World Academy of Science, Engineering and Technology
International Journal of Economics and Management Engineering
Vol:15, No:2, 2021

A Context-Centric Chatbot for Cryptocurrency Using the Bidirectional Encoder Representations from Transformers Neural Networks

Qitao Xie, Qingquan Zhang, Xiaofei Zhang, Di Tian, Ruixuan Wen, Ting Zhu, Ping Yi, Xin Li

Abstract—Inspired by the recent movement of digital currency, we are building a question answering system concerning the subject of cryptocurrency using Bidirectional Encoder Representations from Transformers (BERT). The motivation behind this work is to properly assist digital currency investors by directing them to the corresponding knowledge bases that can offer them help and increase the querying speed. BERT, one of newest language models in natural language processing, was investigated to improve the quality of generated responses. We studied different combinations of hyperparameters of the BERT model to obtain the best fit responses. Further, we created an intelligent chatbot for cryptocurrency using BERT. A chatbot using BERT shows great potential for the further advancement of a cryptocurrency market tool. We show that the BERT neural networks generalize well to other tasks by applying it successfully to cryptocurrency.

Keywords—BERT, chatbot, cryptocurrency, deep learning.

I. Introduction

WITH the massive growth of the internet, we have a large amount of data and only some text data are annotated. For a task in a field like Natural Language Processing (NLP), we require lots of annotated data for supervised learning or unannotated data for unsupervised learning. BERT is one such pre-trained model developed by Google which can be fine-tuned to new data and used to create NLP systems like question answering, text processing, and sentiment analysis [1]. As BERT is pre-trained on huge amounts of data, the process of language modeling is easier. The main benefit for using a pre-trained model of BERT is substantial accuracy improvement compared to training on these datasets from scratch.

One important trend in NLP research is implementing pre-trained models. They have gained huge popularity in neural NLP tasks in recent years. Pre-trained word vectors such as word2vec [2] and GloVe [3] have been successful in capturing the meaning of single words, but by themselves they generally do not help in capturing the meaning of words in their context. Pre-trained contextual models such as ULMFiT [4], ELMo [5], OpenAI transformer [6] and BERT [1] have led to a dramatic shift in NLP research. Instead of training everything from scratch, Pre-trained contextual models have a big advantage of using pre-trained representations that encode contextual information. Another important trend in NLP research is developing systems that can perform bi-direction tasks. ELMo [7] uses a concatenation of right-to-left and left-to-right Long Short Term Memory Networks (LSTM), which means that the representation couldnt take advantage of both left and right contexts simultaneously. On the other hand ULMFit [4] uses a unidirectional LSTM. While the openAI transformer [8] gave us a fine-tunable pre-trained model based on the Transformer, it only trains a forward language model and outputs one token at a time. Having bidirectional contexts should, in theory, generate more accurate word representations.

BERT features both pre-training and bi-directional models. It is the first language model applying deeply bidirectional, unsupervised language representation, and pre-trained using only a plain text corpus [1]. It represents contextual representation with both left contexts and right contexts. The main reason why BERT is better than the previous methods is that it combines unsupervised and bidirectional characteristics for pre-training NLP.

A chatbot is a computer program used to conduct an on-line chat conversation via text, or text-to-speech, in a conversational manner [9]. The application usually runs on the server and then talks to people. The machine may respond in text or voice through various channels such as web sites, applications, or chat programs. A simple chatbot may provide basic information while a complicated chatbot can learn and develop itself [10]. Historically, the ELIZA program was the first chatbot [11]. Thereafter ALICE, Watzon, and Amazon Alexa were developed. Nowadays, with the popularity of chat or messaging applications, we use every application on a regular basis. Chatbots have been used in the fields of commerce, education, finance, and news for answering questions and providing information to serve customers in terms of virtual assistants [12].

Artificial intelligence (AI) plays a crucial role in the advancement of information technology to improve financial service quality and efficiency. Chatbots in particular are one of the most popular AI technologies for this purpose. Chatbots have been rapidly developed, especially in the financial field. Some financial chatbot systems have been proposed over the years. Typical applications of chatbots include financial assistants that help clients to answer their questions, financial service front desks that direct the client to suitable service departments, i.e., agents or brokers, and so on [13].

