Smart Investor

PRACTICAL LANGUAGE PROCESSING

GROUP MEMBERS	*STUDENT ID*
Tao Xiyan	A0215472J
Li Jiayi	A0215492E

Contents

1. Project Overview

1.1 Problem Statement

Making investment decisions in stock market is risky sometimes because it is the fastest and also easiest way of making money as well as losing money. Therefore, investing on stock market needs careful planning with deep analysis which now a days is possible using advanced technologies with large computational power, neural network, relational database etc.

Stocks market analysis can be separated into two categories. One is Technical and another one is fundamental. Technical analysis looks at the price trend of a security and uses this data to forecast its future price movements where fundamental analysis, on the other hand, looks at economic factors, known as fundamentals. Though both type of analysis is important but technicians believe that all the information they need about a stock can be found in its charts and therefore technical traders, believe there is no reason to analyze a company's fundamentals because these are all accounted for in the stock's price.

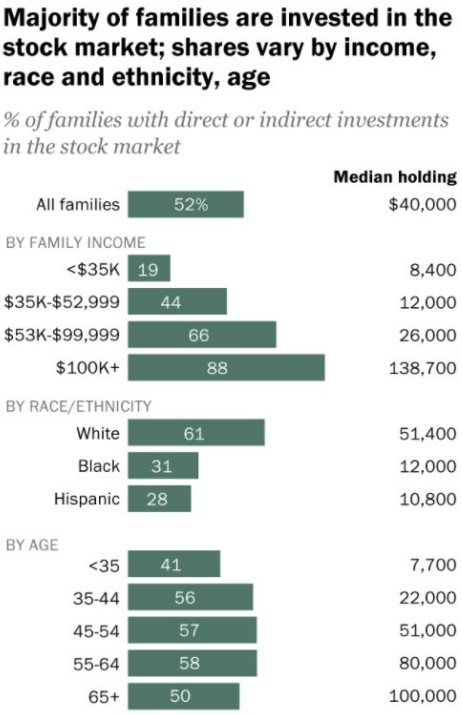

Figure 1. A statistical chart showing the increasing activity of the stock market

It can be seen from the figure above that the stock investment market is becoming more and more active. Due to the participation of more and more people, the stock market information including data information and text information has become more and more complicated. It has a great impact, but it takes a lot of people's time to collect and analyze it. Considering this problem, NLP technical analysis can be applied to this scenario to solve this problem.

1.2 Project Objective

In this project, we applied NLP techniques and other Machine learning algorithm on trading index and news, including International news and corporate news so as to have a comprehensive understanding of the stock environment.

Considering all the issues above, therefore, this project aims to apply NLP tools to analyze major international news and the company related news so as to make reasonable forecasts of one specific company's stock trend-Amazon's stock price trend. Also, combining with the other trading index as input numerical data to more comprehensive prediction.

At the same time, the system could give users with recent Financial reports written by professional Analyst. Additionally, the pre-trained model will be used firstly to do summarization, which dataset consist of CNN Finance news, so that this trained model can be used to do the AMZN analysis report summary.

A friendly User interface will also be designed, real-time forecasting of stock trends based on current numerical trading data and text information. On the meantime, giving professional investment analysis, so that target users can have one-stop control of all relevant investment information.

The highlight of this project lies in its real-time analysis and comprehensive investment service.

1.3 Business Value

While humans remain a big part of the trading equation, AI plays an increasingly significant role. According to a recent study by U.K. research firm Coalition, electronic trades account for almost 45 percent of revenues in cash equities trading. And while hedge funds are more reluctant when it comes to automation, many of them use AI-

powered analysis to get investment ideas and build portfolios.

"Machine learning is evolving at an even quicker pace and financial institutions are one of the first adaptors," Anthony Antonucci, vice president of global business development at IntelCenter Global Services, recently said.

