Università di Pisa

Human Language Technologies
Academic Year 2021/2022

Sentiment Analysis on Crypto Comments

Luca Santarella

Contents

1 Introduction **2**

2 Cryptocurrency community **4**

3 Sentiment Analysis and Transformers **6**

4 Experiments **9**

4.1 Using Transformers as classifiers 9

4.2 Changing the classifiers on top of the Transformers 12

4.3 Using FFNN/CNN/LSTM classifiers with GloVe 13

4.4 Using Transformers as encoders 16

4.5 Using the model for sentiment analysis 16

5 Conclusions **18**

Chapter 1

Introduction

In this project we will conduct a sentiment analysis (sometimes referred also as *opinion mining*) of comments made by users of the crypto community. The models used will be the Transformers models: BERT (Bidirectional Encoder Representations from Transformers), DistilBERT and RoBERTa, a classical FFNN (Feed Forward Network) with multiple layers, a CNN (Convolutional Neural Network) and a LSTM (Long Short-Term Memory) network. The aim of the project is to determine the effectiveness of fine tuning on a specific domain such as the crypto community, which is rich of peculiar terms and idioms commonly used by its users. Further, we compare the performances of the standard FFN with the CNN and LSTM with and without pretrained embeddings and finally with the state of the art models represented by BERT, DistilBERT and RoBERTa models used either as stand-alone classifiers or as encoders for the previous models.

The fine tuning process was applied to pre-trained Transformers, the models used are BERT, DistilBERT and RoBERTa which were provided by the Python library `transformers` from HuggingFace. This library allows to load various pre-trained models, import labeled datasets and also to use a useful API which ease the work on NLP tasks. The labeled dataset used for the training, validation and testing of the models was provided by SocialGrep which was responsible for the collection, aggregation and labeling of the comments taken from various subreddits (i.e. forums on Reddit) about crypto for the period of August 2021. A second unlabeled dataset with comments coming exclusively from the "r/cryptocurrency" subreddit was used to conduct experiments on the model. This dataset was built using the PushShift API for Reddit which is used to collect data of posts and comments from Reddit and pmaw which is a wrapper for the PushShift API to make multiple requests.

Finally we experimented with this unlabeled dataset to try to understand the correlation between the price of Bitcoin (the most popular cryptocurrency) and the sentiment of the community, which we believed to be a strong one. The results demonstrated that in various crashes of the market during the month of November 2021 the sentiment score dropped significantly with respect to the previous days. Thus showing that a strong price action of Bitcoin and more general of cryptocurrencies implies a strong response in sentiment of the community.

Chapter 2

Cryptocurrency community

A cryptocurrency is defined as a digital currency designed to work as a medium of exchange through a computer network that is not reliant on any central authority, such as a government or bank, to uphold or maintain it.

The first and most popular crypto is by far Bitcoin, which was invented in 2008 by an unknown person or group of people using the name Satoshi Nakamoto[1]. The vision of Satoshi was to have *"A purely peer-to-peer version of electronic cash [which] would allow online payments to be sent directly from one party to another without going through a financial institution"*. In this way no centralized party had to be involved in the exchange of goods or for pure payments, this was achieved by using cryptography and a peer-to-peer network which is able to validate the transactions through a Proof of Work system. A public ledger records all bitcoin transactions and copies are held on servers around the world. Anyone with a spare computer can set up one of these servers, known as a node. Consensus on who owns which coins is reached cryptographically across these nodes rather than relying on a central source of trust like a bank.

Figure 2.1: Bitcoin and Ethereum

The introduction of Bitcoin and more in general of crypto, have been vital to a lot of people who couldn't rely on the classical banking and financial system. Further, Bitcoin proved to be, not just a medium of exchange, but also a deflationary store of value which could prevent the effects of the constant rising inflation of traditional fiat currency (e.g. USD, EUR).

The role of crypto reflected also in our society and culture giving importance to the decentralized aspects of our data, one such example is the introduction of the so called "Web 3.0", an idea for a new iteration of the World Wide Web based on blockchain technology, which incorporates concepts such as decentralization and token-based economics. One of the greatest revolution in the crypto sphere was the introduction of smart contracts, a transaction protocol which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement. Ethereum has been one of the first cryptocurrency to integrate smart contracts in their blockchain by leveraging the EVM (Etherem Virtual Machine), a Turing Complete state machine which allows to run code written for example in Solidity, which is the most popular programming language using the EVM.