Qitao Xie is with the Department of Computer Science and Electrical Engineering, UMBC, Maryland, USA (e-mail: qxie1@umbc.edu).
Qingquan Zhang is with University of Illinois.
Xiaofei Zhang is with Carnegie Mellon University.
Di Tian is with Culver Academies.
Ruixuan Wen is with University of San Diego.
Ting Zhu is with UMBC.
Ping Yi is with Shanghai Jiaotong University.
Xin Li is with Wells Fargo Bank.

Cryptocurrency is a virtual currency that is transacted digitally and secured virtually by cryptography [14]. Starting in the year 2009, cryptocurrency has been getting more and more popular [15]. Since lots of the new terms regarding cryptocurrency are unfamiliar to lots of people, building a chatbot that can support the exciting crypto-market is interesting [16]. BERT can speed up retrieving the intended information well after data training. Here we are building a intelligent chatbot for Cryptocurrency using the BERT Neural Networks. With a chatbot, our goal is to transform the knowledge retrieval process to a better end-to-end conversational system. Since BERT has become a new NLP baseline, we also investigate the use of BERT as a language model in a question answering system.

The rest of the paper is organized as follows. Section II introduces why we want to design and implement a chatbot system on the cryptocurrency topic. Section III presents the main system architecture of the chatbot framework. Section IV illustrates how BERT is implemented and applied. Section V shows our results and discussion, and we will conclude in section VI.

II. Motivation and Background Study

At the center of the chatbot system, we have to face two major problems. The first problem is to understand the natural language question a user intends to present. The other problem is to retrieve the information pertaining to the question correctly and efficiently.

1) Understanding the user question: utilize NLP to parse and interpret a user's intent.
2) Retrieving the response: extract the best-fit response to the user's intent.

A. Chatbot

We focus on the machine learning (ML) based chatbots. Existing ML-based chatbots can be intention-detection or generative type. An intent-detection model exploits a text classification task [17], where a classifier learns how to predict intents from a set of questions incoming from the user, with mainly using deep learning (DL). Sometimes, the correct intent cannot be determined from one incoming question and must be clarified in further conversation. Furthermore, another large group of chatbots, called generative chatbots, typically function in the machine translation manner. The input sequence (question) is mapped into the output sequence (answer) by sequentially generating text elements (usually words). This particular family of ML approaches is called sequence-to-sequence (abbreviated as seq2seq). Seq2seq chatbots can be created using either statistical machine translation (SML) or the recently popular neural machine translation (NMT) approaches. The research in Reference [18] presented a conversational model focusing on previous queries/questions. These models can address the challenge of a variable input and output length.

B. BERT Model

BERT, as a language model in the use of Machine Learning for the field of NLP, takes the previous and next tokens as the inputs when processing. Unlike the conventional seq2seq model using the RNN based model, BERT uses the Transformer architecture as its underlying structure. The Transformer is an architecture that relies solely on the self attention mechanism and removes the need for recurrence by processing the entire input string at once instead of in a word-by-word fashion. One Transformer contains two parts: encoder and decoder. The encoder processes the text input and the decoder generates predicted results. The Transformer architecture is designed to reduce the complexity of the conventional sequence transduction models [19]. Fig. 1 shows the structure of the input-output flowchart. The question answer pairs are loaded into embedding process. Then the pre-trained BERT fine-tunes those embeded texts. Ideally, BERT could be used to train on sentence pairs that take question and answer pairs as input and then learn a specific embedding for the sentences to help the model distinguish them.

BERT utilizes only the encoder from the Transformer architecture, as the goal of the network is to generate representations. These representations are trained bidirectionally, which means that for the encoding of each word, the words to the right and left of it in the input sequence are evaluated. The training of a BERT model is similar to the traditional training of word embedding models. In this traditional setting, training often has a specific goal that has to be determined beforehand. For example, many models try to predict the next word in a certain sequence, which limits the usage of these models to that specific task.