When Wall Street statisticians realized they could apply AI to many aspects of finance, including investment trading applications, he explained, "they could effectively crunch millions upon millions of data points in real time and capture information that current statistical models couldn't."

From these points, we can see that NLP can deliver those transcriptions in minutes, giving analysts a competitive advantage and in the market now, lots of Intelligent stock price analysis system's are mostly use fundamental analysis and their service are single and inefficient, so from this perspective, our project do have a certain business value.

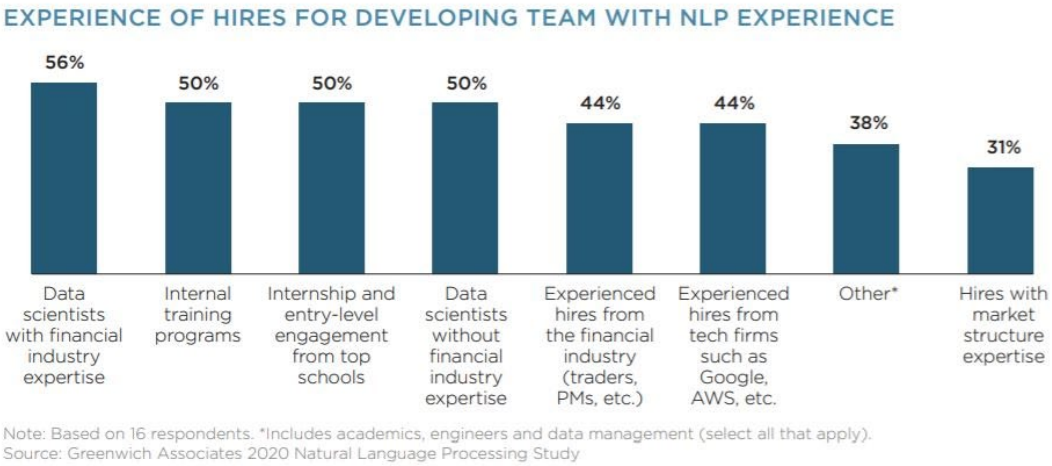

Figure 2. More people with NLP experience join Morgan Stanley

1.4 The AMZN&DJI&NEWS Dataset

1) **International News:** Historical news headlines from Reddit WorldNews Channel. They are ranked by reddit users' votes, and only the top 25 headlines are considered for a single date. (Range: 2008-08-11 to 2020-07-01)
2) **AMZN major News:** Historical AMZN major news headlines from Seeking alpha Website. It ranges also from 2008-08-11 to 2020-07-01.
3) **AMZN Trading index:** Historical AMZN Trading index including Open value, High value, Low value and Volume (numerical data) from Seeking alpha Website.

It ranges also from 2008-08-11 to 2020-07-01. Use the comparison result of the close value of the next day and the previous day as the forecast target.

