Hence we performed contextual lexicon tests to determine if these words have any impact in our classifier. In Table 24. , we populated the top fifteen positive words, and observed the change in accuracy when these words were shifted to the negative lexicon list or removed from both lists. It was noted that all of the words in this list except "better" had higher changes in accuracy when they were shifted to negative list. "Better" showed better accuracy when we removed it from the list altogether.

Lexicon	Change in Accuracy from Positive to Negative	Change in Accuracy from Positive to Neutral	Higher change in Negative/Neutral Polarity
like	6.13%	2.78%	Negative
rich	4.90%	2.00%	Negative
work	2.23%	1.34%	Negative
good	1.89%	1.11%	Negative
well	1.89%	1.23%	Negative
right	1.11%	1.00%	Negative
better	1.00%	1.23%	Neutral
clear	1.00%	0.22%	Negative
enough	1.00%	0.00%	Negative
comfort	0.89%	0.11%	Negative
cheaper	0.89%	0.11%	Negative
strong	0.78%	0.33%	Negative
skilled	0.78%	0.22%	Negative
comfy	0.67%	0.11%	Negative
talents	0.67%	0.56%	Negative

Table 24. Top 15 Positive Lexicons that Have Impact on Sentiment Analysis Accuracy.

In Table 25. and Table 26. we populated the top fifteen words that had improvements on the positive and negative precisions. It is interesting to note that although the word “like” had a significant improvement in positive precision when it was shifted to the negative lexicon, it had adverse effects that decreased the negative precision.

Lexicon	Change in Precision for Negative	Increase/ Decrease in Precision for Negative	Change in Precision for Positive	Increase/ Decrease in Precision for Positive
Like	0.16%	Decrease	1.99%	Increase
Rich	0.42%	Increase	1.86%	Increase
enough	0.23%	Increase	0.64%	Increase
Clear	0.23%	Increase	0.51%	Increase
cheaper	0.20%	Increase	0.51%	Increase
wonder	0.00%	No change	0.51%	Increase
comfort	0.20%	Increase	0.44%	Increase
Strong	0.18%	Increase	0.38%	Increase
Work	0.18%	Increase	0.32%	Increase
welcome	0.22%	Decrease	0.31%	Increase
Comfy	0.15%	Increase	0.31%	Increase
protect	0.13%	Increase	0.31%	Increase
Gains	0.10%	Increase	0.31%	Increase
Right	0.38%	Decrease	0.28%	Increase
Well	0.11%	Increase	0.28%	Increase

Table 25. Top 15 Positive Lexicons that Have Improvement on the Positive Prediction on Sentiment Analysis.

Lexicon	Change in Precision for Negative	Increase/ Decrease in Precision for Negative	Change in Precision for Positive	Increase/ Decrease in Precision for Positive
Rich	0.42%	Increase	1.86%	Increase
enough	0.23%	Increase	0.64%	Increase
Clear	0.23%	Increase	0.51%	Increase
cheaper	0.20%	Increase	0.51%	Increase
comfort	0.20%	Increase	0.44%	Increase
Strong	0.18%	Increase	0.38%	Increase
Work	0.18%	Increase	0.32%	Increase
Comfy	0.15%	Increase	0.31%	Increase
protect	0.13%	Increase	0.31%	Increase
Faster	0.13%	Increase	0.19%	Increase
Well	0.11%	Increase	0.28%	Increase
Gains	0.10%	Increase	0.31%	Increase
trusting	0.10%	Increase	0.25%	Increase
charitable	0.10%	Increase	0.25%	Increase
Super	0.10%	Increase	0.25%	Increase

Table 26. Top 15 Positive Lexicons that Have Improvement on the Negative Prediction on Sentiment Analysis.

We consolidated the words, shown in Table 27. that were common across the three tables and shifted them to the negative lexicon to test their impact on our Phrasal sentiment analysis classifier.

Positive Lexicon
rich
work
enough
well
clear
comfort
comfy
strong
cheaper

Table 27. Positive Lexicon.

Our results showed that there was an improvement of 8% to the accuracy of 43.7% when these positive words shifted to the negative list. While the precision for the negative polarity increased by only 0.2%, the positive precision increased by 5%.