The contribution of this paper can be summarized as follows:

1) We develop an enhanced model based system for the end-to-end response selection. We use multiple word embeddings and the most recent context focusing as our methods.
2) We implement the method for selecting the utterance from the dataset (from a pool of 50,000 sentences), by using a sentence-endcoding based method and obtaining a good performance with a reasonable computational cost.
3) For effective model resembling, we averaged the output from the models with different parameter initializations.
4) The final system achieves the best performance overall by optimizing the end-to-end response selection.
5) BERT has the advantage of context-sensitive and task-agnostic, which means that it requires minimal architecture changes for a wide range of tasks.

Our research is interested in developing a context-centric chatbot in the field of cryptocurrency. The BERT technique is used for training to help the chatbot be more powerful. The pretrained BERT model is applied to enhance the performance of the context-centric chatbot in the field of cryptocurrency.

III. System Architecture

The steps to implement the Q&A model are:

1) To develop a common sense reasoning model that mimics human reasoning.
2) To prepare FAQs from a knowledge base, product manual, or documentation.
3) To ceate a smart chatbot that can answer FAQs for the cryptocurrency industry.

We perform a fine-tuning task on Q&A using Google Colab. The following steps are used to perform BERT fine-tuning on Google Colab:

1) Change Runtime to a Tensor Processor Unit (TPU).
2) Clone the BERT github repository.
3) Download the BERT pretrained model.
4) Preprocess the Q&A datasets, including training/testing datasets.
5) Train the dataset on TPU.

Google released two types of BERT models, including base and large ones [1]. Each one has an uncased and cased version. Uncased version converts the texts to lowercase before tokenizing. Our system uses a BERT pretrained model with BERT-Base, and Uncased: 12-layer, 768-hidden, 12-heads, and 110M parameters.

As shown in Fig. 1, BERT representations are fed into an added output layer with minimum changes to the model architecture, predicting for every token or predicting for the entire sentence. All the parameters are fine-tuned, while the additional output layer will be trained from scratch.

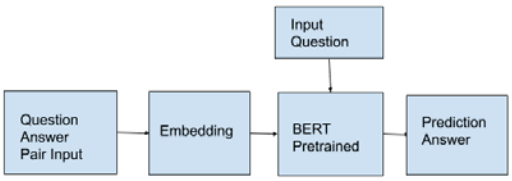

Fig. 1 Input-output flowchart

A simple model was developed, illustrated in Fig. 2. Further, Fig. 3 shows the embeddings of the BERT input sequence.

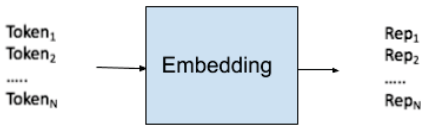

Fig. 2 Tokens through the embedding layer

Given the questions Q and the answers A, the word embeddings for the tokens in Q are the matrices $E^{\mathrm{q}} \in \Re^{\mathrm{d*I}}$ and the tokens in A are $E^{\mathrm{a}} \in \Re^{\mathrm{d*J}}$, where d is the dimension of the word embeddings. Place two LSTMs in both directions, respectively, and then concatenate the outputs of the two LSTMs.

$$H_i^q = LSTM(E^q, i) \tag{1}$$

$$H_j^a = LSTM(E^a, j) \tag{2}$$

Hence, we obtain a matrix H^{q} as a contexual representation of the questions Q and a matrix H^{a} as a contexual representation of the answers A.

The attention weight, which signifies which query words are most relevant to each words in the response, is calculated in the equation [20] below:

$$S_{ij} = Sim(h_i^q, h_j^a) \tag{3}$$

BERT differs from other models in that it is designed to pre-train deep bidirection representations by jointly crossing on both left and right context in all layers.

