Bachelor thesis

Artificial Intelligence

Radboud University 

Exploring the sentiment analysis performance of BERT models on domain specific Twitter data when combined with an intelligent pre-processor

Author:
Laurens Daan Pieter de Bruin
1002199

Supervisor:
Erkan Başar
Communication Sciences, Radboud University
erkan.basar@ru.nl

Second reader:
dr. Roel Boumans
Communication Sciences, Radboud University
roel.boumans@ru.nl

June 19, 2022

Contents

1 Abstract

Bidirectional Encoder Representations from Transformers (BERT) are deep learning language models, used to understand the meaning of language based on context. BERT models are widely used in Natural Language Processing (NLP) research for tasks such as Sentiment Analysis (SA). Social media platforms such as Twitter offer a large quantity of data to run SA on. However Twitter data is very noisy, due to an extensive use of hashtags, emojis, abbreviations, and slang. This noise impairs the performance of BERT models on the SA task. There are BERT models that are pre-trained on Twitter data, however the features labeled as noise are not included in the pre-training. Another problem arises when Tweets contain a high count of niche vocabulary words that did not occur in the pre-training of the BERT models. We propose a fine-tuned pre-trained BERT model combined with a pipeline of pre-processing methods called the "intelligent pre-processor" to overcome the challenges. The intelligent pre-processor is used to translate Twitter noise into a language structure that optimizes the models performance. Domain knowledge is used to help the intelligent pre-processor detect niche vocabulary and replace it with a common language alternatives. The proposed model outperformed the baseline pre-trained Twitter-based BERT model on a sentiment analysis task and confirmed findings of earlier research.

2 Introduction

In the current digital age, social media is an increasingly popular place for people to share their opinions with the rest of the world. As of 2022, there are 217 million active users on Twitter, posting around 500 million tweets every day.

Companies see the vast amount of data produced on Twitter as an opportunity to learn from people's thoughts and opinions. One way to achieve this is with social media monitoring. A company can monitor and store messages posted about a particular subject. Natural language processing models can then be applied to the data to help a company with things as streamlining repetitive tasks, overcoming language barriers, and automating costumer service with chatbots.

A popular NLP task for social media monitoring is sentiment analysis. During this task, a NLP model classifies the sentiment of a given text by assigning a label to it. Labels that are often used are "Positive" and "Negative" [2, 26]. In some cases, the labels "Neutral" and "Mixed" are also used [14, 27]. Sentiment analysis can be used by companies to monitor their social media and detect the opinion of consumers on their products. The ability to detect negative messages about the company can be used to swiftly respond to complaints and become aware of any issues. Another implementation could be for politicians to measure their popularity by running SA on news articles and social media outlets.

The transformer is a novel architecture in the NLP field, that relies on self-attention to compute representations of its input and output. The BERT model builds on the self-attention architecture of transformers and demonstrated great results for downstream tasks including sentiment analysis. BERT is a multi-layer transformer encoder, based on the original transformer implementation and the bidirectional embedding of the ELMo model [22, 32]. The training of BERT models is done in two separate phases. In the first phase called the pre-training, the model gains an understanding of the meaning of language. This is achieved by splitting or "tokenizing" the input sequences into tokens and computing a vector called an embedding that represents the meaning of the token. All tokens in the input sequence are read at the same time, and context left and right of every token is used to encapsulate the meaning of each token in an embedding vector. Other factors such as position in the sequence also influence the embedding. BERT uses the attention mechanism of a transformer encoder to "focus" on the most important parts of the input sequence to base the meaning of a token on. Pre-training uses masked language modelling and next sentence prediction tasks to update the weights in BERTs layers, and the result of this phase is called a base model. A large selection of pre-trained base models can be found online. This reduces the training time of the model during the implementation phase. When we wish to use a BERT model for a downstream task, we only need to update the last layer of the pre-trained BERT model in the fine-tuning phase. Examples of downstream tasks are neural machine translation, question answering, text summarization, and sentiment analysis.

Various pre-trained BERT models exist with each their own unique pre-training datasets. The structure of the language in a dataset can vary a lot depending on the data source. Text collected from Wikipedia often consists of long paragraphs that are formal and objective. Contrary to this, text collected from Twitter has a character limit and is often informal and subjective. It is important for the performance of BERT models, that the pre-training dataset language structure aligns with the language

structure of the dataset on which the model is applied. For example, using a BERT model pre-trained on Twitter data yields greater performance scores in Twitter text mining tasks than using a BERT model trained on Wikipedia data [24].

Although there are BERT models pre-trained on the Twitter language structure, a lot of features that occur in tweets are seen as noise and are removed prior to the training phase [30]. This noise can be in the form of emojis, hashtags, slang, abbreviations and more. However we argue that these features contain information that can improve a models ability to classify sentiment.

An additional problem is that the performance of a BERT model is limited to the vocabulary present in the pre-training dataset. The amount of out-of-vocabulary words may have a negative effect on BERT performance.

Companies often focus on a specific subject when using BERT models (such as a particular product line or service that they provide). The subject specific datasets can contain a niche vocabulary. BERT models are not often pre-trained on any specific domain, so the probability that niche vocabulary words from a certain domain occur in the pre-training dataset is small. Out-of-vocabulary words can only be handled by BERT models if the words contain sub-words. Understanding these niche words is vital, not only for handling the words correctly [10], but also for understanding the sentiment that these words hold.

For this experiment, we used cryptocurrency tweets. Cryptocurrency is a relatively new concept that has gained a lot of traction in the last decade [28]. It bears resemblance to the stock market but with a higher volatility. The crypto market started as a niche which has resulted in a unique vocabulary in the community. We use a BERT model to apply sentiment analysis on the crypto related tweets and classify the tweets using the labels "Positive", "Neutral", and "Negative".

To improve the sentiment analysis performance of a BERT model, we propose an "intelligent pre-processor". The intelligent pre-processor is a pipeline of extensive pre-processing methods that we use to transform features such as hashtags, emojis, slang, and niche vocabulary into a language structure that aligns with pre-training dataset of the pre-trained BERT model. We can apply the intelligent pre-processor on the cryptocurrency Twitter dataset that we collected and use the dataset to fine-tune a pre-trained BERT model for sentiment analysis. With this experiment we attempt to answer the question:

Can the sentiment analysis performance of a fine-tuned pre-trained BERT model be improved using an intelligent pre-processor?

