Earnings Transcripts Impact on Analysts' Recommendations

Michael C. Hayes, Grayson Niehaus, and Frank Whitesell

1 Abstract

Natural Language Processing extends into many applications, particularly fields with rich text data. A prominent source of text data in finance are quarterly earning transcripts provided by public companies. At the end of each fiscal quarter, key executives hold a conference call with research analysts from major financial institutions. Based on market conditions and the quarterly performance of the company, research analysts will update their buy, hold, or sell recommendation (using the standard recommendation rating which is on a 1-5 scale).

In our research, we analyze 6400 earning transcripts utilizing common NLP procedures in order to predict how research analysts will respond to earnings calls. Our transcripts are web-scraped from Motleyfool.com and our analyst recommendations are obtained from a Georgia Tech Bloomberg Terminal using an academic license. We explore different feature representations, models, and predictors to conclude that our best model uses a Word2Vec embedding and a Neural Network model. This resulted in 69.9% accuracy and 64.6% F1 score.

Project code repository can be found at: https://github.com/gniehaus/NLP-Earning-Transcripts

2 Introduction & Related Work

What is the motivating problem for your research project? What is the goal of the project? If it has changed since the project began, how has it changed and how close is it to the original goal? The overall goal of the project has not changed significantly from the original proposal. We still aim to accurately quantify relationships between earnings call transcripts and changes in investment analyst ratings following these calls, but our focus has shifted to try and achieve better model predictiveness instead of creating an entirely streamlined process. When we first started our project, we were trying to predict the change in analyst ratings from one quarter to the next. In our final approach, we predict the actual analyst recommendation in the same quarter post earning transcript release. This is a difference of predicting a percentage delta between ratings and the actual rating. This adjustment brought significantly better results. By creating a structure for extracting text from transcripts and modelling on this data, we believe that this work has the potential for scaled-up use by non-technical audiences in the field of finance.

Related Work: Please cite 3-5 related studies and explain their contribution to the problem space (e.g. " Yang and Prasad (2018) found that attention-based models outperform HMM models for part-of-speech tagging "). Compared to prior work, how is your project trying to address these gaps? What's your contribution compared to the previous work? Keith and Stent (Keith and Stent, 2019) use pragmatic features and semantic content for testing both continual and discrete prediction cases. They found in their modeling, they were able to realize a 25 percent error drop in classification accuracy, compared to only using market data as a prediction. Additionally, they were able to determine that measuring the semantic feature of the whole document was much more powerful than measuring only the Q/A portion of the call, as well as the combination of pragmatic features. In their work, they measured the change in stock price before and after a earnings calls. Consequently, they quantized the percentage change for their labels and this resulted in lower F1 scores. Furthermore, (Jha et al., 2015) has focused on analyzing how transcript tones have changed over different economic environments. (ratio of net positive sentences to total sentences). Lastly, (Loughran and McDonald, 2019) implement a sentiment lookup of word:sentiment for each earning transcript.

We explore and expand these concepts in our project. For instance (Keith and Stent, 2019) use Logistic Regression, LSTM, and Emsemble methods for their classification models. They train these models on both the whole document and Q/A portion of the call. We used a word2vec model with tf-idf weighting for each word to represent the whole document as a vector. Finally, we use this embedding with a neural network for our classification task.

3 Methods

3.1 Data

What data did you use for the project? If you did not do so in the midway report, please provide a table of summary statistics for the data to help us understand the scope of your project, e.g. number of documents, average document size, unique number of labels. Please also provide several short examples of the data, e.g. for sentiment analysis, a post labelled for positive sentiment and negative sentiment. Our complete dataset consists of two main components: transcripts and investment analyst ratings. Each transcript and average investment analyst rating pertains to a specific company and quarter. After our original cleaning process, we obtained 3,376 total transcripts with an associated rating change. We have changed our goal to not focus on the analyst rating change, but the actual rating. Since we don't need to shift our predictors, this provided us an additional 1,061 transcripts bringing our total to 4,377. The transcripts of 997 different companies are included in this final dataset, which covers a time period from September 2017 to December 2019. There are a total of 39,000,201 tokens in the corpus, which equates to an average of 8,910 tokens per transcript. Our vocabulary contains 118,364 distinct words.