In another experiment, we used only the positive words from Table 24. and shifted them to the negative list. The result gave a strong increase from 35.5% to 53.8% in accuracy. It was noted that there was a 2% drop in negative precision but a 7% increase in the positive precision.

D. ANALYSIS OF OUR SENTIMENT ANALYSIS RESULTS

In our sentiment analysis, the accuracy results we obtained were poor due to the fact that our dataset was skewed to negative polarity with negative target phrases taking up 85% of all phrases. People were unhappy about some implementations that were suggested in the Singapore white paper, and this unhappiness was widely discussed in the Facebook pages from which we obtained our dataset. Although positive comments had also been given for the Singapore white paper, there were not as many as the negative ones. Due to the lack of the positive comments, the result produced was not satisfactory. However, we did obtain a remarkable result for precision of 94.3% for negative polarity, showing that 94% of time, when we classified a target phrase as negative, it was negative. This also showed the accuracy of our lexicon in producing the negative polarity.

The results from the two window sizes showed that by taking all the unigrams in the post into account, it had more context over using only the few surrounding unigrams of the target phrase. However, this also meant that multiple target phases in same post would obtain the same polarity result which may lead to inaccuracy of our classifier. We would need to find an optimal window size that could correctly represent the context of the target phrases.

We also noted that the words used in the Singapore white paper context had different meanings when compared to their definitions in the dictionary. In the Facebook pages, the income gap between the rich and poor was widely discussed, so were the cheaper labor from the foreign workers and high cost of living. Hence it was no surprise that words like “rich,” “comfort,” “talents” and “cheaper” had negative polarity in them. In our case, by putting these words into the right context, we helped to improve the system accuracy as well as to remove “noise” from our positive polarity. Instead of having a generic lexicon, a context-related list should be built for the sentiment analysis classifier.

V. FUTURE WORK AND CONCLUSIONS

A. SUMMARY

Our goal was to perform topic detection and sentiment analysis on Facebook posts that were in *Singlish*. We performed supervised topic detection on 2374 Facebook posts using various methods which included naïve Bayes, SVM, Maximum Entropy and Labeled LDA. In our two-class classification for pure polarity and argumentative posts, our naïve Bayes classifier gave the best accuracy results of 79.1%. We also performed boosting on our minority class using SMOTE to evaluate if a balanced dataset helped in improving our classifier. Our results after applying SMOTE, gave us a high accuracy of 89%, while the precisions for pure polarity and argumentative posts were 91.3% and 86.1%, respectively. This shows that it is important to have a balanced dataset in order to achieve accurate classification. In our multi-class classification for the six argumentative topics, we obtained an accuracy of 57% compared to the baseline of 28.4%, with the minority class having the lowest precision. This result once again showed the importance of a balanced dataset.

In our supervised sentiment analysis experiments where we classified 898 positive and negative target phrases from 425 posts, we received a result of 35.5% accuracy compared to a baseline accuracy of 14.3%, and we were able to achieve a high precision of 94.3% for our negative polarity. Our results were due to the skewed dataset in which 85% of the target phrases had negative polarity. The experiments also showed the target phrases were sensitive to the number of surrounding words and that the polarity words were subjected to the context of our dataset. The posts that were collected showed many negative sentiments; hence, words that had positive dictionary meanings ended up having negative polarity in our context. In our last experiment where we shifted these positive words with negative polarity to the negative lexicon, we were able to achieve an accuracy of 53.8%, again compared to the baseline of 14.3%. This showed that the lexicons are context dependent, and they should be customized accordingly to the context or domain of the dataset.

B. FUTURE WORK

This research suggests that there were good signals within our *Singlish* posts, and much work can be done to improve our classification. Some of the work that can be done in the future includes the following:

- Balance the dataset

  It is clear in this research that a balanced dataset plays an important role in obtaining accurate result. Hence more data can be collected and annotated to improve the dataset. In real world applications where having a balanced dataset is not possible, other methods of boosting can also be implemented to see how sensitive they are to the dataset and how they can improve the system.

- Entropy Analysis on SVM, Maximum Entropy and Labeled LDA

  In this research, entropy analysis was done only on naïve Bayes classification. Entropy analysis can be further extended to SVM, Maximum Entropy and Labeled LDA to test if they have an effect on these classifier since it produced an improvement in our classification.