We hypothesize that the intelligent pre-processor has a positive effect on the sentiment analysis performance, as the pre-processor enables the BERT model to successfully process information that is otherwise lost in noise.

2.1 Background

2.1.1 Embedding models

The language modelling field has seen a lot of advances in the past decade. One breakthrough was in 2013, with the development of the word representation model Word2Vec [17]. Word2Vec was shortly followed by the GloVe model in 2014 [21]. These models transform words into vector representations based on semantic value. The vector representations are dimensional coordinates in an embedding space. An embedding space can be pre-trained when used in a computer network. This reduces the time required for computing word vectors. Before the embedding models, one-hot encoding was used to create word vectors. However one-hot encoding is most effective on small cardinality, and there exist so many words that one-hot encoding resulted in sparse vectors. With models such as Word2Vec and GloVe, word vectors could be created more efficiently with lower computational and spacial complexity.

2.1.2 Recurrent Neural Networks and memory models

In 2015, Recurrent Neural Networks (RNNs) became popular for NLP. RNNs created the possibility to process sequences of textual data. Although this was a great step for the NLP field, a problem emerged when sequences became too long. Every input in the sequences would update weights in the network, however the longer the sequence lasted the more the weights of earlier inputs decreased with back propagation. This problem is called the vanishing gradient problem. The problem was partially solved by combining the RNNs with Long Short Term Memory models (LSTM), in which the weights can flow through the model unchanged [34]. A second problem with RNNs, and even more so in LSTMs, is that the models were slow to train. This is because input data is passed sequentially from one state to the next, which does not line up with the parallel processing architecture of modern graphics processing units.

2.1.3 Attention networks

The next step for models were the attention networks. Attention networks use an attention mechanism to dynamically highlight relevant features in input data. One of the attention networks is the Transformer model developed in 2017[32]. A big advantage over RNNs is that the Transformer model supports parallel passing of the input sequence. A transformer model consists of an encoder part and a decoder part. Before the input is fed into the encoder, the input is transformed into vectors with the use of (pre-trained) embeddings. Using positional encoding, the vectors are given information about the position of the word in the input sequence. In the encoder, the transformer uses an attention layer to determine how much focus should be given to each word in the sequence, which is stored in attention vectors. This layer is important to capture contextual relationships between words in a sentence, which can affect the sentiment of a word. The attention vectors are prepared for the next encoder or decoder in a simple feed forward network. The decoder has three main components; an attention block, a feed forward network, and a linear layer. The attention block and feed forward network are similar to those in the encoder. The linear layer expands the dimensions to those of the targeted output, depending on the task of the transformer. A final layer transforms the output into a probability distribution, based on which the most probable outcome is returned by the transformer. Popular tasks for transformer models are language translation, document summarization, and biological sequence analysis.

2.1.4 Bidirectional encoder representations from Transformers

BERT is a multi-layer bidirectional Transformer encoder [6]. It is a deep learning model that is used to learn the meaning of language. BERT models are trained in two phases that can be performed separately in time. In the first phase called the pre-training phase, the model learns the meaning of words from the language in a chosen dataset, Pre-training is done using masked language modelling and next sentence prediction tasks. Pre-training results in a base BERT model which can then be fine-tuned for a specific NLP task using transfer learning. Before a model processes its input, tokenization is performed, splitting input sequences input separate tokens or words. BERT models read all words in text sequences at once. This allows the models to base the meaning of words on context from an entire sentence. The attention mechanism of a transformer is used to "focus" on the most important parts of the input sequence. The meanings of words are represented in vectors called word embeddings. The hidden states of the final layer of the BERT model use the embeddings to compute an output for a particular task. BERT outperformed the state-of-the-art on 11 NLP tasks when the original paper was published.

2.1.5 State-of-the-art

In the current situation, the classic BERT implementation is no longer relevant, however adapted versions are still state-of-the-art for several tasks. RoBERTa is a BERT model with an optimized pre-training approach. ALBERT is a scaled down version of the original BERT implementation that reduces power consumption and increases training speed. StructBERT extends a BERT model by including language structures in pre-training. Finally, DeBERTa builds on RoBERTa with disentangled attention which requires half the data RoBERTa needs for pre-training. Other models that dominate the NLP field at this time are the transformer-based models GPT2 [25] and GPT3 [4] developed by OpenAI. Finally XLNet is a state-of-the-art model which combines the bidirectional capability of BERT and autoregressive technology.

2.2 Related work

Since BERT models can be used for numerous NLP tasks, there are a vast amount of studies using BERT models. Studies on Twitter based sentiment analysis are done on a lot of different domains.

One study explored the Twitter-based sentiment analysis performance of a BERT model on covid-related tweets [30]. No pre-processing steps were performed on the Twitter data, except for simple feature selection. Using Minimum redundancy maximum relevancy (mRMR), the tweet_id, likes, retweets, timestamp, and text_html were selected. In our experiment, we use feature selection to select the tweet_id and the text field of every tweet.

Another study used a BERT model to run sentiment analysis on news articles about the stock market [31]. The news articles were acquired using data crawling. Before feeding the articles to a BERT model, WordPiece was used to tokenize the articles. The performance of BERT was compared to naive Bayes, support vector machines, and TextCNN. BERT outperformed the other models with 15% f1-score difference.

Finally, a study attempted to use sentiment analysis on cryptocurrency tweets to predict the two-hour price of cryptocurrencies [11]. Since the ex-

periment had a goal beyond sentiment analysis, additional to collecting twitter data as done we did in our experiment, the study also collected information about the prices of the currencies. It was not stated that any pre-processing steps were performed on the twitter data.

We observe that extensive pre-processing is not a popular approach when working with BERT models on sentiment analysis. There are several studies that aimed to improve the performance of models on the sentiment analysis task. We encountered two studies that implemented a form of pre-processing to achieve performance improvement when using a language model.