Table of Data Statistics	
Number of Docs.	4,377
Average Doc. Size (Tokens)	8,910
Unique Num. Labels	3

Investment analyst ratings are based on an integer scale from 1 to 5. A rating of 1 is a "strong sell" recommendation, which indicates that the analyst believes that the price of the stock will increase in the future. A rating of 3 is considered a "hold" recommendation, which means that the investment analyst does not expect the price of the stock to change in the future, or at least not in this quarter. Finally, a rating of 5 is a "strong buy" recommendation. In these cases, the investment analyst believes that the price of the stock is likely to decrease in the future. We have the ability to consider these labels as continuous (e.g. the average analyst recommendation will be 3.8 after this earnings call) or as discrete cases (e.g. the average analyst rating will be a buy, hold, or sell). Currently, we model the changes as discrete cases, where a analyst rating between 1 and 3.49 is considered a negative class (-1), 3.5 to 4.49 is considered a neutral class (0), and 4.5+ is considered a strong buy positive class (1). Our counts of labels are in the below figure.

Table of Label Counts	
Number of Positive Labels (1)	2,367
Number of Neutral Labels (0)	1,144
Number of Negative Labels (-1)	866

An example of a sentence in a negative class document is "Looking into 2019, if the tariffs remain in place, we plan to aggressively reduce the remaining effect in potential change in our prices and further adjustments to our supply chain." An example of a sentence in a positive classified document is "So we feel really good about our channel reach and relationships with the commercial accounts and we're seeing in the numbers." Keep in mind our documents have nearly 9000 tokens, so there's many positive and negative sentences in each document; especially since management is trying to convince analysts of a positive outlook on the company.

In addition, we incorporate a financial sentiment word dictionary (Loughran and McDonald, 2019) into some tasks of our analysis. This dictionary consists of 86,486 different words, with sentiment categories such as "negative", "positive", "uncertain", and "litigious". Examples of frequently-occurring negative words in our dataset include "question", "decline", and "closing". Common positive words include "good", "strong", "better", and "opportunity". "Could", "maybe", and "believe" are examples of uncertain words, and "contract" and "regulatory" are common litigious words.

Obtaining our data was quite messy and required lots of wrangling before we could perform any analysis. Since we web-scraped our data, we had to rely on the similarity of web-pages to standardize each data pull. Some web-pages had different date formats and titles. Once we obtained the data we utilized datetime objects in order to standardize our dates to "YYYY-MM-DD". We stored all of our data into a SQL Lite DB (690KB). Since our labels came from a preformatted database (Bloomberg), this required little formatting changes. Once we had both data and labels we needed to shift our labels one period since we are using time-series data. Our primary key is stock ticker concatenated with data ("AAPL-2019-12-31"). For a quarterly earnings call in January, we predict how analysts will react to a earning call 10-90 days after that call.

3.2 Models/Analysis

What NLP/ML model did you propose or extend? How did you adapt to address the problem? Please describe your model in detail. If necessary, provide a figure to show your model architecture. We tried a variety of models throughout our experiments. All model results are expressed in the results section later in this paper. However, the architecture that performed the best (under a reasonable time constraint - we did not use any services for hardware) was a feed-forward deep learning model using the following details and hyper-parameters:

- We first tokenize all of the transcripts using nltk's word_tokenize function.
- We then use word2vec on the tokenized data to create word embeddings (a word embedding of 500 length worked best).
- We then run the word2vec embedded transcripts through a feed-forward network which has the following architecture: a fully connected layer with an output dimension of 100 with a sigmoid activation function, another fully connected layer with an output dimension of 100 with a relu activation function, and finally an output layer that uses a softmax activation function, which calculates the probability of each individual class (there are three possible class labels for our classification task). We found that it took around 1,500 epochs to train until accuracy started to stagnate (and in some cases, decrease likely due to overfitting). Note that we used a batch size of 32 for both training and testing, "Adam" for our optimizer with a learning rate of 0.001, and a categorical cross-entropy loss function.