2. System architecture

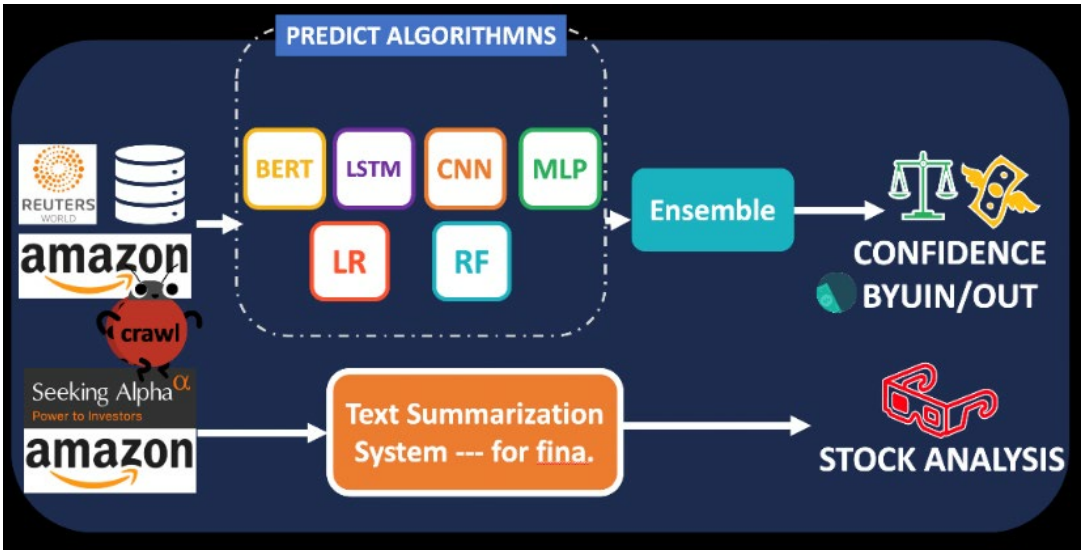

Figure 3. System Architecture

2.1 AMZN Stock Predict part

The first part is stock price prediction and investment recommendation part, where we use six algorithms to deal with information including the Stock trading indexes, top 10 international news and top three Amazon corporate news.

2.1.1 Final Performance

The performance of each algorithm is: For Bert, 61%; LSTM, 54%; CNN, 52%; MLP, 55%; LR, 57%; RF, 52%. Different character of each algorithm has been taken into consideration, so the Ensemble strategy is used to create final model. And the final performance reach to 62%. Although the result is not satisfactory as well, our Smart Investor system perform above the average in the related project in the relevant field (Comparing chart can be seen in the project video). Taking advantage of trained final model, combined with trading index, international and corporate new crawled in real

time, the Smart Investor can give users today stock price trend prediction, confidence probability and the final recommendation.

ACCURACY IS:	**0.607732**		
F1 SCORE IS:	0.608963		
	precision	recall	f1-score
0	0.59	0.58	0.56
1	0.60	0.59	0.61
ACCURACY			0.61
MACRO AVG	0.60	0.59	0.59
WEIGHTED AVG	0.60	0.61	0.60

Table 1. The performance of BERT model

ACCURACY IS:	**0.5240629630543424**		
F1 SCORE IS:	0.5018437772197421		
	precision	recall	f1-score
0	0.52	0.51	0.53
1	0.50	0.49	0.47
ACCURACY			0.52
MACRO AVG	0.52	0.50	0.52
WEIGHTED AVG	0.52	0.51	0.50

Table 2. The performance of RF model

ACCURACY IS:	**0.5715822245683545**		
F1 SCORE IS:	0.5649163965346942		
	precision	recall	f1-score
0	0.57	0.55	0.59
1	0.50	0.52	0.54
ACCURACY			0.57
MACRO AVG	0.53	0.50	0.53
WEIGHTED AVG	0.56	0.49	0.54

Table 3. The performance of LR model

ACCURACY IS:	**0.5395274689415679**		
F1 SCORE IS:	0.562491639653469		
	precision	recall	f1-score
0	0.49	0.48	0.49
1	0.54	0.57	0.55
ACCURACY			0.54
MACRO AVG	0.52	0.50	0.51
WEIGHTED AVG	0.54	0.50	0.52

Table 4. The performance of LSTM model

ACCURACY IS:	**0.5582458222456821**		
F1 SCORE IS:	0.5649163965346934		
	precision	recall	f1-score
0	0.56	0.54	0.57
1	0.51	0.50	0.52
ACCURACY			0.55
MACRO AVG	0.54	0.50	0.53
WEIGHTED AVG	0.55	0.51	0.55

Table 5. The performance of MLP model

ACCURACY IS:	**0.5238589544242854**		
F1 SCORE IS:	0.5782477481250522		
	precision	recall	f1-score
0	0.52	0.53	0.59
1	0.51	0.52	0.50
ACCURACY			0.52
MACRO AVG	0.53	0.49	0.52
WEIGHTED AVG	0.53	0.50	0.51