The Bitcoin code is open-source and it is powered by the community which proposes and discusses new features and also maintains the source code, the whole ecosystem thrives on the actual efforts of developers and enthusiasts, this is one of the reason why decentralization has been a prominent feature. The community does not just affect the development of a cryptocurrency but it also influences its price, one of the main factors is the popularity behind the coin or in some cases how much the investors participate in the protocol. Communities are often gathered in online social spaces such as Twitter, Discord or Reddit. In our experiments we will use datasets which comprehend comments coming from subreddits such as: "r/CryptoCurrency", "r/SatoshiStreetbets", "r/CryptoMoonshots" and many others. In these social spaces people often write and discuss about crypto news or the general state of the crypto market, thus comments collected from this subreddits seem ideal to analyze the sentiment of the crypto market.

Chapter 3

Sentiment Analysis and Transformers

Sentiment analysis is the task of classifying the polarity of a given text at the document, sentence, or feature/aspect level, whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. This kind of classification task is commonly used for analyzing reviews, survey responses, online and social media to have an insight on a particular product or idea.

In this project the text analyzed will be comments coming from a crypto forum where users participate in discussions about crypto news, ideas, analysis and more in general the state of the crypto ecosystem. The comments are taken from an already labeled dataset from SocialGrep which encompasses a month's worth of posts and comments from selected cryptocurrency subreddits, the dataset is indexed for the whole month of August 2021, in order to preserve users' anonymity and to prevent targeted harassment, the data does not include usernames. This first dataset used for training, validation and testing had a total of roughly 3.7 million rows which have been reduced to almost 2 million rows, the preprocessing phase deleted comments that had a "[deleted]" or "[removed]" body, comments that were made by bots and also comments which were longer than 512 characters. This dataset was composed by the *body* of the comment, which is the text collected and by the *sentiment* which is a value that could be 0 (negative comments), 1 (positive comments) or 2 (neutral comments).

The actual number of rows used for training and validation was around one hundred thousand, this reduction was done due to the great amount of time that the fine-tuning process takes to train a model with so many parameters, so for this reason we chose to just use a small subset of the data available.

The BERT model is a Transformer model which uses a mechanism called *Attention* first introduced by Google Brain in 2017[2], BERT is a bidirectional transformer pretrained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia. The DistilBERT model is a distilled version of BERT that was proposed in the blog post *"Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT"* and the paper *"DistilBERT, a distilled version of BERT:*

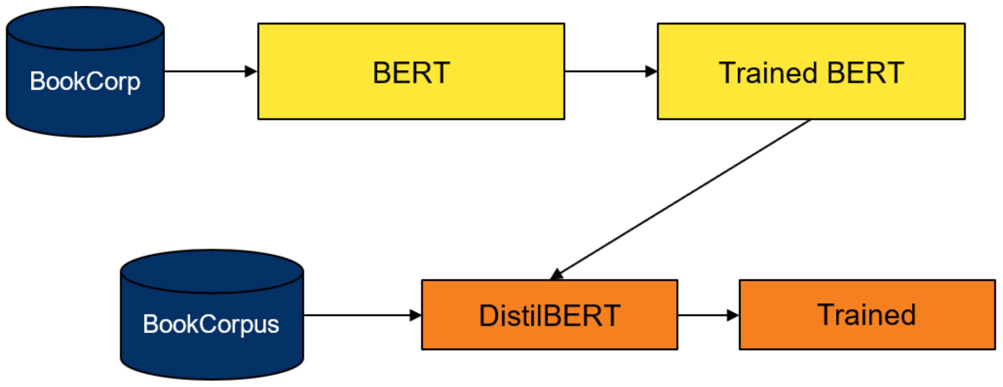


Figure 3.1: The DistilBERT model

smaller, faster, cheaper and lighter"[3]. This model is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than the BERT model, runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark. The distillation process is a compression technique in which a compact model (the student) is trained to reproduce the behaviour of a larger model (the teacher) or an ensemble of models, in DistilBERT the teacher is the BERT model which is bigger in size, with 110 million parameters whereas the distilled version has only 66 million parameters. Training a massive language model (such as BERT) learns millions of latent features which are then picked by the finetuning process improving them over the domain provided (in our case the crypto related discussions). This requires an oversized model, because only a subset of the features are useful for any given task , distillation allows the model to only focus on those features. Finally we used also RoBERTa which is a Transformer model introduced in 2019[4] which is directly inspired from the BERT model, in fact it builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates.

