Final Report For 11-785

Bhumika Kapur, Jun Tao Luo, Nathan Luskey, Amey Patel
Carnegie Mellon University
Pittsburgh, PA 15213
`{bkapur, jtluo, nluskey, ameypate}@andrew.cmu.edu`

Abstract

Working with Akshat Gupta, our team's project for 11-785 is to generate financial word embeddings[1]. Efficient word embeddings can improve performance on downstream tasks by more effectively modeling relationships between words in a denser vector space than simple one hot encoding; however, a general English embedding might have spurious or ineffective relationships for financial downstream tasks. We reviewed several popular word embedding models and evaluated their performance on tasks in the financial domain. We then implemented word2vec and GloVe on a subset of SEC Filings from 1994. Lastly, we tuned general English BERT and BERT based architectures on SEC Filings from 1993 - 2002. Furthermore, we performed block coordinate descent to optimize our model choice, parameters values, and combination of the hidden and attention layers. Through these ablation studies we improved the accuracy on the downstream task provided by the financial phrase bank of sentiment classification from 71% to 80% on distil BERT and we improved area under the curve (AUC) from 0.8066 to 0.8993. This demonstrates the value of training on domain specific knowledge, and future work could expand on hyper parameter tuning with larger architectures.

1 Introduction

Our team's project for 11-785 is generating word embeddings that capture the idiosyncrasies of the financial domain. Efficient embeddings improve performance on downstream tasks by more effectively modeling relationships between words in a lower dimensional space than simple one hot encoding; however, a general English embedding may encode spurious relationships that aren't relevant to the financial domain or fail to encode domain specific knowledge. For example, in plain English a word similar to "freeze" may be "ice", but for the financial domain, a similar word to "freeze" may be "asset". Working with Akshat Gupta, we trained word2vec, GloVe, and BERT based word embeddings on a cleaned and tokenized subset of SEC Filings from 1993-2002. To assess whether these embeddings are better able to capturing the financial domain specific knowledge, they are used to train a simple model on a downstream task against state of the art general English embeddings from BERT.

2 Literature review

Below are reviews of several popular word embedding models from 2013-2020, and how these embeddings have been applied to the financial domain.

2.1 Word embedding models

The oldest model we researched is Google's **word2vec** [1]. The paper proposed two new architectures for generating embeddings: continuous bag of words (CBOW) and skip-gram. The CBOW model predicts the current word based on the context, and the skip-gram model predicts the context based off the current word.

[1]For our code please visit our github repository at: `https://github.com/nathanluskey/CMU_11785_Project`.

There is no information on ordering or distance, just whether words are in the same sliding window. For training, the task is modelled as logistic regression with pairs of words either being neighbors or not; this speeds up training due to not needing a large projection layer from hidden to vocab size. For training models, the authors used a Google News corpus for training the word vectors with about 6B tokens and restricting the vocabulary size to 1 million most frequent words. This model improved upon the recurrent neural network and feed forward neural network models of the time in syntactic and semantic tasks as well as computation complexity for training.

The next model we researched is Stanford's **GloVe** [2]. GloVe is a log-bilinear model with a weighted least-squares objective which learns encodings of words using ratios of word-word co-occurrence probabilities. This improves upon window based model (such as word2vec) by operating on a matrix to take advantage of repetitions. This also improves upon matrix factorization methods, because these suffer from having most frequent words contributing too much. On word analogy, word similarity, and named entity recognition tasks, GloVe outperformed other models (included word2vec's skip-gram and CBOW) even with smaller vector sizes and corpora.

Another model we researched is **Embedding from Language Models (ELMO)** [3]. This model's key improvement was the addition of contextualized information to richness of representation. Richness of representations is that the model can contain both syntactic and semantic similarities. Contextualized information is that a word can have very different meanings based on context; the example given in the paper is "play" as in a musical show or a a part of a baseball game. These improvements were accomplished by the model's architecture of a 2 layer bidirectional LSTM with residual connections from first to second layer and 4096 units of 512 dimension projections. The first layer captured simple features, like syntax, and the second layer captured contextual features, like semantics. The model was trained on 30 million sentences [4], and then is fine tuned for specific tasks. Across 6 different benchmark NLP tasks, ELMO improved upon the state of the art at the time.

The final model and current state of the art that we researched is Google's Bidirectional Encoder Representations from Transformers (**BERT**) [5]. BERT improved upon ELMO by using attention rather than an RNN structure; this also improves ability to paralleling and scaling training. Through self-attention and multi-headed attention, the transformer architecture is able to outperform RNN based architectures. The model is trained on a large corpus, then fine tuned for specific tasks. In the paper, BERT improved upon the state of the art for eleven NLP tasks, and now a large number of BERT models tuned to specific tasks are available from hugging face [6]. Furthermore, upon analyzing how BERT processes information, it behaves as a classical NLP pipeline [7]. This pipeline consists of building up through: part of speech tagging, constituent labeling, dependency labeling, named entity labeling, semantic role labeling, co reference, semantic proto-role, and relation classification as shown in Figure 1. The pipeline holds in aggregate, but can adapt to ambiguous decisions based on higher level information. Through this, we can confidently train BERT on raw sentences without tokenization or tagging often done for classical NLP pipelines.

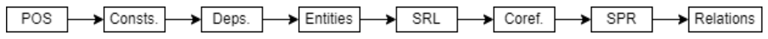

Figure 1: The parts of the classical Natural Language Processing Pipeline that BERT replicates.

BERT revolutionized the landscape of NLP-domain tasks by then achieving the state of art results on several NLP tasks. The power of BERT lies in that a pretrained model could be used for any downstream task just by fine-tuning it. This speaks volumes of the quality of the embeddings that it generates. Hence, BERT was a natural choice for us to move on to.

This suggests that pre-training BERT teaches it about the structure of the language and is able to "learn" the language. However, it raises the question, "What linguistic features do they learn?". Extensive research has been done on using hidden states to learn what the transformer layers learn. "What does BERT look at?" [8] takes an extensive look at what the attention layers are able to capture. They show results suggesting the attention layers do a good job in understanding the syntax of a language. BERT learns its own "syntactical English" which is an side-effect of self supervised training. They notice that the syntax that is learnt by BERT is pretty similar to the "syntactical English" that a human learns and uses.

We also explored many variations of BERT, including Roberta, XLNet, Albert, and DistilBERT.

Roberta [9]: Facebook's improvement on BERT, this model is a Robust and optimized version of BERT. The first difference in the model is dynamic masking. In the original BERT, 15% of words are masked in a sentence during pretraining, but it is the same 15% of words every time. In dynamic masking, each sequence is masked in 10 different ways during pretraining. The second difference is the removal of next sentence prediction in pretraining, which was the second pretraining task in the original BERT. Roberta is also trained on more data and for longer, which leads to outperforming BERT in many tasks.

