How To Talk To Models: A Discussion On Why Details Matter For Text Analysis

March 11, 2021

Bryce Earner **Chan Hyuk Yoon** **Rylan Greer** **Weilin Fu**

1. Introduction

Market prices for assets reflect investors' collective views and expectations for the future stream of cash flows for that asset. Information is continually being released in the forms of news articles, tweets, and SEC filings, among others, that have the potential to change these expectations and thus the prices of these assets. The volume of this information is so great that no person could possibly read it all, and the rate at which it affects market prices is so rapid that nobody could read it fast enough for it to be actionable. Thus, applying natural language processing (NLP) to financial market problems is practically a necessity for any financial market participant who needs to act on text information quickly, or who needs to get information from a large volume of text.

A relative beginner to the field who wants to learn about NLP is immediately faced with a set of challenges. First, most machine learning methods operate on vectors. It's not immediately obvious how to represent a piece of text as a vector, let alone a vector that retains its important features. Even if it were clear how to do this, similar challenges show up as we see in other fields of machine learning: which models are appropriate for which tasks? How can we ensure we do not overfit to our training data? Which parameters in the model will provide the best performance? Our intention is, using Renault's paper as a guide, to provide a beginner with an understanding of the impacts of some of these decisions so that they may know where to begin in performing NLP on text data.

In our project, we mainly studied preprocessing methods on text data and the generalization of the predicting models. Preprocessing methods crucia in NLP, because we need to transform raw text data to features represented by numbers so that we can train supervised models with these data and make predictions. Our research shows unigrams and bigrams perform well, tf-idf works better than simple counter vectorization. Dimension reduction makes models simpler without sacrificing too much accuracy.

Generalization outside of training data is another important aspect. The data we used are labeled as positive or negative by human beings based on their own view, so labels from different people may vary. If a model is robust among datasets labeled by different people, it is more useful. Our experiment shows the generalization to other data sources is hard to achieve. However, if we can mix some data from the target dataset, performance improves significantly.

2. Overview of Report

Thomas Renault's paper (Renault, 2020) offers a walkthrough of transforming data from StockTwits to appropriate inputs to a few machine learning models, as well as some discussion of his results. In it, he reveals a set of decisions that typically lead to higher

performance (as measured by accuracy) on four machine learning models: a multinomial naive Bayes model, a maximum entropy classifier, a support vector classifier, a random forest classifier, and a multi-layer perceptron.

We used 5000 financial tweets, labeled by "positive" or "negative" sentiment, from Kaggle[1] (as well as a separate set in later tests[2]), as opposed to the one million messages Renault used. Using his paper, we determined a "baseline" group of settings, selected as they were indicated to be high-performing in the paper, to modify and compare against. Specifically, we used the following (with definitions to be provided in the appropriate sections.)

Preprocessing Toolkit: Natural Language Toolkit (NLTK)
Data: 5000 tweets
ngram: unigrams and bigrams
Stop Words: Included
Lemmatization[3]: Yes
Punctuation: Yes
Vectorization: CountVectorizer (Bag of Words)
Prediction Models: Naive Bayes, Logistic Regression, SVC, RF
Minimum Word Frequency: Enforce minimum word frequency of 3
Dimension Reduction: None

In the text preprocessing stage, we considered the impact of using different types of vectorization, how many ngrams to include, and whether or not to include stop words. We considered the impact of training on one set of financial tweets and predicting on another, as well as training on a mixed set of financial tweets (from two different sources.) We also considered the impacts of dimension reduction.

It is worth noting that since we chose the "baseline" to be the best performing settings in the paper, we expect that any changes will lead to worse performance; our goal is to see how the performance changes and elucidate why this is happening.

In order to see whether our model provides interpretable output, we observed the relationship between coefficients and words. The result shows that our model learned useful knowledge from the data set, which can also be understood by human beings. For example, we list the top ten "positive" words and top ten "negative" words based on the scale of the coefficients on the logistic regression model. As hoped, words with a positive connotation are associated with "positive" labels, while words with a negative connotation are paired with "negative" labels.

