Sentiment Analysis of Tweets Using Naive Bayes and Support Vector Machines

CS 5824
Dvijen Mahesh Trivedi
Virginia Polytechnic Institute and State University
Department of Computer Science and Applications

1. Abstract

This paper will investigate various approaches for performing “Sentiment Analysis” of Tweets. Based on the sentiment behind the tweets, they are categorized as either positive or negative. This finds application in a wide array of domains including news articles, movie, tv shows or book reviews or product reviews from customers. This paper focuses on applying novel techniques for performing sentiment analysis on a dataset which was gathered by scrapping Twitter APIs using emoticons as noisy labels. The paper intends to compare performance of Naive Bayes classifier and Support Vector Machines for the task of sentiment analysis. Preprocessing steps like bag of words approach is intended to be explored in the classification task. The idea behind using tweets with emoticons for distant supervised learning is explored in this paper [1].

2. Specific Aims

The aim is to classify tweets according to their sentiment viz. positive or negative. Two novel approaches using Naive Bayes Classifier and Support Vector Machines will be compared for the classification task. Impact of feature selection using techniques like bag of words will be explored on the model performance and classification accuracy.

3. Background

Twitter is a well known microblogging and social networking service. Users interact and post messages via tweets. People tweet about a plethora of topics. These tweets are often opinionated regarding the some reviews or feedbacks. People tweet about various consumer products like some electronic devices they had recently bought, latest movie, a new season of a popular tv series, most recent novel from a popular author etc. In a sense such tweets serve as a formal review or feedback for the entity of interest (product, movie, tv series, book etc) from the consumers perspective. Important point to note here is such an important piece of information is readily available on twitter without any complex survey systems managed by organizations. The user’s sentiment is clearly outlined in such opinionated tweets and can be easily interpreted by other human readers. As result analyzing such opinionated tweets to understand the sentiment becomes important for businesses. Obviously corporations cannot rely on human intervention to analyze and understand the sentiment behind such tweets. The underlying process of understanding the sentiment behind an opinionated tweet needs to be automated and the plethora of machine learning algorithms fall right in the ballpark of such design requirements.

A lot has been done in the area of sentiment analysis. A vast majority of such approaches have previously focused on longer texts like reviews [2]. Tweets are however structurally different from long texts such as movie reviews. One of the key difference is the 140 character limit in tweets. Movie reviews are usually longer, summarizing the viewers opinions. Another key difference is reviews are a tad bit formal, well formatted and structured as opposed to the casual and on the go status update nature of tweets. Tweets are however a good source of gathering feedback for products and services by corporations and could offer new exciting avenue in contrast to corporate feedback surveys. One reason why tweets might serve as a richer source of the true sentiments or feelings of the consumer is that, in this case the consumer was not asked to enter a well routined set of survey questionnaires, rather the consumer willingly broadcasted opinions on the microblogging site. As a result sentiment analysis of tweets offers an exciting avenue for understanding the consumer needs and requirements, feedback,

demands, degree of satisfaction, engagement and loyalty to the brand etc. Some intriguing work has been done in the area of phrase and sentence level sentiment analysis in past [3].

One of the striking similarity between the works of Pang et al. [2] and Alec et al [1] is that both use a polarity signal in training data, star rating and emoticons respectively. This works as a good predictor of underlying sentiment. Techniques mentioned in paper of Pang et al. [2] serve as fundamental guiding blocks for the sentiment analysis research. This paper aims to apply novel machine learning techniques of Naive Bayes and Support Vector Machines to a dataset built on the guiding principles as described in Alec et al [1].

4. Research Design and Method

This paper tests the performance of Naive Bayes and Support Vector Machines on sentiment analysis of tweets. I have worked on Naive Bayes Classifier and my partner, Aman Sarawgi has worked on Support Vector Machines. Accuracy for both the models are compared to find a superior classifier for the problem statement. The underlying approach (feature selection and pre-processing etc.) is kept similar as the dataset used for training and testing either classifiers is same. This is done to ensure that model performance and accuracy for Naive Bayes and Support Vector Machines is directly comparable to one another. This leads us to objectively recommend a better fit for the underlying application, sentiment analysis of tweets. Supervised learning is used as the class labels for the positive or negative sentiments are known in the dataset.