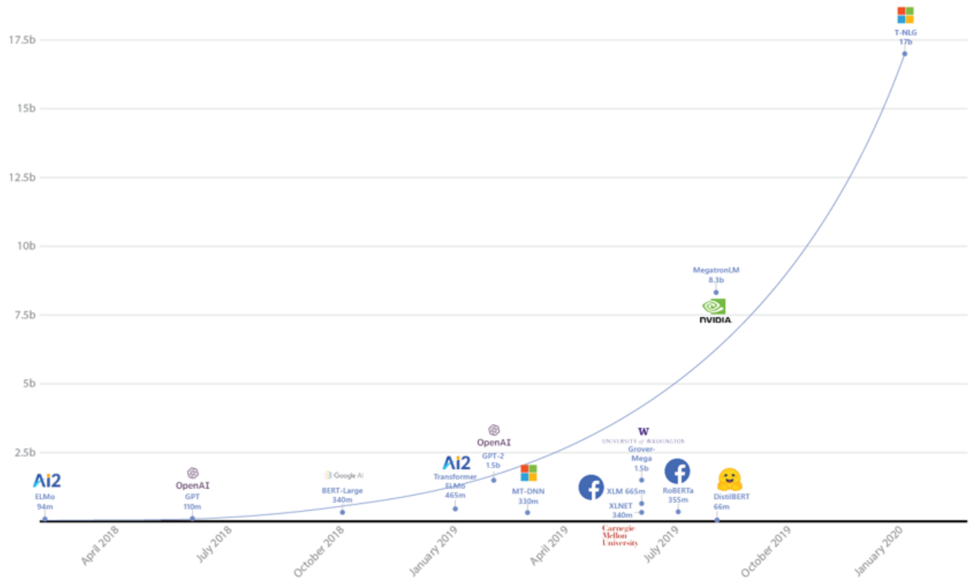


Figure 3.2: Number of parameters of various Transformers models

Chapter 4

Experiments

The frameworks which were used for the training phase of the models is PyTorch and TensorFlow (by using the Keras interface) and the optimizers used were Adam and AdamW [5], which is a slight modification of the adaptive learning algorithm Adam. The first step was to preprocess the dataset by removing irrelevant columns and automatic comments (e.g. comments made by bots) from the dataset, then the dataset needed to be balanced by keeping the same amount of negative, positive and neutral comments and finally the dataset was converted to the `Dataset` format accepted by the HuggingFace model and to the format accepted by torchtext. Furthermore, the dataset had to be tokenized using the tokenizer associated with the models, in this way the data was ready for the experimental phase.

4.1 Using Transformers as classifiers

The first experiment that we tried was:

- compare BERT and DistilBERT as direct classifier using different learning rates and amount of data for the model selection phase
- overall comparison including the implementation of RoBERTa

This aim of this comparison is to test the claim that with Transformers the language modeling performance improves smoothly as we increase model size, training data, and compute resources. Note that in this case the architecture of the models are the default ones provided by HuggingFace implementation. Specifically for the BERT model we have that, after a run over the base BERT Transformer model the output, which is the embedding of the CLS token (also called ”sequence output”) is given to the BertPooler, which is a Feed Forward Neural Network which is composed by a single dense layer and a tanh activation function. The result of the pooler is the pooler output value that will be given to a dropout followed by a dense layer with shape (768,3) which is trained during the fine tuning process to give as output the logits of the 3 possible labels, the process is described in Figure 4.1. The architecture of DistilBERT and RoBERTa are quite similar to the one described for BERT with the exception that they use their base Transformers model respectively and that they take directly the embedding of the CLS token and then process it through a FFNN classifier composed of a dense layer (768 units) and activation function tanh, a dropout and finally an output layer.

