NLP DRIVEN LARGE SCALE FINANCIAL DATA ANALYSIS

BY

ZHE XU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Adviser:

Assistant Professor Bo Li

ABSTRACT

Stock market volatility is influenced by several factors including but not limited to information release, dissemination, and public acceptance. In terms of information, traders would read financial reports released by companies, 8-K reports, mass but informal social media information like tweets, as well as financial topic or market related reports from professional news agencies such as Reuters, Bloomberg, Yahoo Finance etc. Among them the financial news is relatively informative and abundant, therefore I take it as source of data.

Natural language preprocessing (NLP), as a classical yet still prosperous research domain, has seen astonishing progress with the development of deep learning and widely used for texts related tasks such as text sentiment analysis and recommendation system. However, currently there is no widely acknowledged framework on utilizing NLP technique to mine news articles under finance or marketing topic for valuable information about stock market. stock price due to several practical issues.

My contribution is as follows:

I analyze influences of several factors in my whole pipeline for NLP based stock trend prediction. In particular, I crawl and preprocess data online and made a new dataset, and compared with a Reuters news dataset. I compared the underlying word embeddings constructed based on different corpora and preprocessing techniques, which are later used as input for my prediction models in terms of prediction accuracy as well as the corresponding models' robustness against adversarial attacks. I conduct adversarial analysis and provide my hypothesis which might be useful to further develop the NLP based framework to mine information for stock market related tasks such as stock trend prediction. My hypothesis on the adversarial analysis is that the current model, despite its success on texts rich in sentiment, is vulnerable to adversarial attacks and not stable enough with low frequency terms such as domain specific multi-words phrases, and numbers whose semantic meaning should not be represented by numbers. These two types of tokens would be paid attention to if a more robust model is desired.

My experiments suggest that raw data quality and underlying embeddings used for NLP models are of vital importance in terms of prediction performance, and although current state of the art NLP models are capable of learning the sentiment-wise semantic meaning of words, it might not be properly structured to utilize more complex and heterogeneous information such as financial or marketing news.

To my families and friends, for their love and support.

ACKNOWLEDGMENTS

Thanks IBM research scientists Dr. Pin-yu Chen and Dr. Yada Zhu for their guidance, discussion, and assistance in acquiring data.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

With the development of natural language processing (NLP) in both academia and industry, vast amount of useful apps and programs are becoming more and more poIrful. From restaurant recommendation to movie ratings, more and more customers are benefiting from the convenience that NLP techniques provide. Stock market, which is a critical source of revenues for many people and organizations, has been gaining attention from news agencies, individuals, and government associated institutions, and thus there are abundant corpora about it as well. From commercial giant institutions to individual traders, everybody on the stock market try to collect information, analyze them, and make wise decisions about selling out or buying in, maintaining a good portfolio to maximize their properties. However, since the stock market, which is influenced by many factors, is to a large extent unstable or even stochastic, it is relatively hard a hard task and there is currently no mature and satisfying solution.

Stock market related corpora could be categorized into several class: reports released by companies, informal corpora on forums or social media, as well as news articles from news agencies. Among them, social media corpora such as tweets used by [1],[**?**], often contain too much noise, such as unrelated slogans or advertising materials, sometimes with hash tags simply to capture attention from people, is too chaotic. In addition, the cost of collecting and maintaining such data everyday is not affordable to all. Documents required by government such as 8-K reports, used by [2] are informative but required only for few types of events such as bankruptcy. News articles are both informative and relatively abundant, therefore chosen by this project as source of data.

News information related to companies listed on the stock market appears constantly, with immediate impact on stock prices. Big trading institutions as well as individual investors all value news information significantly and make their decision based on the important news report. However, due to issues including the lack of publicly acknowledged dataset, different information processing techniques, different selection of underlying word embeddings etc., research in this domain still has a long way to go.

As pointed in [**?**], there are many difficulties, both practically and conceptually, which leads to the lack of sufficient trustworthy work to compare with. Practical difficulties includes issues such as the natural sparsity of important financial news, website coding issues etc. Conceptual issues include the neutrality of news reports, difficulty to set up a reasonable pattern to focus on, etc. Also, I find other issues such as the imbalance of news reports associated with different companies. the However, the return is worthwhile: if I can gain

useful information about stock price trend of certain set of companies in future, I can utilize it for further tasks such as portfolio management. Also, to extrapolate, stock market related news is representative for a type of corpora: in such articles, there are named entities, numbers as well as words indicating sentiment (such as "fall", "rise"), and thus is a good and extrapolatable entry point to study specially structured models that is proper for this type of corpora.

Since previous works either 1)used improper data that is too casual such as tweets as in [1],[?] or too sparse as in [2] that can not assist traders on daily analysis for individual stocks 2) mixed text information with heterogeneous information such as historical price information that diluted the weight of pure text information as in [?] 3) failed to further explore the factors that influence the predictability or performance on NLP-stock tasks as in [?] or 4) made analysis on macro-level sentiment analysis which is not helpful for traders who are usually interested in individual stocks as in [?]. I decide to conduct my own full-pipeline analysis to further figure out the impact of various factors and gain a deeper understanding.

I analyze influences of several factors in my whole pipeline for NLP based stock trend prediction. I collect and process my own dataset from both publicly accessible website and IBM research to figure out the influence of dataset. I try different set of word embeddings from pretrained embeddings which is based on mass text corpora, to self-trained AutoPhrase embeddings which captures semantically meaningful multi-words phrases, on financial news only corpora in order to denoise the word context information. I use hierarchical attention network to predict the stock trend given the news on previous trading day, and compare the feasibility of using attention obtained during the evaluation process to detect important words, v.s. using adversarial attacks which convert the prediction results to help my analysis.

I have observed several interesting phenomenons and provided my interpretation as well as some hypothesis, which I believe are useful for further study. I find that clean news are of vital importance to predict the stock price, and different from what is proposed in highly cited previous work [?] which indicated that sparsity of articles is harmful for prediction so I should try my best effort to collect articles, my experiment suggests that the quality of news weigh more than the quantity. By experiment on different embeddings, my experiment suggests that for NLP-stock task, embedding trained on small size of topic-specific corpora with phrase mining techniques can lead to equally good or even better performance, compared with embedding that is trained on massive but noise text data. My experiment on attention and adversarial analysis suggested that tokens with high attention are usually not semantically significant, and using more deliberate techniques such as adversarial analysis can help uncover "important" words in texts that match human intuition better and are of higher interpretability. I also found that despite of good-quality data and reasonable model

I adopt, financial articles are pretty vulnerable to adversarial attacks, which indicates more complicate model that can handle heterogeneous information.

To summarize, my contribution is as follows:

- I manually crawl a dataset via a third-party API intrinio to get the url of news articles and crawl news from Yahoo, which consists of more than 20,000 articles of 9 companies within 1 year, and also conduct experiment with a dataset from IBM which consists of more than 20,000 articles of 9 companies within 7 years.

- I compared the underlying word embeddings constructed based on different corpora and preprocessing techniques, which are later used as input for my prediction models, from the perspective of prediction accuracy as well as the corresponding models' robustness against adversarial attacks.

- I conduct adversarial analysis and provide my hypothesis which might be useful to further develop the NLP based framework to mine information for stock market related tasks such as stock trend prediction.

The remaining part of this thesis is organized as follows: Chapter 2 describes related works from several perspectives, including the previous works on utilizing texts information and NLP techniques for stock analysis, embeddings which map discrete word to vectors, the neural networks that have used on similar tasks, adversarial attack and corresponding background knowledge. Chapter 3 explains details of the data I use for experiment, as well as the acquisition of intrinio dataset and the analysis of some intrinsic properties of my data. Chapter 4 describes the methods I use in different phases of the pipeline, including the whole framework, the neural network I use, AutoPhrase embedding training procedure, adversarial analysis etc. Chapter 5 shows the result of my experiments and some analysis, and lists the case study using adversarial analysis to figure out the important words that the learned models emphasize, my conclusion and hypothesis on the adversarial attack results and key factors to improve the models. Chapter 6 summarizes my conclusion and possible future works.

CHAPTER 2: BACKGROUND AND RELATED WORKS

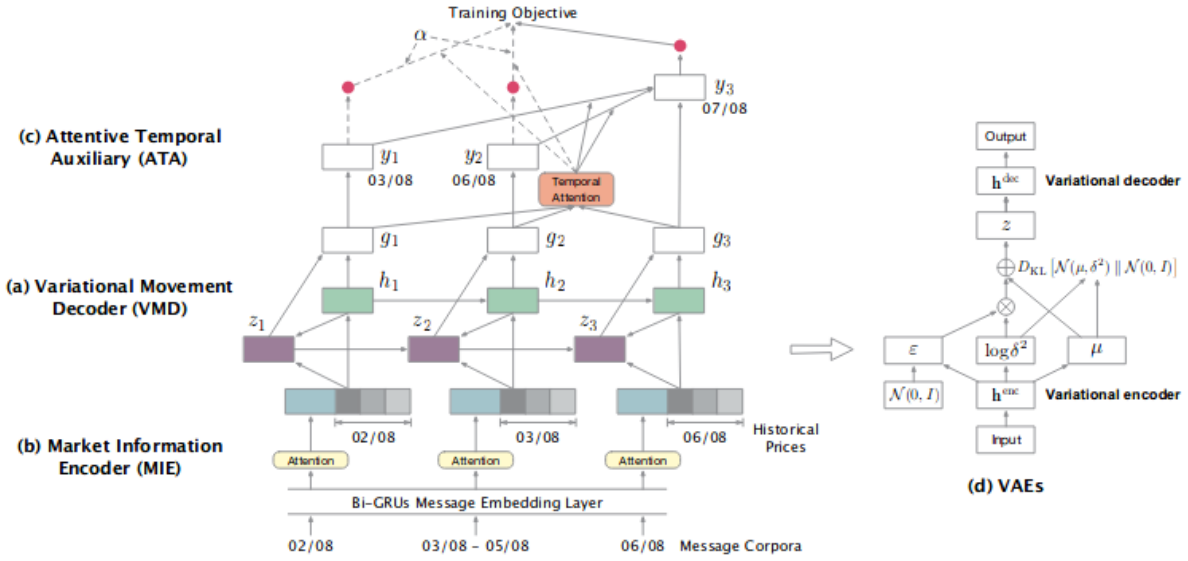
2.1 PRIOR WORKS ON NLP-STOCK TASKS

There are some interesting papers discussing utilizing NLP information for stock market analysis, such as [2], [1], [3].

