Sentiment Analysis of Filipino Cryptocurrency Tweets using the combination of LSTM-MCNN and Naive Bayes

Greo Ochoa

Gabriel Quibod

Rudolph Christian Razul

ATENEO DE DAVAO UNIVERSITY

SCHOOL OF ARTS AND SCIENCES

DAVAO CITY

MARCH 2022

Table of Contents

List of Figures

Figure 22: Decoder layer.

Figure 23: Sample code on the implementation of the decoder model.

Figure 24: Sample code for training the decoder model.

Figure 25: Sample code to initialize the Pipeline with varying n-gram features.

Figure 26: Sample code to train the classifier.

Figure 27: Accuracy score formula.

Figure 28: Script to determine accuracy score and classification report.

Chapter 1

Introduction

1.1 Background of the Problem

A cryptocurrency is a form of digital currency that utilizes blockchain technology: a decentralized and cryptographic technology that warrants the digitalization of transactions, thereby gaining the trust of its users. It replaces the traditional idea of governments producing currency and an intermediary third-party that verifies transactions with its digitally inclusive methodology. The idea began with a single cryptocurrency, Bitcoin (2009), and has since exploded in interest and thus led to the development of other cryptocurrencies in the market. The growth and interest of crypto are attributed to news stories and articles that reported unprecedented returns fueling a modern gold rush. Given that crypto is relatively new, global regulations on crypto are still limited. In combination with high popularity and the lack of a proper regulatory guarantor, the crypto market is ridiculously volatile that even the CEO of Binance(largest global cryptocurrency exchange service), Zhao Changpeng (2021), called it a "*wild west that needs a sheriff*."

The extreme volatility of the cryptocurrency market is greatly attributed to news and posts on social media. Due to cryptocurrency still being a relatively new and mysterious topic, traditional news outlets do not continuously broadcast new information regarding crypto. As a result, crypto investors rely on social media as the primary information source. *Twitter* is a micro-blogging website/service popular and widely used source for cryptocurrency information. Statista.com [14] estimates around 10.2 million active users in the Philippines alone. Not only is it able to provide live updates on crypto news, it can also be utilized as a source for emotional data, as users and investors frequently express their sentiment through tweets. Studies of behavioral economics have found that sentiments and emotions can profoundly affect individual behavior and decision-making.

Utilizing the vast amount of data from Twitter containing sentiment tweets from investors, this study aims to give insight into the perspective that Filipinos currently have regarding cryptocurrency. This study is unique in many ways. First, it tackles a specific demographic of Filipino tweets. Secondly, it will produce a sentiment analysis tool specifically for crypto-related tweets in both Filipino and English. Moreover, many previous studies have only focused on Bitcoin alone. This study is one of the few that will work on cryptocurrency in general and go beyond the scope of Bitcoin.

1.2 Objectives of the Study

The main objective of this study is to understand the conceptions of Filipinos about the topic of cryptocurrency with the application of deep learning techniques and sentiment analysis on a dataset gathered from Twitter tweets. On the topic of gathering tweets, this study also aims to explain how Twitter is utilized to gather specific tweets about cryptocurrency, specifically on the demographic of Filipino tweets. Due to the overwhelming corpora of tweets in Twitter's database, this study further aims to explain how tweets are filtered so that they become valuable data. Finally, in conjunction with tweet filtering, the study also aims to identify the methods of tweet post-processing.

1.3 Significance of the Study

The researchers believe that the data collected from this study will not only yield helpful results for them, but also to the following groups of people:

Data Analysts. The results of this study will offer Data Analysts, who are specifically concerned about the statistics of the Philippines and cryptocurrency, a general understanding as to the stance that current Filipinos have in regards to crypto. By doing so, they will be able to use this information to predict future trends and observe patterns through this study.

Filipinos citizens and interested investors. The findings of this study will offer uneducated citizens information about the current state of cryptocurrency in the country and citizens who are starting to entertain the idea of looking into cryptocurrency, giving them a better grasp into dealing with crypto in general. Interested investors could also use this information to further understand the Philippine market and help choose which currency to invest in.

1.4 Scope and Limitations of the Study

This study will focus on how the LSTM-MCNN-decoder combined with Naive Bayes can be used to determine cryptocurrency-related tweets' polarity. This study will perform Sentiment Analysis using the proposed LSTM-MCNN-decoder combined with Naive Bayes only. Comparisons may be made in the evaluation chapter of the paper between different models but the primary focus will be on this study's framework. This study will only focus on crypto-related tweet data from users in the Philippines. The study will only focus on tweets that contain the predetermined hashtags and keywords (see 3.2.1 for reference). The study will not focus on other cryptocurrencies, such as SOL and LUNA that may undergo a "bullish" scenario out of nowhere. The dataset studied will include text only. The tweet data in this study will only be the tweets from public accounts, using the Filipino(Tagalog) and English language. This study covers Twitter's streaming API for data mining text data in tweets and both the automatic and manual classification of these tweets.