XLNet [10]: This model was developed by CMU and Google AI, and combined autogressive and autoencoder language modeling. The model aims to maximize the log likelihood of a sequence with respect to all possible permutations of the sequence. It uses an attention mask to mask out the words not in the context of a word in a given permutation and two streams of self attention. It also outperforms BERT in most tasks.

Albert[11]: A lighter version of BERT developed by Google AI. It uses cross layer parameter sharing, which prevents parameters from growing as the layers increase, and factorized embeddings to decrease the number of parameters. It also uses a sentence order prediction loss rather than the next sentence prediction loss. This results in an 18 times reduction of parameters compared to BERT, and also improved performance.

DistilBERT[12]: A distilled version of BERT which is distilled using the teacher student method. It has half the number of layers as the original BERT, and half the number of parameters. It is trained on large batches, and uses dynamic masking, which results in 95% of the performance of BERT.

We also explored some ideas explored in **Google's Text-to-Text Transformer**[13]. The main motivation of the studies conducted in this paper was to survey existing state of the art NLP techniques in a unified way and gain insights into their relative impact. The paper recommends using a full transformer model for NLP tasks, for which the encoder side can be construed as the portion of the model that provides an embedding of the input text. This confirms our motivation to experiment with BERT-like models for generating embeddings as these models are just the encoder portions of transformers. The recommendation of the paper to use denoising objectives during training also suggest that Masked Language Model should perform well for training embedding models. Finally, their experiments with multi-task training suggests that training on unlabeled data that's more similar to the downstream task often produces significant improvements in the specific downstream task, suggesting training on unlabeled financial data as an avenue of improvement for finance related NLP tasks.

2.2 Financial domain specific embeddings

Several papers have focused specifically on collecting financial data for training or tuning models. "Learning Word Embeddings from 10-K Filings for Financial NLP Tasks" [14] compiled a corpus of 10-k filings by corporations in the U.S. to the SEC from 1993-2018, and they generated word embeddings using word2vec. Their skip-gram architecture, implemented as a two-layer neural network, was used for this embedding since it produced more accurate word embeddings. Cosine similarity was then used to determine relationships between word embeddings in tasks such as sentiment analysis; although they did not compare their performance to models trained on plain English embeddings.

"EDGAR-CORPUS: Billions of Tokens Make The World Go Round" [15] compiled a dataset from annual 10-k SEC Filings from 1993-2020. This dataset was then used to train a word2vec skip-gram model. The model outperformed generic GloVe on the downstream financial tasks of: finsim, financial tagging, and FiQA. FinSim-3 provides a set of business and economic terms and the task is to classify them into the most relevant hypernym from a set of 17 possible hypernyms from the Financial Industry Business Ontology (fibo) [16]. Financial Tagging was an internal task to annotate financial reports with word-level tags. Lastly, the FiQa Open Challenge [17] will be discussed more in the downstream task data section.

"FinText" [18] was compiled from Dow Jones Newswire's Text News Feed from 2000-2015 and used to train word2vec and FastText (a word embedding model from Meta). The embeddings were qualitatively found to be more sensitive to financial jargon. On the downstream task of volatility forecasting for NASDAQ stocks, the model trained with these embeddings outperformed the model trained with generic word embeddings.

Lastly, two papers have fine tuned BERT for the financial domain. "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models [19]" had two approaches: pre-trained on a financial corpus, and pre-trained on sentences from training classification dataset. To prevent failure, trained with all layers but the classifier layer was frozen, and then gradually unfroze layers. Results were compared with LST and ULMFit on financial phrasebank dataset with FinBERT outperforming both baseline methods by about 15%. "FinBERT: A Pretrained Language Model for Financial Communications" [20] was trained on a large corpora of financial data that was representative of the financial domain which included SEC filings, earnings call transcripts, analyst reports, and overall corpora statistics; the overall size was about 4.9 billion tokens. Results were compared with BERT model and FinBERT outperformed it with an improvement of 15.6% for uncased models and over 29% for cased model on FiQA dataset.

3 Dataset description

3.1 Training Data

For training, Akshat has already generated a dataset from SEC filings from 1993-2002, and the dataset has been cleaned, standardized, and sentence tokenized; the overall dataset size is huge (around 100 gigabytes), so we have also generated subsets ranging in size from 1% to 75% of the overall size. These filings are long standardized regulatory documents filed quarterly for publicly traded companies and information on a company's earnings, assets, and key management[21]. As such they are full of information and key to financial analysis of a company. Therefore, they are prime candidates for generating financial domain specific word embeddings. Initially, to remove stop words and unnecessary punctuation, we used the Natural Language Toolkit (NLTK) [22] and trained on the subset from 1994; however for BERT based architecture training full sentences from the full dataset were used for training.

3.2 Downstream Task

Initially, for our downstream task, we used our word embeddings to train a simple bi-directional LSTM on the Opinion Mining and Question Answering task from the Financial 2018 competition [17] (FiQA); the task is to predict sentiment score in the range of $[-1, 1]$ for mentioned targets given English text from the general financial domain (such as blogs or news). We assess how these simple models trained using our word embeddings perform against state of the art plain English embeddings by looking at mean squared error. FiQA contains 1111 samples which we split into 900 training samples and 211 test samples.

We decided to switch to financial phrasebank [23] because of the larger dataset size of 4840. The task is similar, of sentiment analysis on sentences, but sentences are put into the class of: neutral, negative, or positive. Furthermore the dataset can be further split by number of experts who agree on the sentiment score as shown in Figure 2.

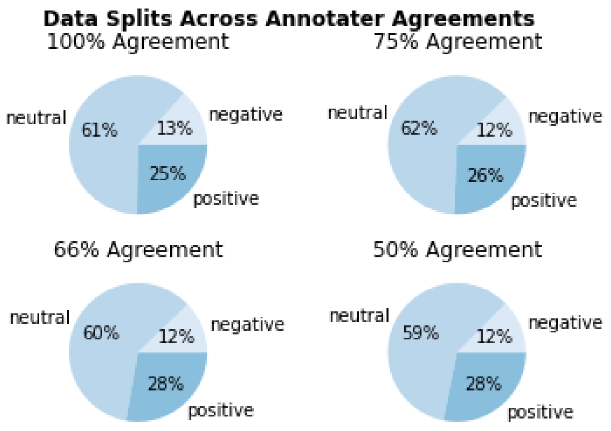

Figure 2: The breakdown of how different subsets with the financial phrasebank dataset are agreed upon by 5-8 annotators.

4 Model description

4.1 word2vec

4.1.1 Mathematical description

While the two word2vec models are similar in that they try to capture correlation between words that are close to each other, their mathematical formulation is different and will be presented separately.