[2] discussed using Form 8-k, a report of unscheduled material events or corporate changes at a company that could be of importance to the shareholders or the Securities and Exchange Commission (SEC). Form 8-k notifies the public of events reported including acquisition, bankruptcy, resignation of directors, or a change in the fiscal year. The dataset was downloaded from from the Securities and Exchange Commission website, and they trained this model using a random forest classifier with 2000 trees, to predict whether the price a stock would go up, go down or stay. Their work used manually crafted features, including recent stock price changes, the volatility index, earnings surprise (the ratio between expected and actual earnings per share), and event category (the reason for filing the form 8-K). It is worth mentioning that some features they used are not NLP information, for example, recent stock price. For each 8-K report, the difference in the companys stock price before and after the report is released was calculated and normalized by subtracting the same difference computed for the entire S&P 500 index (stock index GSPC) for the same period.

[1] and [3] instead used neural network and took advantage of casual and ubiquitous corpus from social media. [1] selected the two-year price movements from 01/01/2014 to 01/01/2016 of 88 stocks, from all the 9 industries in stock market. They presented a novel deep generative model StockNet to jointly exploit text and price signals for this task to predict whether a stock's price would go up or down. StockNet comprises three primary components following a bottom-up fashion: 1) Market Information Encoder (MIE) that encodes tweets and prices to random variable X, which encodes observed market information; 2) Variational Movement Decoder (VMD) that infers latent driven factor Z with X,y and decodes stock movements y from X,Z ; 3) Attentive Temporal Auxiliary (ATA) that integrates temporal loss through an attention mechanism for model training. The architecture of their model is illstustrated here:

Figure 2.1: StockNet architecture, figure cited from [1]



Despite the good performance of StockNet which achieves over 58% accuracy of prediction, I think it improper for my project based on several considerations. First of all, StockNet combines both text and price signals, which is reasonable, but it is necessary to point out that their architecture is based on the fact that the text information – twitter data – is available everyday, and might need revising if the input text data is sparse. the experiment result of [1] should that the generative StockNet using only tweet information behaved equally good as the fully equipped version, which indicated that the price information is not a key factor of the success of this model. Second, StockNet makes it harder to separate out the influence of text information.

In comparison with the extremely official Form 8-K which is required by law and filed by companies, and the casual style corpus, I instead try to use news report filed by news agencies, such as Reuters news and yahoo finance.

2.2 NLP & DEEP LEARNING TECHNIQUES ON GENERAL TEXT CLASSIFICATION TASKS

Since my goal is to predict whether the stock price would go up or down using text data, it could be considered as a sequence classification problem. Some other tasks, such as sentiment analysis, of which there are good publicly available datasets (IMDB, yelp etc.), could be categorized into the same domain. There are many algorithms for text classification,

here I mainly consider two types of neural networks – CNN (convolutional neural networks) and RNN(recurrent neural networks).

Before I introduce representative previous works using these two types of models, I want to first explain the main difference between stock movement prediction task with respect to other text classification tasks. Traditional text classification tasks usually adopt datasets like IMDB (movies ratings and comments) and yelp (restaurants ratings and comments), however, it is relatively easy to classify these comments as positive or negative in terms of sentiment, since these comments are rich in personal opinions. In contrast, financial news is a type of news, and one of the most important principle for journalism is neutrality. Famous and responsible news agents such as Reuters seldom publish news with strong sentiment. Also, the movies or restaurants comments are basically made on the same entity that the customers give their rating. In comparison, there are news about the competitors of certain companies in my financial news dataset. The news indicating the stock movement of commercial opponents usually implies the opposite stock movement of a company itself.

Despite the natural attribute of text classification tasks – input are sequential, which indicates that the models should also be with sequential architecture, such as , in particular LSTM (long short-term memory), [4] argued the dominant architectures are rather shallow in comparison to the deep convolutional networks which have pushed the state-of-the-art in computer vision. VDCNN (very deep convolutional neural networks) was proposed for text processing, which operated directly at the character level and used only small convolutions and pooling operations. VDCNN model begins with a look-up table that generates a 2D tensor of size (f_0, s) where f_0 is the size of character embedding and s is fixed size representing number of characters. Despite the success it achieves, as pointed out in [5], RNN architecture models outperform CNN models in terms of comprehension of global/long-range semantics, I decide to use RNN based model for my current experiment, since my data are mainly composed of relatively complex sentences and neutral words.

[6] is a representative model that belongs to the RNN family, and meanwhile one of the best architectures which can capture word level and sentence level attentions and behaves pretty well on sentiment classification tasks. Especially, I are interested in HAN because I think the prediction it gives for a document is to some extent interpreted by the attention weights. HAN architecture has a hierarchical structure that mirrors the hierarchical structure of documents: it consists of several parts: a word sequence encoder, a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer. This two levels attention mechanism enables HAN to attend differentially to more and less important content when constructing the document representation.

Considering the performance mentioned above and the fact that it can further help us

check the attention information to analyze (which would be explained in detail in case study part), I adopt HAN architecture as my model.

2.3 WORD EMBEDDING AS INPUT TO NEURAL NETWORKS

2.3.1 Word Embeddings

Unlike pixels in computer vision tasks, in NLP domain, the atom input – tokens – are discrete and not contiguous, therefore to feed them into neural networks, I usually need to first map these tokens to embeddings. Embedding is a low dimensional vector that is usually pretrained and used to represent a token. It is common to use embeddings pretrained with algorithms as mentioned in [7], [8].

Since all embeddings I used are trained by word2vec which is proposed in [7], I give a brief introduction here. word2vec tries to model the relationship between words by modeling the word-context distribution. There are 2 methods: CBOW (The Continuous Bag of Words) which tries to predict the target center word given its context, as well as skip gram which predicts the context words around a center word. I only introduce the 2nd method considering length of this thesis. given a sequence of training words $w_1, w_2, ...w_T$, the objective of the Skip-gram model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.1}$$

and $p(w_{t+j}|w_t)$ is modeled as

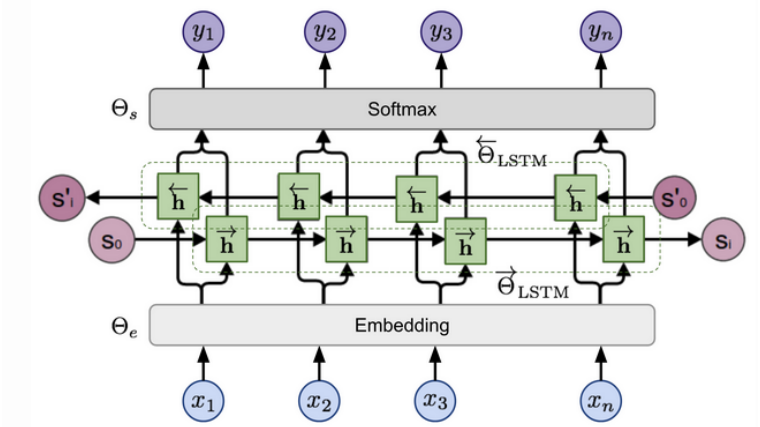
$$p(w_O|w_I) = \frac{exp({v'_{W_O}}^{\intercal} v_{W_I})}{\sum_{w=1}^{W} exp({v'_w}^{\intercal} v_{W_I})} \tag{2.2}$$

where v'_w is the embedding of a word when it is considered as a context word, and v_w is the embedding of a word when it is considered as a center word.

[9], [10] etc. developed another type of embedding. Taking [9] as example, different from the static word embedding strategy described above, a dynamic type of embedding – ELMo(Embeddings from Language Models) representations – was proposed. The intrinsic drawback of "static" word embedding is that the embeddings are fixed after the training stage, which makes it incapable of dealing with ambiguous words issue. For example, "bank" could be associated with contexts containing either "money" or "river", but with embeddings trained using algorithms like word2vec, the word embedding of "bank" would be the

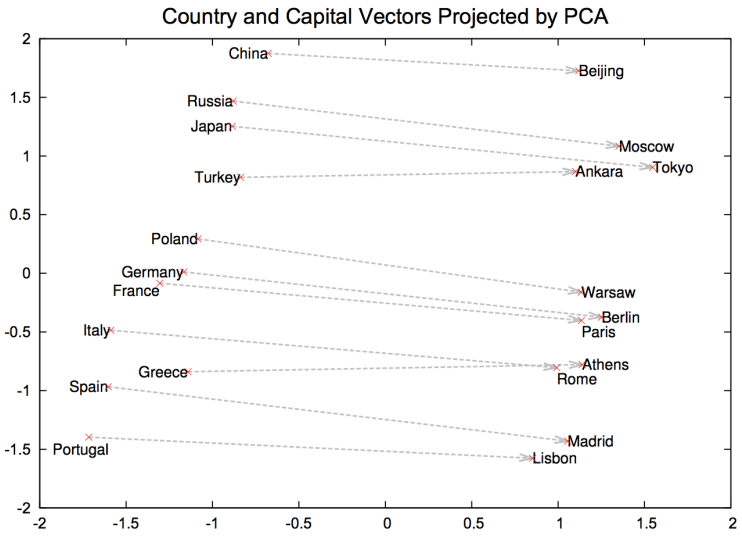
same, no matter what its context is. Elmo tries to solve this issue with a 2 phase method: during the pretraining phase, every word gains 3 embeddings, a word embedding, an embedding corresponding to the 1st layer of bidirectional LSTM which contains more syntactic information, as well as an embedding corresponding to the 2nd layer of bidirectional LSTM which contains more semantic information. During the second phase when these embeddings are used, each embedding would be assigned a weight, and the weighted sum of the 3 embeddings are used as the final word embedding. This mechanism is capable of adjusting the word embedding according to context, and has been more and more popular in recent years. However, limited by time and computational resources, and the fact that ambiguous words is relatively trivial for financial news, I have not incorporated this kind of dynamic embedding into my pipeline.