The normalized attention weights through softmax are represented as:

$$\alpha_{ij} = \frac{exp(s_{ij})}{\sum_k exp(s_{kj})} \tag{4}$$

Then summarize information for each word in the answer via

$$\widehat{h}_j^a = \sum_i \alpha_{ij} h_i^q \tag{5}$$

Thus we obtain a matrix $\widehat{H}^a$ as the question-aware representation of the response A. Next, the working memory M is formatted as

$$M = f_\theta(H^a, \widehat{H}^a) \tag{6}$$

where f_θ is a function of fusing its output matrices, parameterized by θ.

To find the answer span over the working memory M, first, a question vector is formed as

$$h_q = \sum_i \beta_i h_i^q \tag{7}$$

where

$$\beta_i = \frac{exp(w^T h_i^q)}{\sum_k exp(w^T h_k^q)} \tag{8}$$

and W is a trainable vector.

The probability distribution of the index over the response is calculated by

$$P_i = softmax((H_i^q)^T W M) \tag{9}$$

To optimize model parameter θ, minimize the loss function defined as the sum of the log probabilities of the predicted distributions, and then average over all training sets:

$$Loss(\theta) = \frac{-1}{|\mathcal{S}|} \sum_i^{|\mathcal{S}|} (log(P_i)) \tag{10}$$

where $\mathcal{S}$ is the training set.

To fine-tuning question answering, one can add additional layers for learning on the main core model. In the context of the response, one vector is for the beginning of the answer and the other vector is the ending of the answer. Thus, the probability that a word w is the beginning of an answer scope is:

$$P_m = \frac{exp(B * T_m)}{\sum_{n=1}^{H} exp(B * T_n)} \tag{11}$$

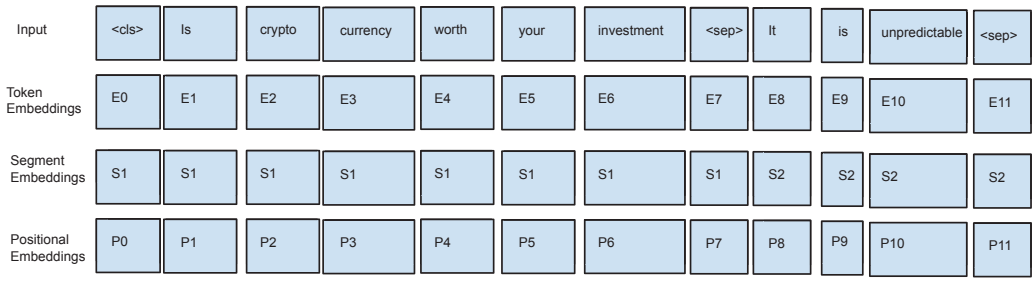

Fig. 3 The embeddings of the input sequenceFig.

To calculate the probability that a word is the ending of an answer, we simply replace the vector B with the vector E. T is the sum of the embeddings of the input sequence. Then the loss function is the log likelihood of the correct beginning and ending position [22].

A. Data Collection

We wrote a Python crawler program to collect multiple Q&As on the topic of cryptocurrency from Quora. We used more than 200 keywords for data collection. For each keyword, we first scraped all the valid questions from Quora. Then we scraped the answers from the questions we collected earlier. Using Python's Selenium module, we are able to simulate the browser's behavior of the JavaScript to move down in the page to get all the available questions on the Quora query. This is very useful for web scrawling. We run the same program repeatedly for each keyword. We collected more than 60,000 question-answer pairs in total (Table I).

TABLE I
STATISTICS OF THE DATASET COLLECTED FROM QUORA

#Keyword	239
Question/Answer Pairs before preprocessing	62K
Question/Answer Pairs after preprocessing	32,511
#Words after preprocessing	0.84M

Algorithm 1: Collecting cryptocurrency related Q&A Data

input : KW: cryptocurrency related keyword
output: question/answer pair list

scrape all the available questions from quora.com based on keyword;

$qList \leftarrow$ http://www.quora.com/search?q=KW;

for $q \in qList$ **do**
  $answer \longleftarrow GetAnswerFromQuora\{q\}$;
  if *unanswered* **then**
    continue;
  $ProcessAnswer(answer)$;
  $qa\_list \longleftarrow (q, answer)$