Top Ten Positive	Top Ten Negative
'nice', 'above',	'short', 'down',
'on', 'higher',	'shot', 'lower',
'long', 'up',	'puts', 'below',
'highs', 'bullish',	'bearish', 'red',
'entry', 'breakout',	'under', 'sell',

3. Methods

A goal of this paper is to explore how text preprocessing, vectorization, and dimension reduction affect how sentiment is classified in our datasets. To that end, we consciously decided not to cross validate the classification models for each perturbation of the default settings. Doing so would have muddied our ability to interpret the output. For example, did the accuracy increase because dimension

1. https://www.kaggle.com/yash612/stockmarket-sentiment-dataset
2. https://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-news
3. We did not adjust lemmatization, but we included it as part of our model. Lemmatization is a procedure that converts words into their base form. For a more complete overview of lemmatization, we direct a reader towards the following resource: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

reduction is beneficial, or because the random grid search happened to find a better optimum? Avoiding questions like these allows us to narrow our focus to the text analysis, and leave the optimal cross validation for the end user.

All four models use the scikit-learn implementations with the default parameters chosen. A brief explanation is given where the default parameters are not necessarily obvious.

Gaussian Naïve Bayes: GaussianNB()

Logistic Regression: LogisticRegression()

A logistic regression with an intercept is applied with an l2 penalty. No class weights to deal with data imbalances.

Support Vector Classifier: SVC()

A radial basis function kernel is used.

Random Forest: RandomForestClassifier()

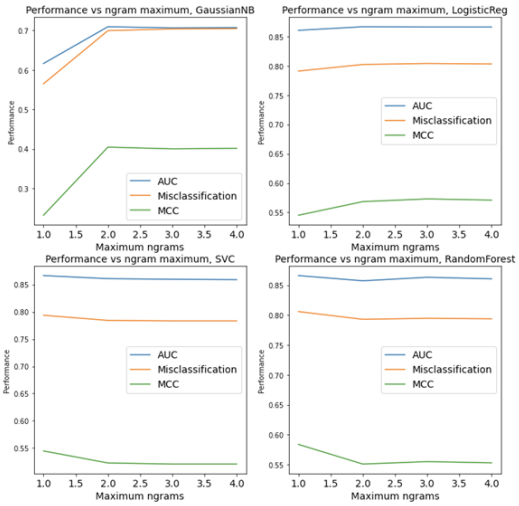
100 trees by default. Using Gini Index as the measure of split quality. Looks at a random sample the size of the square root of the number of features at each split. No class weights to deal with data imbalances.

4. Limitations

Our report is based on a much smaller dataset than Renault's, which could be an unrepresentative sample of stock tweets. In general, following the paper, our approach is to summarize the impact of a lot of decisions, but as a result, it is not feasible for us to study any one impact or method with great depth. Lastly, we consider only the impacts of each change on the base model; that is to say, we do not consider how any of these methods work in tandem. It is possible that, for example, having an increased number of ngrams, plus dimension reduction, leads to much better performance. Considering all possible combinations of changes leads to much too large a search space, however, and does not necessarily serve our educational goals. Similarly, the models we use are not hypertuned and use default settings; this is similar to the approach taken by Renault in his paper.

5. Impact of changing ngrams

The number of features in the text depends strongly on the ngrams selected for the text vectorization. Put simply, an ngram is n adjacent words in the text; e.g. a 2-gram representation of the sentence "Hello, how are you?" would be ("Hello, how", "how are", "are you?"). Increasing the ngrams theoretically allows more meaning to be extracted from the text, at the cost of increasing computational complexity. In the paper, it is suggested that bigrams are a significant improvement over unigrams, but that there is not much performance increase beyond using unigrams plus bigrams.



In our testing, we see similar results. The difference between using unigrams and bigrams is relatively large, but there does not appear to be much advantage beyond this.