Table 6. The performance of CNN model

2.1.2 Some Experiments Results

	EP1	EP3	EP7	EP10	EP20
7X_TOP10	0.537964	0.562036	0.589581	0.527964	0.462036
7X_TOP25	0.587126	0.595569	0.576886	0.517126	0.495569
17X_TOP10	0.60001	0.607732	0.544266	0.48001	0.507609
17X_TOP25	0.589693	0.589595	0.500937	0.489693	0.499595

Table 7. The performance of BERT with different num. world news (combined top10 or top25) and different EDA parameter (7x or 17x) and different epoch parameter

	RF	LR	LSTM	MLP	CNN	PREPROCESSING METHODS
TEXT_DATA_TOP10	0.5240	0.5100	0.4673	0.5582	0.5238	text(glove)
TEXT_DATA_TOP25	0.5511	0.4974	0.5050	0.5621	0.5026	text(glove)
NUM_DATA_HOL_D	0.5218	0.5715	NA.	NA.	NA.	nums(stanscale)
TEXT_DATA_TOP25_NUM	0.4926	0.5632	NA.	NA.	NA.	text(glove)+nums(stanscale)
TEXT_DATA_TOP10_NUM	0.5025	0.5026	NA.	NA.	NA.	text(glove)+nums(stanscale)

Table 8. The performances of 5 algorithms with different data combined methods (text_data_top10: all text dataset with top10 word news; text_data_top25: all text dataset with top25 word news; num_data_hol_d: number data only; text_data_top25_num: text data combine with num. data)

2.2 Financial research report Summarization part

The second part is the Financial research report Summarization part. The Financial research report written by professional analyst has a huge impact on the future stock trend. But sometimes in order to elaborate more, the report tend to be very verbose and explain in great details. In order to solve this problem, the smart investor also provides Summarization of latest professional Analyst research report for users. As for our summarization model, we use pre-trained model as base model. And this summarization function also be fine-tuned to better fit our real case, we use 109,110 Financial News from Reuters and then get the final Text Summarization System --- for finance. One thing needs to be noted is that all the summarization of the report are updated in real time, which means user can just use our system to capture the main point

of the latest authoritative financial report.

3. Tools & Techniques

3.1 Algorithms

3.1.1 EDA

Introduction

We use EDA: easy data augmentation techniques, for boosting performance on our dataset. EDA consists of four simple but powerful operations: synonym replacement, random insertion, random swap, and random deletion.

1. **Synonym Replacement (SR):** Randomly choose n words from the sentence that are not stopwords. Replace each of these words with one of its synonyms chosen at random.
2. **Random Insertion (RI):** Find a random synonym of a random word in the sentence that is not a stopword. Insert that synonym into a random position in the sentence. Do this n times.
3. **Random Swap (RS):** Randomly choose two words in the sentence and swap their positions. Do this n times.
4. **Random Deletion (RD):** Randomly remove each word in the sentence with probability p.

3.1.2 Logistic regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression is estimating the parameters of a logistic model (a form of binary regression).

In a binary logistic regression model, the dependent variable has two levels (categorical). Outputs with more than two values are modeled by multinomial logistic regression and, if the multiple categories are ordered, by ordinal logistic regression (for

example the proportional odds ordinal logistic model). The logistic regression model itself simply models probability of output in terms of input and does not perform statistical classification (it is not a classifier), though it can be used to make a classifier, for instance by choosing a cutoff value and classifying inputs with probability greater than the cutoff as one class, below the cutoff as the other; this is a common way to make a binary classifier. The coefficients are generally not computed by a closed-form expression, unlike linear least squares. The logistic regression as a general statistical model was originally developed and popularized primarily by Joseph Berkson, beginning in Berkson (1944), where he coined "logit".[1]

3.1.3 Random forests

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees[2][3]. Random decision forests correct for decision trees' habit of overfitting to their training set.:587–588 Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees.