Or what analysis/testing did you propose to address the problem? What insights could this kind of testing provide into the problem? If necessary, provide a figure to show your analysis pipeline. Our goal for this problem was not to create a new state-of-the-art architecture for a model, but to apply different NLP techniques to see if we can solve a real world problem. Once we obtained our data, we outlined different models, and document representations in order to obtain our best results. We initially started with a bag-of-words representation, tf-idf, and different ngram values. We used shallow multinomial classifiers such as Logistic Regression, and Naive Bayes. Additionally we attempted basic feed-forward neural network, BERT, and LSTM models. By testing multiple different models, we can determine the best way to represent each document, and if deep learning is necessary to tackle this problem.

3.3 Baseline Models

What baseline models did you choose to use to compare your model or tests against? Please pick at least two baselines for the comparison with your model or tests and describe them in detail. You may find it useful to consider models from prior work or an original model if appropriate for the task, e.g. a logistic regression classifier. We use the shallow algorithms in the above section as our baseline models. We compare the accuracy and F1 score for each of these models against our deep learning model.

Perhaps the most simplistic (or most trivial) model of all, a model that predicts the majority class for all test data, would achieve the following results:

Model	Accuracy	F1 Score
Majority Class Classifier	0.560	0.239

Our baseline classification models utilize tf-idf embeddings with relatively basic machine learning algorithms such as multinomial naive Bayes or multinomial logistic regression. Additionally, we analyzed a classic bag-of-words model as a comparative baseline. We use these models to determine whether the increased complexity of neural network algorithms is needed to accurately model our problem.

TF-IDF Model	Accuracy	F1 Score
Multi. Log. Regression	0.650	0.485
Multi. Naive Bayes	0.551	0.237

The multinomial logistic regression model performed better than the simple majority class classifier, while the multinomial naive Bayes model performed worse. In fact, the multinomial naive Bayes model predicted the majority class for every observation in the test set. The difference in the accuracy is due to the train-test dataset splitting. Overall, the multinomial logistic regression classifier provides a strong baseline level for accuracy, but there is definite room for improvement in the F1 score through the use of higher-complexity models. Given the unbalanced nature of our labels in the data, it is important to note that the F1 Score is a better indicator of model quality than accuracy in this case.

Within this class of baseline models, we tested a number of different variations. First, we compared the performance of stemmed models versus un-stemmed models, and found nearly identical accuracy and F1 scores for each. We also reduced the size of the vocabulary to approximately 11,000 words by removing words which did not appear at least 500 times in the corpus. This reduced both the accuracy and the F1 score by 1-2%.

BOW Model	Accuracy	F1 Score
Multi. Log. Regression	0.566	0.516
Multi. Naive Bayes	0.449	0.447

The results in the above table utilize a bag-of-words representation and each model performed better than our majority class classifier. However, the results were poor compared to approaches using more sophisticated methods of feature representation and deep learning. We tested two different count vectors. The first was with a 1-gram and the second with a 2-gram. We saw slightly better results using a 2-gram model.

We obtained similar results in rating classification using our sentiment dictionary and word frequencies. The logistic regression classifier performed at a level similar to the models listed above. It was able to correctly classify 55.3% of the data points in the test set based on frequency of sentiment words in each transcript, with an F1 score of 0.252. The linear regression we performed using frequency counts of sentiment words resulted in an R-squared (coefficient of determination) value of approximately 0.01, which indicates that the linear regression model based on frequency of sentiment words explains very little of the variance in ratings.

4 Results

4.1 Experiment Setup

What's your data/model configuration details? E.g., you can include the details like the size/ratio of train, development and test set; the learning rate; optimizer. Our deep learning model is tuned based on trial and error attempts, guided by theoretical frameworks. Our model consists of 3 hidden dense layers. Our activation functions are Sigmoid, Relu, and Softmax. Our learning rate is 0.001 (default for the Adam optimizer) and iterate 1,500 times with a batch size of 32. Since this is a classification problem, the Softmax activation is applied to the last layer and the label is assigned to the highest probability.

