

Using BERT for sentiment analysis on SEC data

John Hancock [jhancoc4@fau.edu](mailto:jhancoc4@fau.edu)

April 4, 2020

## 1 Introduction

- Transformers
- BERT
- RoBERTa

## 2 Datasets

## 3 Methodology

## 4 Results

## 5 Conclusions

- We use Bidirectional Encoder Representations from Transformers (BERT) to perform sentiment analysis on quarterly (10-Q) reports public corporations filed with the United States' Securities and Exchange Commission (SEC).
- This project includes a tool for creating datasets similar to the one we use in this study for future research.

- The Transformer is the architecture BERT uses. Vaswani *et al.* [4] introduce the transformer architecture in the paper “Attention is all you need.”
- BERT provides a language representation. When the representation is incorporated into larger models researchers obtain strong results. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” [5] by Devlin *et al.* is the study that introduces BERT. [5]
- RoBERTa is an optimization of BERT. Liu *et al.* introduce RoBERTa in “RoBERTa: A Robustly Optimized BERT Pretraining Approach” [2]

We find papers on BERT or RoBERTa on arXiv.org, only.

Vaswani *et al.* introduce the Transformer architecture as one that gets the best results for English to German translation BLEU benchmark.

- Attention computed as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad (1)$$

- Attention is a way to transform input vectors or tensors. For other purposes one might decide to use a convolutional layer instead. <sup>1</sup>
- Every input and *desired* output token is mapped to a query, a key, and a value. During training an optimizer adjusts the components of the Transformer according to the Transformer's output.

- Once the Transformer is fit, then it can be applied to tasks where it can predict output tokens.

# Transformers III

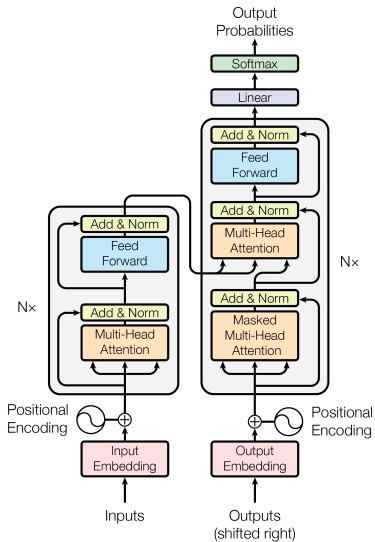
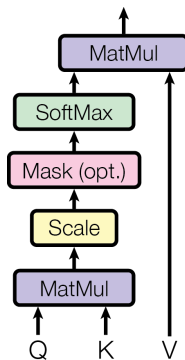


Figure: Copied from Figure 1 of Vaswani *et al.* [4]

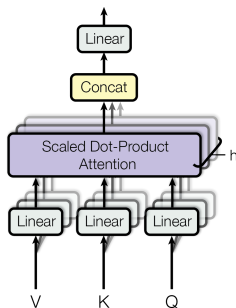
## Scaled Dot-product Attention I



**Figure:** Copied from Figure 2 of Vaswani *et al.* Scaled dot-product attention layers are key components of the Transformer. The optional masking is for preventing leftward information flow from tokens in the desired output (decoder) vectors. [4]



# Multi-head Attention I



**Figure:** Copied from Figure 2 of Vaswani *et al.* Multi-head attention layers compute attention in parallel. Vaswani *et al.* use the linear transformations because, “... we found it beneficial to linearly project the queries, keys and values...” [4]

- Equations copied from Vaswani *et al.* [4]
- $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$
- where  $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$
- “Where the projections are parameter matrices  
 $W_i^Q \in R^{d_{\text{model}} \times d_k}$ ,  $W_i^K \in R^{d_{\text{model}} \times d_k}$ ,  $W_i^V \in R^{d_{\text{model}} \times d_v}$  and  
 $W^O \in R^{hd_v \times d_{\text{model}}}$ .”

Transformers and attention are weighty subjects. These web pages are currently good supplementary resources for gaining understanding.

- “The Annotated Transformer” [7] by Alexander Rush and others  
<http://nlp.seas.harvard.edu/2018/04/03/attention.html>
- “The Illustrated Transformer” [6] By Jay Alammar  
<https://jalammar.github.io/illustrated-transformer/>
- “Stanford CS224N: NLP with Deep Learning — Winter 2019 — Lecture 14 – Transformers and Self-Attention” lecture video featuring Christopher Manning, Ashish Vaswani, and Anna Huang [8]

BERT is an acronym for “Bidirectional Encoder Representations from Transformers.” The paper that introduces BERT is “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” [5], by Devlin *et al.* Here are some key points from this paper.

- “BERTs model architecture is a multi-layer bidirectional Transformer **encoder** based on the original implementation described in Vaswani *et al.* . . .” [5], emphasis ours
- “The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks.”
- Since it only uses the encoder, BERT is simpler than the original Transformer architecture.