Chapter 2

Review of Related Literature

2.1 Cryptocurrency

A cryptocurrency is a type of digital currency. A virtual coinage system that works similarly to a traditional currency and allows users to send and receive money. It allows for virtual payment of goods and services without the need for a central trustworthy authority. As an example, *Bitcoin* is one of the cryptocurrencies used worldwide. It is a peer-to-peer digital money that was first presented in a white paper in 2008. Satoshi Nakamoto was the author of a paper that was published under his name. The goal of the digital currency was to create an alternative payment system that was free of central control and could be used in the same way as existing currencies. In this study, the researchers will be analyzing different sentiments from Twitter using cryptocurrency as a context. This would allow the researchers to gather different views and opinions, whether positive, negative, or neutral.

2.1.1 Cryptocurrency around the World

In 2020, the worldwide cryptocurrency market was valued at USD 826.6 million. Based on the analysis done by Fortunebusinessinsights.com [36], the global market exhibited a significant growth of 10.0% in 2020 as compared to yearly growth during 2017-2019. The growth of distributed ledger technology and increased digital venture capital investments are driving the market's growth. Developing countries have begun to use digital currency as a financial exchange. The rising popularity of digital assets such as Bitcoin and Litecoin is expected to drive industry expansion in the future years. However, because of the restrictions and economic challenges caused by the coronavirus, numerous token sales have been canceled, while others have been postponed. Large blockchain companies, including Elliptic, Chainalysis, and CipherTrace, have stated that they have reduced their employment or budgets to mitigate the economic effects of the COVID-19 outbreak. North America, Europe, Asia Pacific, the Middle East, Africa, and Latin America are the five key regions in which the market is divided. North America held the highest proportion of the global market in 2020, owing to the fact that most countries in the region used bitcoins as a means of trade for tax purposes rather than as money. Despite the fact that the government does not have any legal authority over it, many developed countries continue to emphasize the use of digital money. The acceptance of digital payment by consumers and retailers is propelling the market forward. Furthermore, bitcoin mining's popularity and the presence of a majority of prominent businesses in North America control the market. The global analysis would help give the proponents ideas as to how other countries view cryptocurrency and how widespread it is being utilized.

2.1.2 Cryptocurrency in the Philippines

The Philippines is one country that receives much money from outside. Many Filipinos get remittances overseas or send remittances to loved ones and families in the Philippines. A growing proportion of Filipinos are shopping online, paying for goods and services online, and transferring money via digital means. Such remittances and transfers, when facilitated through the use of cryptocurrency or blockchain technology–compared to typical remittance and payment schemes, Virtual Currencies via licensed VC exchanges are more convenient, faster, and less expensive. The typical Filipino stands to benefit in this scenario. The advent of VC exchanges increases competition in the remittance and payments business, potentially lowering transaction costs and improving service [33]. According to a study by Franciso et al. [34], Bitcoin, the world's first decentralized cryptocurrency, gained popularity in the Philippines in 2017 [35]. Despite the breakthrough, the creation of bitcoin had little impact on the socio-economic aspect of Filipino households because there were insufficient resources and understanding about the market at the time. When it came to investing in the cryptocurrency market, the major concerns highlighted were the market's unpredictability and volatility. The overview of cryptocurrency in the Philippines is a factor that would help our study determine different sentiments of Filipinos.

2.2 Twitter

Twitter is an online microblogging service that delivers short messages among a community of users. Twitter allows users to post messages publicly (which are referred to as “tweets”) with a maximum length of 140 characters. In November of 2017 that limit was doubled to 280 characters. In addition, users can add “hashtags” to a tweet, which is the ”#” symbol followed by a consecutive string of characters.

It is one of the most popular social media outlets in the world. According to Dean [2], Twitter on average has about 206 million active users daily across the globe. In the Philippines alone, Statista.com (2021) has recorded about 10.2 million users. Twitter has also become a top platform for discussion regarding cryptocurrency and market trends [1]. Because of its ability to instantaneously deliver content as well as quickness in terms of information circulation, virality is the biggest asset that Twitter can offer for cryptocurrency projects.