6. Vectorization

One important step that must be taken is to convert from human readable tweets to machine readable numbers while preserving as much important information as possible. Vectorization is typically the name given to this process. One common approach, called Bag of Words[4], is to simply count the number of times a term (a word or ngram) occurs in a given string. The resulting output is a matrix where each row is the original tweet, each column is a unique term, and each entry is the count of a particular term in a particular tweet. Notice that this implies that each term has the same importance, since each column is just the count. In scikit-learn, this is implemented as CountVectorizer and serves as the "baseline" vectorizer mentioned above.

Tf-idf (term frequency-inverse document frequency)[5] takes a different approach. Term frequency measures how frequently a given term t appears in a tweet relative to the number of unique terms. Inverse document frequency measures a term's importance by relating the number of total tweets to the number of tweets with a given term t.

$$\text{tf(t)} = \frac{\text{\# of times the term t appears in the tweet}}{\text{\# of unique terms in the tweet}}$$

$$\text{idf(t)} = \ln\left(\frac{\text{\# of tweets in the dataset}}{\text{\# of tweets with term t in it}}\right)$$

Consider a user tweeting "Apple will go up" and another user tweeting "Tesla might go down". Tf-idf helps in a scenario like this by effectively penalizing the word "go". In the first tweet, tf("go") will return a value of 0.25 whereas idf("go") will return $\ln(2/2) = 0$ leading to a tf-idf(t) score of $0.25 \times 0 = 0$. In this extreme case, since "go" appears in every tweet (both positive and negative sentiment), then it contains no additional information to aid in classifying sentiment.

Looking at the table below, we see that tf-idf produces very promising results. For example, the AUC for all models increases when compared to the baseline, indicating that overall the models are able to better predict the correct class, although depending on the use case, a specific part of the curve may be of more interest. Comparing tf-idf to just term frequency we see that again the AUC is strictly larger for tf-idf, although results vary when measured by accuracy.

Accuracy	GaussianNB	LogisticReg	SVC	RandomForest
Baseline (Bag of Words)	78.95%	80.24%	78.43%	79.12%
TF-IDF With Baseline Parameters	75.75%	78.95%	79.55%	79.29%
TF (Removing IDF)	74.37%	78.60%	79.81%	79.12%
AUC				
Baseline (Bag of Words)	85.78%	86.68%	86.10%	85.94%
TF-IDF With Baseline Parameters	86.39%	87.62%	87.68%	86.74%
TF (Removing IDF)	85.83%	86.77%	87.13%	86.26%

Overall, tf-idf appears to be a useful change to make when compared to the simple bag of words approach. It adds (almost) no complexity to the modelling process, improves results for most classifiers, and aligns well with our intuition that words should have less predictive power if they are included in almost every tweet.

7. Stop Words

Stop words are words which are commonly used in a language that provide relatively little information[6]. For example, given sentences "how is the market condition?" articles such as "the" provide less information compared to words such as "how", "market" and "condition". While stop words are generally removed in preprocessing, the paper states that inclusion of stop words increases the classification accuracy. This is

4. https://scikit-learn.org/stable/modules/feature_extraction.html
5. http://www.tfidf.com/
6. https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

because words such as “up”, “down”, “below” or “above” could be more useful in a financial context than typical applications.

Model	GaussianNB	LogisticReg	SVC	RandomForest
Classification Acc.				
Included	69.974%	80.242%	78.430%	79.897%
Excluded	66.178%	80.155%	78.171%	78.516%
Difference	3.796%	0.086%	0.259%	1.381%
AUC				
Included	70.960%	86.682%	74.093%	86.141%
Excluded	68.833%	86.533%	74.132%	85.165%
Difference	2.127%	0.149%	-0.039%	0.977%

