# Financial Sentiment Analysis using FinBert

Introduction

FinBert is an opensource pre trained Natural Language Processing (NLP) model, that has been specifically trained on Financial data, and outperforms almost all other NLP techniques for financial sentiment analysis.

Doing sentiment analysis on financial data is more complicated than normal use cases. Let’s take an example:

*Food companies doing well despite the global markets downturn due to covid*

Normal NLP techniques will not be able to determine that the above sentence is good news and has positive sentiment for food companies. This is because normal NLP techniques, like word2vec, look at each word separately and don’t have context for the words. To understand the sentiment of the above sentence we would need to be context aware.

BERT

BERT enables context awareness for sentences. BERT stands for Bidirectional Encoder Representation from Transformer. It is one of the most popular state of the art text embedding model published by Google. BERT has caused a revolution in the world of NLP by providing superior results on many NLP tasks, such as question answering, text generation, sentence classification, and many more compared to other methods.

One of the reasons BERT is more successful is that it uses a context based embedding model. Consider the example below

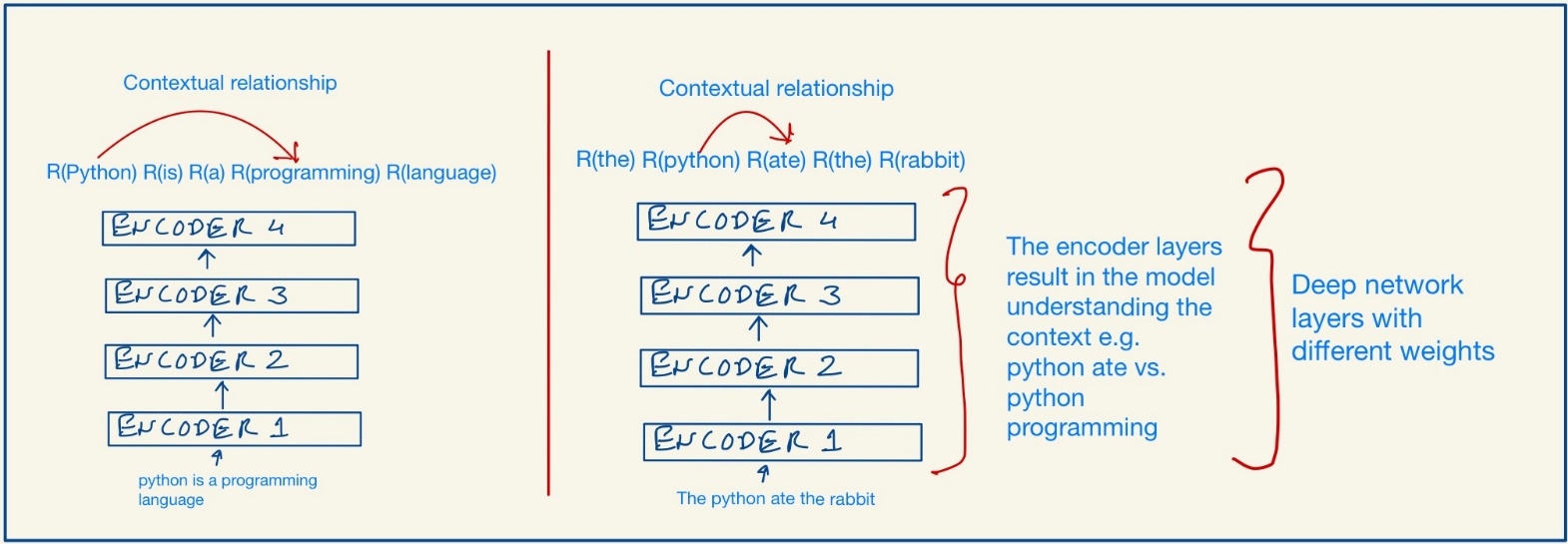
*Sentence 1: The python ate the rabbit*

*Sentence 2: Python is one of the most popular programming languages*

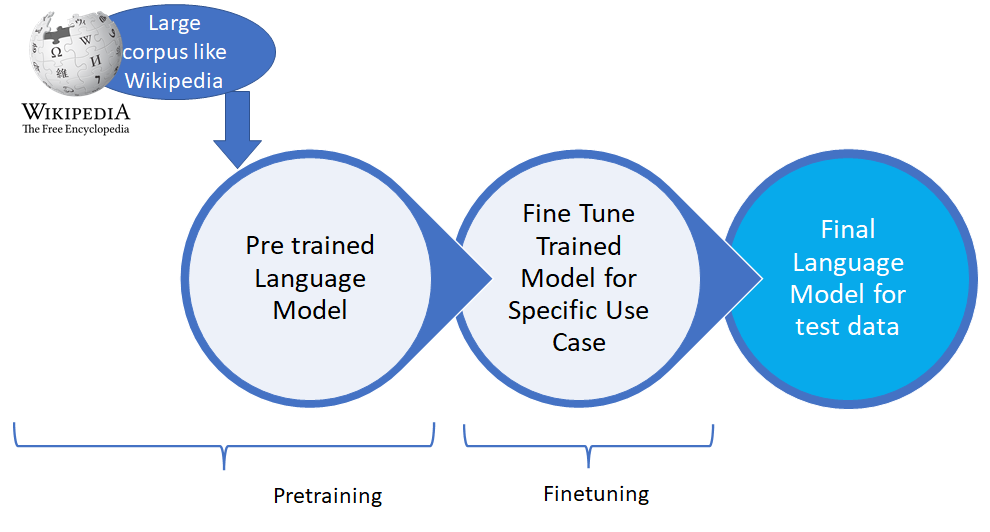
Without context, the word python would have the same meaning in both sentences. BERT looks at the sentence and figures out what words python is related to in the sentence, and will create embedding of the word python based on the context. BERT does this by using transformers, which is a state of the art deep learning architecture, that is mostly used for Natural language processing. The architecture uses encoder-decoder paradigm.

The encoder takes the input sentence and learns its representation and then sends the representation to the decoder. The decoder generates the output sentence. The transformer architecture uses many layers of encoders to generate the representation. BERT can be thought of as a transformer, but only with encoders. BERT has different configurations based on how many encoder layers it uses.