Naseem et al. [19] reported an improvement in performance on a twitter-based sentiment analysis task. The paper emphasizes how noisy twitter data can be. Naseem et al. proposed a new model architecture called "DICE$_T$", which was developed to be able to process the noisy data. DICE$_T$ encodes the representation from a transformer and uses a deep intelligent contextual embedding to elevate the quality of twitter data. The input layer of the model proposed in the paper is referred to as the "Intelligent pre-processor". In this portion of the model, the twitter data is transformed with an extensive pre-processing pipeline tailored to the features that often occur in tweets. We have implemented most of the steps in our version of the pipeline, however there are some differences. Naseem et al. used the Potts tokenizer to translate emoticons, apply word segmentation on hashtags, and replace slang with actual words. Since emoticons are slowly becoming obsolete and replaced with emojis, we ignore emoticons in our experiment and instead focus on the translation of emojis. Additionally, we empirically decided to use a BERT tokenizer instead of the Potts tokenizer since it was optimized to prepare data for a BERT model. Finally, Naseem et al. implemented spelling correction as part of their pre-processing pipeline. Spelling correction is left out in our implementation since it was not feasible in the given time period but has potential in future research and is further discussed in the limitations section. DICE$_T$ was evaluated using three different datasets and considerably outperformed the state-of-the-art in sentiment classification. The effect of the intelligent pre-processor was tested on several different models and datasets and improved the accuracy of models between 0.25% and 2.5%.

A study by Husain et al., explored the effect of pre-processing on offensive language classification using a BERT model [9]. The study used datasets consisting of around 6000 Arabic tweets of varying dialects. Extensive pre-processing was performed including emoji conversion, dialect normalization, word categorization, letter normalization, hashtag segmentation, and miscellaneous steps. Despite the effort, Husain et a. concluded that classifiers based on BERT models do not benefit from pre-processing. The paper did however indicate that the dataset that was used during their experiment, was highly imbalanced and that the results could benefit from validation on a larger dataset.

Although the studies by Naseem et al. and Husain et al. yield different outcomes on the effect of pre-processing on model performance, we can't compare their results. Naseem et al. used English datasets whereas Husain et al. worked with Arabic tweets. Language and culture related differences could have an influence on the effectiveness of pre-processing. To provide comparable research, we explore the effect of the intelligent pre-processor in this experiment.

3 Methods

3.1 Dataset analysis

We summarize the features that are present in the dataset that was used in our experiment. During the dataset analysis we focus on emojis, tags, and slang, as these features influence the effect of pre-processing. The features are analysed for the total dataset as well as separately for every class. The results of the analysis are presented in tables 13 and 14 in appendix B.

In the total dataset, 134 unique emojis were encountered. The dataset contained 938 unique tags. By comparing the words found in the dataset to the dictionary of the WordPiece tokenizer used by BERT models, we detected 143 out-of-vocabulary slang words.

3.1.1 Emoji distribution

In the dataset, the average amount of emojis per tweet was 0.96. The positive class had the most emojis per tweet on average with an average of 1.39. The least amount of emojis per tweet were found in the neutral class.

The most popular emojis in the dataset were 🚀, 📉, and 🔥, which accounted for 17, 7.12, and 5.03 percent of all emojis respectively. The most popular emojis for the separate classes are very true to our expectations. The 🚀 emoji is by far the most popular emoji in the positive class, taking up 28% of all emojis. The rocket represents prices shooting up to the stars and has a strong association to positive sentiment. The neutral class

contains a mixture of 📈 and 📉 but has no emoji that is significantly more used than others. The 📉 emoji is the most popular emoji in the negative class.

3.1.2 Tag distribution

The tweets in the dataset have an average amount of 4.005 tags. The neutral class has the most amount of tags per tweet and the negative class the least. In the total dataset, as well as for all classes, #Bitcoin, #BTC, and #Crypto were the most used tags in tweets contributing to almost a third of all tags.

3.2 Data Gathering

The Twitter API was used to gather tweets based on queries. The queries were used to select only the tweets related to cryptocurrencies. To find tweets with a certain sentiment, we created a list of keywords to expand the query. This list was made as long as possible and was updated iteratively to limit the chance of creating a bias for a certain sentiment. Using the 'Essential' access level of the Twitter API, we could collect one hundred tweets from the last seven days with each pull request. The output of the API is a JSON file with both the text and id of the tweets. The tweet id is later used to match tweets to their labels. Pull requests were done using different queries, on different dates and differing moments of the day. We merged all separate JSON files into one final raw data collection JSON file.

Twitter has a lot of bot accounts that are used to post almost identical messages in large quantities onto the platform. Initially to prevent these duplicate tweets from dominating the sample population, we only stored tweets that did not already occur in the total collection. However an interesting phenomena was found after implementing this. The tweets posted by bots had a designated portion in which a short string was placed which was unique for every post. Most likely, this is done to counter attempts of filtering out duplicate bot tweets. To counter this strategy, we used character-based near duplicate detection. Newly gathered tweets that had a resemblance of 90% or higher to any tweet in the collected dataset, were discarded. To compute the resemblance, we used the formula $2*M/T$. Here T is the total number of elements in both sequences, and M is the number of matches. We collected a total of 17.900 tweets from Twitter. After near duplicate detection, only 6.188 tweets remained in the final dataset for the experiment.

3.3 Annotation

In order to find significant results, we set a goal to collect at least two hundred tweets for every label, six hundred tweets in total. During the data gathering phase we noticed that tweets with positive sentiment were the most common on the crypto side of Twitter. We ran data gathering code in the period between March 22 2022 and April 12 2022. This period was relatively stale for the crypto community, based on the price trends of the largest currencies; Bitcoin, Ethereum, Cardano, Solana, and Ripple. During the first half of this period, the cryptomarket saw a slight increase in prices. In the second half of the gathering period, the prices reset back to their original values from the start of the selected period. Although the second half saw a decrease in prices, the price drop was not large enough to initiate a negative response from the community.

Prior to the annotation process, we created an annotation protocol in which definitions of the labels were set in order to eliminate room for misinterpretation and to keep annotation consistent.

The annotation protocol maps all tweets that infer positive sentiment toward cryptocurrencies or state an increase in price, to the positive label. Tweets that infer negative sentiment toward cryptocurrencies or state a decrease in price, are considered negative tweets. Tweets without sentiment that do not state a direction of a price are assigned a neutral label.

More in depth definitions and examples can be found in tables 15 to 17 of Appendix C.

It is important to note that this annotation protocol is set up from the perspective of a cryptocurrency holder. The holder is only happy with an increase in the price of cryptocurrencies, or if other people express positive sentiment about the currencies. This behavior where a holder aims to sell their assets at a higher price is called longing. Although there also exists a method called shorting with which can be profited from a decrease in price of assets, this method is not considered positive in this experiment.