Classifier is trained on a document term matrix generated from selected words from the text column of the dataset which stores tweets from various users. Each selected word in the document term matrix hence serves as a feature for the classifier to be trained on. These features are considered as asymmetric attributes i.e. marked as a 1 if the word occurs in the record (for a particular tweet from dataset) or 0 otherwise. Now the model is trained using these features and the respective target class is used. This approach tends to learn most important words in terms of being inherently positive or negative in sentiment. Thus while testing, such a model would look for the presence of such words to determine the winning sentiment prediction.

Count Vectorizer method from scikit-learn library is used to tokenize and create a sparse representation of document term matrix to be used by the Naive Bayes classifier.

4.1 Naive Bayes

Naive Bayes calculates the posterior probabilities based on the prior probabilities (from dataset or domain knowledge) and the likelihood based on the given attributes under the confinement of a class universe. This can be mathematically represented as:

$$P(Y|X)=\frac{P(X|Y)P(Y)}{P(X)}$$

Here Y is the target class and X is the set of feature attributes. The denominator, Evidence is merely a scaling factor obtained by marginalizing the joint probability of attributes and target class over all possible target class outcomes. It is called the marginal probability of attributes or feature set. As it is a scaling factor we can easily ignore it for comparison and class predictions.

A naive assumption that the features (attributes) are conditionally independent of each other given a target class helps to calculate likelihood which is nothing but the conditional probability of feature set given the target class. This can be formulated as follows:

$P(X|Y)=\prod_{i=1}^{d} P(X_i|Y)$ The prediction is made on basis of the principle of “maximum a posteriori” i.e. assign the class which has the maximum posterior probability.

4.2. Dataset

The dataset consists of 1.6 million tweets in the record database.
It contains the following 6 fields:

1.**target**: the polarity of the tweet (*0* = negative, *4* = positive)
2.**ids**: The id of the tweet
3.**date**: the date of the tweet
4.**flag**: The query. If there is no query, then this value is NO_QUERY.
5.**user**: the user that tweeted
6.**text**: the text of the tweet

Following is a snapshot of the dataset

```
[ ]  1 df_twt_data.head()
```

	target	ids	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.