Figure 2.2: pretraining network architecture of ELMo, figure cited from Christopher Olah's blog



Many pretrained embeddings have been prooven to be semantically reasonable, and could be visualized, as indicated in figure below:

Figure 2.3: word2vec word embedding 2D PCA projection, figure cited from [7]

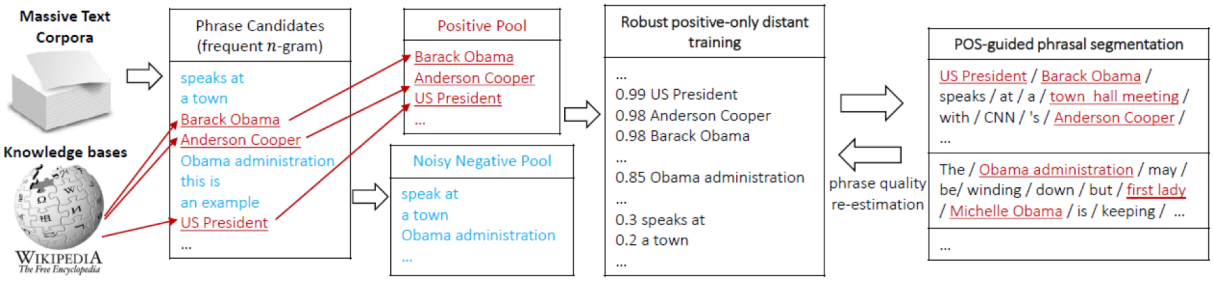


2.3.2 Phrase Mining Assisted Word Embedding Techniques

Despite of the advantages of using pretrained embeddings, considering that generally trained embeddings often failed to capture phrases which are so important as to alter the semantic meaning of a sentence, and that news and articles with financial topic share more similar terms, I also conduct experiment where the corpus is first segmented into single or multi-word phrases. [11] is a good tool and I use that.

This semi-automatic algorithm has several stages: first of all, I take massive text corpora, which would be combined with knowledge bases such as wikipedia to select the positive ones from all phrase candidates, and a phrase list would be generated. Then, I use this generated phrase list and segment the corpora accordingly. The whole procedure is as illustrated in Figure 2.2.

Figure 2.4: AutoPhrase framework, figure cited from [11]



2.4 ADVERSARIAL ATTACK

I use adversarial attack to analyze the important words in the corpus that dominate the predicted stock trend. Initially I have considered using the attention weights from the HAN model to interpret the significance of words in news articles, but the result does not make sense, so I want to further explore this issue by introducing adversarial analysis.

2.4.1 Adversarial Attack on Computer Vision Tasks

Adversarial attack is a popular topic during recent years which originated from the research in computer vision with deep neural networks. Adversarial examples in CV are images delicately constructed images used to fool computer vision systems. They could be gotten by adding perturbations to original queried images, and could hardly be distinguished from the original/benign ones.

Since adversarial examples was first proposed in 2014 by [12], it has been a hot topic because the fatal threat it brings to the widely used deep networks. The procedure of constructing adversarial examples for given outputs are called adversarial attacks, and the attempt to get results consistent to benign counterparts from adversarial examples are referred to as adversarial defenses. To give an intuitive introduction, the image shown below are the original images and adversarial images that fools the model to give a wrong prediction.

Figure 2.5: original images, perturbations, and adversarial images,figure cited from [12]

(a) (b)

2.4.2 Adversarial Attack on NLP Tasks

Recent years, many impactful works have appeared and showed that RNN based models are vulnerable to adversarial attacks as well. Since my model is a sequential model. I mainly

consider works attacking sequential model for reference. There are several works attacking RNN architecture models. Here I only introduce the most related ones.

[13] mainly discussed methods of crafting adversarial examples for sequence-to-sequence models. Group lasso and gradient regularization are combined to project the gradient so as to get a valid result in a discrete set from a continuous space. Non-overlapping attack and targeted keyword attack are used as criterion. This work attacks sequence to sequence model, and the candidate adversarial token is specified to guarantee that the derived sequences (i.e., sentences) make sense. An important and one of the earliest work is [14] discussing fooling NLP models but it is about fooling a reading comprehension model, adding some sentences in the question part to let the model derive a wrong answer.

The most relevant work that I refer to is [15], which aims in attacking sequence classification models, and also need to deal with discrete output space issue. In their work, attacking algorithms for character level LSTM (Long Short-Term Memory) and CNN architecture models. Obviously, since CNN is not sequential model, it can not capture sequential information, and is more straight forward to attack: the only difference with CV tasks is the difference between pixel matrices v.s. stacked embedding matrices. The character level LSTM they used is of higher degree of freedom, since the sequences are processed character by character, so the hidden state changes more frequently, and other constraints must be applied to guarantee that the concatenation of generated characters forms meaningful words. In comparison, my model is word level RNN based model and the adversarial sentences are generated by replacing original words with adversarial ones that are close to them in embedding space, and since I only search from the given vocabulary, I can guarantee that the generated tokens are meaningful.

CHAPTER 3: DATA

There are several stock indexes, and since I mainly consider the high-tech companies, I adopt Nasdaq index.

3.1 STOCK PRICE USED AS SUPERVISING INFORMATION

Here I first want to give a brief introduction about some fundamental stock market knowledge as well as some basic assumptions I have adopted. Every company corresponds to a ticker, and every day I can find several prices associated with this ticker: opening price, closing price, adjusted closing price etc. Opening and closing price are the price of a stock when the trading time in a trading day starts and ends respectively, and are adopted in my experiments.

In terms of ground truth or labels, it's kind of complex due to the complexity of stock market settings. Every trading day any stock would have an open price, close price, adjusted close price, and highest/lowest price of the day. I compared some settings in [**?**] etc. and adopt the following rule: I use the relative difference of stock price between 2 trading days. If the news happened on a trading day, I use the difference between close price and open price on that day as the price difference. Otherwise, I use the difference between the open price of the closest trading day after the news happening day and the close price of the closest trading day before the news happening day. Also, I considered the general market as a factor – a stock might go up even if there is negative news but the general market trend is going up. Therefore I also introduced the concept of relative stock price movement. Based on that absolute price change, I compute the labels: I mark the label as 1 if the price change is positive, otherwise negative:

$$retR_{tk}(d) = retA_{tk}(d) - retA_{ndx}(d) \tag{3.1}$$

$$retA_{tk}(d) = \begin{cases} \log(CP_{tk}(d)) - \log(OP_{tk}(d)), \forall \quad d \in Tr \\ \log(CP_{tk}(Tr_{last}(d))) - \log(OP_{tk}(Tr_{next}(d))), \forall \quad d \notin Tr \end{cases} \tag{3.2}$$

$$label_{tk}(d) = \begin{cases} 0, \forall d \quad s.t. \quad retR_{tk}(d) > 0 \\ 1, \forall d \quad s.t. \quad retR_{tk}(d) \leq 0 \end{cases} \tag{3.3}$$

Where $retR_{tk}(d)$ is the relative return of ticker tk at date d, $retA_{tk}(d)$ is the absolute return of ticker tk at date d, where $retA_{tk}(d)$ is a special case representing the absolute

return of ndx 100 which is used as a representative for the general industry.$label_{tk}(d)$ is the stock movement label for ticker tk on date d, and OP/CP stands for opening/closing price respectively. Tr denotes the set of all trading dates, and $Tr_{last}(d))/Tr_{next}(d)$ is the last/next trading date before date d.

My consideration behind using relative return as criterion for price change is that even if some good news happened, the company stock might still go down if the general economic or stock market situation is pessimistic, and vice versa.

3.2 DATA ACQUISITION

Currently there is no publicly available and widely acknowledged dataset for stock analysis as dataset like ImageNet and CIFAR for computer vision. I have considered several sources of corpus. I decided to respect the robots.txt files for the websites I considered, and I found that websites like seekingalpha.com, nasdaq.com etc. would ban even pretty friendly web crawlers. Also, since I want to crawl news data within 1 year period of time, I hope it could provide querying by dates. I only consider 10 companies under Technology sector (a sector is a domain of industry) so I would want ticker based query support. I finally combined intrinio which keeps record of urls of news articles and supports ticker and time based query, as long as the source site that intrinio keep records from – yahoo finance. Yahoo Finance and Google Finance stopped their historical data service in 2017 and 2012 respectively, otherwise they would both be good source of news articles.

The intrinio dataset includes news for 9 companies within 1 year (2017 Oct to 2018 Oct): Apple, AMD, Amazon, Baidu, Facebook(not used in later experiments since I remove FB in my Reuters data), Google, Intel, Microsoft, Nvidia, Qualcomm – all high trade stocks of frequently reported technology companies. I crawled the data ticker by ticker.

The Reuters dataset I obtained from IBM later spans from 2006 Oct to 2013 Nov. Since FB IPO time is in 2008, I filter it out and keep the other 9 companies as my training set. Reuters is relatively sparse since it only includes the relatively important news (the top stories).

CHAPTER 4: METHOD

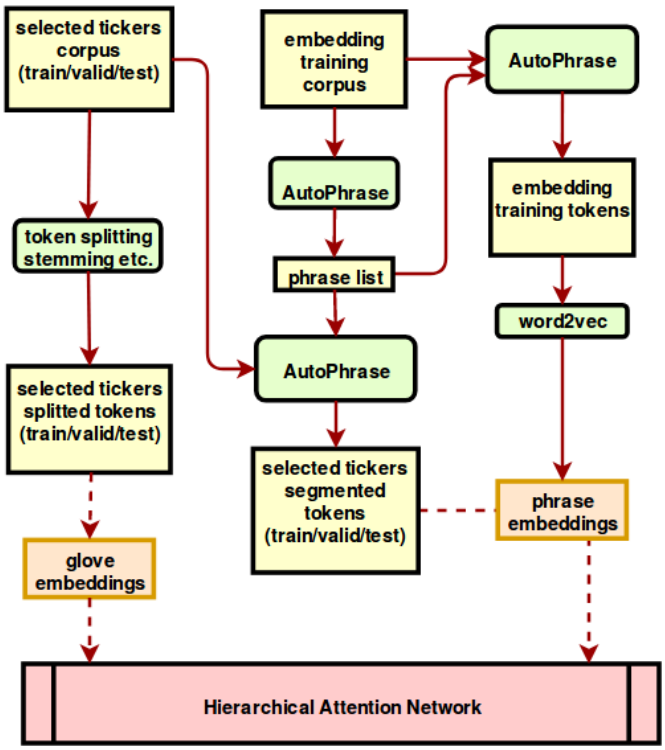
My pipeline consists several modules:

1. pretraining: the embeddings are trained for later usage; I download pretrained glove embedding from Stanford NLP group website, and use AutoPhrase to parse multi-words phrases from my financial news dataset and run word2vec to train embedding from the phrase-segmented corpora.