We broke out data into train, validation, and test sets. The validation set is critical in our mode learning because we implement early stopping. Early stopping computes accuracy on our validation set for each epoch. If the accuracy on the test set does not improve after 1500 epochs, the model stops learning on our train set, and the weights that performed the best on our validation set are returned. We can then use these weights on our test set to obtain valid performance metrics for our model. The below table contains the size of each of our train, validation and test sets.

Table of Train/Val/Test Size	
Train Set	3,545
Validation Set	394
Test Set	438

4.2 Result Comparison

How well did your models perform on a given task? If you are using a standard dataset, report performance on the standard test split to compare your performance with prior work. Use significance testing (e.g. t-tests) to determine significant differences in performance across models. The results we reported for multinomial logistic regression, multinomial naive Bayes, and SVM were all trained and tested on standardized training and testing sets. However, it was not possible use the standard sets for the deep learning models, because they require a validation set. Thus, we used confidence intervals to compare model performance on different test data sets. See the computation below for the 95% confidence interval of the true F1 score difference d between our tf-idf multinomial logistic regression model (M1), and our Feed-Forward$_2$ model (M2).

$$n_{M1} = 876, e_{M1} = 0.515$$

$$n_{M2} = 438, e_{M2} = 0.354$$

$$\hat{\sigma_d} = \frac{.515(1-.515)}{876} + \frac{0.354(1-.354)}{438}$$

$$\hat{\sigma_d} = 0.0008072$$

The Z-score for the 2-sided 95% confidence interval is 1.96, so:

$$d = .515 - .354 \pm 1.96\sqrt{0.0008072} = [0.105, 0.217]$$

Since the interval does not contain zero, we conclude that the difference in F1 score between the two models is significant at the 95% confidence level.

Model	Accuracy	F1 Score
Multi. Log. Regression$_4$	0.650	0.485
Multi. Naive Bayes$_4$	0.551	0.237
Multi. Log. Regression$_3$	0.566	0.516
Multi. Naive Bayes$_3$	0.449	0.447
SVM$_4$	0.554	0.244
Random Forest$_2$	0.568	0.297
Feed-Forward$_1$	0.571	0.562
LSTM$_1$	0.556	0.553
1D CNN$_1$	0.589	0.559
Feed-Forward$_2$	0.699	0.646

1 Uses word2vec embedding (20 dimensions)
2 Uses word2vec embedding (500 dimensions)
3 Uses bag of words embedding (500 dimensions) Note that embedding past 500 dimensions began to reduce accuracy; additionally, deep learning models more sophisticated than a few fully connected layers took a significant amount of time past 150 dimensions (days).
4 Uses tf-idf embedding

How well did your models perform in comparison to at least several baselines? In terms of F1 Score, our advanced models universally performed better than the baseline models. The Feed-Forward$_2$ model in particular obtained an F1 score which was nearly equivalent to the accuracy score, despite the imbalance of labels in the data set. While the accuracy of the multinomial logistic regression model is better than a number of the advanced deep learning models, this does not tell the full story of model quality. The F1 scores of the deep learning models are all better than that of our best baseline model. We believe the F1 score provides a better representation of a model's ability to classify text, given the distribution of labels in our dataset.

If relevant, which hyperparameters did you test in your model and why? Although this is not typically described as a hyperparameter, the type of word embedding we used and the number of features (or dimensions) used as an output after word embedding significantly changed model performance. We can see from the results in the previous section that increasing the number of embedding features increased accuracy / F1 score of the same model by at least 10%. Using word2vec embedding instead of using a default embedding layer also increased accuracy / F1 score by at least 10%. It seems that the way we represent our data into models has the largest impact on improving model efficacy.

For the deep learning models, we typically tuned the number of epochs and the learning rate until convergence (when accuracy was maximized, typically dropped due to overfitting when the number of epochs were too large). We found that using a small learning rate and using a larger number of epochs allowed us to find a decent local optimal for accuracy / F1 score.