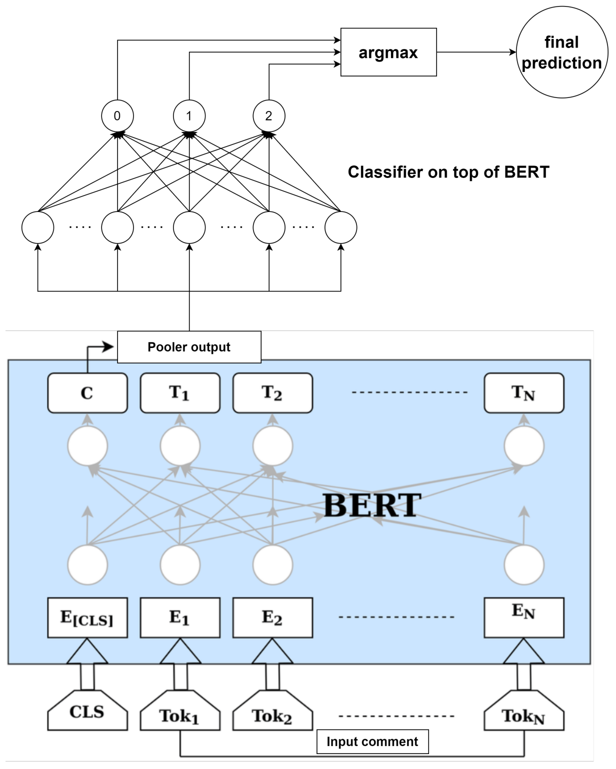


Figure 4.1: Architecture of BERT as classifier

The different amount of data used are described in table 4.1, the partition of the dataset is 70% for the development set (TR+VL) and 30% for the test set.

During the model selection we tested two different learning rates which are: 5e-5 and 3e-5, these values are taken from the BERT’s authors suggestion for the fine tuning for the BERT model, every configuration had 3 epochs for the learning process. As already stated the model has to perform a classification task where the input is a text taken from crypto forums and the output is an integer which could be 0 in case of a negative comment, 1 for a positive comment and 2 for a neutral one. All the experiments have been conducted using a Nvidia RTX 3070 GPU on a local machine with a Jupyter notebook.

Next we started the finetuning process by trying different configurations of learning rates, dataset size and type

Total comments	Development set (TR+VL) (70%)	Test set (30%)
120K	84K	36K
90K	63K	27K
30K	21K	9K
15K	10.5K	4.5K

Table 4.1: Different sizes of dataset

of model, either original BERT or DistilBERT. The results of the model selection are in the table 4.2.

Model	TR Time	Learning Rate	Dataset Size	Accuracy	Precision (avg)	Recall (avg)
DistilBERT	1h20m	5e-5	120K	0.9559	0.9559	0.9559
BERT	2h30m	5e-5	120K	0.9567	0.9568	0.9567
DistilBERT	1h20m	3e-5	120K	0.9538	0.9538	0.9538
BERT	2h30m	3e-5	120K	0.9578	0.9578	0.9578
DistilBERT	1h	5e-5	90K	0.9497	0.9497	0.9496
BERT	2h15m	5e-5	90K	0.9501	0.9501	0.9501
DistilBERT	1h5m	3e-5	90K	0.9507	0.9506	0.9506
BERT	1h55m	3e-5	90K	0.9537	0.9537	0.9537
DistilBERT	20m	5e-5	30K	0.9316	0.9319	0.9316
BERT	40m	5e-5	30K	0.9297	0.9301	0.9297
DistilBERT	20m	3e-5	30K	0.9214	0.9216	0.9214
BERT	40m	3e-5	30K	0.9236	0.9235	0.9236
DistilBERT	10m	5e-5	15K	0.9006	0.9014	0.9005
BERT	20m	5e-5	15K	0.9000	0.9008	0.8999
DistilBERT	10m	3e-5	15K	0.8866	0.8868	0.8865
BERT	20m	3e-5	15K	0.8991	0.8993	0.8989

Table 4.2: Summary of the comparison between BERT and DistilBERT

The results confirmed the authors' claim, which were that:

1. the distilled version of BERT maintains 95% of the performances, in fact in the experiments DistilBERT

maintained the 99,6% of performance w.r.t. the same configuration of the BERT model

2. DistilBERT runs 60% faster than the BERT model, this was also observed in our experiments with a 100% speedup since the DistilBERT version of the model ran roughly in almost half of the time.

We also found out that the performance of the models improved as a greater amount of data was provided, with a limit encountered around 100K comments, we also observed that the two learning rates compared with the same configuration gave out similar results.

Finally, we chose the configuration for the next experiments with the DistilBERT version which was trained with a learning rate of 3e-5 and the dataset of 90K rows, this choice is a trade-off between the accuracy in the validation set (0.9507) and the amount of time required for training and testing. As a further comparison we also tried the implementation of RoBERTa which gave an accuracy of 0.947 on the validation set which is slightly less w.r.t. DistilBERT. The final overall comparison using the dataset of 90K comments is shown in Table 4.3.

Model	**Accuracy (VL set)**
DistilBERT	0.9507
BERT	0.9537
RoBERTa	0.947

Table 4.3: Overall comparison of the Transformers model used as classifiers

4.2 Changing the classifiers on top of the Transformers

The transformers models have a sequence classifier head attached on top of the model which takes as input the sequence output of the base model and performs the classification described. For the DistilBERT and RoBERTa model the default architecture is a classifier with a single hidden layer with 768 (the same dimensionality of the encoder) hidden units with the ReLU activation function, a dropout module (with probability p=0.2) and the final output layer with shape (768,3). The BERT model has a default architecture composed by a dropout module (with probability p=0.1) and a single output layer with the shape (768,3).