- Multi-class in Labeled LDA

  In our annotated dataset for topic detection, we only assigned one topic to each Facebook post. However, multiple topics can exist for each post. Labeled LDA can be used to perform this classification to determine how accurate a post can be represented and the weight of each topic in each post.

- Contextual Lexicons

  It is shown in this research that the lexicons in our sentiment analysis are context related. Hence, instead of using the generic lexicons that are made available by other research, a customized lexicon should be used to evaluate the performance of the classification. Non-English or non-formal words that might have polarity effects can also be used to test their impact on the classification.

C. CONCLUSION

Our hypothesis for this research was that we could apply the state-of-the-art methods of social media sentiment analysis and topic detection on Colloquial Singapore English (*Singlish*). In this thesis, we presented various classification methods on our *Singlish* Facebook posts. Our evaluation using these classification methods shows that there were signals in our *Singlish* Facebook posts on which we could potentially perform classification.

The following figure presents the results of our experiments.

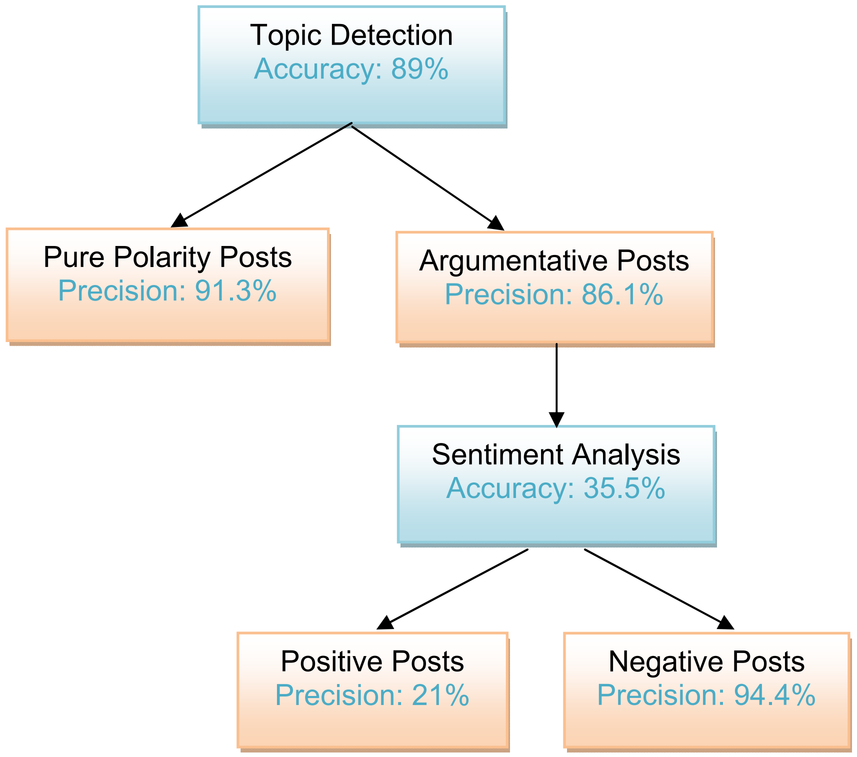


Figure 12. Summary Results of Topic Detection and Sentiment Analysis.

Using our imbalanced dataset and removing some “noise” from it, we were able to achieve 79.1% accuracy, compared to a baseline of 59.5%, for topic detection on our pure polarity versus argumentative posts. From the result of SMOTE, we could conclude that we would be able to achieve a better result if the dataset was balanced. From our experiments using the various classifiers, we also observed that we need to perform tuning on the classifiers' parameters in order to achieve the best result; hence, a general classifier is not applicable to all datasets.

In our sentiment analysis, we performed classification on our annotated target phrases using a lexicon list. We achieved a remarkably good result for our precision of 94.5% on negative polarity. This showed that our dataset contained signals that were suitable for classification and also showed how accurate our lexicon list was in providing the polarity sentiment. However, due to the imbalanced dataset, we were not able to achieve satisfactory accuracy results for our dataset. This research has also shown that although a general classifier can be used to perform classification, a classifier which is related to the context of the dataset can further improve the result. The extensive need for human knowledge to make such a context-related system and to develop a classifier will be essential.