If your research problem required ablation (e.g. "TF-IDF+word2vec" versus "TF-IDF only"), what features did you choose to change and why? How did the different ablations compare in terms of performance? For our problem we modeled both a regular tf-idf scheme, as well as the combination of tf-idf and word2vec. We discuss our tf-idf implementation in the baseline portion of this report. With solely tf-idf, we ran across a problem dealing with the amount of columns. This led us to try and reduce the amount of features in our data by implementing a word2vec representation. We believe that with fewer dimensions, our models could better capture patterns in the transcripts. Since word2vec provides a feature vector for each word, we had to average together all the words in a document to represent that document. Instead of taking a simple average of all word2vec vectors, we use the tf-idf weighting for each word. Our eventual representation for a earning transcripts is a tf-idf weighted average of all the words in the document.

If you found negative results, what do you think might have led to these results? How would a different model or test have achieved positive results instead? You may find it useful to include diagnostics including: a plot of training loss; a plot of test performance across different training schemes (e.g. more/less training data), hyperparameter tuning results. Note that previously (in our midway report), our results were disappointing. Current results are significantly better than the results posted in the midway report which we believe the changes to the label (the label is no longer significantly in the future), as well as using different word embeddings were the primary cause of increased performance. We believe current results can potentially be further improved (by trying an even larger variety of models or further tuning hyperparameters), but think that an approximate 70% accuracy and 65% F1 score is sufficient for this particular classification task, since predicting financial market trends is usually a difficult task.

Please include a brief error analysis. This may include features such as a confusion matrix, a table of common errors made by the model, feature weights, or a picture of attention scores over example text. What kinds of errors does your model make, are there any examples, and why do you think it made them? For the model with the highest F1 score, the confusion matrix is as follows:

Label = -1		
Actual / Prediction	Label != -1	Label = -1
Label != -1	436 (TN)	59 (FP)
Label = -1	56 (FN)	106 (TP)

Label = 0		
Actual / Prediction	Label != 0	Label = 0
Label != 0	199 (TN)	90 (FP)
Label = 0	75 (FN)	293 (TP)

Label = 1		
Actual / Prediction	Label != 1	Label = 1
Label != 1	481 (TN)	49 (FP)
Label = 1	67 (FN)	60 (TP)

It seems that the distribution of correct classifications (TN or TP) is pretty balanced throughout each possible label. Additionally, the incorrect classifications (FP, FN) also seem to be near equal values for each label. This seems to imply that our classification algorithm does not perform too poorly in one particular area, but in general sometimes naturally misclassifies new data.

We think that the classification errors are likely due to the difficulty of the data, as well as the label is affected by other factors than just board of director meeting transcripts. As opposed to some other common types of sentiment analysis (e.g. short movie reviews), we noticed that the distribution of words across all labels are typically rather similar. It makes sense that even struggling companies would be using words to promote optimism even if their financial situation is bad. Additionally, some topics during these meetings are ubiquitous among all companies. This makes classification more difficult. Secondly, the analyst ratings for a company (our label) are based on more information than just the quarterly board of directors meetings. Since that is not captured in our data (since the focus of this project is the natural language processing component), our models are almost guaranteed to lose some accuracy since our features do not capture all of the data required for accurate prediction.

4.3 Work Division

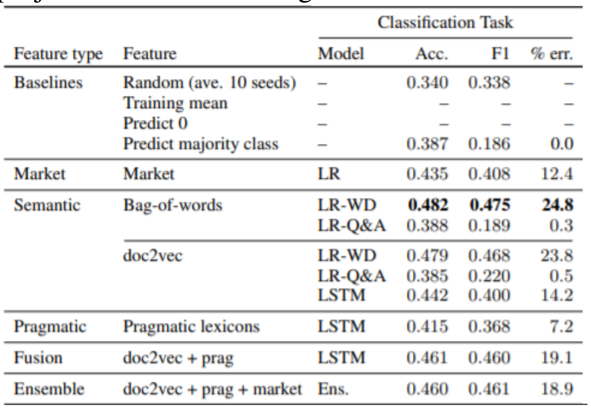
Which team members were responsible for which part of the results?