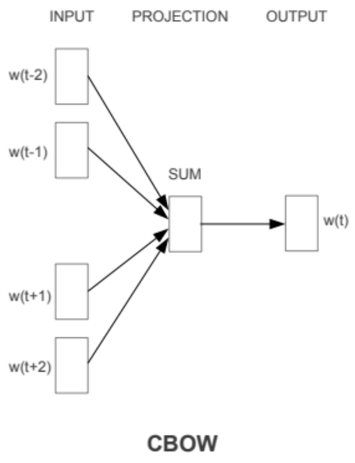

Figure 3: The relationship between surrounding words and target word for the CBOW word2vec model

Continuous Bag of Words (CBOW) This version of the word2vec model aims to predict the target word $w(t)$ given some context of surrounding words $w(t-m), w(t-m+1), ...w(t-1), w(t), w(t+1), ..., w(t+m)$ for some embedding parameters θ, context size m and some word in the corpus $t \in 1, T$ where T is the size of the corpus. The recommended context is 4 [1]. In more rigorous terms we want to maximize the likelihood:

$$L_{CBOW}(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_t | w_{t+j}; \theta)$$

To simplify the computation, we will instead minimize the negative log likelihood:

$$\ell_{CBOW}(\theta) = -\sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_t | w_{t+j}; \theta)$$

The probability of a word w given a word its context w_c defined as a softmax over all words in the vocabulary V where q, q_c are the embedded vectors of w, w_c, repectively:

$$P(w | w_c; \theta) = \frac{\exp(q \cdot q_c)}{\sum_{i \in V} \exp(q \cdot q_i)}$$

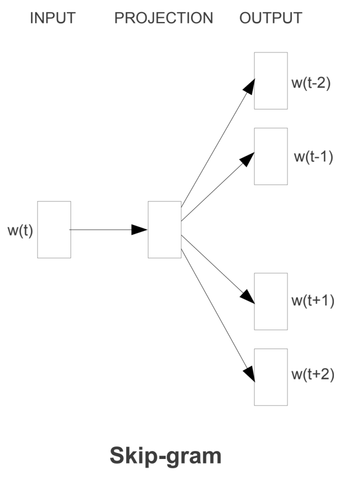

Figure 4: The relationship between surrounding words and target word for the CBOW word2vec model

Skip-gram This version of the word2vec model is slightly different in that it aims to predict the context words $w(t-m), w(t-m+1), ...w(t-1), w(t), w(t+1), ..., w(t+m)$ given some target word $w(t)$ for some embedding parameters θ, context size m and some word in the corpus $t \in 1, T$ where T is the size of the corpus. The recommended context is also 4 [1]. In more rigorous terms we want to maximize the likelihood:

$$L_{Skip-gram}(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \leq j \leq m \\ j \neq 0}} P(w_{t+j} | w_t; \theta)$$

To simplify the computation, we will instead minimize the negative log likelihood:

$$\ell_{Skip-gram}(\theta) = -\sum_{t=1}^{T} \sum_{\substack{-m \leq j \leq m \\ j \neq 0}} \log P(w_{t+j} | w_t; \theta)$$

The probability of a context word w_c given a target word w defined as a softmax over all words in the vocabulary V where q, q_c are the embedded vectors of w, w_c, repectively:

$$P(w_c | w; \theta) = \frac{\exp(q_c \cdot q)}{\sum_{i \in V} \exp(q_c \cdot q_i)}$$

4.1.2 Implementation description

Again we will discuss the model specific implementation details below. Common to both implementation is the vocabulary extraction and training process. The vocabulary is built using `torchtext.vocab.build_vocab_from_iterator` but we've limited the vocabulary to words that appear at least 50 times in the corpus since rare words rarely produce good embeddings due to their small sample size, so we will ignore them during training and inference. Training for both models is performed using a `torch.nn.Linear` layer with `torch.nn.CrossEntropyLoss` on the parsed training samples.

Continuous Bag of Words (CBOW) We prepare the data for unsupervised training by parsing the training corpus into `([list of context words], target word)` pairs. The word embeddings are implemented via `torch.nn.Embedding` layer with the appropriate vocabulary size and embedding dimension and taking a sum of the embedded vectors of the list of context words.

Skip-gram For Skip-gram we prepare for unsupervised training by parsing the training corpus into multiple `(target word, context word)` pairs, for each target word context word combination. Similar to CBOW, the word embeddings are implemented via `torch.nn.Embedding` layer with the appropriate vocabulary size and embedding dimension but we can directly use the embedding for training without taking any sums.

4.2 GloVe

4.2.1 Mathematical description

As the base of the model, GloVe uses the ratio of word occurrence probabilities. Rather than calculating the probability of the relationship between two words, i and j, directly, GloVe uses a variety of "probe" words, k. The model makes use of P_{ik} and P_{ij}, the probability of word k appearing in context of word i and word j respectively, and computes the ratio of these two probabilities to deduce the similarity between word i and word j.

The derivation of the model begins with the ratio of these two probabilities: F(w_i, w_j, $\tilde{w}_k$)=$\frac{P_{ik}}{P_{jk}}$. Then, a series of transformations is applied including: taking the difference between w_i and w_j, transposing that difference and taking the dot product with w_k, setting F=exp, and applying the log function and adding a bias. This results in the final model which is in a least squares regression form. The model is shown below:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logX_{ij})^2$$

Where, $\tilde{w}_j$ is a context vector, w_i is a word vector, V is the size of the vocabulary, X_{ij} is the number of co-occurences of i and j, and f(x) is the activation function. The activation function takes the form:

$$f(x) = \begin{cases} (x/x_{max})^\alpha & x < x_{max} \\ 1 & otherwise \end{cases} \tag{1}$$

The complexity of the model is around $O(|C|)$, where C is the corpus.

4.2.2 Implementation description

Our implementation of GloVe was heavily adapted from [24]. The model is composed of two embedding layers, one for w_i, and one for $\tilde{w}_j$, two parameter tensors for b_i and $\tilde{b}_j$, and the weighting function. The forward function of the model multiplies the two word vectors, adds the two biases, and subtracts the log of the number of co-occurences of the two words, and passes that result into the weighting function. The implementation begins by calculating the co-occurrence matrix from the corpus, creating a matrix of size $|V| * |V|$ and using a window size of 10. Then the model is trained for 20 epochs on the parsed 1994 SEC filings dataset, using AdaGrad as the optimizer. The final output of the model includes the word embeddings for each word in the vocabulary.

4.3 BERT and BERT based architectures

4.3.1 Mathematical description

BERT [5] is a language model that stands for Bidirectional Encoder Representations from Transformer. The main issue with the models such as word2vec and GloVe is that they fail to capture the relationships based on the context in which a word appears. BERT addresses this shortcoming as it uses transformer encoders to generate dynamic bidirectional representation for a word. It uses bidirectional conditioning which allows it to see future context as well as the past context to generate meaningful embeddings.