To manually assign labels to the tweets, LightTag[1] was used. This site requires a csv file as input. Annotation was done in batches of fifty tweets. Batches were sampled from the tweet collection and converted into csv format to upload and annotate on the site. To optimize the efficiency of the annotation process, we ranked the tweets in the collection on usability by a score based on the number of crypto specific terms that were concluded in the posts. By doing this, tweets with a low

[1] LightTag website: lighttag.io, accessed on 16 june 2022.

score that solely included 'crypto' or 'bitcoin' without any other relevance to cryptocurrencies, could be discarded. The output of LightTag is another JSON file containing the tweets and their respective labels. By matching the tweets and labels using tweet ids, we inserted the labels into the tweet collection file.

We continued the annotation of tweets until we achieved the goal of at least 200 for every class. When this was achieved, we had annotated 224 positive tweets, 200 neutral tweets, and 204 negative tweets. In order to have a balanced dataset, we manually selected 24 positive and 4 negative tweets to be discarded from the final dataset. The tweets that were removed had weaker expression of positive and negative sentiment than the ones that were kept. This resulted in 600 tweets for the final dataset that was used during the experiment.

3.4 Pre-processing

Previous studies have found that implementing an intelligent pre-processor and combining it with a BERT model has a positive effect on the sentiment classification performance of the model [19]. Tweets often contain a lot of informal language including abbreviations, slang, and hashtags. A usual pre-processing approach is feature selection, where only certain features are considered and others are labeled as noise and discarded. In this study, we propose an intelligent pre-processor which enriches data by transforming "noise" into usable information.

All steps in the pre-processing pipeline are described in the order that they are applied to the data. This order is important to enrich the Twitter data as much as possible. In some cases, the application of one method is a prerequisite for another further down the pipeline. Altering the order of the methods applied can also lead to incorrect results, as we discuss later in this section.

3.4.1 Text normalization

As the result of the Twitter API sending an HTML request to Twitter and converting the result of that into a JSON format, HTML entities are not represented correctly anymore (examples given in table 1).
These entities need to be restored to text characters before using the collected data. This corruption is a result of the data gathering process, and since this restoring process is not a part of our experimental comparison, we always apply it to the data.

Starting from this point in the process, a distinction is made between a raw data set and a clean data set. The raw data set is finalized after text normalization. The clean data set is a copy of the raw data set, to which the following pre-processing steps are applied.

HTML entity	Character
&amp;	&
&gt;	>
&lt;	<

Table 1: Examples of HTML entities and their respective characters.

3.4.2 Translating mentions and links

Tweets often contain user mentions, a string starting with an at sign or "@". Mentions are used to address a post to a certain account. URLs are also often used in Twitter posts to embed images or to share sites. In the implementation of Naseem et al. [19], the pre-processing layer removes both mentions and hyperlinks. However in our experiment, mentions and links are instead replaced with the placeholders "@USER" and "HTTPURL" respectively. The reason for this, is that this same approach was done during the pre-training of the "BERTweet" model which we use as base model in our experiment [20] [2].

3.4.3 Translating emojis

Emojis are a very important feature for sentiment analysis [29]. As shown by Churches et al. [5], the human brain perceives emoticons, simpler versions of emojis, in the same way as it reacts to real facial expressions of humans. The use of emojis has a wide range of advantages during communication including processing speed of messages, improved expression of feelings, and an intensified perception of negative or positive sentiment [3]. Emojis can also clarify the intended tone of a message [13]. All these advantages of emojis however are visual, and are lost when BERT models process the emojis in textual form. Emojis collected from Twitter with the Twitter API are encoded in a format called Java Escape (see table 2), and BERT is not pre-trained on this format. Because of this, the Java escape strings are considered out-of-vocabulary words. Since the strings do not contain any sub-words, tokenization fails to assign a meaningful vector to the strings.

[2] https://github.com/VinAIResearch/BERTweet

Emoji	Java escape
📉	\ud83d\ude80
🚀	\ud83d\udcc9
😨	\ud83d\ude28

Table 2: Examples of emojis and their respective java escape notation.

To solve this issue, we use the emoji python module to translate emojis to a format that describes the content of an emoji in words (examples in table 3). The emoji module essentially replaces the emojis with a text format that a pre-trained BERT model is familiar with. The textual output can then be correctly tokenized by a BERT model, which improves the ability to learn the sentiment that emojis hold during the fine-tuning phase of the BERT model.

Emoji	Translation
📉	:chart_decreasing:
🚀	:rocket:
😨	:fearful_face:

Table 3: Examples of emojis before and after translation.

This method of translating emojis was also used in the pre-training of the base model for our experiment called "BERTweet" [20]. Following this method during the development of the model that we propose, optimizes the knowledge that our model has learned about the sentiment of translated emojis earlier on.

3.4.4 Word segmentation on hashtags and cashtags

Hashtags (#) and cashtags ($) are used to give context to a tweet and to allow users to find tweets about certain topics. Cashtags are a special type of hashtags dedicated to currencies. An example of finding tweets of a certain topic is how, among others, we used the hashtag "crypto" during the data gathering phase of our experiment to find tweets about cryptocurrency. A characteristic feature of hashtags is that they do not contain spaces. This means that it is possible that words in hashtags are not recognised because they are concatenated to other words in a long string. There are a lot of studies that underline the importance of hashtags in sentiment analysis [16, 18, 33]. To optimize the use of the sentiment embedded in the tags, we apply word segmentation on hashtags and cashtags. The first character of the tag is removed and the rest of the string is split up into separate words. The word segmentation algorithm that we used, made use of a unigram, a list of words similar to a dictionary. This unigram is used to detect English words. We updated the unigram to include abbreviations and slang from the niche vocabulary of the crypto community. Without adaptation of the unigram, segmentation of the tags is ambiguous due to the number of abbreviations and slang that make up the tags along side "normal" words. Incorrect segmentation can distort the content of the tags, resulting in a loss of information.

3.4.5 Processing abbreviations and slang

In the next step of preprocessing, the abbreviations and slang that are in the tweets, and that resulted from the segmentation of tags, are translated into more common language representations. All cryptocurrencies have a name and an abbreviated version similar to fiat currencies as seen in table 4.

Currency	Abbreviation
Bitcoin	BTC
Ethereum	ETH
Ripple	XRP
Euro	EUR
US Dollar	USD

Table 4: Examples of cryptocurrencies and their respective abbreviation.