3.1.4 Long short-term memory

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture[4] used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between

important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

3.1.5 Convolutional Neural Network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels that shift over input features and provide translation equivariant responses. Counter-intuitively, most convolutional neural networks are only equivariant, as opposed to invariant, to translation[5].They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain-computer interfaces, and financial time series.

3.1.6 Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of perceptron (with threshold activation). Multilayer Perceptron are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training[6][7]. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

3.1.7 Attention mechanism

Self-attention-based models

There are three categories of tasks on self-attention-based models. For single and pair sequence tasks, we fine-tune pre-trained BERT model[9] on the downstream task. In pair sequence tasks, instead of independently encoding each text, we concatenate both separated by a delimiter and pass it to BERT model. Finally, the embedding corresponding to token is fed to a feed-forward network for prediction. For neural machine translation, we use Transformer model proposed by Vaswani et al. (2017) [8]with base configuration.

3.1.8 Transformer model

Introduction

Recurrent neural networks, long short-term memory and gated recurrent neural networks, in particular, have been firmly established as state-of-the-art approaches in sequence modeling and transduction problems such as language modeling and machine translation. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences. In all but a few cases, however, such attention mechanisms are used in conjunction with a recurrent network.

In the paper, “Attention Is All You Need”, the authors propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output.

Model architecture

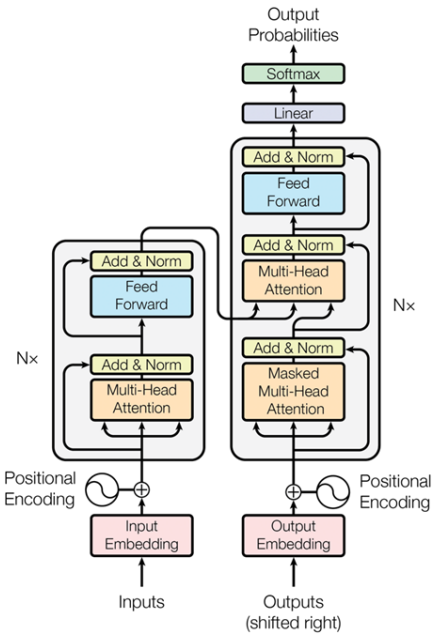

Figure 4. The Transformer - model architecture.

Most competitive neural sequence transduction models have an encoder-decoder structure. Here, the encoder maps an input sequence of symbol representations $(x1, \ldots, xn)$ to a sequence of continuous representations $z = (z1, \ldots, zn)$. Given z, the decoder then generates an output sequence $(y1, \ldots, ym)$ of symbols one element at a time. At each step the model is auto regressive, consuming the previously generated symbols as additional input when generating the next.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 4, respectively.

Encoder and Decoder Stacks

- **Encoder:** The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is $LayerNorm(x + Sublayer(x))$, where $Sublayer(x)$ is the function implemented by the sub-layer

itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $dmodel = 512$.

- **Decoder:** The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Scaled Dot-Product Attention

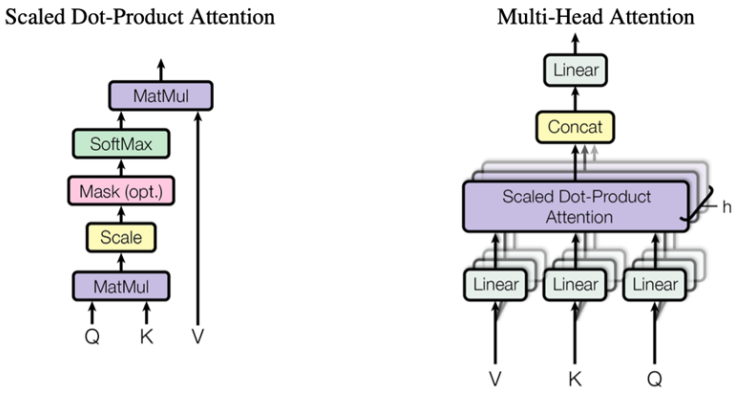

Figure 5. (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

We call our particular attention "Scaled Dot-Product Attention" (Figure 5). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We

compute the dot products of the query with all keys, divide each by$\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