The novel development in transformer [25] lies in that it got rid of recurrent neural layers entirely and relied entirely on attention. This form of attention was called "Scaled Dot-Product Attention". The attention is usually computed on matrices of Queries Q, Keys K and Values V in practice to leverage fast matrix multiplication. The attention, thus computed, is scaled down by $\sqrt{d_k}$ as they achieve more stable results.

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

The transformer paper [25] proposes the use multiple attention heads and concatenting the results. Muti-head attention enables the model to attend to information that is present in different subspaces more effectively and it is computed as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_m)W^O$$

where there are m heads and W^O is used to project the result of the concatenation to produce the final output of the attention layer.

As the model does not contain any recurrence, the information contained in the order of the sequence is lost. Thus, positional encoding is used to feed this information into the model. The popular form of encoding includes using sine and cosine functions of different frequencies. These encodings are computed as follows:

$$PE_{(x,2i)} = sin(\frac{x}{10000^{2i/d_{model}}})$$

$$PE_{(x,2i+1)} = cos(\frac{x}{10000^{2i/d_{model}}})$$

where x is the position, i is the dimension and d_{model} is the dimension of the embeddings.

We also worked with distilled models, distilbert and distilroberta. These models were created using knowledge distillation, which is a method where the student (the distilled models) learns to behave like the teacher (BERT and Roberta). The student models are trained using a combination of the distilation and the training loss. For BERT and Roberta the training loss is the masked language modeling loss. The distilation loss is defined by

$$L_{ce} = \sum_i t_i * log(s_i)$$

where t_i represents the probability generated from the teacher. This allows the student model to learn from the probability distribution that the teacher generates.

4.3.2 Implementation description

In order to experiment with the range of models, we used hugging faces API for Transformers [26]. We experimented with: AlBERT, BERT Uncased, DistilBERT Cased, DistilBERT Uncased, DistilRoBERTa, and BERT Cased. The tokenizer also adapted to the model being used through Hugging Face's Auto Classes. The API allows us to reuse much of our code across experiments while training vastly different models and ensure that our experiments don't contain basic errors.

The process for training these models is discussed further in Section 5.

5 Data processing pipelines

We have slightly different configurations of our data processing pipeline from our midterm report to the final experiments with BERT and BERT based architectures. This is shown in Figures 5 & 6.

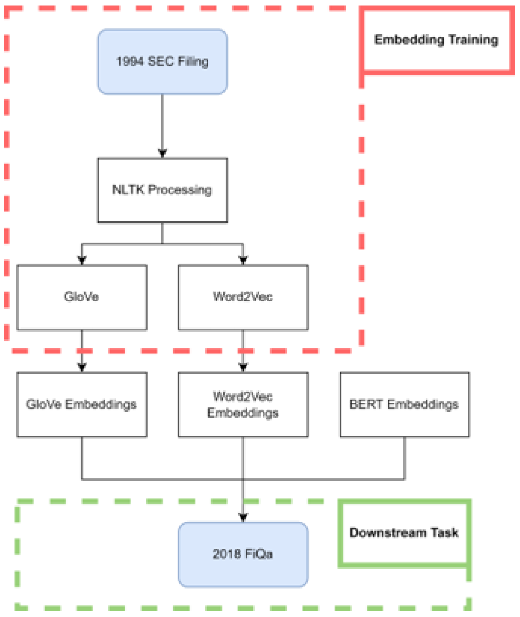

Figure 5: The initial pipeline starts with cleaning the 1994 SEC filings to train both GloVe and word2vec. These models are used to generate word embeddings to train a biderectional LSTM on the FiQA task comparing performance to word embeddings from a pre-trained BERT.

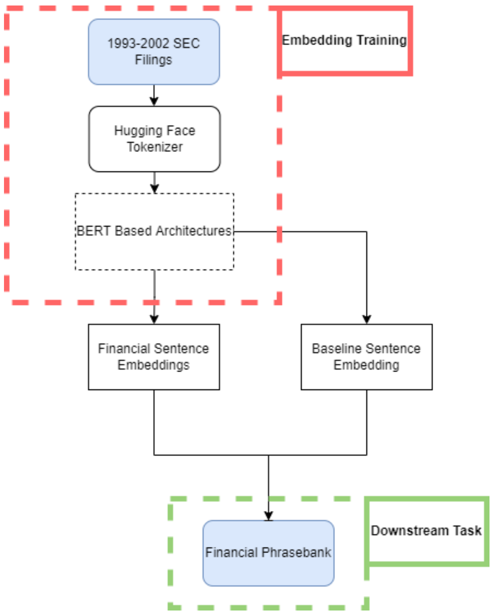

Figure 6: The final pipeline starts with training BERT based architectures on tokenized sentences from SEC Filings 1993-2002. These models are then used to generate embeddings for the Financial Phrasebank task and are evaluated against the pre-trained plain english baseline.

6 Research methods

Initially, we compared the performance of GloVe and word2vec trained on the 1994 SEC dataset against embeddings that are generated using BERT as our baseline. We thought BERT would be a good comparison since it can perform contextualized word embedding whereas GloVe and word2vec are both non-contextualized. For the evaluation, we use the squared sum of residuals as our metric. This is a good metric to use for our initial set of experiments as the labels are fractional values and we wish to penalize larger differences in predictions more heavily. The loss is calculated over all FiQA dataset samples $d \in D$, where y_i is the true sentiment score and $\hat{y}$ is the sentiment score computed by our model, as follows

$$\text{Loss} = \sum_{i=1}^{D} (y_i - \hat{y_i})^2$$

The results of these initial experiments can be seen below in Figure 7. From this we saw that training on financial data improved downstream performance significantly. Due to domain specific knowledge, word2vec even outperformed the plain English BERT embeddings which demonstrates the value of domain specific training. From our literature review, we decided to proceed with tuning BERT based architectures on the SEC data set.

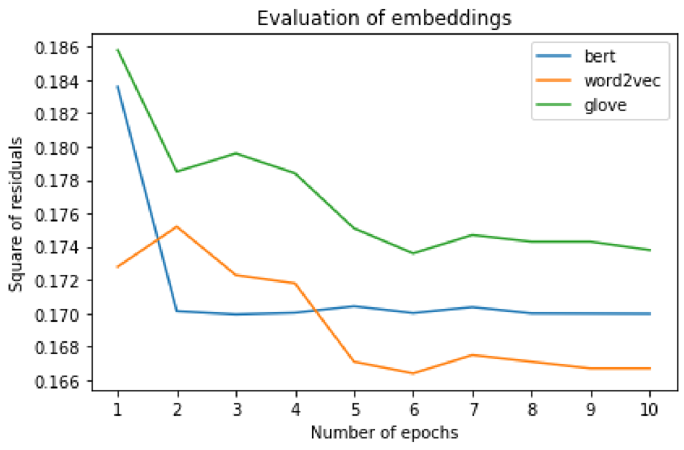

Figure 7: As the LSTM trains, the MSE on FiQA of financial domain specific embeddings (word2vec and GloVe) versus generic BERT.