The result of all of these statistics is that Twitter can be a very rich source of data on how people feel about nearly any given topic. Given that the ability to see a tweet’s date and time of posting is also available, this opens up the possibility of tracking when people’s feelings change over time. This makes Twitter an excellent resource to collect text data on a topic like cryptocurrencies to explore the possible sentiments of Filipino tweets.

2.3 Tweepy

Tweepy is an open-source Python library that will enable the proponents to gather tweets by communicating with the Twitter API. It is simple to use and is one of the few recognized Python libraries accepted by Twitter's developer platform, meaning the use of Tweepy is legal. Twitter requires any library to utilize the OAuth authentication in order to enable the following tasks:

Lookup Tweets, Manage Tweets, Timelines, Search for Tweets, View Like count, Retweet count, Manage Replies, Lookup Users, and so many more tasks used in the study.

Without Tweepy, dealing with low-level tasks such as HTTP requests, data serialization, authentication, and rate limits is very time-consuming and prone to error [2]. Tweepy gives the proponents a very convenient way to work with Twitter's API; hence utilizing it is the primary data gathering method.

2.4 Convolutional Neural Network

Convolutions are a type of sliding window function used to produce specific results on a matrix (e. g., image blur, edge detection.) [13]. A CNN, or convolutional neural network, is a type of artificial neural network that mimics human visual activity. It has a distinctive function in computer vision. Yann LeCun and Yoshua Bengio first proposed the idea in 1995. Early versions of CNN's were utilized in

banking, anomaly detection, drug development, health risk assessment and biomarkers of aging, checker game, time series forecasting, and postal services, among other applications. In addition, it can be used to perform video analysis. With successful applications in computer vision, pattern recognition, speech recognition, natural language processing, and recommendation systems, CNN techniques have been proven to be ideal for extensive data analysis. Convolutional Neural Network can extract an area of features from global data using the convolution process, which allows it to extract a piece of data information as a feature and consider the relationship between these features. When transferring the text into matrix, it can also be considered as the same as an image pixels' matrix, therefore the same operation can be performed to the text data, which is important in this study since the researchers are gathering data from Twitter using texts.

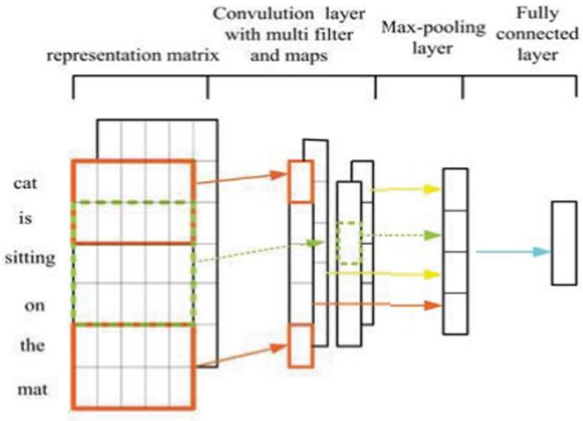


Fig. 1 CNN for Sentence Classification

Assume that the sentence input S is with length n, S is represented as a matrix $\mathbf{x} = [x_1, x_2, \quad, x_i, \quad x_n]$, i_{th} element x_i is a k dimension vector representing the word embedding of i_{th} word in S. Afterwards, the matrix $\mathbf{x} \in R^{n \times a}$ is fed into the convolution layer which is made up of multiple filters which will then capture different features. After the convolution process, the output is then fed to the pooling layer to be sub-sampled into smaller dimensions. When all is said and done, the output is fed to a fully connected layer [23, p. 2].

2.5 Encoder-Decoder Model

These models understand the meaning and emotions of the input sentence and output a sentiment score. The encoder-decoder model is a method of using recurrent neural networks, which also has proven successful at related sequence-to-sequence prediction problems such as text summarization and question answering [11]. Includes two recurrent neural networks, one to encode the input sequence (encoder) and the other to decode the encoded input sequence (decoder) into the target sequence. The proponents of this research will be using the decoder model to further process the collected data from LSTM and CNN.

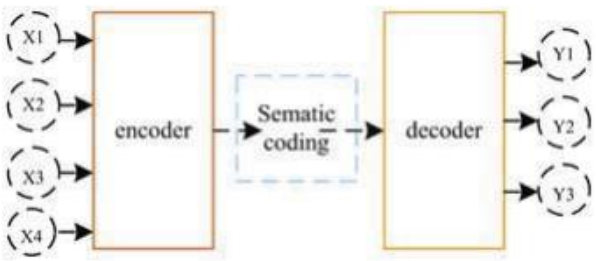


Fig. 2 Encoder-Decoder Framework

According to Chen and Wang [23], The encoder neural network reads and encodes the input sentence into a fixed-length vector then the decoder outputs a translation from the encoded vector [23, p. 2].