APPENDIX A. CONFUSION MATRICES FOR NAÏVE BAYES RESULT

A. CONFUSION MATRICES FOR LAPLACE SMOOTHING RESULT

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	92	18
Argumentative Posts	1718	2622

Table 28. Confusion Matrix for Laplace Smoothing α=1.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	280	65
Argumentative Posts	1530	2575

Table 29. Confusion Matrix for Laplace Smoothing α=0.1.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	927	294
Argumentative Posts	833	2346

Table 30. Confusion Matrix for Laplace Smoothing α=0.01.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1088	426
Argumentative Posts	722	2214

Table 31. Confusion Matrix for Laplace Smoothing α=0.001.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1034	398
Argumentative Posts	776	2242

Table 32. Confusion Matrix for Laplace Smoothing α=0.0001.

B. CONFUSION MATRICES FOR LAPLACE AND WITTEN BELL SMOOTHING

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1088	426
Argumentative Posts	722	2214

Table 33. Confusion Matrix for Laplace Smoothing α=0.001.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1038	309
Argumentative Posts	772	2331

Table 34. Confusion Matrix for Witten Bell.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1027	386
Argumentative Posts	783	2254

Table 35. Confusion Matrix for Laplace Smoothing α=0.001 and Class Prior.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	970	302
Argumentative Posts	840	2338

Table 36. Confusion Matrix for Witten Bell and Class Prior.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1213	334
Argumentative Posts	597	2306

Table 37. Confusion Matrix for Laplace Smoothing α=0.001 and Entropy Exclusion List.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1098	253
Argumentative Posts	712	2387

Table 38. Confusion Matrix for Witten Bell and Entropy Exclusion List.

C. CONFUSION MATRICES FOR UNIGRAMS, WORD-BIGRAMS, WORD-TRIGRAMS AND CHARACTER-TRIGRAMS

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1213	334
Argumentative Posts	597	2306

Table 39. Confusion Matrix for Unigrams using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1503	783
Argumentative Posts	307	1857

Table 40. Confusion Matrix for Word-Bigrams using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1715	2245
Argumentative Posts	95	395

Table 41. Confusion Matrix for Word-Trigrams using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1071	321
Argumentative Posts	739	2319

Table 42. Confusion Matrix for Character-Trigrams using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1098	253
Argumentative Posts	712	2387

Table 43. Confusion Matrix for Unigrams using Witten Bell Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1199	370
Argumentative Posts	611	2270

Table 44. Confusion Matrix for Word-Bigrams using Witten Bell Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	574	282
Argumentative Posts	1236	2358

Table 45. Confusion Matrix for Word-Trigrams using Witten Bell Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1027	246
Argumentative Posts	783	2394

Table 46. Confusion Matrix for Character-Trigrams using Witten Bell Smoothing.

D. CONFUSION MATRICES FOR SMOTE USING LAPACE AND WITTEN BELL SMOOTHING

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1213	334
Argumentative Posts	597	2306

Table 47. Confusion Matrix for 0% SMOTE using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	2141	298
Argumentative Posts	489	2342

Table 48. Confusion Matrix for 45% SMOTE using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	2540	287
Argumentative Posts	450	2353

Table 49. Confusion Matrix for 65% SMOTE using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	3190	298
Argumentative Posts	430	2342

Table 50. Confusion Matrix for 100% SMOTE using Laplace Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	1098	253
Argumentative Posts	712	2387

Table 51. Confusion Matrix for 0% SMOTE using Witten Bell Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	2120	272
Argumentative Posts	510	2368

Table 52. Confusion Matrix for 45% SMOTE using Witten Bell Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	2545	261
Argumentative Posts	445	2379

Table 53. Confusion Matrix for 65% SMOTE using Witten Bell Smoothing.

CATEGORIES	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	3243	310
Argumentative Posts	337	2330

Table 54. Confusion Matrix for 100% SMOTE using Witten Bell Smoothing.

E. CONFUSION MATRICES FOR SIX ARGUMENTATIVE TOPICS USING LAPACE AND WITTEN BELL SMOOTHING

CATEGORIES	Marriage	Identity	Immigrant	Economy	Cost	Livability
Marriage	100	75	102	141	104	72
Identity	3	140	31	28	18	22
Immigrant	3	14	235	21	21	31
Economy	5	5	12	409	30	40
Cost	1	5	16	46	295	17
Livability	8	31	64	105	62	328

Table 55. Confusion Matrix for Six Argumentative Topics using Laplace Smoothing.