6.1 Ablation studies

To ensure efficient use of computing resources in our search for optimal hyperparameter settings, we decided to take the block coordinate descent approach [27] where we experiment with one parameter at a time while keeping remaining settings constant. The best setting is used for that parameter as further experiments are performed on a different parameter. This proves to be a useful and systematic method for searching through the hyperparameter space especially given we are tuning the values for a set of highly related BERT-based models.

6.1.1 Learning rate

The first parameter we experimented with was learning rate of $1e-6, 5e-6, 1e-5, 2e-5$. Using a Distil BERT model without label smoothing and using the linear scheduler, it was found that a setting of $1e-5$ was optimal for producing embeddings as evaluated by the downstream task, see Figure 8, and this value is used in all later experiments.

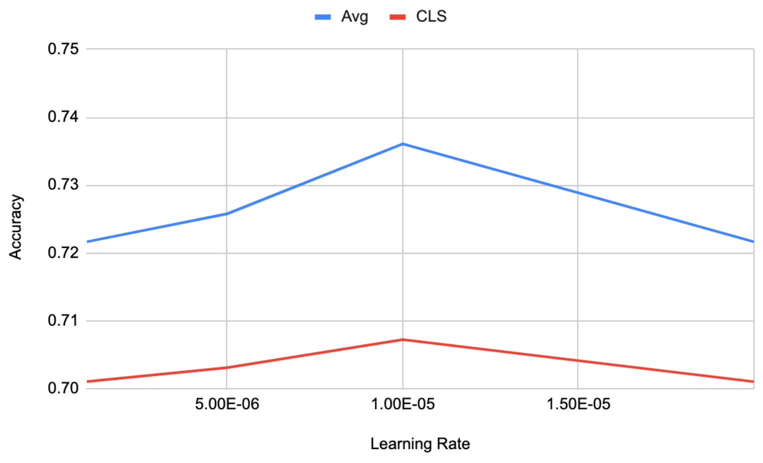

Figure 8: Learning rate experiments

6.1.2 Label smoothing

Next we tested label smoothing values of $0, 0.1, 0.15$. Using a Distil BERT model and using the linear scheduler, it was found that any amount of label smoothing above 0 was detrimental to the embeddings used in downstream task performance, see Figure 9. For this particular experiment, we think the results can be explained with the fact that label smoothing is a form of regularization introduces noise for the labels to prevent the model from being overconfident in its predictions. The trade off is the introduced noise could also reduce the significance of the signal in the original data. Our experiments suggest that the second case dominates in our model so we set label smoothing to 0 all later experiments.

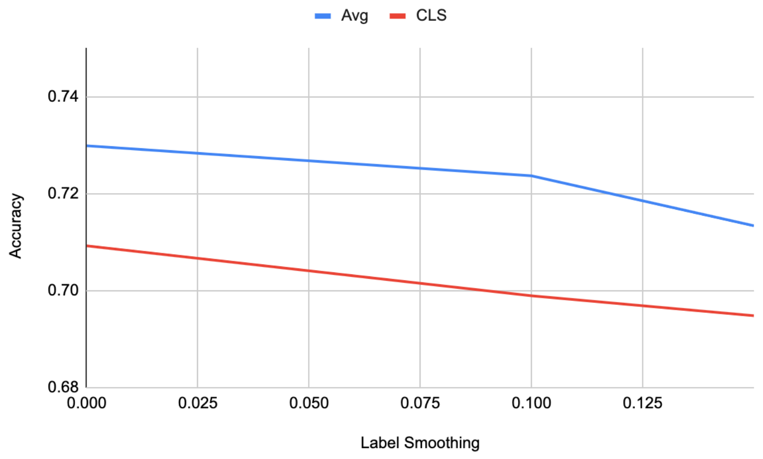

Figure 9: Label smoothing experiments

6.1.3 Learning rate scheduler

The next experiment involved testing different schedulers, for which we tried the default value of linear scheduler, cosine annealing scheduler and constant scheduler. Using a Distil BERT model, it was found that the default linear scheduler produced the best embeddings for downstream task performance, see Figure 10. The linear scheduler was used in all subsequent experiments.

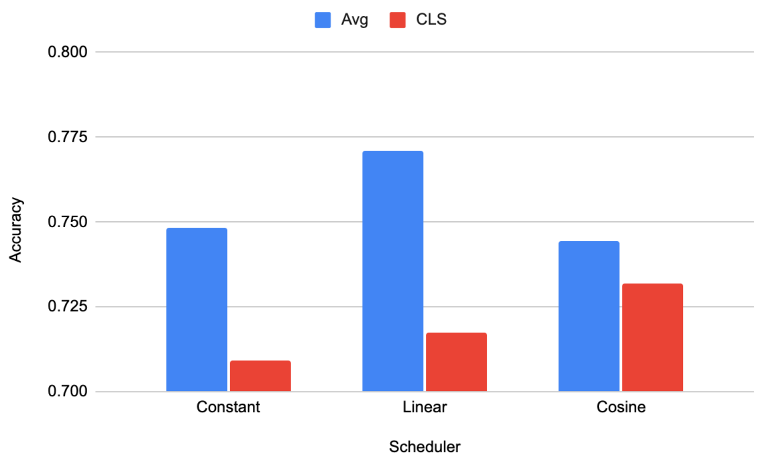

Figure 10: Scheduler experiments

6.2 CLM vs MLM

Masked Language Modeling (MLM) and Causal Language Modeling (CLM) are both pre-training approaches used in current NLP models [28]. In both approaches, a word in the given sequence is masked, and the model must predict the masked word. In MLM, words on both sides of the masked word are used as context, whereas in CLM only words on the left side are considered. MLM is used by BERT during pretraining, and CLM is used by other transformer based models like Gpt2.

We ran a few tests to compare CLM and MLM for our task. Using distilgp2 as the CLM model, we found that the pre-trained model achieved 62.7% accuracy on our downstream task, and the fine tuned model achieved a 65.8% accuracy rate. While this was impressive performance, we observed that CLM performed poorly compared to MLM, and decided to focus our efforts on MLM models.

This is expected, as MLM models have more information about the masked word, and thus the prediction is more accurate. Additionally, MLM models are typically preferred when the goal is to generate a representation of the input, whereas CLM models are more useful for tasks such as predicting text.