- Michael: LSTM, BERT, CNN, Deep Learning Classification
- Grayson: Webscraping and Data Wrangling, Bag-of-Word and w2vec representations.
- Frank: Tf-idf, Naive Bayes and Logistic Regression Classification, Sentiment dictionary analysis

5 Conclusion

What are the low-level and high-level conclusions that we can draw from your work? E.g. low-level would be "contextualized embeddings consistently improve QA performance on factoid questions", high-level would be "contextualized models capture a wider range of semantic patterns in typical questions." Representing a document as a feature vector is a difficult task because of the sparsity of the data. This resulted in poor model performance on bag-of-word and tf-idf representations. However, when we implemented w2vec, we were able to capture the signal from the document with 500 features. Compared to bag-of-words which may of had 9000+ dimensions. The deep learning models were able to pick up on trends from the 500 features. Another prominent low-level conclusion is the results from our test. Compared to well studied datasets such as MNIST or IRIS, it may not be reasonable to expect 90+ accuracy for this task. The data is messy, and patters are difficult to discover because each management team is motivated to always spin negative results and push positive results.

How do your results compare with the related work in the problem space (e.g. "in contrast to Yang and Prasad (2018), we find that attention-based models do not outperform HMMs in part-of-speech tagging")? If you find different results from related work, what factors do you think could explain the difference? The paper that is most similar to our project has the following classification results:



		Classification Task			
Feature type	Feature	Model	Acc.	F1	% err.
Baselines	Random (ave. 10 seeds)	–	0.340	0.338	–
	Training mean	–	–	–	–
	Predict 0	–	–	–	–
	Predict majority class	–	0.387	0.186	0.0
Market	Market	LR	0.435	0.408	12.4
Semantic	Bag-of-words	LR-WD	**0.482**	**0.475**	**24.8**
		LR-Q&A	0.388	0.189	0.3
	doc2vec	LR-WD	0.479	0.468	23.8
		LR-Q&A	0.385	0.220	0.5
		LSTM	0.442	0.400	14.2
Pragmatic	Pragmatic lexicons	LSTM	0.415	0.368	7.2
Fusion	doc2vec + prag	LSTM	0.461	0.460	19.1
Ensemble	doc2vec + prag + market	Ens.	0.460	0.461	18.9

from (Keith and Stent, 2019)

Note that our numbers (for both accuracy and F1-score) in the results section are actually better than the numbers in this table. Keith and Stent also had three, somewhat evenly distributed, classes to perform classification on, but a significant difference between our implementation and theirs is the fact that Keith and Stent attempted to predict future stock prices instead of analyst rating. Actually predicting stock performance has even more factors (and is generally a very difficult thing to predict) than analyst ratings; which is likely a large reason why our accuracy and F1 scores are better (their best model has an F1 score of 0.475 while ours is 0.646. We think that this is evidence towards the fact that analyst ratings are more directly influenced by the transcripts of board member meetings, while the stock price is affected by a number of different economic factors.

If someone wants to continue your work, what are some potential future directions for the project? This may include further model development, model analysis or data collection. Our biggest limitation for this project was the access to earning transcripts. When we pivoted our project goal from predicting the change in analyst rating to predicting the rating, this awarded us 1000 extra transcripts. Our model saw a noticeable uplift in accuracy and F1 score with increased training data. Obtaining more transcripts than we used is feasible, but may require a fee or purchasing these from a third party supplier.

Besides collecting more transcripts, our work can be improved by gathering analyst recommendations on a weekly basis. Our labels are were obtained on a quarterly basis. This results in a lag effect where a company could report earnings at the beginning of the quarter, and our prediction is based on the recommendation at the end of the quarter. This is a 3 month lag, which could remove any signal contained in the earning transcript.

Lastly, we had high expectations for our project when we first started. We wanted to incorporate an automated pipeline that would web scrape a new earning transcript, feed that into our trained model,make a prediction for how analysts would react, and ultimately visualize these results. Modeling this problem was more difficult than originally imagined, so we spent our time tuning our models to an acceptable level of accuracy and F1 score. However, future work could take our original plan and create a full pipeline to be implemented in practice or research.

References

Vinesh Jha, John Blaine, and Will Montague. 2015. Finding value in earning transcripts data with alphasense.

Katherine A. Keith and Amanda Stent. 2019. Modeling financial analysts' decision making via the pragmatics and semantics of earnings calls. *CoRR*, abs/1906.02868.

Tim Loughran and Bill McDonald. 2019. Master dictionary.