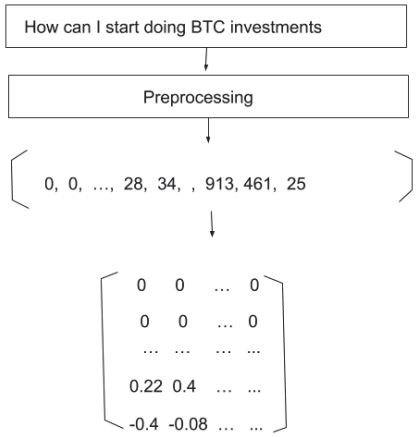

Fig. 4 The example of an embedding process

B. Data Preprocessing

First we cleaned up spaces and special chars and then removed questions if more than 20 words. We kept the answers only with the first 100 words. We did a little bit of further preprocessing before loading the data into the BERT model. The format of the data files is "ID, Question, Answer". The files include 3 columns, separated by tabs. The header of the file explains each of these columns. To use the pretrained BERT model, we needed to pre-process the same way each time. We used the panda API to feed the data for data training.

Below we summarize the major steps for data pre-processing.

1) Removed extra spaces. This was done using regular expressions.
2) Eliminated extra special characters. Necessary word contracting is processed as well.
3) Limited the length of questions and answers.
4) Converted all texts to lower-case in order to avoid duplicates.

The above steps applied Python's re module and string operations.

IV. EXPERIMENT

We used Google Colab (Tesla K80 GPU, 12 GB RAM) for the deep learning framework. We applied the

TABLE II
FORMATTED INPUT EXAMPLES

ID	Question	Answer
1	Which are the best cryptocurrency tokens to buy now and hold them for a minimum of 2 years?	That's like asking who will win Super Bowl in 2 years.
2	Will BTC get back up to 15k this year?	Ask yourself.
3	What is the best cryptocurrecy to buy other than BTC and LTC?	That depends on what you mean by the best.
4	What is the target price of BTC in the next six months?	The price of BTC has been violent and volatile over the last few months.
5	What is your favourite platform for BTC and ETH purchase?	I use Coinbase for ETH because of convenience and ability to integrate my European bank account.

TABLE III
A FEW EXAMPLES OF QUESTION-ANSWER PAIRS

Conversation 1:
Question: "How can I start doing BTC investments?"
Answer: "I would like to buy a movie ticket."
Conversation 2:
Question: "What is BTC?"
Answer: "Bitcoin."
Conversation 3:
Question: "Why do people keep saying BTC is dead?"
Answer: "Dying LOL Bitcoin is not dying."

TABLE IV
THE RANGES OF BERT MODEL HYPERPARAMETERS

Hyperparameter	Range
Train Batch Size	16 to 48
Learning Rate	2e-3 to 2e-7
Train Epochs	0.5 to 5
Max Seq Length	32 to 512

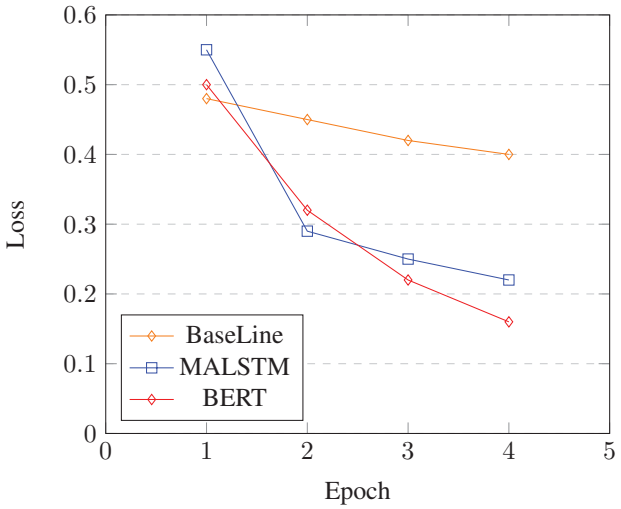

Fig. 5 Training Epochs vs Loss

BERT-Base pre-trained model which contains 24 layers for data training/testing/predicting. Then we studied the accuracy and consuming-time by adjusting the learning rate, epoch of training, and maximum sequence length. Different type of machines such as TPU and GPU were also studied. For the performance research, we further applied different sample sizes. We collected nearly 70,000 cryptocurrency-related records from the Quora dataset. For training the model faster and easier, we randomly selected 10,000 records as the initial experimental data. Some examples are listed in Table III.