6.3 BERT variants

We also ran tests to compare the different variants of BERT. Our initial experiments used the same experimental setup. However, we noticed that BERT took a long time to train on the dataset. For example, it took **46 hours** for one epoch of training on the entire dataset. As such training was not feasible in an experimental setup, we decided to reduce the dataset and **normalized** it across different models based on the number of hours it takes. Thus, we **clipped** the dataset such that the BERT model could be trained in a feasible amount of time. The variants we tested include: DistilBERT, DistilRoberta, ALbert, cased and uncased BERT. The cased model differentiated between capitalized and uncapitalized words in the text, whereas the uncased model does not. Our results for the variants can be seen below:

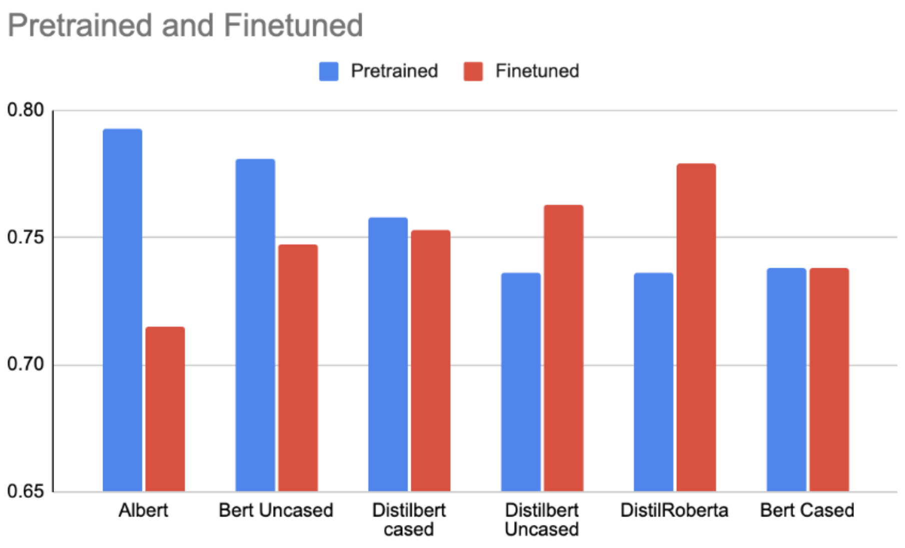

Figure 11: Label smoothing experiments

Overall we found that the distilled models performed much better on the downstream task after training the model on financial data. This is expected behavior, as the larger models (BERT, Albert) require more resources, so we were able to train the distilled models more effectively. These results are promising, as they indicate that performance on the downstream task does improve with fine tuning.

6.3.1 Hidden states and attention layers

Similar to how deep Convolutional Neural Networks capture information pertaining to the input (such as image), intermediate hidden and attention layers can capture useful representations of the data. Thus, we carried out several experiments by concatenating different combinations of hidden and attention layers to generate vectors for our embeddings.

We observed that using attention layers to generate embeddings gave us results poorer than our baseline. However, embeddings generated using concatenation of the hidden states performed significantly better than the baseline. The figure 12 shows the results of our experiments. We observe that our combinations of hidden states outperform the baseline significantly, giving us an improvement of 8% when we use the last 6 hidden layers.

Most of the powerful results from BERT come by using the model outputs to perform a downstream task. This leads us to think that the model outputs form very good embeddings. In *BERT rediscovers the classical NLP Pipeline* [7], upon analyzing how BERT processes information, it behaves as a classical NLP pipeline (further details of which can be found in the paper or our report). Thus, the model outputs from BERT are a good replacement for the entire pipeline that generates representations of the input. This corroborates why BERT generates good embeddings. This is supported by not only literature but also by our experiments.

What Does BERT look at? An Analysis of BERT's Attention [8] looks at what the attention maps in BERT capture. Most attention heads are able to learn some aspects of linguistic knowledge. For example, some of them learn about the relationships between objects and the verbs they relate to. However, such information is scattered across attention maps. Thus, our hypothesis (which is backed by the results that we got through our experiments) is that attention maps are able to attend to capture the nuances of a language to a good degree, however, when used together to generate a meaningful single representation (i.e embedding), it fails to do a good job.

On the other hand, what do the hidden states capture? Similar to how the hidden states in RNN capture the essence of input and encode it in a single representation, the hidden states in BERT do a similar task. Note that BERT comes from the transformer [25] architecture, which replicates Seq2Seq architecture (with an encoder and decoder component). The success of the transformer architecture has shown that transformer encoders do a great job in encoding the information of an input and the transformer decoders do a great job in decoding this information. BERT, being a language modeling technique that arises from the transformer encoders, is therefore very good at encoding this information. BERT is composed of several transformer blocks and each block generates a hidden state which is passed on to the next block. The final hidden state is able to encode a lot of this information. However, inspired by this reading [29], we hypothesized that the intermediate hidden states too could have information that our embeddings could benefit from.

The vector that provides the best contextualized embedding is usually specific to the downstream task itself. However, an 8% improvement in our accuracy strongly advocates that the using different combinations can significantly improve the quality of embeddings.

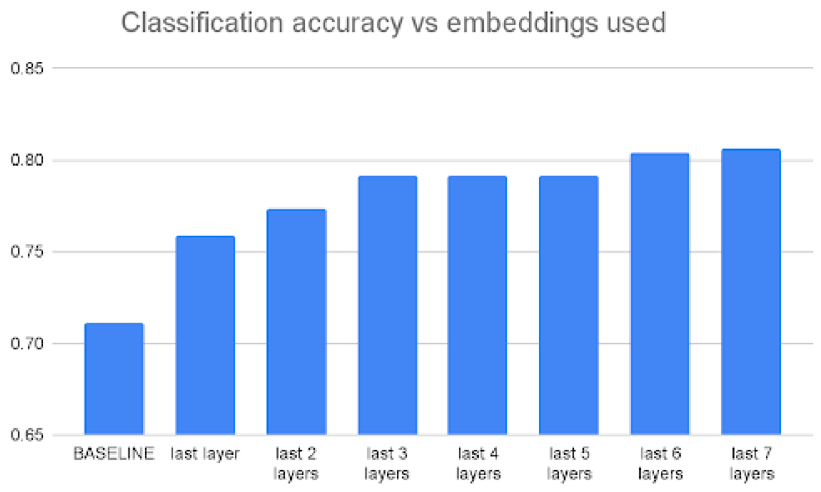

Figure 12: The classification accuracy of the model varies depending on which hidden layers we use to generate the embeddings vector

Combining hidden layers together actually combines the information encoded in various layers together. Similarly, combining attention maps is also a form of ensemble. [8] shows that different layers learn different parts of the sentence, generating different forms of embeddings. We thought this was an interesting avenue to perform cross model ensembling and examined the performance of cross model ensembles and the results of our experiments as shown in Figure 13.