2. training and evaluation: Every news article is tokenized, parsed into hierarchically structured sequences, mapped to word embeddings specified and fed into HAN model to predict the stock price movement.

3. result analysis: attention weight and adversarial attack based analysis are conducted to figure out words and phrases in a document that are important for the model to give its prediction.

Figure 4.1: data flow of the NLP driven stock market prediction. If we use pretrained word embeddings, then the corpus is tokenized directly and embedded, otherwise, we filter out qualified phrases and store them into a phrase list, based on which the corpora are tokenized. More detailed description is in the section below.



The data flow for my pipeline is as in Figure 4.1, and in the remaining part of this chapter I describe each module in detail.

4.1 AUTOPHRASE BASED AND DISTANT SUPERVISED EMBEDDING TRAINING

4.1.1 Embedding Pretraining via AutoPhrase and word2vec

As mentioned in related works chapter, before feeding the tokens into neural networks, I need to first convert them into embeddings. Limited by the enormous size of the intrinio dataset, I could not segment the intrinio dataset. Therefore, I try this technique only on the Reuters dataset.

I first specify a phrase-filtering corpora, which should be as big as possible, in theory. Then I use AutoPhrase to mine high quality phrases (including single word and multi-words phrases, where each single word phrase is technically just a word) from this phrase-filtering corpora and store these phrases in a database **PB**.

It is worth mentioning here that the phrase-filtering corpora for embedding pretraining is different from training set for stock trend prediction, which is used to train the stock movement predicting model from preprocessed tokens and underlying embeddings. Obviously, to extract as many phrases of good quality as I can, the phrase-filtering corpora could include not only the train set of my selected 9 companies, but also the financial news that is not associated with any selected companies.

Then, I segment the all the news in this phrase-filtering corpora, and append every multi-words phrase together, taking it as a unique token. Then I apply word2vec to obtain the embedding for each word or multi-words phrase.

During training/testing phase, each news article in the corresponding train set/validation set/test set of 9 selected companies would be segmented according to the phrases database **PB**. Then I map these words to their corresponding pretrained embeddings.

4.1.2 Distant Supervised Embedding Training

To improve the quality of embeddings, I also try to include part of the intrinio dataset. My intuition is that the articles are all financial news, so they share some frequently used financial terms. Since this kind of corpus does not appear in the Reuters dataset, I consider it as a kind of distantly supervised embedding training. I call the embedding trained with only Reuters data as the Reuters AutoPhrase embedding, and I call the embedding trained with data merged from intrinio and Reuters as distant AutoPhrase embedding.

4.2 PREDICTION MODEL

4.2.1 Documents Preprocessing

I use typical preprocessing techniques such as stemming and tokenization with [16] to preprocess the news articles. When encountering a word that is not in the vocabulary, the word would be marked as 'unknown' and might be ignored, which depends on the experiment setting. After the preprocessing procedure, each document in my dataset would be in the format of a 2-level list:every item in the first level list is a second level list representing a sentence, and every item in the second level list represent a word. To be more specific, a document (news article) with L sentences is represented as $D = [S_1, S_2, ...S_L], S_i = [w_{i1}, w_{i2}, ...w_{iT_i}]$, where T_i is the number of tokens(words or phrases) in the i-th sentence.

4.2.2 GRU Based Sequence Encoder

Both the word encoder and the sentence encoder are based on GRU(Gated Recurrent Units), which uses a gating mechanism to track the state of sequences without using separate memory cells. There are two types of gates: reset gate and update gate, denoted as r_t and z_t respectively. At time t, the new hidden state is updated following the following formula:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h_t} \tag{4.1}$$

where h_{t-1} is the previous state, $\widetilde{h_t}$ is the new current state at time step t, and z_t is the gate that balance the proportion of these two states and combine them to get the new state, and is updated as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{4.2}$$

where x_t is the input vector at time step t. New current state $\widetilde{h_t}$ is updated in following way:

$$\widetilde{h_t} = tanh(W_h x_t + r_t \odot (U_t h_{t-1}) + b_h) \tag{4.3}$$

W, U, b in above formulae are parameters to be learned. Reset gate r_t is updated as:

$$z_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{4.4}$$

4.2.3 Hierarchical Attention Mechanism

The weighted sum of hidden vectors would be treated as upper level representation via hierarchical attention mechanism. First, word or phrase embedding would be encoded to sentence embedding through word encoder and word attention layer. Denoting the word embedding matrix as W_e, the word embedding sequence of a given sentence with tokens $w_{it}, t \in [1, T]$ is extracted as

$$x_{it} = W_e w_{it}, t \in [1, T] \tag{4.5}$$

Then, bidirectional GRU is used to get contextualized annotations of words by summarizing information from both directions along the token sequences. The bidirectional GRU contains the forward GRU $\overrightarrow{f}$ which reads the sentence s_i from w_{i1} to w_{iT} and a backward one $\overleftarrow{f}$ which reads the sentence s_i from w_{iT} to w_{i1}:

$$\overrightarrow{h_{it}} = \overrightarrow{GRU}(x_i t), t \in [1, T] \tag{4.6}$$

$$\overleftarrow{h_{it}} = \overleftarrow{GRU}(x_i t), t \in [1, T] \tag{4.7}$$

The concatenation of $\overrightarrow{h_{it}}$ and $\overleftarrow{h_{it}}$ are used as the annotation of word w_{it}:

$$h_{it} = [\overrightarrow{h_{it}}, \overleftarrow{h_{it}}], t \in [1, T] \tag{4.8}$$

Upon getting the annotation of words, attention mechanism is used to extract words and phrases that are important to the meaning of the sentence:

$$u_{it} = tanh(W_w h_{it} + b_w), t \in [1, T] \tag{4.9}$$

$$\alpha_{it} = \frac{exp(u_{it}^\intercal u_w)}{\sum_\tau \in [1, T] exp(u_{i\tau}^\intercal u_w)}, t \in [1, T] \tag{4.10}$$

$$s_i = \sum_t \alpha_{it} h_{it}, t \in [1, T] \tag{4.11}$$

where h_{it} is the annotation of word w_{it}, and u_{it} is its hidden representation. u_w is a word level context vector to which I match the similarity of u_{it} and get α_{it} the normalized importance weight of h_{it}.

The context vector u_w is a symbolic representation of "the important words of the sentence", which is randomly initialized and learned in a latent manner during the training process.

Sentence level attention layer and sequence encoder is of the same structure to their counterparts for word level. Bidirectional GRU are used to encode the sentences to get their

annotation:

$$\overrightarrow{h_i} = \overrightarrow{GRU}(s_i), i \in [1, L] \tag{4.12}$$

$$\overleftarrow{h_i} = \overleftarrow{GRU}(s_i), i \in [1, L] \tag{4.13}$$

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}], i \in [1, L] \tag{4.14}$$

where $\overrightarrow{h_i}$, $\overleftarrow{h_i}$ and h_i are forward part, backward part and the whole annotation of sentence s_i respectively.

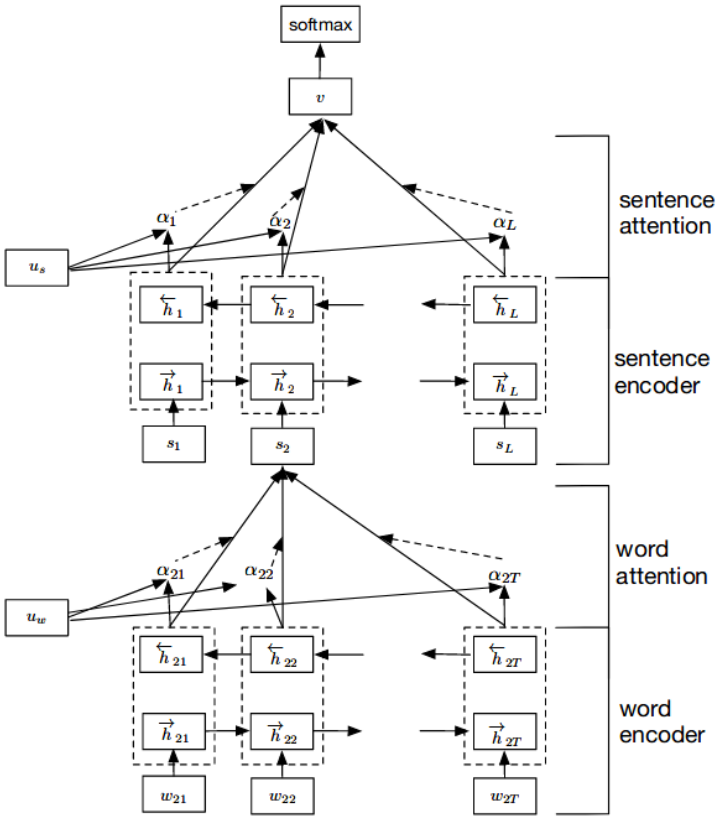
Sentence level context vector u_s is then used to measure the importance of sentence annotations and compute sentence level importance weights α_i:

$$u_i = tanh(W_s h_i + b_s), i \in [1, L] \tag{4.15}$$

$$\alpha_i = \frac{exp(u_i^\intercal u_s)}{\sum_j \in [1, L] exp(u_{ij}^\intercal u_w)}, i \in [1, L] \tag{4.16}$$

$$v = \sum_i \alpha_i h_i, i \in [1, L] \tag{4.17}$$

Figure 4.2: HAN architecture, figure cited from [6]



Finally, v is used as the representation or embedding of the whole news article, and fed into a linear layer W_p to get the prediction p of the news article:

$$p = W_p v \tag{4.18}$$

I use cross entropy loss to optimize my model:

$$loss(\mathcal{D}) = \sum_{d \in \mathcal{D}} -[y_d log(p_d) + (1 - y_d) log(1 - p_d)] \tag{4.19}$$

where $\mathcal{D}$ is the whole train set, y_d is the ground truth label of the d-th article, and p_d is the predicted result of it.