A grid search for the classifiers of both models has been performed to find a good configuration for the hyper-parameters, the ones that were changed are the number of hidden layers, the number of hidden units and the type of activation function. The range of values used for the grid search are shown in 4.4.

The results obtained had similar performances among them on the validation set, the best configurations of hyper-parameters coincide with the default given by the HugginFace library except for the DistilBERT model

Hidden layers	Hidden dim	Activation Function
[1,2]	[768, 500, 256]	[ReLU, Tanh, Sigmoid]

Table 4.4: Hyper parameters values used for the grid search

which performs slightly better with hidden dim set to 500 instead of 768. The final best models with the accuracy for the validation set are shown in Table 4.5

Model	# Layers	Hidden Dim	Act. fun.	Dropout	Accuracy (VL)
BERT (default)	1	n.a.	ReLU	0.1	0.956
DistilBERT	2	500	ReLU	0.2	0.951
RoBERTa (default)	2	768	ReLU	0.2	0.947

Table 4.5: Final sequence classifiers chosen

4.3 Using FFNN/CNN/LSTM classifiers with GloVe

An implementation of FFNN, CNN and LSTM models is provided for a further comparison, in this case the model selection was performed on a broader range of hyper-parameters values for the grid search which is shown in 4.6.

Embedding dim	Hidden dim	Num units	Activation Function	Dropout
[50, 200]	[64, 768]	[64, 768]	[ReLU, Tanh, Sigmoid]	[0.1, 0.2]

Table 4.6: Hyper parameters values used for the grid search

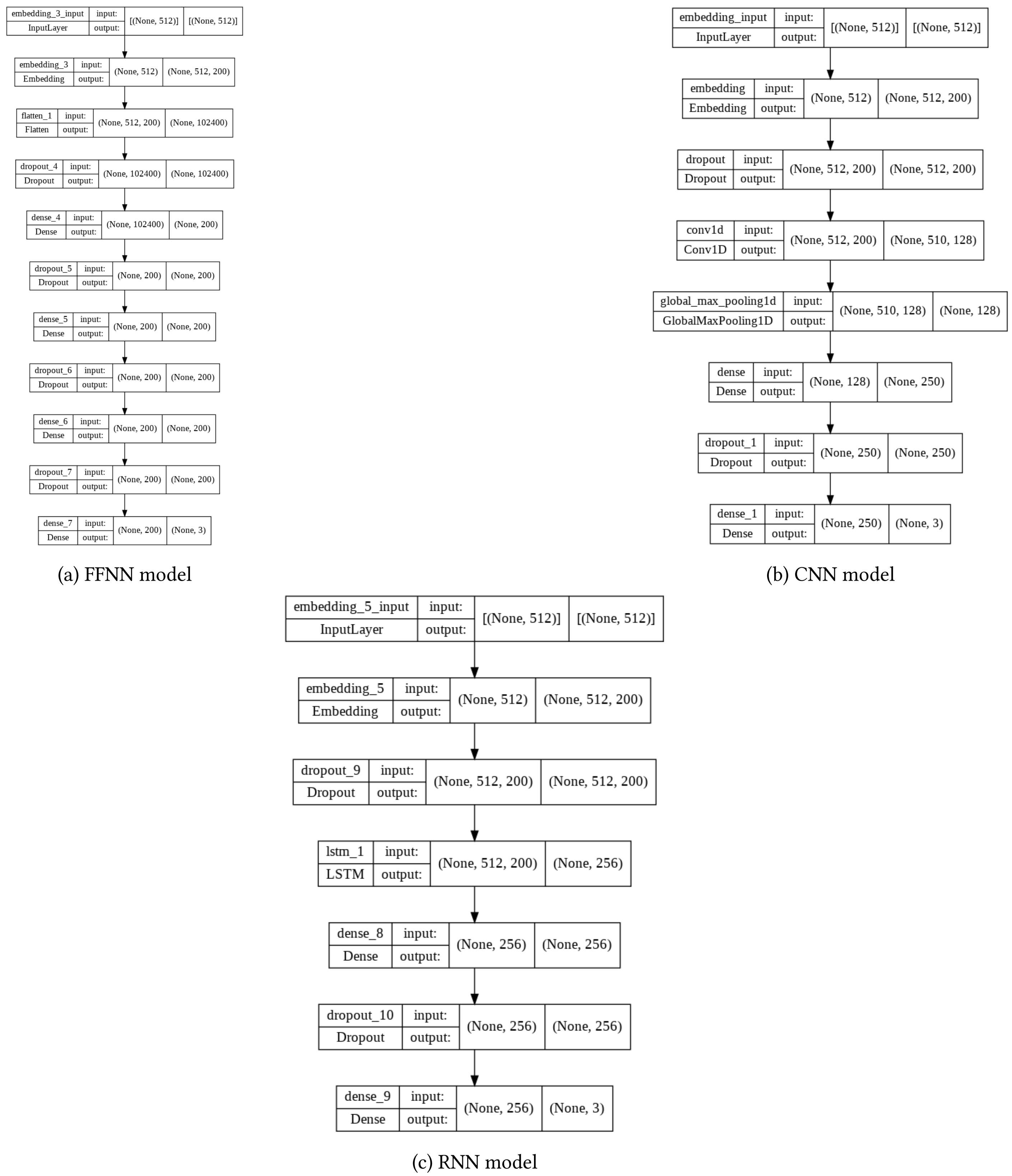
These models have been trained initially with an embedding layer which was learned through the process and then after the model selection phase the same configuration have been tested with the pretrained embedding GloVe. The dimension of the embedding ranged from 50d to 200d, the maximum sentence length of the input sequence was 512 (a padding phase was necessary in the preprocessing phase), the batch size set to 32, the optimizer chosen is Adam with the default learning rate of 0.001. The CNN model after the embedding layer has a dropout of p=0.2, then a 1d convolutional layer with 128 output filters and a kernel size of 3, then a max pooling layer followed by an hidden layer with dimensionality set to 250 and a dropout and finally the output layer with shape (250,3). The FFNN model after the embedding layer has flattening layer which gives the output for the dropout set to p=0.2, then there are 3 hidden layers with dimensionality set to 200 alternated with a dropout, the final output layer has the shape (200,3). The LSTM model has an initial embedding layer followed by a LSTM layer with a hidden dimensionality of 256 and dropout of 0.2 and a dense layer made up by 256 units with ReLU as activation function and a final output layer with shape (256,3).