Figure 1: Snapshot of Dataset

Attribute data types are as follows:

```
[ ]  1 df_twt_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   target  1600000 non-null  int64
 1   ids     1600000 non-null  int64
 2   date    1600000 non-null  object
 3   flag    1600000 non-null  object
 4   user    1600000 non-null  object
 5   text    1600000 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

Figure 2: Attribute info in dataset

4.3. Preprocessing and Feature Extraction

Working with tweets comes with a set of challenges. Hence a great deal of effort goes into preprocessing the text and choosing the appropriate features to build the document term matrix for training the model and testing new records using the trained model. Twitter allows users to mention other users, groups or trending topics with various hashtags in their tweets. Additionally hyperlinks to various blogs, websites etc are embedded in users tweets. Such at-mentions and/or hyperlinks along with punctuations used do not convey any information about the underlying sentiment of the tweet and hence must be discarded. This was achieved by using regular expressions. Additionally certain words are characteristics of the language itself and they commonly occur in any text. Such words are called as stop-words and hence must also be removed.

It is imperative to note that different forms of the same word can be used in various tweets for e.g. *organize*, *organizes*, and *organizing*. Additionally, there are families of derivationally related words with similar meanings, such as *democracy*, *democratic*, and *democratization*. Hence having multiple such words as tokens (features) in our document term matrix makes it redundant. Hence it is essential to normalize words to kind of a common denominator or a root word per say before using it as a feature.

Stemming and Lemmatization are methods to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:

Original Set of Words	Root or Base Word
am, are, is	be
car, cars, car's, cars'	car

Hence such mappings can reduce the texts, somewhat in the following fashion:

Original Text	Changed Text
the boy's cars are different colors	the boy car be differ color

4.3.1. Stemming

Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. Stemming is the process of reducing the word to its word stem that affixes to suffixes and prefixes or to roots of words known as a lemma. Affix is a grammatical element that is combined with a word, stem, or phrase to produce derived or inflected forms. Affixes may be derivational, like English -ness and pre-, or inflectional, like English plural -s and past tense -ed. There are three main types of affixes: prefixes, infixes, and suffixes. A circumfix (CIRC) consists of a prefix and a suffix that together produce a derived or inflected form, as in the English word enlighten.
In simple words stemming is reducing a word to its base word or stem in such a way that the words of similar kind lie under a common stem. For example – The words care, cared and caring lie under the same stem ‘care.
There are two popular English stemmers, the original Porter stemmer, and an improved stemmer which has been called Porter2 aka Snowball Stemmer. Snowball Stemmer is more aggressive than Porter Stemmer and it also fixes some of the previous issues in Porter Stemmer. Lancaster Stemmer is another more aggressive stemming algorithm.
I have used Snowball stemmer from the popular Natural Language Toolkit (NLTK) library.

4.3.2. Lemmatization

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. Inflectional morphology is the process by which a root form of a word is modified by adding prefixes or suffixes that specify its grammatical function but do not change its part-of-speech. Lemma (root form) is inflected (modified/combined) with one or more morphological features to create a surface form.. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. This is done by identifying the part of speech of a word and applying different set of rules based on it to achieve the true root word or lemma. So it links words with similar meanings to one word. Lemmatization is usually backed up by a part-of-speech tagger in order to disambiguate homonyms.
Natural Language Toolkit (NLTK) library offers a WordNetLemmatizer. But it does not use part of speech tagger and hence performs inferior to other lemmatizers.
I have used the industry standard SpaCy library to achieve lemmatization. It is more robust and offers a detailed parts-of-speech tagging, syntactic dependency (relation between tokens), shape of the word (capitalization, punctuation, digits) and a check for word being alphanumeric and/or a common English stop-word.