4.3 IMPORTANT WORDS ANALYSIS

Since I are interested in what words or phrases are important for the model to make prediction on financial news articles, I conduct important word analysis.

Since the attention weights of HAN model show the importance of corresponding sentences and words, I use that as a baseline method. However, the experiment result shows that highly weighted words are almost all punctuation or stopping words, I introduce adversarial attack to further figure out this issue.

I conduct text adversarial attack, following similar philosophy as in [**?**]. I develop an NLP attack algorithm based on FGSM(Fast Gradient Sign Method)[17], a widely cited and efficient adversarial attack algorithm used in CV tasks.

Given a sequence of tokens, I map them into embeddings and then feed them into the HAN model and compute the gradient, then add this as adversarial perturbation to the input embeddings, and search for the nearest neighbor. Different from [**?**], which attacked character level sequential model and word level convolutional model, my adversarial attack is applied to word level convolutional model. I use KDTree implemented in standard scipy package [18], to search for the closest token after adversarial attack.

4.3.1 FGSM and BIM Attacks on Texts

For a given documentation d, I extract its word embedding matrix $\mathbf{X_d}$ by concatenating individual word embeddings $[x_{it_i}], i \in [1, L], t_i \in [1, L_i]$ which is obtained as described in formula (4.5). Let $\mathbf{\Theta}$ be the parameters of a model, the loss of predicting $\mathbf{X_d}$ as the ground truth label y_d with my model is denoted as $\mathcal{L}(\mathbf{\Theta}; \mathbf{X_d}; y)$ Then I compute the gradient of these

word embeddings with respect to this loss, take the sign of the computed gradient vector and scale with coefficient ϵ to get the perturbation for d:

$$\eta_{\mathbf{d}} = \epsilon sign(\nabla_{\mathbf{X_d}} \mathcal{L}(\mathbf{\Theta}; \mathbf{X_d}; y_d)) \tag{4.20}$$

ϵ is a hyper-parameter that balance the attack strength and difference between the adversarial example and original input. To increase the loss, I add this perturbation to the original input matrix to get the adversarial example:

$$\mathbf{X_d^{adv}} = \mathbf{X_d} + \eta_{\mathbf{d}} \tag{4.21}$$

FGSM is an efficient attack, since the gradient vector is computed once and there is no loop. If I want stronger attack, I can further modify the algorithm such that it becomes BIM(Basic Iterative Method):

$$\begin{aligned} \mathbf{X_{d_0}^{adv}} &= \mathbf{X_d} \\ \mathbf{X_{d_t}^{adv}} &= \mathbf{X_{d_{t-1}}^{adv}} + \epsilon sign(\nabla_{\mathbf{X_d}} \mathcal{L}(\mathbf{\Theta}; \mathbf{X_{d_{t-1}}^{adv}}; y_d)) \end{aligned} \tag{4.22}$$

This is, every time I move toward the direction that can maximize the loss within limited magnitude of perturbation. After certain amounts of steps, the result would certainly be altered. This is actually a pretty strong attack, but could be time consuming, so I take it as an alternative plan in case FGSM failed to attack the documents. During the experiment, FGSM is actually strong enough to attack the documents.

4.3.2 The Whole Attack Pipeline

It is worthwhile to discuss the document set that I apply adversarial attack as well. Since adversarial examples are meaningful only if they can alter the prediction made by model, I need to first filter out the correctly predicted articles.

After I conduct adversarial attack as described in last subsection, I need one more step to get the adversarial text – I need to map the adversarial embedding back to discrete tokens in the vocabulary. One of the most important difference between CV and NLP tasks, as mentioned in previous chapters, is that the input for NLP is in discrete space, so it is not enough to simply keep the adversarial example above as the final result. I do this via KDTree, a widely used algorithm for search in multidimensional space. After mapping embeddings into tokens, when evaluated, the word embedding matrix of adversarial text is not guaranteed to be the same as the raw adversarial matrix, i.e., $\mathbf{X_{d_{adv}}}$ is not necessarily

equal to $\mathbf{X_d^{adv}}$. If the prediction result on $\mathbf{X_{d_{adv}}}$ is identical to that of $\mathbf{X_d}$, I consider the attack as unsuccessful.

The whole procedure is as follows:

Algorithm 4.1: Adversarial Attack on Texts

Input : Document Dataset $\mathcal{D}$, Model Parameters $M_{\mathbf{\Theta}}$
Output: Adversarial Document Set $\mathcal{D}_{adv}$ for Dataset $\mathcal{D}$
$\mathcal{D}_{correct} = \{(\mathbf{X_d}, y_d) | d \in \mathcal{D} s.t. M_{\mathbf{\Theta}}(\mathbf{X_d}) = y_d\}$;
$\mathcal{D}_{adv} = \{\}$;
for $d \in \mathcal{D}_{correct}$ **do**
  $\eta_{\mathbf{d}} = \epsilon sign(\nabla_{\mathbf{X_d}} \mathcal{L}(\mathbf{\Theta}; \mathbf{X_d}; y_d))$;
  $\mathbf{X_d^{adv}} = \mathbf{X_d} + \eta_{\mathbf{d}}$;
  $d_{adv} = KDTreeMap(\mathbf{X_d^{adv}})$;
  if $\mathbf{X_{d_{adv}}} == \mathbf{X_d^{adv}}$ **then**
    $\mathcal{D}_{adv} += d_{adv}$,
  end
end

CHAPTER 5: EXPERIMENT RESULTS AND ANALYSIS

5.1 EXPERIMENT SETUP

5.1.1 Dataset Split

For intrinio dataset, since I can almost guarantee that everyday I have news for every ticker in my selected ticker list, I split the dataset by dates. Since in the early stage I found I cannot learn a model that converges on the dataset, I turn to Reuters dataset, and didn't continue to split the validation set. The news between 2017 Oct 01 and 2018 Jun 30 (in total 273 days) are split into train set and news between 2018 Jul 01 and 2018 Sep 30 are split into test set. The counts of news of intrinio data by company tickers is as follows:

Table 5.1: data split for intrinio dataset

Ticker	#. of news in train set	#. of news in test set
BIDU	218	104
AAPL	2809	1651
MSFT	978	545
INTC	832	442
QCOM	276	199
AMD	243	198
AMZN	2785	1540
NVDA	693	347
GOOGL	1396	979
Total	13192	7208

For Reuters dataset, I conduct pioneer experiment and have observed that I are able to train a model whose performance on validation set converges, so I split news for each ticker companies into train/validation/test set. The news articles in Reuters dataset are pretty sparse, and I definitely cannot guarantee 1 article per day for any company. However, since I don't incorporate recent stock price into my model and I model my task as a document classification problem, the temporal order does not actually matter. So I fix the data split rate and split articles into train/validation/test set proportionally for each company in the ticker list.

Table 5.2: data split for Reuters dataset

Ticker	#. of news in train set	#. of news in validation set	#. of news in test set
BIDU	225	32	65
AAPL	3122	446	892
MSFT	1066	152	305
INTC	891	127	256
QCOM	332	47	96
AMD	308	44	89
AMZN	3027	432	866
NVDA	728	104	208
GOOGL	1662	237	476
Total	14276	2037	4087

5.1.2 Hyper Parameters for AutoPhrase Based Embedding Training

For intrinio dataset, limited by computational resource and the huge size of corpora, I are not able to apply AutoPhrase to segment the corpora and filter robust and meaningful phrases. For Reuters dataset, the embedding pretraining parameters are as follows, where the first two parameters are for AutoPhrase and remaining ones are for word2vec:

Table 5.3: parameter for embedding pretraining

multi-words phrase highlight threshold	0.5
single-words phrase highlight threshold	0.9
threshold for occurrence of words	10^{-3}
training model	skip-gram
length of skip window	5
size of output embedding	25

I follow this setting for training both the Reuters AutoPhrase embedding and the distant AutoPhrase embedding.

5.1.3 Hyper Parameters for Neural Network

The structure of my HAN model is as listed in the previous chapter. In the experiment, I set my network as follows:

Table 5.4: parameter setting of each layer of HAN model

dropout rate to feed word embedding into word encoder	0.5
input size of word encoder	25
output size of word encoder	50
input size of sentence encoder	100
output size of sentence encoder	50

Notice here I turn off dropout for the GRU cells in bidirectional encoders. I used PyTorch to build up the code framework. For all the experiment, I use Adam optimizer, and set step learning scheduler, where I turn off the learning rate halve interval and set $\gamma = 0.9$. The batch size is set as 4.

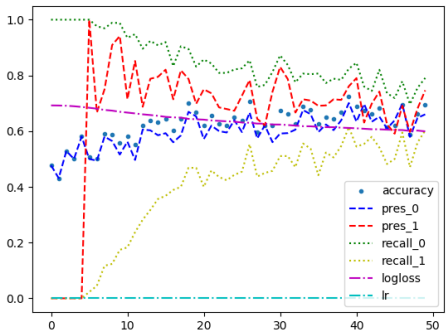
5.2 EXPERIMENTAL RESULTS

I show the result of prediction accuracy, attention weight based importance words mining, and adversarial analysis respectively.