We also tried a pretrained embedding layer using GloVe (Global Vectors for Word Representation) [6] with various dimensionalities (50d, 100d, 200d) the word representations used are from the Wikipedia 2014 and Gigaword 5 corpus. The results show that using the pre-trained embedding has increased slightly the performance of the classifiers, with the LSTM models the best accuracy on the validation set (0.9322) is comparable with the best one achieved using the Transformers (0.95).

The final hyperparameters for the models are shown in table 4.8, whereas figure 4.2a and figure 4.2b show the graphical representation of the FFNN and CNN models.

Model	**Embedding dim**	**GloVe**	**Hidden Dim**	**Num units**	**Act. fun.**	**Dropout**	**Accuracy (VL)**
LSTM	50	✗	256	256	ReLU	0.2	0.9069
LSTM	100	✗	256	256	ReLU	0.2	0.9129
LSTM	200	✗	256	256	ReLU	0.2	0.9129
LSTM	50	✓	256	250	ReLU	0.2	0.928
LSTM	100	✓	256	250	ReLU	0.2	0.9322
LSTM	200	✓	256	256	ReLU	0.2	0.9292
CNN	50	✗	128	250	ReLU	0.2	0.9010
CNN	100	✗	128	250	ReLU	0.2	0.8974
CNN	200	✗	128	250	ReLU	0.2	0.8968
CNN	50	✓	128	250	ReLU	0.2	0.9040
CNN	100	✓	128	250	ReLU	0.2	0.9080
CNN	200	✓	128	250	ReLU	0.2	0.9006
FNN	50	✗	n.a.	128	ReLU	0.2	0.8384
FNN	100	✗	n.a.	128	ReLU	0.2	0.8435
FNN	200	✗	n.a.	128	ReLU	0.2	0.8320
FNN	50	✓	n.a.	128	ReLU	0.2	0.8335
FNN	100	✓	n.a.	128	ReLU	0.2	0.8358
FNN	200	✓	n.a.	128	ReLU	0.2	0.8235

