Financial Sentiment Analysis using Natural Language Processing

Name: **Nitin Kumar Singh**
Registration No./Roll No.: 18MA20028

1 Introduction

Text analysis, as a whole, is an emerging field of study in Artificial Intelligence domain. Arenas like Stock Marketing, Media, Product Management, Academia, and Governance are already dwelling into the process of analyzing and extracting information from textual data. Text classification is an important segment of text analysis. In general, Text classification can be defined as a process in which texts are classified into categories depending upon its feature. Text generally are composed of words, adjective, noun etc whose structure is human understandable but computer cant process it. In order to make the text readable to the computers for the classification or any other task, there is a need for extraction of statistics/features from the data. These statistics can range from the fundamental frequency count of each word to more complex sentiment count. A variety of methods are used to extract such features/statistics from the text also called as pre-processing task. The end result of a pre-processing task is a count matrix consisting of all the information in the text data provided by the user. Several previous research suggested that pre-processing is one of the most important part of text classification as the results widely depend upon the feature/ sentiments extracted from the data. A good pre-processing is one which is capable of collecting all the information accurately from the text data with minimum scope of error from the text for classification or any other task. In this report, we use a few popular pre-processing techniques which are mentioned in the section below along with their comparison. After the pre-processing the data is converted into a count matrix whose column acts like features(which is essentially the word count, in this case) and rows are the no of text examples we had. The dataset used is based on the a financial news corpus consisting of a collection of 2264 sentences with manually annotated class labels from [1]. The class labels are either "positive" or "negative" or "neutral". The significance of the class labels is that the news i.e, the given sentences may have positive, negative or neutral influence on the stock price. As a result, sentences which have a sentiment that is not relevant from an economic or

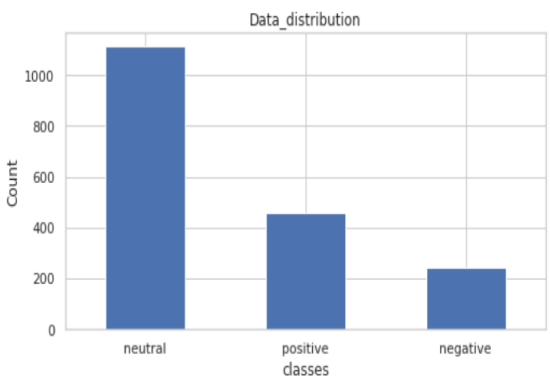

Figure 1: Overview of Corpus

financial perspective are considered neutral. The distribution of the data over class labels in depicted in the Fig. 1. The count matrix thus obtained after pre-processing has 2264 rows and

5205(Bag of Words) or 300(Word2Vec) columns. The pre-processed data is then divided into test and train set using sklearn train ans test split function into an 80:20 split ratio of train ans test sets. The train set is then used to train the classifiers(mention in sections below). To get the best performance of from the classifier parameter tuning is done using gridsearchcv. Then the performance of these classifiers are calculated in form of macro and micro F1 score using the test set(comparison and analysis are given below). The best pipeline is then used to obtain the labels on the test data which is logistic regressions on BOG pipeline.

2 Methods

2.1 Prepossessing Techniques:

As described in the above section preprocessing technique used is one of the important deciding factor towards the classification accuracy. The preproessing mainly includes cleaning(removing unwanted character) and organizing the data(lemmetization, stemming, n grams,etc) followed by computing the statistics(word count, sentiment count, etc.) using feature extraction methods on the data. In this section be briefly describe about different methods used for cleaning and structuring the financial news corpus data.

2.1.1 Tokenization:

We use NLTK tokenizer for breaking the scentences into a list of strings. NLTK (Natural Language Toolkit) is an open-source Python library for NLP. For example: "God is Great! I won a lottery." is broken into ['God', 'is', 'Great', '!', 'I', 'won', 'a', 'lottery', '.']

2.1.2 StopWord Removal:

Stop words are those words in the text which does not add any meaning to the sentence. Example in English e.g., a, an, the etc. Removal of these words helps in reducing the feature dimension of the text making it easier to process the data further.

2.1.3 Ngrams:

n-gram is a contiguous sequence of n items in a given text. Where n can range from 1 to any natural number. Sometimes two words together signify better context rather than their individual representation. n gram helps to capture these meaningful interpretations. For Example "Let's go for shopping". If n=2 (i.e., bigrams), then the n-grams are: 'Let's go', 'go for', 'for shopping'.

2.1.4 Stemming:

The goal of stemming is to convert the different variational grammatical words into their base word. For example 'playing', 'played', 'plays' are converted into 'play'.