Spacy's lemmatization is computationally expensive due to its robustness and various capabilities as highlighted above. Applying lemmatization naively to such a huge dataset with 1.6 Million records would have taken days to run. A very useful feature of pipeline offered in Spacy was therefore used here to speed up the pre-processing by providing records from dataset in batches through a pipeline.

4.3.3. Stemming vs Lemmatization

Stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma. Stemming identifies the common root form of a word by removing or replacing word suffixes (e.g. "flooding" is stemmed as "flood"), while lemmatization identifies the inflected forms of a word and returns its base form (e.g. "better" is lemmatized as "good") [4].
Both stemming and lemmatization was applied to the tweets in dataset. Figure 3 below depicts the difference in which Snowball Stemmer and Spacy Lemmatization changes the original text in tweets from the dataset.

	target	ids	date	flag	user	text	lemma_text	stem_text
0	NEGATIVE	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D	awww bummer shoulda got david carr day d	awww bummer shoulda got david carr third day
1	NEGATIVE	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by texting it... and might cry as a result School today also. Blah!	upset update facebook texting cry result school today blah	upset updat facebook text might cri result school today also blah
2	NEGATIVE	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds	dived times ball managed save 50 rest bounds	dive mani time ball manag save 50 rest go bound
3	NEGATIVE	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire	body feels itchy like fire	whole bodi feel itchi like fire
4	NEGATIVE	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all. i'm mad. why am i here? because I can't see you all over there.	behaving mad	behav mad see
...	...	...	...	...	...	...	...	...
1599995	POSITIVE	2193601966	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	AmandaMarie1028	Just woke up. Having no school is the best feeling ever	woke having school best feeling	woke school best feel ever
1599996	POSITIVE	2193601969	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	TheWDBoards	TheWDB.com - Very cool to hear old Walt interviews! â « http://blip.fm/~8bmta	thewdbcom cool hear old walt interviews â	thewdb com cool hear old walt interview
1599997	POSITIVE	2193601991	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	bpbabe	Are you ready for your MoJo Makeover? Ask me for details	ready mojo makeover ask details	readi mojo makeov ask detail
1599998	POSITIVE	2193602064	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	tinydiamondz	Happy 38th Birthday to my boo of alll time!!! Tupac Amaru Shakur	happy 38th birthday boo alll time tupac amaru shakur	happi 38th birthday boo alll time tupac amaru shakur
1599999	POSITIVE	2193602129	Tue Jun 16 08:40:50 PDT 2009	NO_QUERY	RyanTrevMorris	happy #charitytuesday @theNSPCC @SparksCharity @SpeakingUpH4H	happy charitytuesday	happi charitytuesday thenspcc sparkschar speakinguph4h

Figure 3: Stemming and Lemmatization applied to tweets in dataset

4.4. Model Building

SVM model which was developed by my project partner Aman Sarawgi was taking a lot of time and due to limited hardware resources on our local machines it was decided to train our models on a subset of 200,000 records from the original 1.6 Million records in the dataset. This was needed so that head on comparison can be made between Naive Bayes and SVM classifier as discussed in the Model Evaluation section below.

Multinomial Naive Bayes model was trained on document term matrix obtained from CountVectorizer, first on stemmed text and then on lemmatized text. A train test split of 80:20 for training and testing set was used. Figure 4 shows the architecture highlighting the entire process.

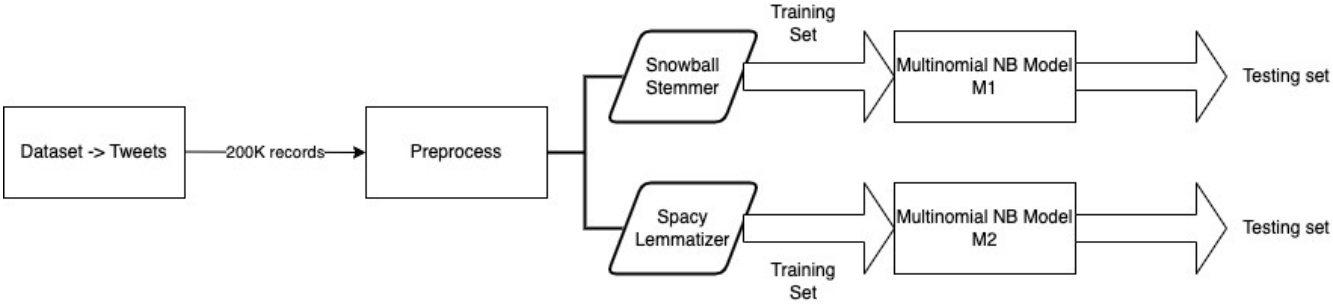

Figure 4: Architecture used for training Naive Bayes Classifier

As part of preprocessing step both stemming and lemmatization was applied to tweets in the dataset one by one and a document term matrix was created from each step. Training and Testing sets were obtained from each of the document term matrix using 80:20 random split. A Multinomial Naive Bayes Model was trained on each of the training set (one which uses stemming and other which uses lemmatization). Each of the trained model was then tested against its respective testing set to evaluate model performance.

4.5. Model Evaluation

Various scoring metrics like accuracy, precision, recall and f1-score were used to evaluate each models performance on the testing set. Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall is the ratio of correctly predicted positive observations to the all observations in actual class. F1-score is the harmonic mean of precision and recall and is generally useful to evaluate performance on a skewed dataset for a combined evaluation of two competing forces in precision and recall.
Figure 5 shows performance of Model 1 which runs on stemmed text.