The hyperparameters of BERT include the batch size, epoch, learning rate, and maximum sequence length. The learning rate defines how quickly a network updates its parameters. The epoch is the number of runs of the training dataset during data training. [23] The batch size is the number of the samples each training loop. The maximum sequence length is the maximum length for the source sequence length. Sequences longer than this length are truncated to this length. The ranges of the hyperparameters studied are shown in Table IV. We compare the model with other state-of-the-art Q&A algorithms, Table V. We also performed the experiment on the remaining cryptocurrency sentence pairs as well. Finally, the first 1000 matched sentence pairs are taken out of each dataset. Here we reuse the models trained from the one above, but test with the remaining datasets.

For the purpose of fine-tuning, we used the following hyperparameters for training and evaluating: batch size (16, 32), learning rate (5e-5, 3e-5, 2e-5), and the number of epochs (2, 3, 4). The training loss and corresponding evaluation loss/accuracy were compared.

V. RESULTS

We used the pretrained model with uncased BERT base model. Initially, 3 train epoch, 128 of max sequence length, and 1000 iteration per loop were used. Different sample sizes are applied. 70% of data is for data training, and 30% of data is for testing. It took 5 minutes for data training on 5000 samples. When the sample size is larger, data training time takes longer.

How to choose the optimal learning rate for model training is an interesting topic. Too small learning rate could lead to a long training process. However, too large learning rate will affect the quality. After investigating different learning rates in configuring BERT model, the results suggest a rather reasonable learning rate of 2e-5 would have good model performance in terms of our train and test data sets. Table VI A lists the relationship between the learning rate and the accuracy after 200 iterations.

Training epoch is another factor. We compare different epochs to determine how many epochs are necessary for a finetuned BERT model. An epoch is one complete presentation of the data set to be learned to a learning machine. Learning machines like feedforward neural nets that use iterative algorithms often need many epochs during their learning

TABLE V
MODEL COMPARISON

Methods	Average Evaluation Accuracy	Deviation
MALSTM	78.4%	(-2.9%, +2.0%)
LSTM + Word2Vec	79.2%	(-2.5%, +1.8%)
BERT	81.4%	(-2.4%, +2.2%)

TABLE VI
A) TEST ACCURACY AFTER 200 ITERATIONS VS LEARNING RATE
B) RUNNING EPOCHS VS LOSS

Learning rate	Test accuracy
2e-3	46%
2e-4	60%
2e-5	80%
2e-6	30%
2e-7	25%

Epoch	F1 score
0.5	2.700e-4
1	1.803e-4
2	8.986e-5
3	8.199e-5
4	6.632e-5
5	5.585e-5

phase. Table VI(b) shows that after four epochs the F1 score doesn't change much. Further, Fig. 5 illustrates that Loss decreases when epoch increases during the training. This indicates that performance may be improved by increasing the number of training epochs. We use the LSTM model as a baseline. Fig. 5 also shows that BERT model is slightly better than LSTM. when considering the relationship between training epoch and loss. The loss decreases a little faster with BERT model than that with LSTM models during the training.

BERT was a powerful model, which could learn most fine-tuning datasets rather easily. This means that it is prone to overfitting of the new dataset when trained with bad settings. The required memory for BERT data training is so huge that the batch size could be lowered to get the better performance. [1] We can consider reducing the Sequence length of the input feature as well, as long as this does not remove significant information. Table VII shows the relationship between the batch size and the training time Table VIII shows the relationship between the sequence length and the training time. Our experiment indicates that the ideal sequence length size is between 100-500. If the maximum sequence length is less than 100, it will result in a memory issue. In addition, if the sequence length is less than 50, the accuracy could be small. Smaller sequences compute faster, but the end of the information could be discarded if it does not fit. The batch size affects the data training time, as well as the loss. The larger the batch size is, the less the data training time; whereas the loss increases. Larger batch size often means lower accuracy but faster in terms of speed.