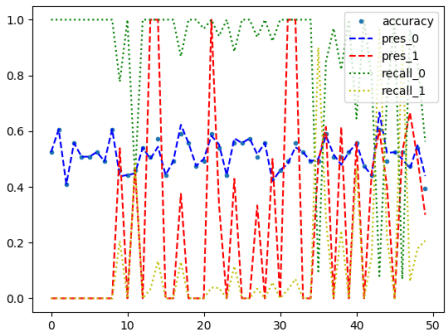
5.2.1 Prediction Accuracy

Below is the training set and testing set performance curve at each training epoch on intrinio and Reuters dataset(horizontal axis is the training epoch, vertical axis is the statistics number):

Figure 5.1: performance at each epoch on different dataset and embeddings

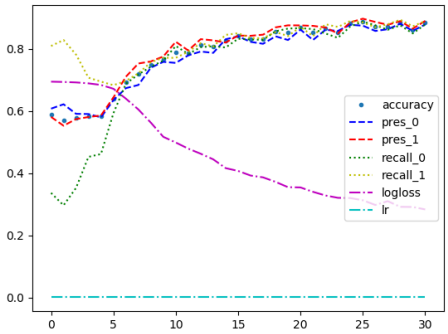


intrinio train set

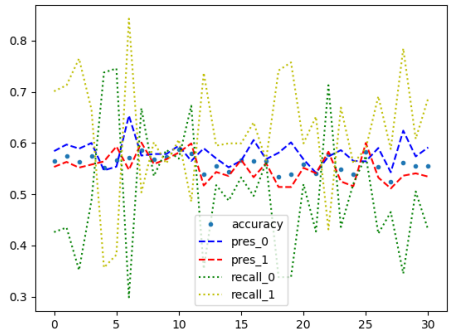


intrinio test set

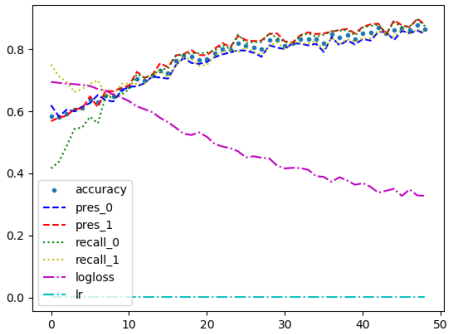
Figure 5.1 continued



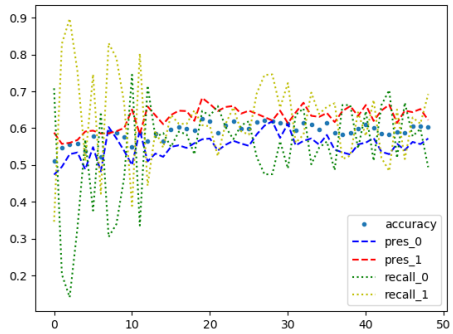
Reuters train set,
distant AutoPhrase embedding



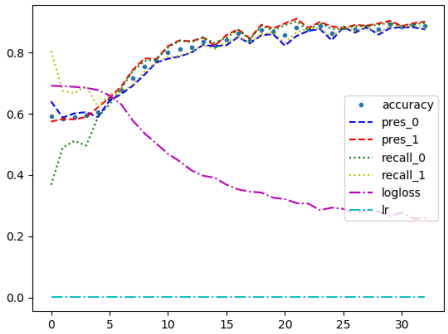
Reuters test set,
distant AutoPhrase embedding



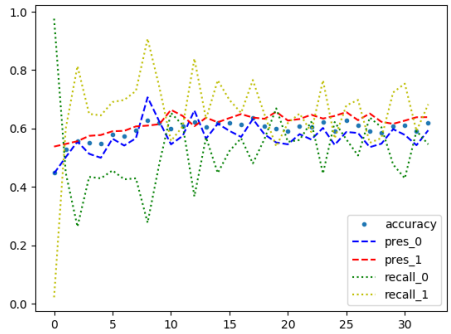
Reuters train set, glove embedding



Reuters test set, glove embedding



Reuters train set,
Reuters AutoPhrase embedding



Reuters test set,
Reuters AutoPhrase embedding

Table 5.5: highest accuracy among all epochs

dataset / underlying embedding	accuracy(%)
Intrinio	61.0
Reuters, with distantly AutoPhrase trained embedding	59.5
Reuters, with glove embedding	61.8
Reuters, with AutoPhrase trained embedding on Reuters corpora	62.2

5.2.2 Attention weight Based Importance Words Mining

For all test set documents, I rank top 100 words with highest attentions, and keep a dictionary of ranking index and the frequency of these words. After evaluating all documents, I rank words by their average ranking indexes. I make statistics on all documents, documents corresponding increasing/decreasing stock prices, and documents correctly/wrongly predicted by my model. For each case, I list the top 20 words with highest attentions.

Table 5.6: words with highest attention

articles labeled as up	',', '.', 'said', 'The', '(', ')', 'percent', 'U.S', '-', 'would', 'Reuters', 'also', 'billion', 'could', 'company', 'last', 'new', 'year', ';', '–', 'government', 'since', 'one', 'In', 'banks', 'million', 'first', 'economy', 'But', 'two', 'business', 'may', 'expected', 'A', 'bank', 'It', 'according', 'market', 'deal', 'still', 'debt', 'next', 'China', 'report', 'investors', 'rose', 'companies', 'including', 'much', 'fell', 'time', '_New_York_', 'shares', 'take', 'I', '_New_York_', 'make', 'sales', 'going', 'end', '_United_States_', 'per', 'week', 'high', 'long', 'like', 'stock', 'back', 'think', 'three', 'likely', 'made', 'He', 'data', 'many', 'see', 'Inc', 'second', ':', 'people', 'interest', 'oil', 'cents', 'top', 'get', 'around', 'even', 'industry', 'cut', 'prices', 'growth', 'well', 'quarter', 'help', 'capital', 'due', 'need', 'part', 'analysts', 'major'

Table 5.6 continued

articles labeled as down	',', '.', 'said', 'The', '(', ')', 'percent', 'U.S', '-', 'would', 'said.', 'Reuters', 'also', 'billion', 'could', 'company', 'last', 'new', 'year', ';', '–', 'government', 'since', 'one', 'In', 'banks', 'million', 'first', 'economy', 'But', 'two', 'business', 'may', 'expected', 'A', 'bank', 'It', 'according', 'market', 'deal', 'still', 'debt', 'next', 'China', 'report', 'investors', 'rose', 'companies', 'including', 'much', 'fell', 'time', '_New_York_', 'shares', 'take', 'I', '_New_York_', 'make', 'sales', 'going', 'end', '_United_States_', 'per', 'week', 'high', 'long', 'like', 'stock', 'back', 'think', 'three', 'likely', 'made', 'He', 'data', 'many', 'see', 'Inc', 'second', ':', 'people', 'interest', 'oil', 'cents', 'top', 'get', 'around', 'even', 'industry', 'cut', 'prices', 'growth', 'well', 'quarter', 'help', 'capital', 'due', 'need', 'part', 'analysts'
articles correctly predicted	',', '.', 'said', 'The', '(', ')', 'percent', 'U.S', '-', 'would', 'said.', 'Reuters', 'also', 'billion', 'could', 'company', 'last', 'new', 'year', ';', '–', 'government', 'since', 'one', 'In', 'banks', 'million', 'first', 'economy', 'But', 'two', 'business', 'may', 'expected', 'A', 'bank', 'It', 'according', 'market', 'deal', 'still', 'debt', 'next', 'China', 'report', 'investors', 'rose', 'companies', 'including', 'much', 'fell', 'time', '_New_York_', 'shares', 'take', 'I', '_New_York_', 'make', 'sales', 'going', 'end', '_United_States_', 'per', 'week', 'high', 'long', 'like', 'stock', 'back', 'think', 'three', 'likely', 'made', 'He', 'data', 'many', 'see', 'Inc', 'second', ':', 'people', 'interest', 'oil', 'cents', 'top', 'get', 'around', 'even', 'industry', 'cut', 'prices', 'growth', 'well', 'quarter', 'help', 'capital', 'due', 'need', 'part', 'analysts'
articles mispredicted	',', '.', 'said', 'The', '(', ')', 'percent', 'U.S', '-', 'would', 'Reuters', 'also', 'billion', 'could', 'company', 'last', 'new', 'year', ';', '–', 'government', 'since', 'one', 'In', 'banks', 'million', 'first', 'economy', 'But', 'two', 'business', 'may', 'expected', 'A', 'bank', 'It', 'according', 'market', 'deal', 'still', 'debt', 'next', 'China', 'report', 'investors', 'rose', 'companies', 'including', 'much', 'fell', 'time', '_New_York_', 'shares', 'take', 'I', '_New_York_', 'make', 'sales', 'going', 'end', '_United_States_', 'per', 'week', 'high', 'long', 'like', 'stock', 'back', 'think', 'three', 'likely', 'made', 'He', 'data', 'many', 'see', 'Inc', 'second', ':', 'people', 'interest', 'oil', 'cents', 'top', 'get', 'around', 'even', 'industry', 'cut', 'prices', 'growth', 'well', 'quarter', 'help', 'capital', 'due', 'need', 'part', 'analysts', 'major'

Table 5.6 continued

all articles	',', '.', 'said', 'The', '(', ')', 'percent', 'U.S', '-', 'would', 'said.', 'Reuters', 'also', 'billion', 'could', 'company', 'last', 'new', 'year', ';', '–', 'government', 'since', 'one', 'In', 'banks', 'million', 'first', 'economy', 'But', 'two', 'business', 'may', 'expected', 'A', 'bank', 'It', 'according', 'market', 'deal', 'still', 'debt', 'next', 'China', 'report', 'investors', 'rose', 'companies', 'including', 'much', 'fell', 'time', '_New_York_', 'shares', 'take', 'I', '_New_York_', 'make', 'sales', 'going', 'end', '_United_States_', 'per', 'week', 'high', 'long', 'like', 'stock', 'back', 'think', 'three', 'likely', 'made', 'He', 'data', 'many', 'see', 'Inc', 'second', ':', 'people', 'interest', 'oil', 'cents', 'top', 'get', 'around', 'even', 'industry', 'cut', 'prices', 'growth', 'well', 'quarter', 'help', 'capital', 'due', 'need', 'part', 'analysts'

5.2.3 Important Word Derived via Adversarial Analysis

I analyze adversarial examples from both macro and case by case perspective. For the latter one, I attach the result in the next subsection a case study manner. Here I list the most vulnerable words – words that are most frequently attacked, hottest target words – words that are most easily found to replace the original words, as well as most frequent original/adversarial word pairs. I list the top 100 for each case, in format of "token:frequency".