```
Accuracy score: 0.7585
Precision score: 0.7771574413782102
Recall score: 0.7277072442120985
F1 score: 0.7516198704103672
              precision    recall  f1-score   support

    NEGATIVE       0.74      0.79      0.77     19915
    POSITIVE       0.78      0.73      0.75     20085

    accuracy                           0.76     40000
   macro avg       0.76      0.76      0.76     40000
weighted avg       0.76      0.76      0.76     40000
```

Figure 5: Performance of Model M1 on stemmed text

Figure 6 shows performance of Model 2 which runs on lemmatized text.

```
Accuracy score: 0.76075
Precision score: 0.7786020878596789
Recall score: 0.7315409509584266
F1 score: 0.7543382277441215
              precision    recall  f1-score   support

    NEGATIVE       0.74      0.79      0.77     19915
    POSITIVE       0.78      0.73      0.75     20085

    accuracy                           0.76     40000
   macro avg       0.76      0.76      0.76     40000
weighted avg       0.76      0.76      0.76     40000
```

Figure 6: Performance of Model M2 on lemmatized text

Figure 7 shows confusion matrix which turned out to be the same for both the models M1 and M2.

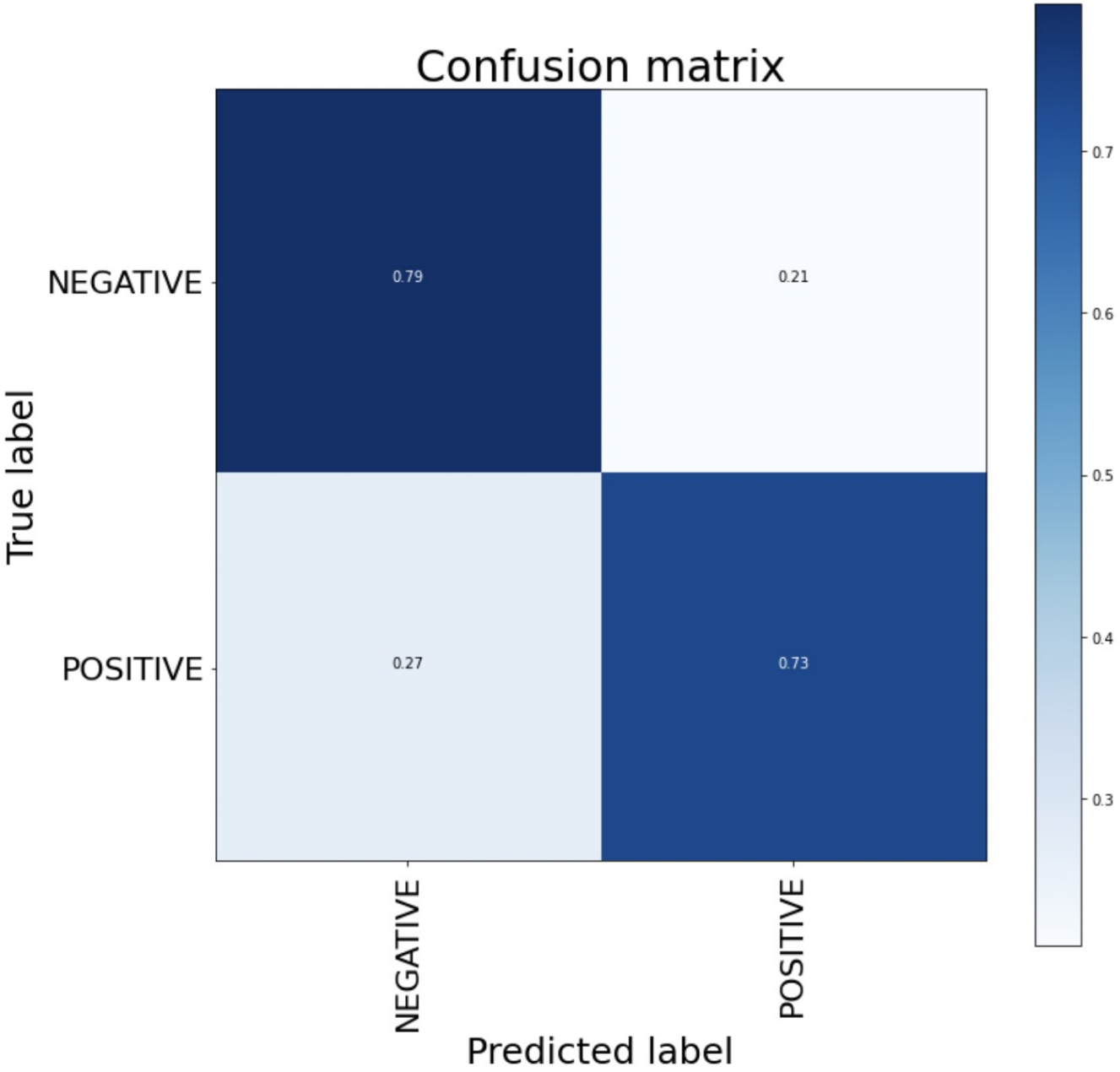

Figure 7: Confusion Matrix was same for both Models M1 and M2

4.6. Naive Bayes vs SVM for Sentiment Analysis of Tweets

SVM models were similarly developed by my project partner Aman Sarawgi. Hence I would like to compare here the performance of Naive Bayes and SVM classifier. SVM takes a lot of time especially on a large dataset such as this with 1.6 Million records. Hence due to limitations of hardware on our local machines it was decided to train SVM model on a subset of 200,000 records from the original dataset.

Figure 8 shows performance of SVM on stemmed text.