We exchange these abbreviations with the spelled out name of the currency to decrease the number of unique terms in tweets that a model needs to learn from and work with. Slang is not included in the tokenizer's unigram used by the BERTweet model and these words are therefore exchanged by more common words. Although BERT models can use sub-word tokenizers to solve out-of-vocabulary tasks [15], almost all slang that we encountered in the collected data did not contain any sub-words to tokenize and thus sub-word tokenization would not suffice.

3.4.6 Restore spacing

During pre-processing, sub-strings are replaced several times. Since some target sub-strings are only a few characters long, such as in the case of abbreviations, we place spaces around a sub-string every time one is placed. This is done to avoid replacement of portions of words that match with the targeted sub-strings. An example would be the cryptocurrency Ethereum. The abbreviated form of Ethereum is eth. To avoid "Ethereum" being

changed to “Ethereumereum”, we set “ eth ” as target sub-string. This method is effective, however after several iterations of replacements the final result contains a lot of excess space characters. Studies show the importance of white space removal, to enhance performance in sentiment analysis [1, 8]. Following their findings, we reduce all white space sequences to a single space.

3.5 BERT Fine-tuning

After pre-processing, the clean data set is ready to be used for the fine-tuning of a custom crypto-domain sentiment analysis BERT model. As base model, we use a model called “bertweet-sentiment-analysis”, pre-trained on a large set of tweets [3] [23]. Prior to fine-tuning, a tokenizer is defined that converts tweets into a list of tokens and computes a numerical representation for every token. To optimize the performance of the tokenizer, we add domain-specific tokens to the token embeddings. Adding these tokens allows the tokenizer to correctly recognize all words in our dataset and prepare it for the BERT model.

Hyperparameters used during every fine-tuning session in the experiment are shown in table 5.

Learning rate	2e-5
Batch size	4
Number of epochs	4

Table 5: Hyperparameters used during the fine-tuning of the models.

The learning rate is the default value of the trainer from transformers. We reduced batch size from the default 16 to 4 to reduce training time. We empirically decided to increase the number of epochs to avoid underfitting.

3.6 Evaluation

After the training phase, we evaluated the performances of a pre-trained BERT model fine-tuned on the raw dataset, as well as that of a pre-trained BERT model fine-tuned on the clean dataset. We also evaluate the sentiment analysis performance of the base model.

The performance of the base model was measured by applying SA on the cryptocurrency data set, and comparing the predicted labels with the true labels, that were assigned to the tweets during the annotation phase. This approach was done to both the raw dataset and the clean data set. There is no need for extra validation steps to avoid overfitting since the cryptocurrency dataset does not intersect with the dataset that was used during the pre-training of the base BERT model.

To measure the performance of the fine-tuned models, 10-fold cross-validation was used to avoid overfitting on the training data. Every fold of the model produced ten percent of the labels which were all concatenated to reconstruct a list of 600 prediction labels. These labels were then compared to the true labels to compute the models' performances.

As metric for performance the f1-score is calculated. The f1-score is able to create a clear insight to how well a model performs on a task. The f1-score is the harmonic mean of the recall and precision of a model. Precision represents how much of the data that was given a certain label was correctly classified and is useful for measuring the exactness of the model. Recall represents how much of the data is assigned to their true label and is useful for measuring the completeness of the model.

3.7 Experimental set-up

To test the hypothesis of this experiment, we compare four versions of a pre-trained BERT model. The models differ in whether they are fine-tuned and whether pre-processing is applied to their input. Performances of the models are compared to each other using the f1-score metric. An overview of the experimental set-up is seen in fig. 1 in appendix A.

3.7.1 Environment

We used Python 3.6 during this experiment. Code for data gathering and pre-processing was written in the PyCharm development environment. Code for implementing and fine-tuning the pre-trained BERT model was created and executed using Google Colab [4] and HuggingFace [5]. The project is published as open source on GitHub [6].

4 Results

We compared four different models, for readability we refer to the models as model 1 to 4. Model 1 is the base model, a pre-trained BERT model without fine-tuning or pre-processing. Model 2 is

[3] https://huggingface.co/finiteautomata/bertweet-base-sentiment-analysis

[4] Google Colab: colab.research.google.com, accessed 16 june 2022.

[5] HuggingFace: huggingface.co, accessed 16 june 2022

[6] https://github.com/laurens88/ThesisProject

the base model that performs sentiment analysis on data that was processed by the intelligent pre-processor. Model 3 is base model that was fine-tuned by and tested on the raw dataset. Model 4 is the proposed model, a pre-trained BERT model fine-tuned by and tested on the clean dataset provided by the intelligent pre-processor.

4.1 F1-score

Looking at the f1-scores of the models in table 6, the models that were fine-tuned on the cryptocurrency dataset returned higher f1-scores than the baseline BERT models. In both base and proposed model cases, higher f1-scores were achieved when applying sentiment analysis on the clean dataset compared to the raw dataset. The difference in f1-scores between no pre-processing and pre-processing is slightly greater for the fine-tuned models than for the base models. Comparing model 1 and 2 shows that pre-processing increases the f1-score of a base model with 2.1%. Comparing model 3 and 4 shows that pre-processing increases the f1-score of the fine-tuned model with 3.5%.

	F1-score
Model 1	51.3%
Model 2	53.2%
Model 3	85.4%
Model 4	89.1%

Table 6: F1-scores of models 1 to 4.

4.2 Confusion matrices

The confusion matrices in tables 7 to 10, show the distribution of predicted versus true labels of the tweets in the dataset. Model 1 assigns 401 out of 600 tweets to the neutral class.

Model 2 assigns less tweets to the neutral class and yields more true positives.

In the confusion matrix of model 3, 512 out of 600 are on the true positive diagonal. The model performs best on the negative class and worst on the neutral class, however the difference is small (difference of 13 tweets).

Finally, model 4 assigns 535 out of 600 labels correctly. The model performs best on the negative class, predicting 192 true negatives out of 200. For both model 3 and 4, the confusion matrices show that false predictions for the positive and negative class leak toward the neutral class.

		Predicted label		
		NEG	**NEU**	**POS**
True label	**NEG**	69	122	9
	NEU	14	172	14
	POS	17	107	76

Table 7: Confusion matrix of model 1.

		Predicted label		
		NEG	**NEU**	**POS**
True label	**NEG**	67	123	10
	NEU	13	172	15
	POS	16	96	88

Table 8: Confusion matrix of model 2.