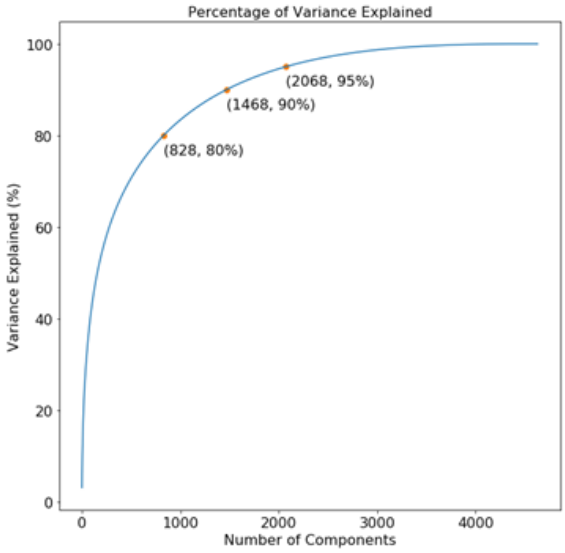
Our analysis shows similar results. Inclusion of stop words in the preprocessing stage increases classification accuracy for all models except for a slight decrease in SVC. Though not tested, we expect further inclusion of punctuations such as exclamation marks and emojis, if handled appropriately, would increase the out of sample accuracy as stated in the paper.

8. Dimension Reduction

After the preprocessing and vectorization, we are left with a dataset containing 5791 rows and 6298 columns. Each row corresponds to a tweet in the original dataset, and each column corresponds to an ngram found in the original dataset. Using a train-test split of 80%-20%, this leaves us 4632 rows and 6298 columns to train on. Since there are many more features than samples, it could be greatly beneficial to reduce the dimension of the data.

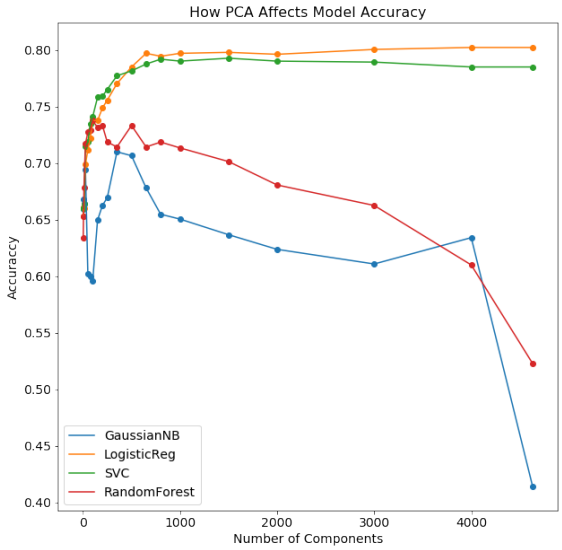
While nonlinear dimension reduction approaches could be explored, we decided to use PCA due to the simplicity in which it can deal with new observations (from the same corpus of words). Needing to identify the nonlinear transformation for a new observation coupled with the potential that the observation contains a word unseen by the model complicates the cause-and-effect relationships we hope to discover.

The graph below shows how the total variance explained by the first k principal components as k increases. We can see that a relatively large number of components are needed in order to preserve much of the variance in the original data. For example, to retain 95% of the variance we need 2068 columns, roughly a third of the original, but would still leave us with only two tweets for every column.



Nevertheless, the next chart shows how the support vector classifier and the logistic regression model both maintain their levels of accuracy as the dimension is reduced from the 4000 range down to around 500, counter to what the variance explained might have suggested. One interpretation of this is that much of the variance in the original data set is not contributing to our ability to classify. This makes sense, as we are using both unigrams and bigrams, meaning that it’s unlikely we truly need all of the variance between columns since some may contain very similar information to others.