```
Accuracy score: 0.760975
Precision score: 0.7511285859909713
Recall score: 0.7772754671488848
```

	precision	recall	f1-score	support
NEGATIVE	0.77	0.74	0.76	20092
POSITIVE	0.75	0.78	0.76	19908
accuracy			0.76	40000
macro avg	0.76	0.76	0.76	40000
weighted avg	0.76	0.76	0.76	40000

Figure 8: Performance of SVM on stemmed text

Figure 9 shows performance of SVM on lemmatized text.

```
Accuracy score: 0.75975
Precision score: 0.7484558965450685
Recall score: 0.7791340164757886
F1 score: 0.7634869068714315
```

	precision	recall	f1-score	support
NEGATIVE	0.77	0.74	0.76	20092
POSITIVE	0.75	0.78	0.76	19908
accuracy			0.76	40000
macro avg	0.76	0.76	0.76	40000
weighted avg	0.76	0.76	0.76	40000

Figure 9: Performance of SVM on lemmatized text

Figure 10 shows confusion matrix for SVM which turned out to be the same for both the models which were trained on stemmed and lemmatized texts respectively.

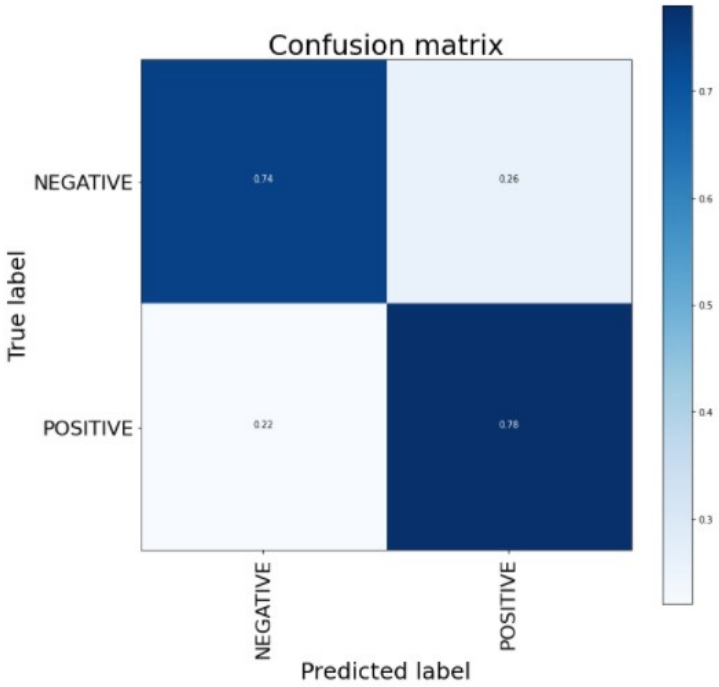

Figure 10: Confusion Matrix was same for SVM run on stemmed and lemmatized text

It can be clearly seen that the Naive Bayes Model outperforms SVM classifier on the given dataset for the task of sentiment analysis of tweets. It can also be noted that for the given dataset the two approaches of stemming and lemmatization do not offer any difference in performance given either classifiers Naive Bayes or SVM.

4.7. Testing Resilience of the Model

One feedback that I had received was how trustworthy are the target class labels in the dataset. I have previously explained that the dataset was labeled with sentiment polarity as either positive or negative by looking at emoticons present in the text. A general rule of thumb used was that if a tweet contains an emoticon which relates to a positive sentiment like happy, smiling etc then the overall sentiment for the tweet was labeled as positive or else negative in other cases. The key idea behind this approach was that such an assumption helps us automate labelling the data which otherwise would be restricted to manual effort and building a large dataset in order of millions of records won't be possible.

The hope here is that even this assumption will play out well and the methods described above would be able to learn important tokens (words) which capture the correct sentiment and hence guide the Naive Bayes classifier to perform well. But this approach certainly raises concern on the authenticity of the target labels in dataset and thereby resilience of my model towards presence of such mislabels in dataset.

For this reason I retrained my model on three additional datasets which had 1%, 3% and 5% mislabeled records in addition to the original 200,000 records.

Figure 11 shows performance of my model applied on stemmed text of 1% mislabeled dataset