Multi-Head Attention

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$MultiHead(Q,K,V) = Concat(head1,\ldots,headh)W^O$$

where $headi = Attention(QWi^Q, KWi^K, VWi^V)$

Where the projections are parameter matrices $Wi^Q \in \mathbb{R}^{d_{model} \times dk}$, $Wi^K \in \mathbb{R}^{d_{model} \times dk}$ and $W^O \in \mathbb{R}^{hdv \times d_{model}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $dk = dv = d_{model}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.1.9 BERT

Introduction

The full name of Bert is bidirectional encoder representation from Transformer model, that is, the encoder of the bidirectional transformer, because the decoder cannot obtain the information to be predicted. The main innovation of the model is pre-trained method, which uses masked LM and next sentence prediction to capture the representation of words and sentences respectively. That is, the Bert is a pre-trained model based on transformer.

A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

Model Architecture

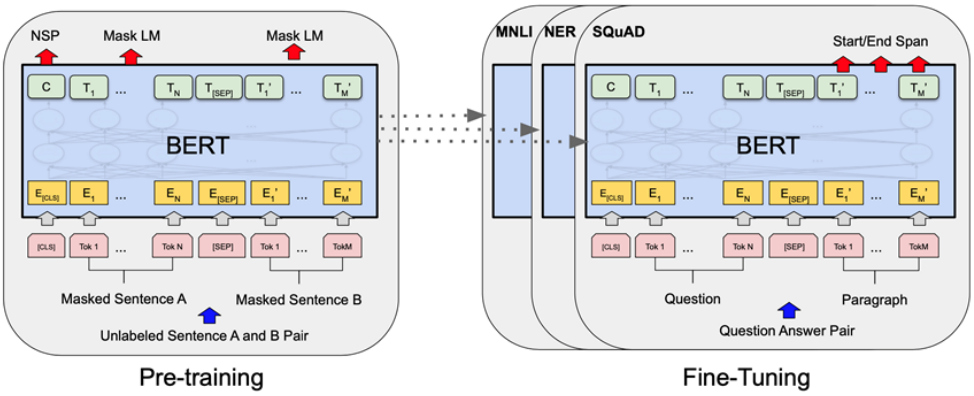

Figure 6. Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Input/Output Representations

To make BERT handle a variety of down-stream tasks, our input representation is able to unambiguously represent both a single sentence and a pair of sentences (e.g., ⟨ Question, Answer ⟩) in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text, rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence, or two sentences packed together.

The authors use WordPiece embeddings (Wu et al., 2016) with a 30,000-token vocabulary. The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. The authors differentiate the sentences in two ways. First, the authors separate them with a special token ([SEP]). Second, the authors add a learned embedding to every token indicating whether it belongs to sentence A or sentence B. As shown in Figure 1, we denote input embedding as E, the final hidden vector of the special [CLS] token as $C \in \mathbb{R}^H$, and the final hidden vector for the i^{th} input token as $T_i \in \mathbb{R}^H$.

For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. *A* visualization of this construction can be seen at Figure 7.