Table 4.7: Comparison of learning with learned and pretrained embedding


embedding_3_input | InputLayer | input: [(None, 512)] | output: [(None, 512)]
embedding_3 | Embedding | input: (None, 512) | output: (None, 512, 200)
flatten_1 | Flatten | input: (None, 512, 200) | output: (None, 102400)
dropout_4 | Dropout | input: (None, 102400) | output: (None, 102400)
dense_4 | Dense | input: (None, 102400) | output: (None, 200)
dropout_5 | Dropout | input: (None, 200) | output: (None, 200)
dense_5 | Dense | input: (None, 200) | output: (None, 200)
dropout_6 | Dropout | input: (None, 200) | output: (None, 200)
dense_6 | Dense | input: (None, 200) | output: (None, 200)
dropout_7 | Dropout | input: (None, 200) | output: (None, 200)
dense_7 | Dense | input: (None, 200) | output: (None, 3)
embedding_input | InputLayer | input: [(None, 512)] | output: [(None, 512)]
embedding | Embedding | input: (None, 512) | output: (None, 512, 200)
dropout | Dropout | input: (None, 512, 200) | output: (None, 512, 200)
conv1d | Conv1D | input: (None, 512, 200) | output: (None, 510, 128)
global_max_pooling1d | GlobalMaxPooling1D | input: (None, 510, 128) | output: (None, 128)
dense | Dense | input: (None, 128) | output: (None, 250)
dropout_1 | Dropout | input: (None, 250) | output: (None, 250)
dense_1 | Dense | input: (None, 250) | output: (None, 3)
embedding_5_input | InputLayer | input: [(None, 512)] | output: [(None, 512)]
embedding_5 | Embedding | input: (None, 512) | output: (None, 512, 200)
dropout_9 | Dropout | input: (None, 512, 200) | output: (None, 512, 200)
lstm_1 | LSTM | input: (None, 512, 200) | output: (None, 256)
dense_8 | Dense | input: (None, 256) | output: (None, 256)
dropout_10 | Dropout | input: (None, 256) | output: (None, 256)
dense_9 | Dense | input: (None, 256) | output: (None, 3)

(a) FFNN model

(b) CNN model

(c) RNN model

Figure 4.2: Final architecture for the FFNN/CNN/LSTM models.

4.4 Using Transformers as encoders

The focus on Transformers now is shifted on the encoding part of the models. The BERT/DistilBERT/RoBERTa Transformers are used as encoders to get the embedding of each one of the tokens of the input sequence, the difference is that in this case the classifier attached on top will get the embeddings of the tokens and not the pooler output. Therefore, a comparison between the performance of using the pooler output and the embeddings produced by the Transformers is made. The architecture that has been found in the previous experiments for the FFNN/CNN/LSTM models will be kept but the input of these models will have the shape of the input sequence length.

Encoder	**Classifier Type**	**Accuracy (VL)**
BERT	FFNN	0.5124
BERT	CNN	0.8099
BERT	LSTM	0.8656
DistilBERT	FFNN	0.54
DistilBERT	CNN	0.8317
DistilBERT	LSTM	0.8871
RoBERTa	FFNN	0.48
RoBERTa	CNN	0.756
RoBERTa	LSTM	0.8472

Table 4.8: Comparison of models with learned and pretrained embedding

In this case the embeddings of the tokens provided by the Transformers as encoders yield worse performance w.r.t. using the pooler output. The FFNN models performed very poorly with an accuracy around 50%, whereas the best performing classifier is yet again the LSTM model, although the accuracy of around 86% is still far behind from the best result obtained using the pretrained embeddings from GloVe (93%).

The final model chosen is the DistilBERT model chosen in the first experiment (see table 4.2), this one was also tested on a separate test set made of roughly 90K rows, the accuracy was 0.9511 which is a close value to the performance on the validation set.

4.5 Using the model for sentiment analysis

The next experiment was to compare the sentiment of the community with the price of Bitcoin, the hypothesis is that a great fall in price for Bitcoin implies also a fall in the sentiment of the community. In this case we used a second unlabeled dataset with 500.000 comments taken only from the "r/CryptoCurrency" subreddit which is the most used forum of all the mentioned before. The timeframe selected for the comments in the dataset is the month of November 2021, we believe this timeframe to be significant because of multiple sudden crashes of the crypto market which happened throughout the month. The dataset has been preprocessed exactly as before by deleting columns and rows which were irrelevant, in this case there was no "sentiment" column since the dataset has just been collected with no labels. The only difference is that the data has been partitioned according to the days, in such way we obtained multiple groups (dataframes of data grouped by day). For each day we predicted the sentiment of each comment classifying it as positive (1), negative (0) or neutral (2) and then we computed a sentiment score which reflected the overall sentiment of the day as $score_i = \frac{count(pos_i)}{count(neg_i)}$ where $i \in [1, 30]$.