To test the performance of our system, we compare it with two other well-known algorithms. All embeddings are trained in uncased mode. The hidden layers are set to 12. MALSTM [21] was proposed by the MIT team in 2016 and has achieved excellent results in calculating the similarity of sentences. In Table V, we display the results of training 20,000 cryptocurrency related question pairs. The BERT model is slightly better in terms of the accuracy rate for our data, which means that our system would have achieved a level that is higher or on par with the current state-of-the-art models. In addition, using a pretrained BERT model is time-saving with data training.

The Levenshtein distance [24] is very commonly used in sequence to sequence comparison. The Levenshtein distance between two strings a and b calculates the length of the shortest sequence of substitutions, insertions, and deletions needed to transform a into b, which can be expressed as the following:

$$Lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0 \\ \begin{cases} Lev_{a,b}(i-1,j)+1 \\ Lev_{a,b}(i,j-1)+1 \\ Lev_{a,b}(i-1,j-1)+1 \end{cases} & \text{else} \end{cases} \tag{12}$$

Here, $Lev_{a,b}(i,j)$ is the distance between the first i characters of a and the first j characters of b. We then calculated the Levenshtein distance between the predicted answer and the actual answer to further prove the accuracy of the cryptocurrency chatbot model. Table IX shows the Levenshtein distance vs the learning rate with the batch size of 32, epoch number of 3, maximum sequence length of 256. The best average Levenshtein distance 0.661 at the learning rate of 3e-5.

TABLE VII
BATCH SIZE VS. TRAINING TIME

batch size	training time	loss
16	5m5s	3.993e-5
24	4m19s	6.799e-5
32	3m53s	8.199e-5
40	3m34s	9.432e-5
48	3m30s	1.125e-4

TABLE VIII
SEQUENCE LENGTH VS. TRAINING TIME

maximum sequence length	training time
32	need more RAM
64	need more RAM
128	3m53s
256	4m26s
512	6m36s

BERT has a constraint on the maximum length of a sequence after tokenizing. As described earlier, the maximum sequence length has a direct impact on the batch size that can be chosen. For the BERT model, the maximum sequence length is 512. But we can set any sequence length lower than that. For faster training, we used 128 as the maximum sequence length. A bigger number may give better results if there are sequences longer than 128. From the test, if the maximum sequence length is lower than 128, then it needs

a machine with higher RAM. Our account couldn't handle that once the maximum sequence length is set to 64 or lower.

Our experimental procedures had a logical flow to the problem, but in hindsight we would have changed our strategy at attacking this problem. Dedicating more time to hyperparameter fine tuning of the BERT baseline was likely a better use of our time than continuing to add different models on top of the baseline. We also should have been more aware of potential over fitting pitfalls when adding layers to the baseline before we added them - and focused more of our time and energy on hypothesizing ways to overcome this issue before implementing models and then trying to fix the overfitting issue.

TABLE IX
LEVENSHTEIN DISTANCE VS LEARNING RATE

Learning rate	batch size	Levenshtein distance
2e-3	32	1.993
2e-4	32	0.904
3e-5	32	0.661
2e-5	32	0.668
2e-6	32	1.337
2e-7	32	1.505

VI. CONCLUSION

In this research, we investigated different hyperparameters on cryptocurrency datasets for the performance of the BERT model. It also demonstrates the advantages of BERT embeddings. This research is important and interesting because the chatbots are trained on the limited domain-specific data and it is the first promising chatbot trained on cryptocurrency data using the BERT model. The BERT implementation uses only a fine-tuning process on top of the BERT-base model, making use of its powerful embeddings. Further the hyperparameters of BERT need to be well tuned for the best performance. It is true that a chatbot using BERT shows great potential for the further development and deployment of such a beneficial cryptocurrency market tool.