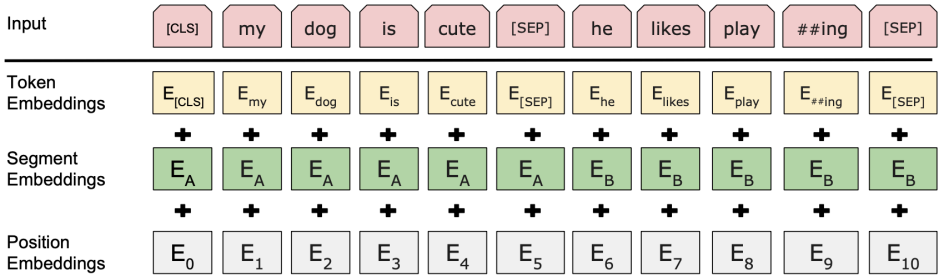

Figure 7. BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

3.1.10 Text Summarization with BERT

Bidirectional Encoder Representations from Transformers (BERT; Devlin et al. 2019) represents the latest incarnation of pretrained language models which have recently advanced a wide range of natural language processing tasks. In this paper, we showcase how BERT can be usefully applied in text summarization and propose a general framework for both extractive and abstractive models. We introduce a novel document-level encoder based on BERT which is able to express the semantics of a document and obtain representations for its sentences. Our extractive model is built on top of this encoder by stacking several intersentence Transformer layers. For abstractive summarization, we propose a new fine-tuning schedule which adopts different optimizers for the encoder and the decoder as a means of alleviating the mismatch between the two (the former is pretrained while the latter is not). We also demonstrate that a two-staged fine-tuning approach can further boost the quality of the generated summaries. Experiments on three datasets show that our model achieves stateof-the-art results across the board in both extractive and abstractive settings[9].

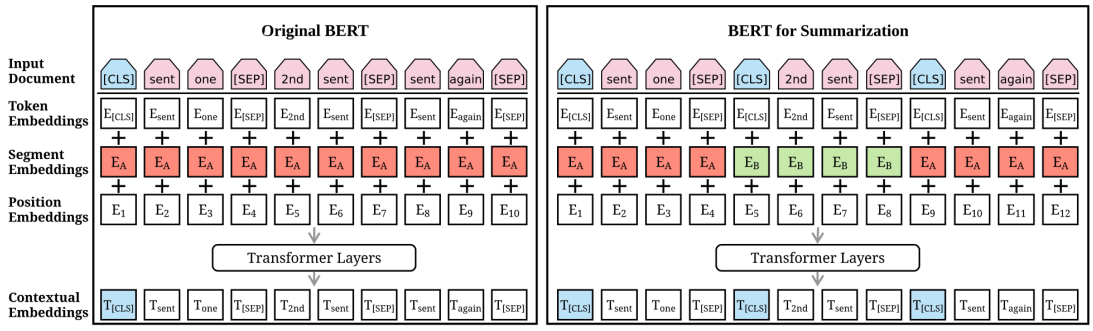

Figure 8. Architecture of the original BERT model (left) and BERTSUM (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences.

3.1.11 Ensemble Learning

Ensemble learning helps improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model. That is why ensemble methods placed first in many prestigious machine learning cases. Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking). Here considering different al has their own pros and cons, the Ensemble strategy is used to enhance the overall performance in Smart Investor.

3.2 Web Design

3.2.1 Front-end

The user interfaces are built using HTML5, CSS3, JavaScript and Bootstrap.

This past is that the user interacts with directly and can intuitively see real-time investment advice and financial report information. It is also referred to as the ‘client side’ of the application. It includes everything that users experience directly: text colors and styles, images, graphs and tables, buttons, colors, and navigation menu.

Responsiveness and performance are two main objectives of this Front End. The important issues need to deal with is to ensure that the site is responsive i.e. it appears correctly on devices of all sizes no part of the website should behave abnormally irrespective of the size of the screen. Here we take advantage of Bootstrap framework to make sure the front-end website is responsive enough.

HTML: HTML stands for Hypertext Markup Language. It is used to design the front-end portion of web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within the tag which defines the structure of web pages.

CSS: Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

JavaScript: JavaScript is a famous scripting language used to create magic on the sites to make the site interactive for the user. It is used to enhancing the functionality of a website to running cool games and web-based software.