CATEGORIES	Marriage	Identity	Immigrant	Economy	Cost	Livability
Marriage	94	77	95	118	82	67
Identity	1	122	39	25	20	20
Immigrant	1	19	219	30	19	24
Economy	6	8	19	436	37	37
Cost	5	9	23	44	304	12
Livability	13	35	65	97	68	350

Table 56. Confusion Matrix for Six Argumentative Topics using Witten Bell Smoothing.

APPENDIX B. CONFUSION MATRICES FOR SVM RESULT

Confusion Matrices for Pure Polarity versus Argumentative Posts

	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	127	68
Argumentative Posts	63	190

Table 57. Confusion Matrix for Pure Polarity versus Argumentative Posts using SVM.

	Marriage	Identity	Immigrant	Economy	Cost	Livability
Marriage	4	0	1	3	3	1
Identity	0	8	0	3	1	1
Immigrant	3	10	12	7	9	3
Economy	0	0	0	15	3	0
Cost	5	8	23	24	62	30
Livability	2	1	5	5	1	13

Table 58. Confusion Matrix for Six Argumentative Topics using SVM.

	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	180	0
Argumentative Posts	171	278

Table 59. Confusion Matrix for Pure Polarity versus Argumentative Posts with SMOTE using SVM.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. CONFUSION MATRICES FOR MAXIMUM ENTROPY RESULT

Confusion Matrices for Pure Polarity versus Argumentative Posts

	Pure Polarity Posts	Argumentative Posts
Pure Polarity Posts	206	43
Argumentative Posts	60	138

Table 60. Confusion Matrix for Pure Polarity versus Argumentative Posts using Maximum Entropy.

	Marriage	Identity	Immigrant	Economy	Cost	Livability
Marriage	2	0	1	4	3	2
Identity	1	9	6	5	2	4
Immigrant	0	6	21	8	7	6
Economy	1	1	5	57	7	8
Cost	0	2	5	14	22	7
Livability	0	2	2	14	4	28

Table 61. Confusion Matrix for Six Argumentative Topics using Maximum Entropy.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. CONFUSION MATRICES FOR LABELED LDA

Confusion Matrices for Pure Polarity versus Argumentative Posts

	Argumentative Posts	Pure Polarity Posts
Argumentative Posts	223	88
Pure Polarity Posts	43	93

Table 62. Confusion Matrix for Pure Polarity versus Argumentative Posts using Labeled LDA.

	Cost	Economy	Identity	Immigrant	Livability	Marriage
Cost	44	10	1	2	22	5
Economy	12	28	6	5	5	0
Identity	2	1	12	10	2	1
Immigrant	9	9	4	18	4	0
Livability	6	5	2	8	15	4
Marriage	1	0	2	3	3	2

Table 63. Confusion Matrix for Six Argumentative Topics using Labeled LDA.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] A. F. Gupta, and L. Alsagoff, “Singapore English” [Online]. Available: http://www.postcolonialweb.org/singapore/people/language/singeh2.html. [Accessed: July 1, 2013].

[2] *The 2012 Rock Publicity Singapore Social Media Study* [Online] November 2012, Available: http://rockpublicity.com/wp-content/uploads/2012/11/2012-RP-SINGAPORE-SOCIAL-MEDIA-STUDY.pdf. [Accessed: January 2013].

[3] *A Sustainable Population for a Dynamic Singapore: Population White Paper*, National Population and Talent Division, Singapore, January, 2013.

[4] A. F. Anta, L. N. Chiroque, and P. Morere, A. Santos, “Sentiment analysis and topic detection of Spanish tweets: A comparative study of NLP techniques,” *La Revista de Procesamiento de Lenguaje Natural*, vol. 50, pp. 45–52, 2013.

[5] F. Batista, and R. Ribeiro, “Sentiment analysis and topic classification: Case study over Spanish tweets,” *TASS 2012, Satellite Event of the SEPLN 2012 Conference*, September 7, Valencia, Spain, 2012.