```
Accuracy score: 0.7532425742574257
Precision score: 0.7719196192490746
Recall score: 0.720732730953439
F1 score: 0.7454485100732834
              precision    recall  f1-score   support

    NEGATIVE       0.74      0.79      0.76     20147
    POSITIVE       0.77      0.72      0.75     20253

    accuracy                           0.75     40400
   macro avg       0.75      0.75      0.75     40400
weighted avg       0.75      0.75      0.75     40400
```

Figure 11: Performance of Model M1 applied on stemmed text of 1% mislabeled dataset

Figure 12 shows performance of my model applied on lemmatized text of 1% mislabeled dataset

```
Accuracy score: 0.7543316831683168
Precision score: 0.7714751340553044
Recall score: 0.7245840122450995
F1 score: 0.7472947167409293
              precision    recall  f1-score   support

    NEGATIVE       0.74      0.78      0.76     20147
    POSITIVE       0.77      0.72      0.75     20253

    accuracy                           0.75     40400
   macro avg       0.76      0.75      0.75     40400
weighted avg       0.76      0.75      0.75     40400
```

Figure 12: Performance of Model M2 applied on lemmatized text of 1% mislabeled dataset

Figure 13 shows performance of my model applied on stemmed text of 3% mislabeled dataset

```
Accuracy score: 0.7464805825242719
Precision score: 0.7584693877551021
Recall score: 0.7224571123098605
F1 score: 0.7400253876596063
              precision    recall  f1-score   support

    NEGATIVE       0.74      0.77      0.75     20623
    POSITIVE       0.76      0.72      0.74     20577

    accuracy                           0.75     41200
   macro avg       0.75      0.75      0.75     41200
weighted avg       0.75      0.75      0.75     41200
```

Figure 13: Performance of Model M1 applied on stemmed text of 3% mislabeled dataset

Figure 14 shows performance of my model applied on lemmatized text of 3% mislabeled dataset

```
Accuracy score: 0.7462621359223301
Precision score: 0.7543083957192384
Recall score: 0.7296010108373426
F1 score: 0.7417490118577075
              precision    recall  f1-score   support