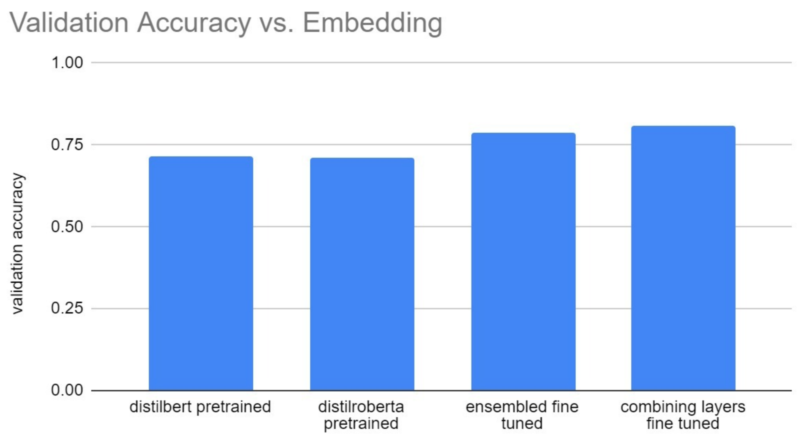

Figure 13: The classification accuracy of the model varies depending on which hidden layers we use to generate the embeddings vector

Though ensembling across models did help, we observed that combining hidden state information across layers outperforms this form of ensembling. A possible explanation for this may be that different layers in BERT pick up different aspects of the sentence so ensembling layers seems to be more effective than ensembling final model outputs.

7 Results

A summary of our results on the midterm and final can be found in Tables 14 and 15. The results from the addition of hidden layers for sentence representation are shown in both Figure 16 and Figure 17. The improvement was from 71% to 80% and an AUC improvement of 0.8066 to 0.8993 by training a Distil BERT model on unlabeled data gathered from SEC Filings from 1993 - 2002. Through these results we see a large improvement over general language models by both training for domain specific knowledge and more effectively capturing this knowledge in our embedding.

Model	Phrasebank Accuracy	FiQA MSE
wor2vec	52.2%	0.166
GloVe	60.7%	0.174
BERT	73.8%	0.170

Figure 14: The table shows results from our interim report and includes the Phrasebank accuracy of these models.

Model	Phrasebank Accuracy Pretrained	Phrasebank Accuracy Finetuned
AlBERT	79.3%	71.5%
BERT Uncased	78.1%	74.7%
DistilBERT Cased	75.8%	75.2%
DistilBERT Uncased	73.6%	76.3%
DistilRoberta Cased	73.6%	77.9%
BERT Cased	73.8%	73.8%

Figure 15: The table shows all models improvement without the addition of using hidden layers.

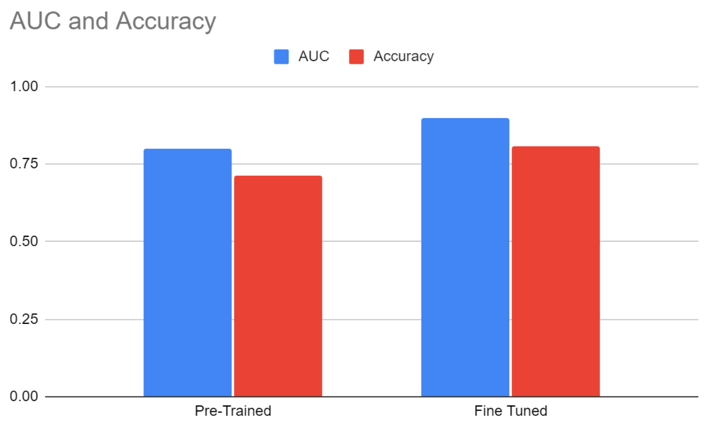

Figure 16: The improvement in both AUC and accuracy of the final pre-trianed and fine-tuned model including the addition of hidden tasks on the Phrasebank sentiment analysis task.

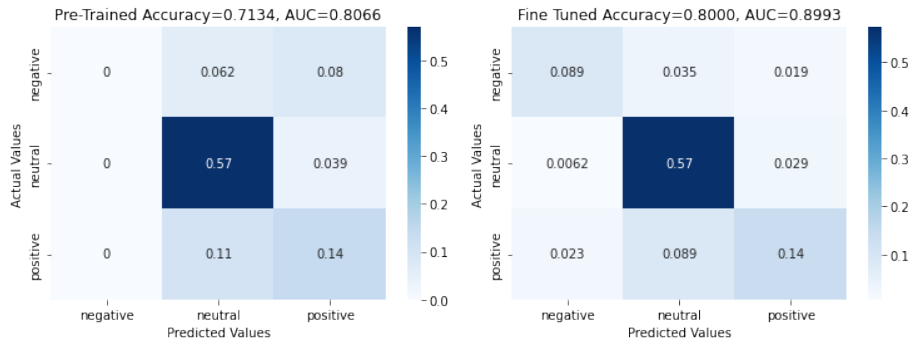

Figure 17: The confusion plots of the final pre-trianed and fine-tuned model including the addition of hidden tasks on the Phrasebank sentiment analysis task.

8 Future work

8.1 Dataset cleaning and augmentation

One of the future works would include using a cleaner dataset that is more specific to the financial domain. Akshat did a great job in providing us with a dataset that is domain-specific. However, we think manual inspection of the dataset to understand the dataset better and improve its quality could help get better performance. Another improvement could come from augmenting the SEC data used for training the embedding model. For training FinBERT [20] they used analyst reports and earnings call transcripts in addition to SEC Filings' "textual components".

8.2 Ensembling of hidden layers for embedding

Exploring the quality of embeddings that are generated by using different combinations of hidden layers is also an interesting future extension. Looking at why attention layers do not do well is also an interesting extension and it would be interesting to see if they can be used in an effective manner by some modifications.

8.3 Alternative downstream tasks

In general, it is common to evaluate the quality of an embedding using a suite of downstream tasks. For general NLP embeddings, this is generally achieved using the SuperGLUE or similar benchmarks. Given more computational resources and access to domain experts, we would construct a suite of downstream financial tasks to evaluate our embeddings to give a more reliable measure of the quality of our generated embeddings. The inclusion of more sophisticated benchmarks, such as question answering, could also be used to gauge the effectiveness of our embeddings for understanding numerical values which is often key in financial documents.

8.4 Distance weighted attention

While using full attention on any particular word in an input sentence is what makes encoder based embeddings so powerful, it is often useful to impose some limits on attention values as was implemented in the T5 model [13]. For example, in the T5 experiments, the attentions are weighed by their distance from the target segment. This would modify the model to take ideas from both encoder based embedding models as well as the original word2vec model. While this is an interesting idea to explore, we did not have the capacity to test it in detail.

8.5 More exhaustive hyperparameter tuning

To reduce the amount of computation required in our ablation studies, we chose to take a block coordinate approach. We understood that this may not yield the best combination of parameters for each specific model we tried and relied on the assumption that the parameters for all BERT-like models we tried are likely to have similar impact on the downstream results. Given more resources, we could have more exhaustively search for the best parameters for each model to quantify their true potential.