[6] S. Narr, M. Hülfenhaus, and S. Albayrak, “Language-independent Twitter sentiment analysis”, *in Proceedings of the Lernen, Wissen, Adaption 2012, Knowledge Discovery and Machine Learning Workshop*, Dortmund, Germany, September 12–14, 2012.

[7] S. Asur, and B. A. Huberma, “Predicting the future with social media,” HP labs [Online] April 21, 2010. Available: http://www.hpl.hp.com /techreports/2010/HPL-2010-53.pdf. [Accessed: January 2013].

[8] A. Pak, and P. Paroubek, “Twitter as a corpus for sentiment analysis and opinion mining,” in *Proceedings of the Seventh International Conference On Language Resources and Evaluation (LREC),* pp. 1320–1326, Valletta, Malta, May 19–21, 2010.

[9] A. Go, R. Bhayani, and L Huang, “Twitter sentiment classification using distant supervision,” Technical report, Stanford Digital Library Technologies Project, Stanford University, 2009.

[10] E. Kouloumpis, T. Wilson and and J. Moore, “Twitter sentiment analysis: The good, the bad and the OMG!,” in *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media,* Barcelona, Spain, July 17–21, 2011.

[11] S. Petrovic, M. Osborne, and V. Lavrenko, “The Edinburgh Twitter corpus,” *in Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pp. 25–26, Los Angeles, California, June 6, 2010.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten, “The WEKA data mining software: An update,” *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[13] M. A. Kachites, “MALLET: A machine learning for language toolkit." [Online] 2002. Available: http://mallet.cs.umass.edu. [Accessed: May 2013].

[14] D. Ramage, D. Hall, R. Nallapati, and C.D. Manning, “Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora,” in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing,* vol. 1, Singapore, August 6–7, 2009.

[15] D. M. Blei, A. Ng, and M. Jordan, “Latent Dirichlet allocation,” *Journal of Machine Learning Research,* vol. 3, pp. 993–1022, 2003.

[16] K. Lee, D. Palsetia, R. Narayanan, M.M.A. Patwary, A. Agrawal, and A. Choudhary, “Twitter trending topic classification,” *in Proceedings of the 11th IEEE International Conference on Data Mining Workshops,* pp. 251-258, Vancouver, Canada, December 11–14, 2011.

[17] S. Asur, B. A. Huberman, G. Szabo, and C. Wang, “Trends in social media: Persistence and decay*,” in Proceedings of Fifth International AAAI Conference on Weblogs and Social Media*, Barcelona, Spain, July 17–21, 2011.

[18] B. Pang, and L. Lee, “Opinion mining and sentiment analysis,” *Foundations and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1-135, 2008.

[19] J. K. Ahkter, S. Soria, “Sentiment analysis: Facebook status messages,” Technical report, Final Project CS224N, Stanford University, 2010.

[20] A. Shrivatava, B. Pant, “Opinion extraction and classification of real time Facebook status,” *Global Journal of Computer Science and Technology*, vol. 12, no. 8, 2012.

[21] N. Cristianini, and J. Shawe-Taylor, “An introduction to support vector machines and other kernel-based learning methods,” *Cambridge University Press*, 2000.

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

[23] G. E .A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter - Special Issue on Learning from Imbalanced Datasets,* vol. 6, no. 1, pp. 20–29, June, 2004.

[24] H. He, E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*," vol. 21, no. 9, September, 2009.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[26] G. Harihara, E. Yang, and N. Chambers, "USNA: A dual-classifier approach to contextual sentiment analysis," *in Proceedings of the 7th International Workshop on Semantic Evaluation*, *Task 1*, Atlanta, Georgia, June 13–15, 2013.

[27] T. Wilson, J. Wiebe, and P. Hoffmann, "Subjectivity lexicon" [Online] 2005. Available: http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/ [Accessed: July 1, 2013].

[28] M. Hu and B. Liu, "Opinion lexicon" [Online] 2004. Available: http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar [Accessed: July 1, 2013].

[29] C. R. Souza, "The Accord.NET framework" [Online] Apr 2012. Available: http://accord.googlecode.com [Accessed: June 1, 2013].

[30] R. Durstenfeld, "Fisher–Yates shuffle" [Online] 1964. Available: http://www.dotnetperls.com/fisher-yates-shuffle [Accessed: June 1, 2013].

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California