That was a mouthful, wasn’t it? Let’s look at it diagrammatically.



The BERT model is pretrained on a large corpus of words. What is pretraining? Pretraining is when we train a model with a huge dataset and potentially a very large number of parameters for a particular task and save the trained model. For any new task, instead of initializing a new model with random weights, we will initialize it with the weights of the trained model, and adjust the weights for the new task. This is helpful since for our work we may not have easy access to huge volumes of training data, and we will save a lot of time and resources that were spent on training the model.



The BERT model is pre-trained using two tasks: masked language modelling and next sentence prediction. BERT models have been trained on BookCorpus and English Wikipedia, which have in total more than 3.5 Billion words. I found this [1] book to be a great reference on BERT

Domain Specific FineTuning of BERT

Many domain specific models have emerged using BERT as the base and are being used for NLP tasks. Some of them are: FinBERT for Finance, BioBERT for Biomedical, VideoBERT for Video captioning categorization, ClinicalBERT for hospitals, and many more continue to evolve. If you are looking for cutting edge, deep learning pre-trained models for any domain, it would be worth researching to see if a DomainBERT model for that area exists. You could save yourself a lot of time and end up with high quality results.

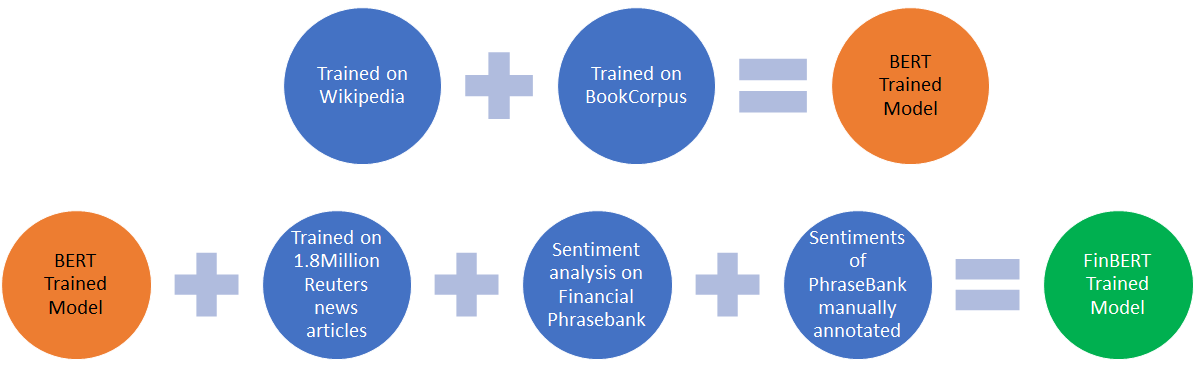
Diagram, schematic

Description automatically generated

I will only focus on FinBERT in this article.

FinBERT

FinBERT is a language model based on BERT. It further trains the BERT model for financial data. The additional training corpus is a set of 1.8M Reuters’ news articles and Financial PhraseBank. As this paper [2] mentions, the main sentiment analysis dataset used is Financial PhraseBank which consists of 4845 English sentences selected randomly from financial news found on LexisNexis database. These sentences then were annotated by 16 people with backgrounds in finance and business.



Implementation

FinBERT sentiment analysis model is available on Hugging Face model hub. You can get the model at [3]. FinBERT implementation is reliant on Hugging Face’s pytorch\_pretrained\_bert library and their implementation of BERT for sequence classification tasks.