- **Pre-train** BERT with masked language modeling and next sentence prediction tasks. From what we can understand of the BERT source code <sup>2</sup>, the optimizer is evaluating BERT on both tasks simultaneously.
- **Fine-tune** BERT: continue optimizing all parameters of BERT for some downstream task. For this project, we are doing the fourth downstream task that Devlin *et al.* mention in [5].
- Devlin *et al.* recommend fine-tuning BERT with the appropriate input and output.
- The input should be "... a degenerate text- $\emptyset$  pair in text classification ...".
- and we should obtain the output as follows, "... the [CLS] representation is fed into an output layer for classification, such as entailment or **sentiment analysis**[emphasis added]"

- We do not find an explanation of what inspires the [CLS] token, but it appears to be useful for several downstream tasks. Devlin *et al.* write that “[CLS] is a special symbol added in front of every input example”

---

<sup>2</sup>[https://github.com/google-research/bert/blob/master/run\\_pretraining.py](https://github.com/google-research/bert/blob/master/run_pretraining.py)

The study that introduces RoBERTa is “RoBERTa: A Robustly Optimized BERT Pretraining Approach” [2], by Liu *et al.*

RoBERTa is a refinement of BERT. Liu *et al.* write, “We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it.” In order to better train BERT, Liu *et al.* make the following enhancements:

- dynamically chose tokens to mask for the language modeling pre-training task – slightly better performance, more efficient implementation
- remove the next sentence prediction pre-training task, and make inputs full sentences from the source documents. “We find that using individual sentences hurts performance on downstream tasks, which we hypothesize is because the model is not able to learn long-range dependencies.”

- increase batch size from 256 sequences to 8 thousand sequences. “We observe that training with large batches improves perplexity for the masked language modeling objective ... also easier to parallelize via distributed data parallel training ...”
- use Byte Pair Encoding (BPE). BPE “... makes it possible to learn a subword vocabulary of a modest size (50K units) that can still encode any input text without introducing any “unknown” tokens.” Liu *et al.* do not use BPE for better performance, but for better compatibility. “... we believe the advantages of a universal encoding scheme outweighs the minor degradation in performance ...”
- With these enhancements, Liu *et al.* report results that show RoBERTa outperforms BERT on several downstream tasks.



- We use two datasets. The first is the Yelp reviews dataset. This dataset is the same dataset the author uses in the example we cite below.
- The dataset is at `https://s3.amazonaws.com/fast-ai-nlp/yelp\_review\_polarity\_csv.tgz`
- *Caveat* this dataset includes a file `readme.txt` that does not accurately describe the data. In particular, the label column of the actual data has two distinct values, but `readme.txt` states that there are four values.
- The archive `yelp_review_polarity_csv.tgz` contains two files: `train.csv`, and `test.csv`.
- As the names of the files indicate we use one for fine-tuning RoBERTa, and one for evaluating RoBERTa.

- Example Yelp input value for RoBERTa: “This place is s l o w....like so slow I have got up and left after waiting 30 minutes and not even being acknowledged by a server. The food is okay. Def not worth the wait. Milkshakes are just okay as well. Boooooooo.”
- **Note:** We do not have to do tokenization, stemming, lemmatization, or remove stop words. The library takes care of all this for us.
- The Yelp dataset labels are values from  $\{1, 2\}$ . The value 1 indicates a negative sentiment, and the value 2 indicates a positive sentiment.

- We derive a second dataset from the United States' Securities and Exchange Commission (SEC). The SEC maintains the Electronic Data Gathering, Analysis, and Retrieval system [10] (EDGAR) database. EDGAR contains information that the SEC either requires, or suggests public corporations to submit. Here, we use the quarterly reports corporations submit to EDGAR as input for RoBERTa.
- The SEC publishes an index of all forms it receives by quarter. The index contains Uniform Resource Indicators (URI's) for these reports. Therefore, one can process this index and automatically download, and parse all the quarterly reports companies file with the SEC.

- TD Ameritrade, Inc. provides a Representational State Transfer (REST) application programming interface (API) we to fetch historical prices of shares of companies that file quarterly reports.
- We use the percent change in share price from the beginning of the quarter to the end of the quarter to determine a value for the label. If the percent change is positive we label the quarterly report with the value 2, and 1 otherwise.
- Quarterly report files are large Hypertext Markup Language (HTML) files, so we truncate HTML code at 200,000 characters, then parse the quarterly report file, and truncate the parsed result at 100,000 characters.

- The source code for generating the dataset is available at [https://github.com/jhancock1975/info-retrieval-cap-6776-project/blob/master/gen\\_dataset.py](https://github.com/jhancock1975/info-retrieval-cap-6776-project/blob/master/gen_dataset.py)

For our experiments, we use “Simple Transformers Introducing The Easiest Way To Use BERT, RoBERTa, XLNet, and XLM” [9] by Thilina Rajapakse.