Table 5.7: statistics of adversarial examples generated on Reuters dataset using AutoPhrase embedding

most vulnerable words	,:179, said:29, .:25, billion:22, new:18, company:17, also:15, year:15, business:13, think:10, would:10, Apple:9, deal:8, The:8, market:7, Tuesday:7, million:7, could:6, fell:6, turn:6, software:6, especially:6, technology:6, large:6, percent:5, good:5, But:5, taking:5, smartphones:5, still:5, continue:5, see:5, industry:5, –:5, added:5, 3:4, ;:4, Simon:4, economy:4, made:4, device:4, given:4, iPhone:4, A:4, _United_States_:4, consumer:4, Microsoft:4, public:4, sales:4, way:4

Table 5.7 continued

hottest target words	stares:104, and:28, _Editing_by_David_Christian-Edwards_:28, zig-zagged:16, Trippitt:14, ago:14, $18.94:13, $9:11, new,:11, guess:9, exorbitant:9, _profit_slides_:9, jumped:8, _AIU_Holdings_:8, _ads_alongside_:8, _Nikko's_shares_:7, statement.:7, cos:7, Initially:7, might:6, _tablet_PC_:6, _Chief_Executive_Henri_Termeer_:6, _Google's_Nexus_:6, _moves_ahead_:6, away:6, hold,:6, bosses':6, fare,:6, $16:6, 4:5, Tuesday:5, _changing_tack_:5, _vantage_point_:5, moneylending:5, Wednesday:5, $17-per-share:5, _premium_pricing_:5, _home_grown_:5, ($1=A$1.30):5, setting:5, 3:5, forever,:5, parts.":5, _grade_corporate_debt_:5, $270:5, mind:4, 10.3:4, much":4, _Edwin_Chan_and_Andre_Grenon_:4, built-in:4
most frequent paris	(,,stares):101, (,,and):28, (,,_Editing_by_David_Christian-Edwards_):28, (.,zig-zagged):16, (said,Trippitt):14, (year,ago):14, (billion,$9):11, (new,new,):11, (think,guess):9, (,,_profit_slides_):9, (said,statement.):7, (business,cos):7, (fell,jumped):6, (company,_Chief_Executive_Henri_Termeer_):6, (deal,_moves_ahead_):6, (turn,away):6, (also,bosses'):6, (billion,$16):6, (Apple,_Google's_Nexus_):5, (Tuesday,Wednesday):5, (business,_premium_pricing_):5, (,,($1=A$1.30)):5, (software,_ads_alongside_):5, (see,forever,):5, (–,_grade_corporate_debt_):5, (million,$270):5, (percent,10.3):4, (company,_AIU_Holdings_):4, (;,_Edwin_Chan_and_Andre_Grenon_):4, (Simon,Jayson):4, (economy,_recovery_prospects_):4, (made,Shenouda's):4, (device,_Chrome_OS_):4, (.,..):4, (iPhone,_tablet_PC_):4, (would,hold,):4, (market,_home_grown_):4, (consumer,consumers'):4, (continue,adjust):4, (industry,parts."):4, (The,Initially):4, (also,_changing_tack_):4, (technology,-volume,):4, (could,might):3, (3,4):3, (said,much"):3, (really,I'm):3, (use,exorbitant):3, (feature,built-in):3, (change,rest"):3

Also, it is worth mentioning that the vulnerable part of corpus varies with underlying embedding. On the AutoPhrase based embedding I have in total 86 successful attacks, among which 85 are attacks on non-number tokens. On a pretrained glove embedding it's easy to alter the prediction on a news article by attacking numbers – I have only 22 out of 112 attacks that converts the prediction only by attacking non-number tokens to non-number tokens.

5.2.4 CASE STUDY: MINING IMPORTANT WORDS VIA ADVERSARIAL ANALYSIS

We conduct adversarial attacks to further and have some interesting result to show. Adversarial attack is a hot topic in current deep learning research, which is helpful in explaining the black box of neural networks. By deliberately constructing some adversarial examples, the attackers can usually fool the pretrained model easily.

Initially we want to figure out the important word by checking the attention weights of each word, since HAN provided both sentence and word level attention. However, we find that the attention of words lie in a latent space and is not bounded. Therefore we conducted ranking and selected the most highly ranked words. It turns out that the words paid high attention to are usually stopping words or some special tokens, which is not what we expected – meaningful words with strong sentiment etc.

Therefore, we turn to adversarial attacks. Since adversarial attacks aims to destroy a model, we only list the adversarial examples that can alter a correct prediction to be a wrong one. Below are some case study examples:

Table 5.8: a phrase with opposite meaning is used to convert the sentiment of the whole sentence.(ground truth: down)

original texts: ... the company said expects moderate year-over-year growth _fourth_quarter_ due _unk_ temporary **_negative_impact_** _unk_ anticipated discontinuing _online_marketing_ classic edition . _unk_ said would complete switch phoenix nest advertising bidding system _unk_ 1 ...	adversarial texts: ... the company said expects moderate year-over-year pace _fourth_quarter_ due $\langle/s\rangle$ _loan_repayments_ **_positive_impact_** $\langle/s\rangle$ anticipated employee-level _online_marketing_ classic edition . $\langle/s\rangle$ said would complete "Site _Krugman_wrote_ scornful _Web_properties_ bids system $\langle/s\rangle$ 1 ...

Table 5.9: tiny perturbations can beat the trained system, which implies that the model is vulnerable to adversarial example.(ground truth: up)

original texts: ... investors beginning wonder _unk_ rally keep going without **taking** break fall...	adversarial texts: ... investors beginning wonder _pad_ rally keep going without **once** break fall ...

Table 5.10: A pair of examples, both used the phrase "Global economic outlook" to mislead the model, no matter what is the ground truth, which might imply that this phrase is capable of altering the sequence label.(ground truth: up)

original texts: ...**_stocks_climbed_ 2 percent** ... **_unk_** . _unk_ _unk_ points , 2.37 percent , finish _unk_ . the _unk_ & _unk_ _unk_ _unk_ points , 2.63 percent , _unk_ . _unk_ _unk_ _unk_ points , 3.45 percent , _unk_ . _consumer_spending_ accounts roughly two-thirds _unk_ . economy . the **_unk_** . _confidence_data_ followed similarly rosy _consumer_reports_ _unk_ _unk_	adversarial texts: ...**_stocks_climbed_ 2 percent** ... **_Global_economic_outlook_** . _unk_ _unk_ points , 2.37 percent , finish _unk_ . the _unk_ & _unk_ _unk_ _unk_ points , 2.63 percent , _unk_ . _unk_ _unk_ _unk_ points , 3.45 percent , _unk_ . _consumer_spending_ accounts roughly two-thirds _unk_ . economy . the **_Global_economic_outlook_** . _confidence_data_ followed similarly rosy _consumer_reports_ _unk_ _unk_ ...
original texts: ...**_stocks_fell** ... **_unk_** . _unk_ 2.4 percent . _unk_ it shows susceptible _bad_news_ right , _unk_ said _unk_ , _unk_ _unk_ _unk_ . _unk_ we _unk_ got extended _stock_market_ feather news enough cascade 100 points . _unk_ **_unk_** . _unk_ _unk_ points , 0.92 percent , end _unk_ ...	adversarial texts: ...**_stocks_fell** ... **_Global_economic_outlook_** . _unk_ 2.6 percent . _unk_ it shows susceptible _bad_news_ right , _unk_ said _unk_ , _unk_ _unk_ _unk_ . _unk_ we _unk_ got extended _stock_market_ feather news enough cascade 100 points . _unk_ **_Global_economic_outlook_** . _unk_ _unk_ points , 0.92 percent , end _unk_ ...

Considering that "_unk_" (the default token we use to substitute those that we cannot find embedding in the dictionary) embedding is all zero, we also conduct experiment where the unknown tokens are skipped and not appended to the sentence sequences.

Table 5.11: Another example that the model might emphasize words with explicit sentiment.(ground truth: down)

original texts: ... But **three** major indexes shed gains release Fed 's statement , left outlook cuts clouded . Short -term _rate_futures_ suggested chance future cut , chance end _Fed_meeting_ . Everything I heard **coming today could signals** pause . It 's seeming like language **, I think** 's people reacting , said _Eric_Kuby_ , _chief_investment_officer_ _North_Star_Investment_Management_Corp_ Chicago . _Stocks_rebounded_ month miserable March , Fed flooded financial system cash wake _Bear_Stearns_ ' collapse _amid_signs_ _frozen_credit_markets_ beginning thaw . _The_Dow_Jones_industrial_average_ finished 11.81 points , 0.09 percent , . The & **fell** 5.35 points , 0.38 percent , . _The_Nasdaq_Composite_Index_ points , 0.55 percent , . For month , Dow gained 4.5 percent , S & P rose 4.8 percent Nasdaq advanced 5.9 percent . Research Motion biggest weight Nasdaq 100 , falling 3.7 percent $. Shares BlackBerry maker notched 8.4 percent advance month . Apple shares , surged 21 percent April , fell 0.6 percent $ No . 3 _drag_on_ Nasdaq Wednesday	adversarial texts: ... But **six** major indexes shed gains release Fed 's statement , left outlook cuts clouded . Short -term _rate_futures_ suggested chance future cut , chance end _Fed_meeting_ . Everything I heard staying **mind might diffuse** pause . It 's seeming like language **stares I guess** 's people reacting stares said _Eric_Kuby_ , _chief_investment_officer_ _North_Star_Investment_Management_Corp_ Chicago . _Stocks_rebounded_ month miserable March , Fed flooded financial system cash wake _Bear_Stearns_ ' collapse _amid_signs_ _frozen_credit_markets_ beginning thaw . _The_Dow_Jones_industrial_average_ finished 11.81 points , 0.09 percent , . The & **jumped** 5.35 points , 0.38 percent , . _The_Nasdaq_Composite_Index_ points , 0.55 percent , . For month , Dow gained 4.5 percent , S & P rose 4.8 percent Nasdaq advanced 5.9 percent . Research Motion biggest weight Nasdaq 100 , falling 3.7 percent $. Shares BlackBerry maker notched 8.4 percent advance month . Apple shares , surged 21 percent April , fell 0.6 percent $ No . 4 _drag_on_ Nasdaq Wednesday . ...