    NEGATIVE       0.74      0.76      0.75     20623
    POSITIVE       0.75      0.73      0.74     20577

    accuracy                           0.75     41200
   macro avg       0.75      0.75      0.75     41200
weighted avg       0.75      0.75      0.75     41200
```

Figure 14: Performance of Model M2 applied on lemmatized text of 3% mislabeled dataset

Figure 15 shows performance of my model applied on stemmed text of 5% mislabeled dataset

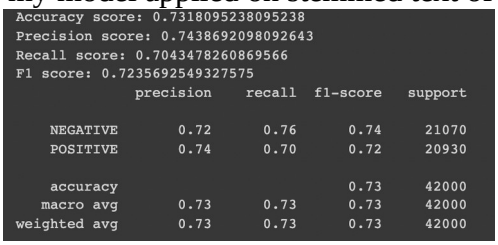

Figure 15: Performance of Model M1 applied on stemmed text of 5% mislabeled dataset

Figure 16 shows performance of my model applied on lemmatized text of 5% mislabeled dataset

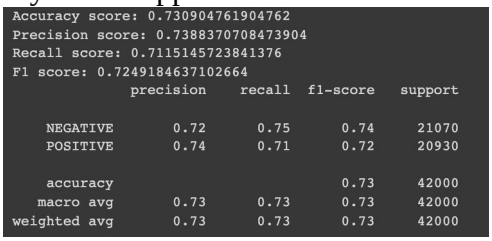

Figure 16: Performance of Model M2 applied on lemmatized text of 5% mislabeled dataset

5 Conclusion

It is evident from the above findings that Naive Bayes performs better for analyzing a document data as such. Hence for the current task of Sentiment Analysis of Tweets Naive Bayes (which is much simpler and somewhat more efficient than SVM) classifier run on a document term matrix formed from lemmatization should be used. Although in our findings for the given dataset, performance of Naive Bayes on lemmatization and stemming was similar, still lemmatization should be used as a pre-processing step. Lemmatization always ensures that the obtained word is a dictionary word which is not always the case for stemming. Lemmatization also takes into consideration the context and part of speech of the text before giving its root word. Stemming behaves greedily by often truncating commonly occurring sets of letters in the end like various participles like ing, ly etc. Hence in general lemmatization would produce better results for a different dataset.

6. Key references

[1] Alec Go, Richa Bhayani and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Stanford, 2009.

[2] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 79-86, 2002.

[3] T. Wilson, J. Wiebe, and P. Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In Proceedings of Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), Vancouver, CA, 2005.

[4] Huang, X., Li, Z., Wang, C., & Ning, H. (2020). Identifying disaster related social media for rapid response: a visual-textual fused CNN architecture. International Journal of Digital Earth, 13(9), 1017–1039

[5] N. Cristianini and J. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, March 2000.

[6] D. O. Computer, C. wei Hsu, C. chung Chang, and C. jen Lin. A practical guide to support vector classification chih-wei hsu, chih-chung chang, and chih-jen lin. Technical report, 2003.

[7] Pouria Kaviani and Mrs. Sunita Dhotre. Short Survey on Naive Bayes Algorithm. International Journal of Advance Engineering and Research Development Volume 4, Issue 11, November -2017

[8] Symeon Symeonidis, Dimitrios Effrosynidis and Avi Arampatzis. A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis. Expert Systems with Applications Volume 110, 15 November 2018.

[9] Bhumika Gupta, Monika Negi, Kanika Vishwakarma, Goldi Rawat and Priyanka Badhani. Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python. International Journal of Computer Applications (0975 – 8887) Volume 165 – No.9, May 2017

[10] Fajri Koto and Mirna Adriani. A Comparative Study on Twitter Sentiment Analysis: Which Features are Good. International Conference on Application of Natural Language to Information Systems. NLDB 2015: Natural Language Processing and Information Systems pp 453-457. 04 June 2015.

[11] Swati Sharma and Mamta Bansal. Stemming and Lemmatization of Tweets for Sentiment Analysis using R. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-2, July 2019