- *We did not realize one must pay Medium.com \$5 for this guide after so many views of it*
- This guide explains how to use the SimpleTransformers Python RoBERTa implementation.
- The SimpleTransformers library reduces using RoBERTa to splitting data into test and train sets, and at a minimum three lines of code:

```
■ model = ClassificationModel('roberta', 'roberta-base')  
■ model.train_model(train_df)  
■ result, model_outputs, wrong_predictions =  
  model.eval_model(eval_df)
```

- We use a docker image with for Pytorch with Graphics Processing Unit (GPU) support to save some time setting up.  
`https://hub.docker.com/r/pytorch/pytorch/tags`
- `docker run -v /git:/git --gpus all -ti --ipc=host pytorch/pytorch:latest`
- `gen_dataset.py` generates a dataset similar to the Yelp dataset Rajapakse's example uses, so we use it as input to the example, directly.
- To generate the data we use in the experiment in this study, we use EDGAR data from the first quarter of 2018.
- We run the program `example.py` using both datasets as input to generate our results.

The implementation we did for this project has some limitations

- The central index key (CIK) must be in the CIK to stock ticker symbol mapping file the SEC provides.
- The TD Ameritrade REST API does not return historical price data for some ticker symbols



- We capture Mathews' Correlation Coefficient (MCC) [11] as well as the number of true positives, false positive, true negative, and false negatives.
- The formula for MCC involves counts of all four types of classifications, *i.e.* true positives, false positive, true negative, and false negatives. MCC takes values in  $[-1, 1]$  where a value of 1 would mean perfect classification, and -1 would mean total misclassification. The MCC value of 0 means our classifier does no better than random prediction.
- We use both of the datasets we describe above as input to RoBERTa as implemented in `example.py`.

Yelp				
TP	TN	FP	FN	MCC
17917	17989	1011	1083	0.8898
EDGAR				
0	729	0	500	0.0






**Table:** Results for running RoBERTa on the fine-tuning task of sentiment analysis on Yelp review data, and SEC EDGAR data; TP, TN, FP, FN stand for true positive, true negative, false positive, and false negative, respectively; MCC stands for Mathews correlation coefficient

Our results are underwhelming to say the least. We have some ideas for improvement.

- The results we get when we use Rajapakse's example [9] are strong. The MCC value of approximately 0.8898 is close to 1, which implies RoBERTa does a good job of classifying Yelp reviews as having positive or negative sentiment.
- One opportunity for future research is to find how algorithms based on other architectures, such as recurrent neural networks (RNN's) do on the same Yelp review data.
- The results we get for the EDGAR data are terrible. With the data we give RoBERTa, RoBERTa ends up classifying all reports in the negative class.
- The EDGAR data is not imbalanced. We have 3,674 instances labeled as negative, and 2,471 instances labeled as positive in the training data

- This implies that we have a data quality issue.
- One issue could be that we do not segment our input data properly. Another opportunity for future research would be to evaluate how RoBERTa does with smaller segments of quarterly reports, instead of the truncated versions of entire reports that we use.
- A third avenue of research we would like to explore in the future is to parse the quarterly reports more intelligently, with a parser that is more aware of the format of the 10-Q quarterly report. The parser we use in `example.py` is capable of extracting text from HTML documents, but not specific elements particular to the 10-Q report.

## References I

-  Hjek, P., 2018. Combining bag-of-words and sentiment features of annual reports to predict abnormal stock returns. *Neural Computing and Applications*, 29(7), pp.343-358.
-  Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.
-  CHEN, C.L., LIU, C.L., CHANG, Y.C. and TSAI, H.P., 2013. Opinion Mining for Relating Subjective Expressions and Annual Earnings in US Financial Statements. *Journal of information science and engineering*, 29(4), pp.743-764.
-  Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, . and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).
-  Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

## References II



Alammar, J. (2018). The Illustrated Transformer [Blog post]. Retrieved from <https://jalammar.github.io/illustrated-transformer/> [Accessed: 4 April 2020]



Rush, A. (2018). The Annotated Transformer. Retrieved from <http://nlp.seas.harvard.edu/2018/04/03/attention.html> [Accessed: 4 April 2020]



Manning, C., Vaswani A., and Huang A. (2019). Stanford CS224N: NLP with Deep Learning — Winter 2019 — Lecture 14 – Transformers and Self-Attention. Retrieved from <https://www.youtube.com/watch?v=5vcj8kSwBCY&t=1293s> [Accessed: 4 April 2020]



Rajapakse T. (2019) Simple Transformers Introducing The Easiest Way To Use BERT, RoBERTa, XLNet, and XLM. Retrieved from <https://towardsdatascience.com/simple-transformers-introducing-the-easiest-bert-roberta-xl-net-and> [Accessed: 4 April 2020]



(2020). Electronic Data Gathering, Analysis, and Retrieval system (EDGAR). Retrieved from <https://www.sec.gov/edgar/about> [Accessed: 4 April 2020]



Matthews, B.W., 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2), pp.442-451.

The End