2.1.5 Lemmatization:

Lemmatization aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. For example 'corpora' is convrted to 'corpus'

2.2 Feature- Extraction

After preprocessing the text we need to extract the feature. Here we describe two method of obtaining The count matrix.

2.2.1 Bag of Words

Bag-of-words model is a simple yet popular method for feature extraction and it based on the intuition that similar sentences have similar content. Bag of words technique can convert variable-length texts into a fixed-length vector. We use CountVectorizer and tdif transformation for obtaining the count matrix which has 2264 rows and 5205 columns. TF-IDF[1] calculates the term frequency and inverse term frequency to capture the importance of each kind of words. To understand the importance of each of the preprocessing techniques we used the count vectorizer followed by tdidftransform along with ngram(2,3) and the option from the first column on logistic regression classifier as shown in table 1.

preprocessing_used (Bag of Words)	precision	recall	micro_f1	macro_f1
no_preprocessing(only tokenizer)	0.79	0.79	0.68	0.79
only_stopword	0.75	0.75	0.61	0.75
only_stemmed_words	0.89	0.89	0.85	0.89
only_lemmatized_words	0.89	0.89	0.84	0.89
stopword + stemmed_words	0.89	0.89	0.85	0.89
stopword +lemmatized_words	0.89	0.89	0.84	0.89

Table 1: Performance Of Different preprocessing methods along with Bag of word feature extraction method on logistic regression classifier

2.2.2 Word2Vec

Word2Vec is a recent breakthrough in the world of NLP which generated word embeddings. Word embeddings is a technique where individual words are transformed into a numerical representation of the word (a vector). Where each word is mapped to one vector, this vector is then learned in a way which resembles a neural network.The effectiveness of Word2Vec comes from its ability to group together vectors of similar words. Given a large enough dataset, Word2Vec can make strong estimates about a words meaning based on their occurrences in the text Thus gives semantic information. These estimates yield word associations with other words in the corpus. For example, words like "King" and "Queen" would be very similar with one another. Its architecture consists of approaches, CBOW and skip gram [2]. Similar to Bag of words model , to understand the importance of each of the preprocessing techniques we used the word2vec feature extraction method from[2] along with ngram(2,3) and the option from the first column on logistic regression classifier as shown in table 2.

preprocessing_used(Word2Vec)	precision	recall	micro_f1	macro_f1
no_preprocessing(only tokennizer)	0.83	0.83	0.76	0.83
only_stopwords	0.80	0.80	0.73	0.80
only_stemmed_words	0.81	0.81	0.73	0.81
only_lemmatized_words	0.82	0.82	0.76	0.82
stopword + stemmed_words	0.78	0.78	0.70	0.78
stopword +lemmatized_words	0.80	0.80	0.72	0.80

Table 2: Performance Of Different preprocessing methods along with word2vec feature extraction method on logistic regression classifier

2.3 Models Used:

After looking into the effectiveness of different pre-processing and feature extraction technique we choose the best preprocessing and feature extraction method to compare the performance of different

[1] https://scikit-learn.org/stable/modules/generated/sklearn.feature$_e$*xtraction.text.TfidfVectorizer.html*

[2] https://code.google.com/archive/p/word2vec/

models. We use sklearn grid search[3] to obtain the best set of hyperparemeters for each classifier. Below, is a brief description of classifier along with the tools used for implementation.

Logistic Regression: to analyze the performance of logistic regression we use tools[4] used to implement the classifiers [3, 4].

MultinomialNB: to analyze the performance of naive Bayes we use tools[5] used to implement the classifiers [3, 4].

SVC: to analyze the performance of SVC we use tools[6] used to implement the classifiers [3, 4].

Random Forest: to analyze the performance of random forest we use tools[7] used to implement the classifiers [3, 4].

Classifiers	precision	recall	micro_f1	macro_f1
logistic_regression	0.90	0.90	0.87	0.90
multinomial_naive_bayes	0.81	0.81	0.73	0.81
svm	0.88	0.88	0.83	0.88
random_forest	0.83	0.83	0.76	0.84

Table 3: Performance Of Different Classifiers Using Bag of words model with best set of preprocessing on All Terms

The table 3 shows the performance of different classifier on best preprocessing technique using bag of words feature extraction method which gives best performance comparison to word2vec model(shown in table 4)

Classifier	precision	recall	micro_f1	macro_f1
logistic_regression	0.82	0.82	0.76	0.82
SVM	0.76	0.76	0.62	0.76
random_forest	0.77	0.77	0.64	0.77

Table 4: Performance Of Different Classifiers Using Word2Vec model with best set of preprocessing on All Terms