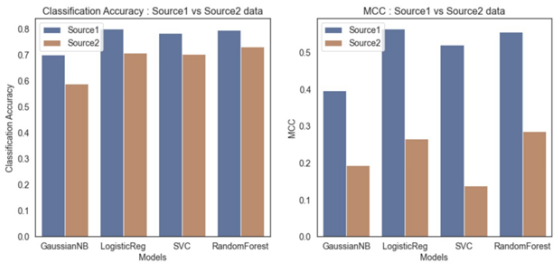
One important thing to note is that these models were not cross-validated at each dimension reduction step. The purpose of this exercise was not to try every combination of dimensions and hyperparameters to find the



"best" settings (on this particular dataset, and as measured by accuracy), but rather to explore greedily on what rules of thumb could be obtained for future sentiment analysis work. We can see that the untuned random forest and Gaussian naive Bayes models struggle to maintain their accuracy as the number of columns tends towards the number of rows. Similar to other sections in this report, the logistic regression and support vector classifiers perform very well, and the smaller dimension greatly reduces training time, allowing for more exhaustive cross-validation procedures in the future. Use of SVD provides nearly identical findings. Lastly, in some applications, it may be noteworthy that models trained with dimension-reduced data train much faster than those trained on the full dataset, although we did not explicitly measure or consider training time.

9. Generalization To Additional Dataset

One important aspect of Machine Learning is the generalizability of the trained model. After all, the choice of "positive" vs "negative" sentiment labels can be subjective and hard to decide in some cases. Thus, we wanted to see if the model can effectively predict labels on another dataset, which has been classified "positive" or "negative" by another user. Specifically, we trained on 80% of the original data (Source 1) and compared the performance of that model on the remaining 20% of the original dataset with its performance on a dataset from a separate source (Source 2).



Our results show that generalization to Source 2 is difficult. At first glance, based on classification accuracy it seems that generalization is possible with accuracy on Source 2 revolving around 70%. However, after inspecting the confusion matrix and the Matthews Correlation Coefficient[7] (MCC), we find opposing results.

$$\text{MCC} = \frac{\text{TP}\times\text{TN}-\text{FP}\times\text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$$

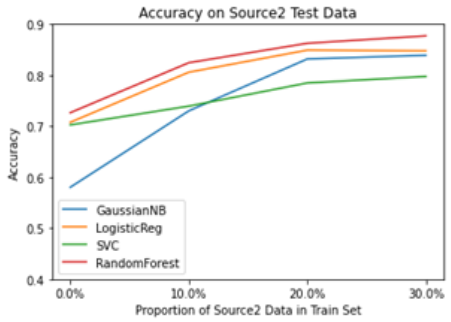
The confusion matrix on Source 2 output shows a high False Positive rate, which indicates the model's inclination to guess Y=1 regardless of the input. Considering the fact that the additional testing data set has about

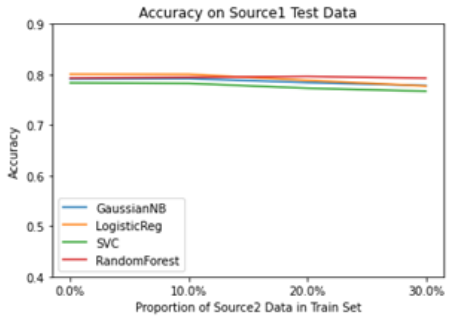
7. MCC is used in the paper. The definition is provided in our report. For further discussion, we direct the reader towards the following resource: https://lettier.github.io/posts/2016-08-05-matthews-correlation-coefficient.html

30% "negative" label, we can conclude that 70% of testing accuracy is achieved by the model guessing a large majority of the inputs as correct; when testing on data from the original source, we see a more balanced confusion matrix.

10. Training With Mixed Data

Although the above result shows it is hard to generalize only using models trained on original data (Source 1), we find the performance could be dramatically improved if we only add a small amount of the data from the separate source (Source 2) to the training set. We fixed the size of the training set but replaced some proportion of the training data with entries from the separate data source (which were then excluded from use as test data.) We then tested on the portion of the held-out dataset which was not used to train the model.





The above two figures show the models' accuracy when we change the proportion of data from another source in the training set. From the two figures, we can see that adding more of the data from another source to the training set will increase the performance on the test set drawn from data from that source. At the same time, the performance on the test set drawn from the original data source keeps around the same. This suggests there are serious benefits and few drawbacks to including a diversity of training data.