Next we retrieved the price of Bitcoin for the month of November and plot them in a candlestick graph. Since it is believed that markets are strongly affected by the emotion of traders, in a phenomenon known as "emotional trading"[7], these candlesticks show that emotion by visually representing the size of price moves with different colors. The comparison of the candlestick graph with the sentiment trends (see figure 4.3) shows clearly how in relation to the great crashes on days 16, 18 and 26 there is a corresponding drop in the sentiment score with the only exception on the day 11. This further proves the correctness of the approach and how the model is able to capture the sentiment of the community.

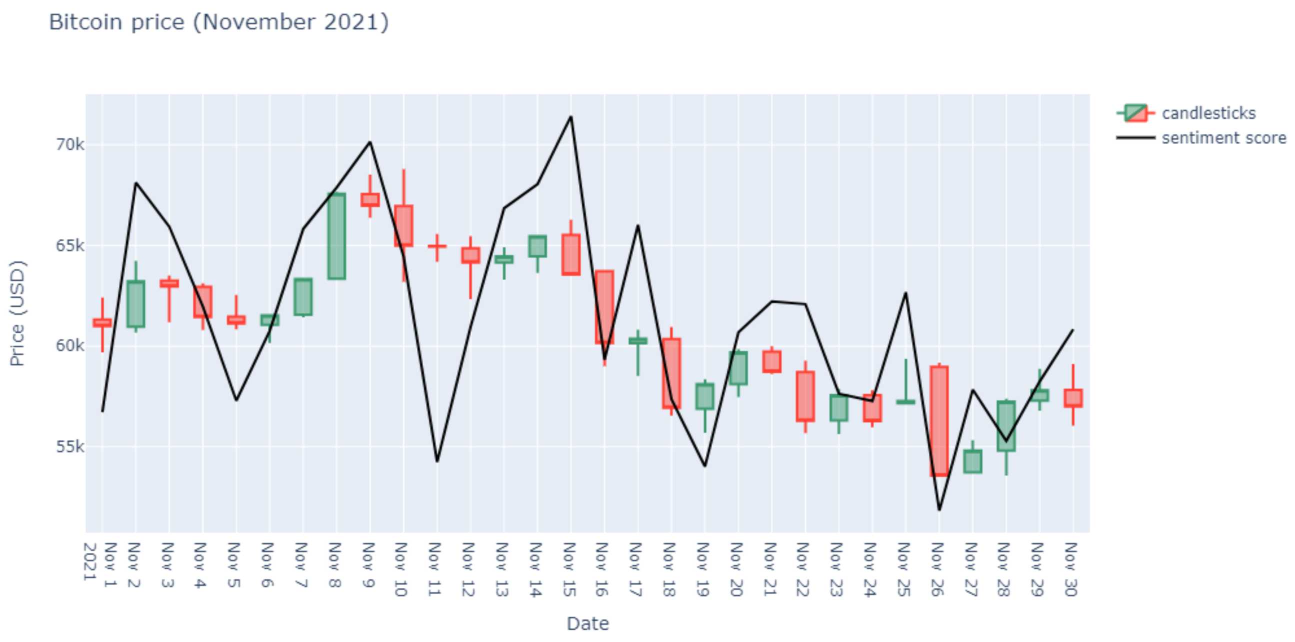


Figure 4.3: Correlation between Bitcoin price (candlesticks) and sentiment score

Chapter 5

Conclusions

We used a labeled dataset of comments made by the crypto community to perform fine tuning on the DistilBERT, BERT and RoBERTa pre-trained models provided by HuggingFace and compared the performance with a standard FFN with multiple layers, with a CNN and a LSTM network. We experimented with different configurations of the models and amount of data, in this way we obtained a final DistilBERT model which achieves an accuracy of 0.9511 on the test set. We observed that in the experiments the DistilBERT version took half of the time and it also had a similar performance with respect to the corresponding BERT model. Further, as expected with a greater amount of data the models performed better on the validation set. Finally we showed a practical example of how to analyze the movements of the crypto market in comparison to the sentiment showing that the approach is well-founded. A further work that could be done is an analysis on a much longer span and possibly on multiple sources of data, this could possibly show if a so called "wisdom of the crowd" could predict rises or falls of the market. More in general the model could help in finding some kind of insight that could be extrapolated from the sentiment of the community.

Bibliography

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[3] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[7] Yongkil Ahn and Dongyeon Kim. Emotional trading in the cryptocurrency market. *Finance Research Letters*, 42:101912, 2021.