8.6 Time weighted models

A hypothesis we did not explore was that the proximity of the date of the SEC data to the data used in the downstream task may have an impact on the quality of the embeddings. For example, given that the data used in the Phrasebank task is collected circa 2018, we may expect embeddings trained on SEC data from the 2000's to be more relevant than the SEC data from the 1990's. Ideally we would train separate embedding models for each decade or similar timespans and have the downstream models learn how to best weigh the embeddings from different time periods.

9 Conclusions

Through this work, we demonstrated that fine-tuning of embedding models on data similar and relevant to the downstream tasks can significantly improve performance. With the right combination of hyperparameters and model selection, we were able to achieve an accuracy improvement from 71% to 80% and an AUC improvement of 0.8066 to 0.8993 by training a Distil BERT model on unlabeled data gathered from SEC Filings from 1993 - 2002, see Figure 16. While optimizing hyperparameter settings through ablation studies is important, we also showed that experimentation with using information from intermediate hidden and attention layers can confer considerable downstream task performance. Future work can be expanded upon with larger models, further hyper parameter tuning, continued experimentation with hidden layers, data augmentation, and distance weighted attention.

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: `1301.3781 [cs.CL]`.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: `http://www.aclweb.org/anthology/D14-1162`.

[3] M. E. Peters, M. Neumann, M. Iyyer, *et al.*, *Deep contextualized word representations*, 2018. DOI: `10.48550/ARXIV.1802.05365`. [Online]. Available: `https://arxiv.org/abs/1802.05365`.

[4] C. Chelba, T. Mikolov, M. Schuster, *et al.*, *One billion word benchmark for measuring progress in statistical language modeling*, 2013. DOI: `10.48550/ARXIV.1312.3005`. [Online]. Available: `https://arxiv.org/abs/1312.3005`.

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. arXiv: `1810.04805`. [Online]. Available: `http://arxiv.org/abs/1810.04805`.

[6] "Hugging face – the ai community building the future." (2022), [Online]. Available: `https://huggingface.co/` (visited on 04/03/2022).

[7] I. Tenney, D. Das, and E. Pavlick, "BERT rediscovers the classical NLP pipeline," *CoRR*, vol. abs/1905.05950, 2019. arXiv: `1905.05950`. [Online]. Available: `http://arxiv.org/abs/1905.05950`.

[8] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, *What does bert look at? an analysis of bert's attention*, 2019. DOI: `10.48550/ARXIV.1906.04341`. [Online]. Available: `https://arxiv.org/abs/1906.04341`.

[9] Y. Liu, M. Ott, N. Goyal, *et al.*, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. arXiv: `1907.11692`. [Online]. Available: `http://arxiv.org/abs/1907.11692`.

[10] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. arXiv: `1906.08237`. [Online]. Available: `http://arxiv.org/abs/1906.08237`.

[11] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," *CoRR*, vol. abs/1909.11942, 2019. arXiv: `1909.11942`. [Online]. Available: `http://arxiv.org/abs/1909.11942`.

[12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. arXiv: `1910.01108`. [Online]. Available: `http://arxiv.org/abs/1910.01108`.

[13] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *CoRR*, vol. abs/1910.10683, 2019. arXiv: `1910.10683`. [Online]. Available: `http://arxiv.org/abs/1910.10683`.

[14] S. Sehrawat, "Learning word embeddings from 10-k filings for financial nlp tasks," *Available at SSRN 3480902*, 2019.

[15] L. Loukas, M. Fergadiotis, I. Androutsopoulos, and P. Malakasiotis, "EDGAR-CORPUS: billions of tokens make the world go round," *CoRR*, vol. abs/2109.14394, 2021. arXiv: `2109.14394`. [Online]. Available: `https://arxiv.org/abs/2109.14394`.

[16] J. Kang, I. E. Maarouf, S. Bellato, and M. Gan, "FinSim-3: The 3rd shared task on learning semantic similarities for the financial domain," in *Proceedings of the Third Workshop on Financial Technology and Natural Language Processing*, Online: -, Aug. 2021, pp. 31–35. [Online]. Available: `https://aclanthology.org/2021.finnlp-1.5`.

[17] "Financial opinion mining and question answering." (2018), [Online]. Available: `https://sites.google.com/view/fiqa/home` (visited on 04/03/2022).

[18] E. Rahimikia, S. Zohren, and S.-H. Poon, "Realised volatility forecasting: Machine learning via financial word embedding," *Available at SSRN 3895272*, 2021.

[19] D. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," *CoRR*, vol. abs/1908.10063, 2019. arXiv: `1908.10063`. [Online]. Available: `http://arxiv.org/abs/1908.10063`.

[20] Y. Yang, M. C. S. Uy, and A. Huang, "Finbert: A pretrained language model for financial communications," *CoRR*, vol. abs/2006.08097, 2020. arXiv: `2006.08097`. [Online]. Available: `https://arxiv.org/abs/2006.08097`.

[21] A. HAYES. “SEC filings: Forms you need to know.” (2021), [Online]. Available: https://www.investopedia.com/articles/fundamental-analysis/08/sec-forms.asp (visited on 03/06/2022).

[22] N. Project. “Natural language toolkit.” (2022), [Online]. Available: https://www.nltk.org/ (visited on 04/03/2022).

[23] P. Malo, A. Sinha, P. Takala, P. J. Korhonen, and J. Wallenius, “Good debt or bad debt: Detecting semantic orientations in economic texts,” *CoRR*, vol. abs/1307.5336, 2013. arXiv: 1307.5336. [Online]. Available: http://arxiv.org/abs/1307.5336.

[24] P. Yan. “A comprehensive python implementation of glove.” (2021), [Online]. Available: https://towardsdatascience.com/a-comprehensive-python-implementation-of-glove-c94257c2813d (visited on 04/03/2022).

[25] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017. arXiv: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[26] H. Face. “Hugging face: Transformers - auto classes.” (2021), [Online]. Available: https://huggingface.co/docs/transformers/model_doc/auto#transformers.AutoModelForMaskedLM (visited on 05/02/2022).

[27] H. Lyu, *Convergence and complexity of block coordinate descent with diminishing radius for nonconvex optimization*, 2020. DOI: 10.48550/ARXIV.2012.03503. [Online]. Available: https://arxiv.org/abs/2012.03503.

[28] rakhar Mishra. “Understanding masked language models (mlm) and causal language models (clm) in nlp.” (2021), [Online]. Available: https://towardsdatascience.com/understanding-masked-language-models-mlm-and-causal-language-models-clm-in-nlp-194c15f56a5 (visited on 05/02/2022).

[29] J. Alammar. “The illustrated bert, elmo, and co.” (2021), [Online]. Available: https://jalammar.github.io/illustrated-bert/ (visited on 05/06/2022).