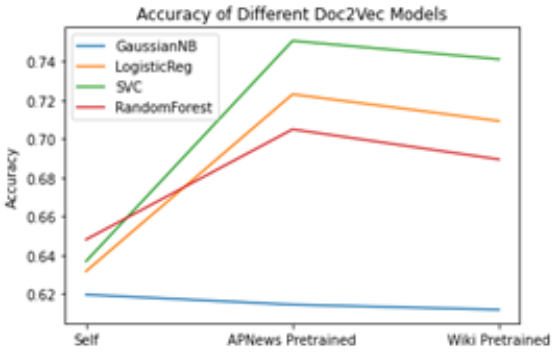
Results on MCC are also in line with this result. Considering the formula below, a high FP proportion decreases the MCC score for additional dataset. In conclusion, generalization of NLP prediction to other datasets cannot be easily achieved.

11. Doc2Vec Method

The Bag of Words and tf-idf vectorization methods focus on the frequency of different words in the corpus. They fail to catch the relationship between different words. However, the relationships between words are very important for both people and machines to understand sentences and their meaning. Word2vec (Mikolov et al., 2013) and Doc2vec (Le and Mikolov, 2014) are two models which are able to capture the relationships between words much better than vectorization methods based on frequency. Word2vec and Doc2vec's theoretical underpinnings are outside the scope of this report; for further information, please see (Mikolov et al., 2013), or any of the variety of brief explainers available online.

As sentences' lengths vary, it is inconvenient to apply the Word2vec model here. Otherwise, the Word2vec model will lead to samples falling into different feature spaces. However, the Doc2vec model is quite suitable for this problem as all titles are transformed to vectors of the same length. We tried three different Doc2vec models. The first model is trained on our training data

set. The other two are pre-trained models. One is based on a corpus from the Associated Press News, and the other is based on text from Wikipedia.



From the above figure, we can conclude the performance of the two pre-trained models are better than the model trained on our data set. The performance does not yet match what we see with the vectorization approach, but we note that the separate pretrained models do show significantly different results, suggesting this could be an area where one may benefit from further exploration.

12. Conclusion

In our task setting, preprocessing on text data is important. Proper preprocessing methods can bring significant improvements in prediction. Unigrams and bigrams are useful, although performance increases from 3grams or higher are marginal. Tf-idf is better than simple vectorization, as it penalizes words that are overused. We retain the interpretability of the bag of words approach, while gaining a more informative measure of a term's relevance. Dimension reduction makes a simpler model without sacrificing too much prediction ability on unseen data. Both our experiment and the paper we followed show including stop words helps form a better prediction.

We also tested our models generalization on an alternate data source, which is labeled by another user. The result shows that the model trained on one data source does not predict as well on an entirely different data source. However, if we add even a small amount of data from that separate data source to the training dataset, the performance on the separate data source will improve significantly.

The Doc2vec model can not achieve as good a result as our models based on simple counter vectorization and tf-idf methods. However, we can see the performance of a pretrained model is much better than the model only trained on our dataset. A few reasons for this are that our dataset is quite small, and that the pretrained models have learned more from larger dataset. It is worth noting that the pretrained models are based on general news and Wikipedia, which are not as focused on the financial industry. Therefore, we anticipate that a pretrained model on financial news will achieve better results.

As with any topic in machine learning, details matter. It may not be correct to state that there are more decisions for a practitioner to make in text analysis compared with other topics, but the fact that fundamental concepts, like how to represent the data as a number, do not have clear answers highlights the variety of decisions that need to be made. In this report, following the lead of Thomas Renault, we have shown that there are a great deal of performance advantages that can be realized by carefully considering the inputs at every step of the process, even without the aid of hypertuning the models. We hope that this report serves as an introduction to the sorts of things a newcomer to the field should consider as they begin learning about natural language processing, and emphasizes the impact that details can have.

References

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

T. Renault. Sentiment analysis and machine learning in finance: a comparison of methods and models on one million messages. *Digital Finance*, 2(1):1–13, 2020. doi: https://doi.org/10.1007/s42521-019-00014-x.