3 Evaluation Criteria

To performance of each classifier is expressed in terms of

Recall: Recall is the ratio of correctly predicted positive observations to the all observations in actual class

$$Recall = \frac{TP}{TP + FN}$$

Precision: Precision is the ratio of correctly predicted positive observations to the total predicted positive observations

$$Precision = \frac{TP}{TP + FP}$$

F1 score: F1 Score is the weighted average of Precision and Recall. this score takes both false

[3] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[4] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html sklearn.linear_model.LogisticRegression

[5] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html sklearn.naive_bayes.MultinomialNB

[6] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[7] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html sklearn.ensemble.RandomForestRegressor

Table 5: Confusion Matrices of LR and MNB Respectively on BOG pipeline Using All Terms

	predicted class		
	Negative	Neural	Positive
Negative	40	5	4
Neural	4	211	8
Positive	7	9	75

logistic regression

	predicted class		
	Negative	Neural	Positive
Negative	26	11	12
Neural	4	211	8
Positive	8	24	59

Naive Bayes

Table 6: Confusion Matrices of SVM and RF Respectively on BOG pipeline Using All Terms

	predicted class		
	Negative	Neural	Positive
Negative	33	7	9
Neural	0	216	7
Positive	2	20	69

SVM

	predicted class		
	Negative	Neural	Positive
Negative	26	13	10
Neural	0	213	10
Positive	6	20	65

Random Forest

positives and false negatives into account. F1 is usually more useful than accuracy, especially if you have an uneven class distribution.

$$F1\_Score = \frac{2 * (Recall * Precision)}{Recall + Precision}$$

Micro f1 score: Calculate metrics globally by counting the total true positives, false negatives and false positives.
Macro f1 score: Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

We also provide **Confusion Matrix** also called as error matrix, is a specific table which helps in evaluating the performance of classification. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa.

4 Analysis of Results

In this section we describe the observation from the evaluation matrices from tables1 to 4 and the confusion matrices and analyse the result simultaneously.

- From Table 1 it can be inferred that using stemming and lematization along with stopword removal both gives best performance in BOG model. Therefore suggest that data cleaning is important as performnace significantly increased from no preprocessing one(row1 of table 1)
- In case of Word2Vec, from Table 1 it can be inferred that removal of stopword harms the classifier performance. Also teh best performance is achieved when no processing or only lemmetization is used. Therefor suggesting stopwords are import for Word2vec model.
- Altogether, from Table 1 and Table 2 we can infer that overall BOG performs better that Word2Vec feature extraction module when different preprocessing techniques are used. Where as when no preprocessing is used WorD2Vec performs better.

- Table 3 shows the performance of different classifier on the best feature extraction method (BOG) as discussed above. Logistic Regression outperforms other classifiers followed by SVM, Random forest and then Naive Bayes in BOG pipeline.

- Though we already know BOG outperform Word2Vec. Further it can be inferred from Table 4 that logistic regression outperform other model followed by Random Forest and SVM in the Word2Vec pipeline. Here we couldn't calculate the f1 score for Multinomial Naive Bayes because as it doesn't accept negative values.

- For the negative class classification Logistic regression performs better than other model. 40 of the text are classifised correctly to negative class and only 5 and 4 text are classified falsely into neural and positive classes respectively as depicted by Table 5 and Table 6. The worst performance in given by Naive Bayes and Random Forest as nearly half of the text is falsely classified.

- SVM outperform other model in regards to neural class classification by classing almost all the text (216 in count) correctly to neural classes and only 7 and 0 text are falsely classified into positive and negative respectively, demonstrated by Table 5 and Table 6. Where as the performance of the other model are in near vicinity only. Which suggest that among the all the classes Neural class is more accurately classified.

- For the positive class classification the outcomes are quite similar to the neagative class classification. Logistic regression performs better than other model where 75 of the text are classifised correctly to positive class and only 9 and 7 text are classified falsely into neural and negative classes respectively as depicted by Table 5 and Table 6. The worst performance in given by Naive Bayes where 1/5 th of text is falsely classified.

5 Discussions and Conclusion

- Irrespective of feature extraction method, Logistic regression outperforms all the models. Upon in depth analysis of the result one can say feature extraction methods play a important role towards classifier performnace. Several preprocessing technique also impact the performnace based on the kind of dataset in use.

- BOG pipeline perform better than Word2Vec in our dataset also suggesting stopword are import for Word2Vec.

- In future several other robust feature extraction methods like FastText can be used for better sentimental analysis along with better preprocessing techniques.

References

[1] Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65(4):782–796, 2014.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[4] I. H. Witten, E. Frank, and M. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, third edition, 2011.