Bootstrap: Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

3.2.2 Back-end

The Web Server backend is developed based on Flask framework. It acts as the middleware serving multiple functions as below:

i Consumed by the front-end. It proves the backing information for the front end.;

ii Showing analyzed consequence from the algorithm to users;

iii Calling other python scripts

Flask：Flask is a micro web framework powered by Python. Its API is fairly small, making it easy to learn and simple to use. But don't let this fool you, as it's powerful enough to support enterprise-level applications handling large amounts of traffic. You can start small with an app contained entirely in one file, then slowly scale up to

multiple files and folders in a well-structured manner as your site becomes more and more complex system performance.

Jinja2: Flask uses Jinja2 template engine for rendering templates. It was created by Armin Ronacher and is licensed under a BSD License. Jinja is similar to the Django template engine but provides Python-like expressions while ensuring that the templates are evaluated in a sandbox. It is a text-based template language and thus can be used to generate any markup as well as source code. It is modelled after Django's templates. It is fast, widely used and secure with the optional sandboxed template execution environment.

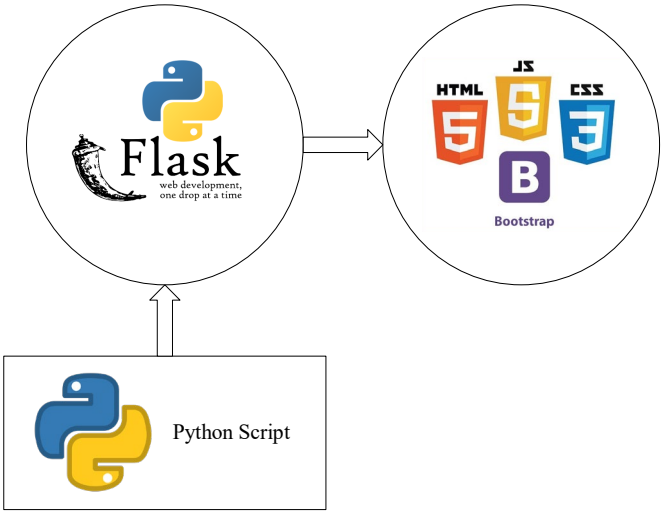

Figure 9. Front& back-end interaction diagram

4. Finding and Discussion

In the training process, the performance of each algorithm is: For Bert, 61%; LSTM, 54%; CNN, 52%; MLP, 55%; LR, 57%; RF, 52%. Different character of each algorithm has been taken into consideration, so the Ensemble strategy is used to create final model. And the final performance reach to 62%. Although the result is not satisfactory as well, our Smart Investor system perform above the average in the related project in the relevant field (Comparing chart can be seen in the project video).

Smart investor also provides Summarization of latest professional Analyst research report for users. As for our summarization model, we use pre-trained model as base model. And this summarization function also be fine-tuned to better fit our real case, we use 109,110 Financial News from Reuters and then get the final Text Summarization System --- for finance. One thing needs to be noted is that all the

prediction result and summarization of the report are updated in real time, which means user can just use our system to capture the main point of stock trend rapidly with the help of Smart Investor.

APPENDIX

Appendix A: Reference

[1] Logistic regression and artificial neural network classification models. Stephan Dreiseitla, and Lucila Ohno-Machadob. Journal of Biomedical Informatics 35 (2002) 352–359.

[2] Ho, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282. Archived from the original (PDF) on 17 April 2016. Retrieved 5 June 2016.

[3] Ho TK (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). IEEE Transactions on Pattern Analysis and Machine Intelligence. 20 (8): 832–844. doi:10.1109/34.709601.

[4] Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014.

[5] Mouton, Coenraad; Myburgh, Johannes C.; Davel, Marelie H. (2020). Gerber, Aurona (ed.). "Stride and Translation Invariance in CNNs". Artificial Intelligence Research. Communications in Computer and Information Science. Cham: Springer International Publishing. 1342: 267–281.

[6] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

[7] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[8] Attention is all you need. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. In Proc.of NIPS, 2017.

[9] Bert: Pre-training of deep bidirectional transformers for language understanding. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. In Proc. Of NAACL, 2019.