		Predicted label		
		NEG	**NEU**	**POS**
True label	**NEG**	177	12	11
	NEU	22	164	14
	POS	8	21	171

Table 9: Confusion matrix of model 3.

		Predicted label		
		NEG	**NEU**	**POS**
True label	**NEG**	192	6	2
	NEU	7	168	25
	POS	10	15	175

Table 10: Confusion matrix of model 4.

4.3 Precision

The precision scores for models 1 to 4 for every class are presented in table 11.

Model 1 has a high precision of 0.86 for the neutral class compared to the other classes with precision scores between 0.3 and 0.4.

Model 2 has the same precision score for the negative and neutral class, and achieves a slightly higher score for the positive class. Precision is increased by 0.06 compared to model 1.

Model 3 scores between 0.8 and 0.9 on precision for all classes. The negative class has the highest score with 0.89 and neutral the lowest with 0.82. Neutral precision has decreased compared to the base model, dropping from 0.86 to 0.82. Precision on the other classes has doubled.

Finally, model 4 has increased precision for all classes compared to model 3. Precision for the negative class has reached 0.96, the neutral class has a precision of 0.84, and the positive class scores 0.88.

Precision	Negative	Neutral	Positive
Model 1	0.34	0.86	0.38
Model 2	0.34	0.86	0.44
Model 3	0.89	0.82	0.85
Model 4	0.96	0.84	0.88

Table 11: Precision scores of models 1 to 4 for every class.

4.4 Recall

Recall scores for models 1 to 4 are shown in table 12.

Recall scores of model 1 are highest for the positive and negative class scoring 0.77 and 0.69 respectively. Neutral class recall is 0.43.

The recall scores of model 2 are exactly 0.01 higher than those of model 1 for all classes.

Model 3 scores almost twice as higher on recall for the neutral class, and recall scores for the negative and positive class increase with 0.19 and 0.09 respectively.

Compared to model 3, model 4 does not improve recall score for the positive class and remains at 0.87. Scores for the negative and neutral class increase with 0.06. This results in a 0.92 score on the negative class, 0.89 on the neutral class, and 0.87 for the positive class.

Recall	Negative	Neutral	Positive
Model 1	0.69	0.43	0.77
Model 2	0.7	0.44	0.78
Model 3	0.86	0.83	0.87
Model 4	0.92	0.89	0.87

Table 12: Recall scores of models 1 to 4 for every class.

5 Discussion

5.1 Experiment assessment

Based on the f1-scores of the different models, it is clear that both fine-tuning as well as the implementation of the intelligent pre-processor have a positive effect on the performance of BERT models on the SA task.

The positive effect of fine-tuning a model on a specific domain was expected. Preparing a BERT model for a specific task increases the performance, because the weights of the base model are updated to account for the characteristics of the domain specific data.

We also expected that implementing the intelligent pre-processor would have a positive effect on the sentiment analysis performance.

The confusion matrices in tables 7 to 10, show the distribution of predicted versus true labels of the tweets in the dataset. Model 1 assigns a little over two thirds of all tweets to the neutral class. Because so many tweets are assigned the neutral label, it is inevitable that there are a lot of true neutral tweets among them. This results in a relatively high precision score for the neutral class compared to the other classes as seen in table 11. However this bias toward the neutral class also causes a decrease in precision for the positive and negative class and decreases the recall score of the neutral class. This is because the bias causes a lot of false neutral predictions.

Comparing model 1 with model 3, we see that fine-tuning improves the distribution of assigned labels. In the confusion matrix of model 3, a large majority are now on the true prediction diagonal instead of the predicted neutral column. This improved distribution of predicted labels slightly decreases the precision of the neutral class but greatly increases the recall score for the neutral class as the number of false neutral predictions drops significantly (from 229 to 33). The improved distribution also improves both the precision and recall scores of the negative and positive classes.

Applying the intelligent pre-processor on the base model improves the classification performance of the positive class, mostly by increasing the precision score. The pre-processor has a minimal effect on the classification performance of the negative and neutral class.

In contrast, applying the pre-processor on the fine-tuned model improves the performance for all classes. The precision score increases for all three classes, and recall scores increase for the negative and neutral class.

The combination of fine-tuning and pre-processing has a greater effect on model performance (37.8% increase in f1-score) than the sum of effects found when fine-tuning or pre-processing was implemented separately (34.1% + 1.9% = 36% increase in f1-score). This means that the methods magnify the positive effect of each other.

From the comparison of model 3 and 4, it can be observed that the negative class classification improves the most from all classes. The intelligent pre-processor improves both the precision and recall scores for the negative class, whereas for the neural class most improvement is seen in the recall score. The classification performance of the positive class is only improved in the proposed model in terms of precision score.

Dataset analysis showed that the positive class contains the most emojis per tweet on average. This

high amount of emojis could explain the greater influence of the proposed intelligent pre-processor on the precision score for the positive class for the base models. In the comparison of model 3 and 4 this effect is not seen anymore, suggesting that the translation of emojis had less effect on fine-tuned models. This could be explained by the fact that fine-tuning on an embedding for one emoji could yield more consistent results, than fine-tuning on multiple words describe the contents of an emoji and possibly disrupt the structure of a sentence.

The improvement in performance from our implementation of the intelligent pre-processor confirms the results found by Naseem et al. [19].

5.2 Limitations

5.2.1 Data gathering

We used the Twitter API to collect tweets for our experiment. The API has three levels of access; "Essential", "Elevated", and "Academic research". With every level of access, the amount of possible features increases. During the experiment it was only possible to use the essential level of the API, since the academic research level is not available for bachelor students. This introduced some limitations on the data gathering process.

To find tweets relevant to the crypto domain, we used a list of hashtags. The API would then select all tweets that contained one or more of the hashtags in the list. Using cashtags in queries is not possible using the essential level API. This might have resulted in finding less relevant tweets than could have been found if cashtags were available in queries.