Despite the search for effective solutions on limited datasets being much more challenging, in future research, we are planning to augment our datasets with newly covered topics, to experiment with different seq2seq architectures by tuning their hyper-parameters. As for future work, we foresee the potential of chatbot technologies to play a much more significant role in the digital currency domain. For example, a software chatbot can be deployed in the real-world to become home financial market inquiry robots. It is important to advance the understanding of how to effectively represent the contextual interactions for the response selection task. Besides, we can combine the data mining method and predict the potential market trend.

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding, *ArXiv preprint arXiv*, 1810.04805, 2018.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. "Distributed representations of words and phrases and their compositionality". *Proceedings of the 26th International Conference on Neural Information Processing Systems* Vol. 2, pages 31113119, 2013.

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing* (EMNLP), pp 15321543, 2014.

[4] Jeremy Howard and Sebastian Ruder. "Universal language model fine-tuning for text classification", *arXiv:1801.06146 [cs.CL]*2018.

[5] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." *arXiv preprint arXiv:1802.05365*, 2018.

[6] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.

[7] Asa Cooper Stickland and Iain Murray. "Bert and pals: Projected attention layers for efficient adaptation in multi-task learning", *arXiv:1902.02671 [cs.LG]*, 2019.

[8] Sina J. Semnani, Kaushik Ram Sadagopan, and Fatma Tlili. "BERT-A: Fine-tuning BERT with Adapters and Data Augmentation." https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15848417.pdf, 2019.

[9] Mauldin, Michael L. "Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition." *AAAI*. Vol. 94. 1994.

[10] Manish Dudharejia. "Chatbots are the next big platform." *Entrepreneur.* www.entrepreneur.com/article/298600. Oct. 4, 2017.

[11] Weizenbaum, Joseph. "ELIZAa computer program for the study of natural language communication between man and machine." *Communications of the ACM 9.1*, 36-45, 1966.

[12] Xu, Anbang, et al. "A new chatbot for customer service on social media." *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems.* ACM, 2017.

[13] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schtze. "Introduction to Information Retrieval." *Cambridge University Press.* 2008.

[14] Ameer Rosic. "What is Cryptocurrency: Everything you need to know." *Blockgeeks*. blockgeeks.com/guides/what-is-cryptocurrency. Accessed Sept. 13, 2018.

[15] *Market cap*, 2020 https://www.ccn.com/cryptocurrencies-are-looking-bullish-as-market-cap-shatters-6-month-high/

[16] Qitao Xie, Dayuan Tan, Ting Zhu, Qingquan Zhang, Sheng Xiao, Junyu Wang, Beibei Li, Lei Sun, and Ping Yi. "Chatbot Application on Cryptocurrency" *IEEE Conference on Computational Intelligence for Financial Engineering & Economics*, 2019.

[17] Devlin, J. und Chang, M.-W. "Google AI Blog: Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing, Google AI Blog." Available here: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html (Accessed: 2. February 2020). 2018.

[18] "The Transformer for language translation." Available here: https://www.youtube.com/watch?v=KzfyftiH7R8&t=1022s (Accessed: 7. July 2020).

[19] Sharan, S. "Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT." Available here: https://medium.com/huggingface/distilbert-8cf3380435b5 (Accessed: 7. February 2020). 2019.

[20] Jianfeng Gao, Michel Galley, and Lihong Li. "Neural Approaches to Conversational AI: Question Answering, Task-oriented Dialogues and Social Chatbots" *Information Retrieval* Vol. 13, No. 2-3, pp 127298. 2019.

[21] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. "Why Does Unsupervised Pre-training Help Deep Learning". *Journal of Machine Learning Research* 11 625-660, 2010.

[22] Jonas Mueller and Aditya Thyagarajan. "Siamese recurrent architectures for learning sentence similarity." *Thirtieth AAAI Conference on Artificial Intelli- gence*, 2016.

[23] Brownlee, Jason. "Difference Between a Batch and an Epoch in a Neural Network". https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/, 2018.

[24] VI Levenshtein Tlili. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf, 1966.