Table 5.12: Since our model is ticker agnostic – this is partially due to the limited data we have – this attack on brand/ticker implies that the learned model perhaps assigns sentiment labels to the words, which might imply that a possible way to improve performance is to label tokens associated with a brand/ticker with "role" information, such as "this company itself", "product of this company", "rival" and "product of rival".(ground truth: up)

original texts: ... Spokespersons **Apple** AT & T _declined_to_comment_ . The higher number worrying Apple company receives cut AT & T 's iPhone _service_fees_ , revenue carries high _gross_margin_ _fueled_optimism_ earnings potential . For example , Sacconaghi said , Apple hit sales goal 10 _million_iPhones_ end fiscal 2008 30 percent result carrier payments , revenue profit would $ 500 million 37 cents per share lower expected . If Apple cracks unlocked phones could preserve high margins miss _sales_target_ , whereas allowing could erode profitability make tough sign carriers similar revenue -sharing deals . Besides **_financial_implications_** , believe prevalence unlocked iPhones presents **significant strategic dilemma Apple** , Sacconaghi wrote	adversarial texts: ... Spokespersons **_Google's_Nexus_** AT & T _declined_to_comment_ . The higher number worrying Apple company receives cut AT & T 's iPhone _service_fees_ , revenue carries high _gross_margin_ _fueled_optimism_ earnings potential . For example , Sacconaghi said , Apple hit sales goal 10 _million_iPhones_ end fiscal 2008 30 percent result carrier payments , revenue profit would $ 500 million 37 cents per share lower expected . If Apple cracks unlocked phones could preserve high margins miss _sales_target_ , whereas allowing could erode profitability make tough sign carriers similar revenue -sharing deals . Besides **_short_and_medium_term_** , believe prevalence unlocked iPhones presents **plague ScreenTonic _problems_facing_ Fire.** , Sacconaghi wrote . ...

Table 5.13: Here are two articles, both altered to predict "up" using adversarial texts that contain the word or phrase "two-fisted".(ground truths: down)

original texts: ... apple inc ( ) board member jerry york hospitalized wednesday near home detroit suffering collapse , person familiar matter said . details york 's condition immediately available . jerry york worked chief finance officer chrysler director general motors , best known adviser billionaire investor kirk kerkorian . he also serves president chief executive **harwinton** capital	adversarial texts: ... apple inc ( ) board member jerry york hospitalized wednesday near home detroit suffering collapse , person familiar matter said . details york 's condition immediately available . jerry york worked chief finance officer chrysler director general motors , best known adviser billionaire investor kirk kerkorian . he also serves president chief executive **two-fisted** capital . ...
original texts: ... microsoft corp ( ) chief financial officer chris liddell said thursday " cautious " economy , wide range products geographic diversity support company 's earnings . " we 're cautious like everybody else , " liddell said interview reuters . his comments came heels earnings estimates current quarter disappointed investors , sending shares 5 percent **after-hours** trade . liddell , declined comment company 's takeover bid yahoo inc ( ) , said windows revenue lower company 's expectations due part inventory **build-up** computers added issues " quarter specific . " (reporting daisuke wakabayashi ; editing **braden**) ...	adversarial texts: ... microsoft corp ( ) chief financial officer chris liddell said thursday " cautious " economy , wide range products geographic diversity support company 's earnings . " we 're cautious like everybody else , " liddell said interview reuters . his comments came heels earnings estimates current quarter disappointed investors , sending shares 5 percent **two-fisted** trade . liddell , declined comment company 's takeover bid yahoo inc ( ) , said windows revenue lower company 's expectations due part inventory **melee** computers added issues " quarter specific . " (reporting daisuke wakabayashi ; editing **looper**) ...

5.3 REMARKS FROM THE EXPERIMENTAL RESULTS

5.3.1 Importance Of Dataset

As shown in the result, the test set accuracy is not converging on the intrinio dataset, therefore my first conclusion is that quality of raw data is of key importance. My hypothesis

which may interpret this result is that the Reuters dataset is relatively sparse, and only includes relatively important news, reported by Reuters reporters, so I believe it is of higher importance. In comparison, the intrinio dataset is obtained and utilized mainly in the early stage of my experiment, and it often includes some trivial reports – it is common to find more than 10 reports for one company everyday, and the article style is relatively casual compared to Reuters news. For instance, some articles are more of advertisement instead of regular financial news report. Therefore later for "hot word" part I mainly use Reuters dataset to carry out experiments.

5.3.2 Impact Of Underlying Word Embedding Selection

The accuracy with AutoPhrase pretrained embedding is 62.2%, better than 61.8%, which is the result with glove embeddings. This is as I expected since phrases can be informative and should be treated as a union instead of broken sequences.

However, I also noticed that the classification accuracy with distant AutoPhrase embedding is lower than that using the pure Reuters AutoPhrase embedding. This is different from my expectation, and the interpretation might be that the distribution of words and contexts are different since the article styles are pretty different.

5.3.3 Important Words and Phrases via Attention weights

It is obvious that the different types of documents, no matter correctly predicted or mispredicted, no matter corresponding to increasing or decreasing stock price, share pretty much same set of high attention words. This might be interpreted from several perspective. The general important words are all frequent words or phrases, which suggest that my model didn't pay attention to unique terms such as product or company names.

In contrast, there are location (_New_York_) or countries (such as U.S, China), which frequently appear in financial news, indicating not are these places ubiquitous in articles, but also emphasized by my model. This further suggested that special entities are not well recognized, which provides us the potential way to optimize the model and strengthen its ability for entity recognition.

5.3.4 Adversarial Analysis on Words and Model

In comparison with the "important" words found using attention weights, the result of adversarial attack is much more interesting. From the very beginning, my goal is just to

mine some important words, but I did find some phenomenon more worth further exploring.

The vulnerable words are more "frequent" or "ordinary" words, while unique terms, especially multi-words phrases and numbers, could be found in the hottest words used to replace original ones. This actually indicates that the model are not so easy to attack, since which phrases would be selected as qualified phrases depends on what is used as the phrase embedding training corpus, and the attack is less likely to be transferable across different models. Numbers are frequently used to replace original symbols, since the model is not capable of mapping numbers to the correct position in the temporal sequence, so these numbers are not fully utilized, and my model is not able to extract useful information from embedding of numbers effectively, which indicates that the homogeneous sequential model which works well for tasks like movie or restaurant embedding might not be enough to tackle documents with relatively heterogeneous dimensions of information.

CHAPTER 6: SUMMARY AND CONCLUSION

This work explores the influence of various factors on the performance of utilizing NLP knowledge to predict stock trend of a company. This is a relatively complex problem since there is no publicly acknowledged dataset and some published dataset are no longer effective. Also, since the stock price in future is not simply decided by news (not even by the traders that read the news), the performance is not as high as other accurately labeled task, such as movie or restaurant rating, where the supervising information – labels – completely depends on corpus (user comments).

My experiment shows that dataset of good quality is a key factor for the performance. Still, we can improve the performance by utilizing some data mining techniques. Underlying embedding is another important issue: the training corpus used to pretrain embedding, some preprocessing techniques to parse valuable phrases etc.

Heterogeneous information is another issue worth considering. As analyzed in the part of adversarial attack case study, it is perhaps useful if we can treat heterogeneous information differently considering their type. One factor is the "role" of tokens – let us assume that there is a news article claiming product of a company A outperforms its rival B, then it is reasonable to predict that stock price of A rise and stock price of B fall, and this is probably the reason that the performance of Reuters dataset is a lot better than that of intrinio – for the former dataset, usually under category of one company we have only the news report associated with that specific company, not its rivals or commercial partner, which is not the guaranteed situation in the later one.

Also, numbers are frequent in those financial news articles. Definitely this is informative but with common models or under common NLP frameworks, if we treat these numbers simply as tokens, they are frequently tossed out since we don't have their embedding or mapped to multidimensional embeddings, which are not necessarily numerically equal to these numbers.

There are several potential directions worth further exploring in the future: for the pre-training phase, we might need to further clean the input tokens to gain embeddings with better quality, and mark the companies and its opponents to capture entity role information. Techniques like sphere embedding or dynamic embedding mechanism mentioned in related work chapter are also worth trying. For training phase, we can split recognized entities, words and add syntactical tagging information to help boost the performance. After the training, we can also conduct further analysis. For instance, we can combine adversarial analysis with the attention weights: filtering out the attacked documents, and compare the

words that gain most attention in the original documents with those in adversarial documentation. Also, we can limit the number of modified tokens and conduct fine granularity experiment to find what is the most vulnerable word in a given article, instead of find groups of adversarial words.

REFERENCES

[1] Y. Xu and S. B. Cohen, "Stock movement prediction from tweets and historical prices," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018. [Online]. Available: http://aclweb.org/anthology/P18-1183 pp. 1970–1979.

[2] H. Lee, M. Surdeanu, B. MacCartney, and D. Jurafsky, "On the importance of text analysis for stock price prediction," in *Language Resources and Evaluation Conference (LREC)*, 2014.

[3] "Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction," *CoRR*, vol. abs/1712.02136, 2017, withdrawn. [Online]. Available: http://arxiv.org/abs/1712.02136

[4] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun, "Very deep convolutional networks for natural language processing," *CoRR*, vol. abs/1606.01781, 2016. [Online]. Available: http://arxiv.org/abs/1606.01781

[5] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," *CoRR*, vol. abs/1702.01923, 2017. [Online]. Available: http://arxiv.org/abs/1702.01923

[6] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," 01 2016, pp. 1480–1489.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

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

[9] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *CoRR*, vol. abs/1802.05365, 2018. [Online]. Available: http://arxiv.org/abs/1802.05365

[10] 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. [Online]. Available: http://arxiv.org/abs/1810.04805

[11] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," *CoRR*, vol. abs/1702.04457, 2017. [Online]. Available: http://arxiv.org/abs/1702.04457

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014. [Online]. Available: http://arxiv.org/abs/1312.6199

[13] M. Cheng, J. Yi, H. Zhang, P. Chen, and C. Hsieh, "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples," *CoRR*, vol. abs/1803.01128, 2018. [Online]. Available: http://arxiv.org/abs/1803.01128

[14] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," *CoRR*, vol. abs/1707.07328, 2017. [Online]. Available: http://arxiv.org/abs/1707.07328

[15] "Adversarial texts with gradient methods," *CoRR*, vol. abs/1801.07175, 2018, withdrawn. [Online]. Available: http://arxiv.org/abs/1801.07175

[16] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[17] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6572

[18] E. Jones, T. Oliphant, P. Peterson et al., "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed ¡today¿]. [Online]. Available: http://www.scipy.org/