Another limitation of the API access level was that we were only able to collect tweets that were posted in the last seven days. Because the sentiment of crypto tweets on twitter is related to the current situation of the crypto market, the seven day limit makes it more difficult to collect tweets of different sentiments. With access to a higher API level, we could specify a period in time to collect tweets from. This would help to collect negative tweets by sampling from a period around a crash of the market, and collect positive tweets by collecting tweets from a period where currencies rose in price. As an alternative to this, we tailored queries to find tweets of a certain sentiment. For example to find negative tweets about cryptocurrencies, we appended a list of keywords associated with negative sentiment to the query. This list however, is limited to the domain knowledge of the experimenter and if this list of possible keywords is too short it can introduce a bias. For example, if only the keywords "crash" and "fear" are included in the list, a bias is created that negative tweets must include either one of those terms. During the experiment we iteratively updated the keyword list for all individual sentiments to avoid or minimize bias. However, in future research it would be better to avoid this method and sample from different time periods with a higher level API instead.

5.2.2 Spelling correction

Spelling errors are a very common occurrence. However especially in an informal setting such as Twitter, several types of spelling errors can be detected. Based on results found during an attempt to implement spelling correction, as well as similar findings by Kaufmann [12], classic spelling correction is not compatible with social media data. The problem with classic spelling correction is that it relies on a database of existing words, and this database does not contain all novel words that are used on Twitter. This can result in incorrect detection of misspellings, as we found when the python module "TextBlob" was implemented for spelling correction during the development phase of our research. For example, when applying spelling correction on "Bitcoin" which is spelled correctly, the corrected output would be "Billion". Opposed to unintentional errors, Twitter also contains a lot of so called "intentional errors". An example for this is phonetic spelling such as "ur" instead of "your". "ur" is grammatically seen as an error but can be semantically congruent depending on the context. Kaufmann has successfully developed a syntactic normalization process that dealt with the challenges mentioned before, increasing BLEU scores by 18% [12].

It was not feasible to implement a version of Kaufmann's normalization process during the time period of our research, and the spelling correcter from TextBlob did not have a positive effect on the data. However based on the high number of misspellings in Twitter data, and the positive results of successful spelling correction as a pre-processing method [7], we suspect that spelling correction has potential and leave this step of the pipeline to future research.

5.3 Future research

5.3.1 Prices of cryptocurrencies

In the current implementation, knowledge is limited to the text in a tweet. If a tweet does not contain any sentiment or indications on the direction of the price of a cryptocurrency, the tweet is

classified as neutral. However this is not always correct. For example, a tweet could state "I believe the price of bitcoin will go back to 40.000 dollars". This tweet indicates that the price of bitcoin will change, but because there is no knowledge about the price of bitcoin at the time of the tweet, it is unknown whether the change is a positive change or a negative change. In future research this could be improved by storing the prices of cryptocurrencies mentioned in a tweet together with the text of the tweet. The prices would be based on the date and time of the tweet.

5.3.2 Expanding to emotion detection

Only using positive, neutral, and negative sentiments is a simplification of human emotion. As tweets about cryptocurrencies can be emotion-packed, research could be expanded toward emotion detection. This approach could provide a more accurate fit to human expression.

5.3.3 Inclusion of the mixed sentiment label

There is a section of the cryptocurrency community that considers the success and value of currencies by comparing them with other currencies. These tweets often list pros and cons such that they contain both positive and negative sentiment. In our experiment these tweets are considered out of scope as they do not fit to one of the positive or negative label. Using the "mixed" label allows for the inclusion of these tweets, and would improve the proposed models ability to be used in a real life application.

6 Conclusions

In this study, we explored the effects of an intelligent pre-processor on the sentiment analysis performance of a pre-trained BERT model. Data from Twitter often contains noise in the form of hashtags, slang, and abbreviations. This noise can impair the performance of a pre-trained BERT model. We propose a fine-tuned pre-trained BERT model combined with an intelligent pre-processor that translates Twitter noise into an optimized language structure. The proposed model outperformed a baseline model without the intelligent pre-processor by 3.5% on the f1-score metric. This result confirms the 0.25-2.5% accuracy improvement found earlier by Naseem et al. Based on the results we conclude that combining a fine-tuned pre-trained BERT model with an intelligent pre-processor has a positive effect on the sentiment analysis performance.

References

[1] Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi. A comparison between preprocessing techniques for sentiment analysis in twitter. In *KDWeb*, 2016.

[2] Seyed-Ali Bahrainian and Andreas Dengel. Sentiment analysis using sentiment features. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 3, pages 26–29, 2013.

[3] Isabelle Boutet, Megan LeBlanc, Justin A. Chamberland, and Charles A. Collin. Emojis influence emotional communication, social attributions, and information processing. *Computers in Human Behavior*, 119:106722, 2021.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[5] Owen Churches, Mike Nicholls, Myra Thiessen, Mark Kohler, and Hannah Keage. Emoticons in mind: An event-related potential study. *Social neuroscience*, 9, 01 2014.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North*, 2019.

[7] Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. The influence of preprocessing on

text classification using a bag-of-words representation. *PLOS ONE*, 15(5), 2020.

[8] Emma Haddi, Xiaohui Liu, and Yong Shi. The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, 17:26–32, 2013. First International Conference on Information Technology and Quantitative Management.

[9] Fatemah Husain and Özlem Uzuner. Fine-tuning approach for arabic offensive language detection system: Bert-based model. *CoRR*, abs/2203.03542, 2022.

[10] Yuuki Iwasaki, Akihiro Yamashita, Yoko Konno, and Katsushi Matsubayashi. Japanese abstractive text summarization using bert. In *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 1–5, 2019.

[11] Arti Jain, Shashank Tripathi, Harsh Dhar Dwivedi, and Pranav Saxena. Forecasting price of cryptocurrencies using tweets sentiment analysis. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*, pages 1–7, 2018.

[12] Max Kaufmann and Jugal Kalita. Syntactic normalization of twitter messages. In *International conference on natural language processing, Kharagpur, India*, volume 16, 2010.

[13] Linda K. Kaye, Stephanie A. Malone, and Helen J. Wall. Emojis: Insights, affordances, and possibilities for psychological science. *Trends in Cognitive Sciences*, 21(2):66–68, 2017.

[14] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the international AAAI conference on web and social media*, volume 5, pages 538–541, 2011.

[15] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics, 2018.

[16] Zongyang Ma, Aixin Sun, Quan Yuan, and Gao Cong. Tagging your tweets: A probabilistic modeling of hashtag annotation in twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, page 999–1008, New York, NY, USA, 2014. Association for Computing Machinery.

[17] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[18] Saif M Mohammad and Svetlana Kiritchenko. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2):301–326, 2015.

[19] Usman Naseem, Imran Razzak, Katarzyna Musial, and Muhammad Imran. Transformer based deep intelligent contextual embedding for twitter sentiment analysis. *Future Generation Computer Systems*, 113:58–69, 2020.