In order to demonstrate FinBert in action, I will use a financial news dataset from Kaggle[4]. Some of my code is inspired by [5] and [6]. I have used Google colab to run this code, primarily because pip installing transformers on my Windows machine proved to be a nightmare due to version specific library dependencies being hard to resolve. It worked with no difficulty on Google colab. I did have to trim the dataset size to 50 rows to stay within RAM limits since I was using the free tier. You can see the code working here: <https://colab.research.google.com/drive/1jEHhU5_x4oQkelW3p__fY2y0m3-z7Y5P?usp=sharing>.

For easy reference, I am pasting the relevant code below.

------------------------------

df = pd.read\_csv("reuters\_headlines.csv", nrows=50)

df\_array = np.array(df)

df\_list = list(df\_array[:,0])

tokenizer = AutoTokenizer.from\_pretrained("ProsusAI/finbert")

model = AutoModelForSequenceClassification.from\_pretrained("ProsusAI/finbert")

inputs = tokenizer(df\_list, padding = True, truncation = True, return\_tensors='pt') #tokenize text to be sent to model

outputs = model(\*\*inputs)

predictions = torch.nn.functional.softmax(outputs.logits, dim=-1)

model.config.id2label

positive = predictions[:, 0].tolist()

negative = predictions[:, 1].tolist()

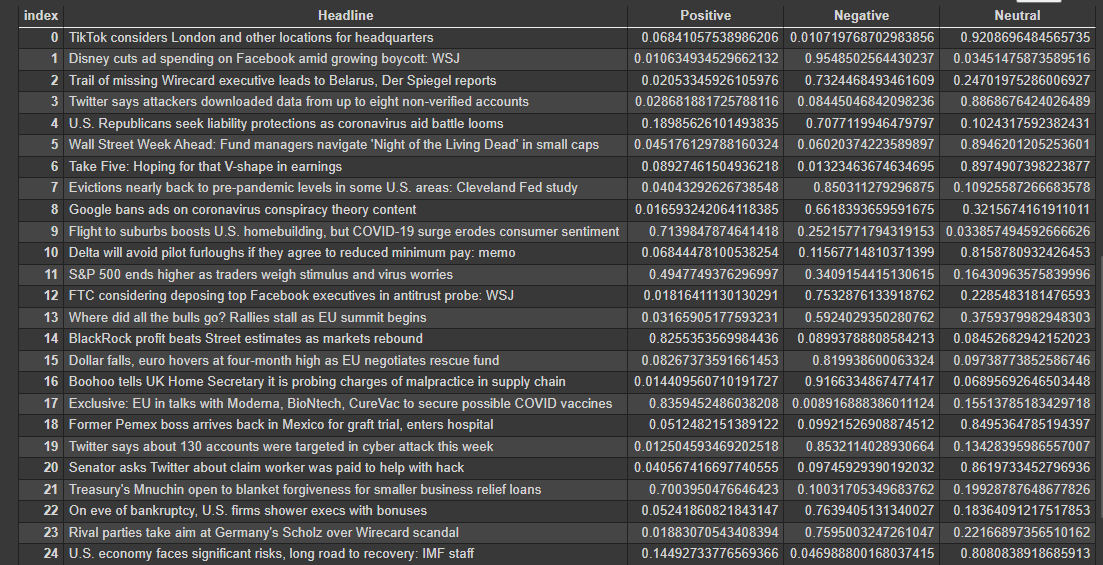
neutral = predictions[:, 2].tolist()

table = {'Headline':df\_list, "Positive":positive, "Negative":negative, "Neutral":neutral}

df2 = pd.DataFrame(table, columns = ["Headline", "Positive", "Negative", "Neutral"])

------------------------------

The result of the sentiment analysis done is shown below.



Looking at the output above we can see that the sentiment analysis is very good, with no training from us. We also managed to do it in about 20 lines of code.

Conclusion

FinBERT makes the job of sentiment analysis for financial feeds very easy, and a lot more accurate. The heavy lifting for training and testing a model on a very large financial corpus has already been done by the researchers, and the model has been made public by Hugging Face. The rest of us can simply use it with very few lines of code to get fairly accurate results for financial sentiment analysis.

References

1. Book: “Getting Started with Google BERT” <https://www.packtpub.com/product/getting-started-with-google-bert/9781838821593>
2. Thesis: FinBERT: Financial Sentiment Analysis with Pre-trained Language Models Dogu Tan Araci (<https://arxiv.org/pdf/1908.10063.pdf>)
3. FinBERT Source implementation: <https://github.com/ProsusAI/finBERT>
4. Kaggle dataset used in this article: <https://www.kaggle.com/notlucasp/financial-news-headlines>
5. Reference implementation 1: [*https://colab.research.google.com/drive/1C6\_ahu0Eps\_wLKcsfspEO0HIEouND-oI?usp=sharing*](https://colab.research.google.com/drive/1C6_ahu0Eps_wLKcsfspEO0HIEouND-oI?usp=sharing)
6. Reference implementation 2: [*https://github.com/ProsusAI/finBERT/blob/master/scripts/predict.py*](https://github.com/ProsusAI/finBERT/blob/master/scripts/predict.py)