[20] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. Bertweet: A pre-trained language model for english tweets. *CoRR*, abs/2005.10200, 2020.

[21] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 02 2018.

[23] Juan Manuel Pérez, Juan Carlos Giudici, and Franco Luque. pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks, 2021.

[24] Mohiuddin Md Abdul Qudar and Vijay Mago. Tweetbert: A pretrained language representation model for twitter text analysis. *CoRR*, abs/2010.11091, 2020.

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al.

Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[26] A Pappu Rajan and SP Victor. Web sentiment analysis for scoring positive or negative words using tweeter data. *International Journal of Computer Applications*, 96(6), 2014.

[27] Eshrag Refaee and Verena Rieser. An arabic twitter corpus for subjectivity and sentiment analysis. In *LREC*, pages 2268–2273, 2014.

[28] Ludwig Christian Schaupp and Mackenzie Festa. Cryptocurrency adoption and the road to regulation. In *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age*, dg.o '18, New York, NY, USA, 2018. Association for Computing Machinery.

[29] Mohammed Shiha and Serkan Ayvaz. The effects of emoji in sentiment analysis. *Int. J. Comput. Electr. Eng.(IJCEE.)*, 9(1):360–369, 2017.

[30] Mrityunjay Singh, Amit Kumar Jakhar, and Shivam Pandey. Sentiment analysis on the impact of coronavirus in social life using the bert model. *Social Network Analysis and Mining*, 11(1), 2021.

[31] Matheus Gomes Sousa, Kenzo Sakiyama, Lucas de Souza Rodrigues, Pedro Henrique Moraes, Eraldo Rezende Fernandes, and Edson Takashi Matsubara. Bert for stock market sentiment analysis. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1597–1601, 2019.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[33] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, page 1031–1040, New York, NY, USA, 2011. Association for Computing Machinery.

[34] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

A Experimental set-up

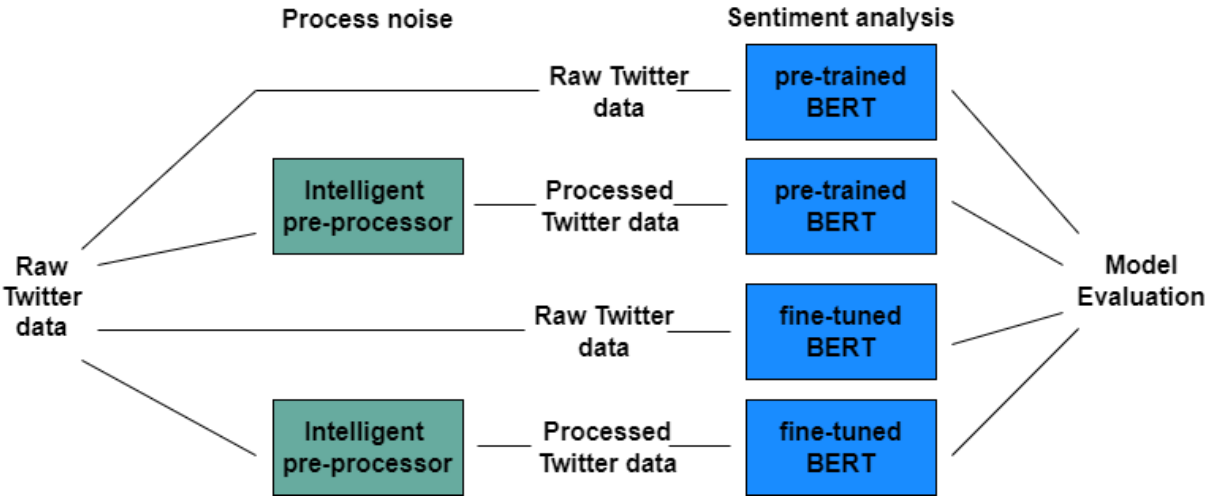

Figure 1: Overview of the four different models that we compare in this experiment.

B Dataset Analysis

	Positive	**Neutral**	**Negative**	**Total dataset**
Avg # emojis per tweet	1.39	0.955	1.095	0.96
Top emojis used (%)	🚀 (28.06) 🔥 (6.12) 📈 (4.32)	🚀 (9.95) 📉 (8.9) 📈 (6.81)	📉 (10) ➡️ (8.68) 🚨 (5.02)	🚀 (17) 📉 (7.12) 🔥 (5.03)

Table 13: Analysis of the emojis used in the dataset of the experiment.

	Positive	**Neutral**	**Negative**	**Total dataset**
Avg # tags per tweet	3.93	4.23	3.855	4.005
Top tags used (%)	#BTC (6.36) #Bitcoin (5.73) #crypto (2.67)	#Bitcoin (12.17) #BTC (7.33) #crypto (4.37)	#Bitcoin (9.6) #BTC (9.1) $BTC (4.8)	#Bitcoin (8.49) #BTC (7.57) $BTC (3.79)

Table 14: Analysis of the tags used in the dataset of the experiment.

C Annotation Protocol

Positive
Possible definitions
1: The tweet infers an increase in price for one or more cryptocurrencies.
2: The tweet contains positive sentiment toward cryptocurrency.
Examples from dataset
1: BNB Price: $445.62 Change in 1h: +0.5126% Market cap: $73 billion Ranking: 4
2: #XRP is getting ready for the biggest pumping!! 🚀 🔥 #crypto

Table 15: Annotation guide for positive tweets.

Neutral
Possible definitions
1: The tweet states a price for one or more cryptocurrencies without an indication of the direction of the price.
2: The tweet does not contain any sentiment toward cryptocurrency.
Examples from dataset
1: #BTC price ➡️ $43437 2022-04-07 17:45
2: Current state of the #Bitcoin and #Crypto markets https://t.co/V394nbd5Rd32

Table 16: Annotation guide for neutral tweets.

Negative
Possible definitions
1: The tweet infers a decrease in price for one or more cryptocurrencies.
2: The tweet contains negative sentiment toward cryptocurrency.
Examples from dataset
1: $BTC #BTCUSD #altcoins #Crypto 🚨 🚨 🚨 🚨 MARKET IS UNSTABLE. BTC IS GOING TO FALL BELOW 45K
2: Lost everything. 😢 #btc #eth #crypto

Table 17: Annotation guide for negative tweets.