2.6 Recurrent Neural Network

People exchange information about things they care about, such as products, services, events, or places, in order to better understand and inform others. A global system like Twitter allows people to express their feelings in a relatively short multimedia message. RNNs (recurrent neural networks) are a type of artificial neural network that can be used to model sequences. Sentiment analysis can be thought of as a sequence to value problems. The text, which is a sequence of tokens, is the input. These tokens can be words or characters, for example.

More than a decade has gone by since the groundbreaking work of Go [4], who were among the first to categorize sentiment in Twitter messages. Their strategy was centered on what is known as "noisy labels." They used emoticons to train machine learning algorithms, and the findings were positive, paving the way for many other researchers with similar research objectives. With that in mind, it is possible for our group to include emoticons in analyzing sentiments in a tweet.

2.7 Long Short-Term Memory

According to Phi [8], reviewing the recurrent neural network can help to comprehend how LSTMs accomplish this. This is how an RNN works: the first words are converted into machine-readable vectors. The RNN then goes through each of the vectors one by one. While processing, it passes the previous hidden state to the next step of the sequence. The hidden state acts as the neural network's memory. It holds information on previous data the network has seen before. A vector is created by combining the input and the previous hidden state. The current and previous inputs are now represented in that vector. The vector is passed through the tanh activation, and the result is the network's new hidden state or memory. A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network.

The control flow of an LSTM is comparable to that of a Recurrent Neural Network. It processes data and passes information on as it moves along. The operations within the cells of the LSTM differ. The cell state and its many gates are at the heart of LSTMs. The cell state serves as a transportation highway for relative information as it travels down the sequence chain. It can be thought about as the network's "memory." In theory, the cell state can carry relevant information throughout the sequence's processing. As a result, information from earlier time steps can make its way to later time steps, lessening the short-term memory effects. Information is added or withdrawn from the cell state via gates as the cell state travels. The gates are several neural networks that determine which information about the cell state is permitted.

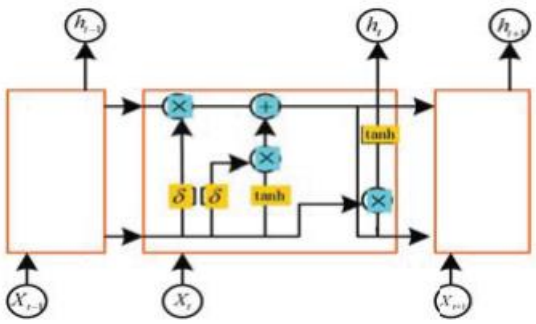


Fig. 3 Unrolled LSTM network

As shown above, LSTM has three gates which can determine if the flow should be remembered or not. The *input gate* controls if the input of new information can be memorized. The *forget gate* determines how long certain values can be held in memory and the *output gate* controls how much the value stored in memory affects the output activation of the block [23, p. 2].

2.8 Sentiment Analysis and other techniques

A subfield of Natural Language Processing (NLP), it is the method of machine learning that attempts to identify and extract opinions or *sentiments* within a given text [15]. Raj (2021) also added that *Sentiment Analysis* is the means to analyze and find the emotion/intent involved in any mode of communication. In order to understand the perspectives of the many Filipinos that are involved in cryptocurrency, it is inefficient to manually go through each individual tweet, therefore methods such as *NLP* and *Sentiment Analysis* will greatly benefit our research.

2.8.1 Twitter Sentiment Analysis Using Deep Learning

According to Neha [6] The multi-task learning method, which was recently introduced, was another effective NLP solution. During this phase, a single model is run through a series of actions in order to attain the highest level of precision in as many fields as possible. According to the theory, the information and matching y- and z-data jobs can improve the efficiency of the x-based model. Modern, cutting-edge accuracy is provided by the capacity to use non-conventional memory and a variety of task weights. The Dynamic Memory Network and the Neural Semantic Encoder are two popular MTL models that have had much success when it comes to sentiment analysis.

2.8.2 Sentiment Analysis of COVID-19 Tweets Using Deep Learning Models

In the article [32], they used data from Indian user tweets from the Twitter website during the COVID-19 shutdown period in-country. The researchers' motive behind this study was to analyze tweets by Indian netizens during the lockdown. They collected tweets between 23 March 2020 and 15 July 2020 and text labeled as fear, sad, anger, and joy. They analyzed data using the new deep learning model Bi-Directional Encoder Representation from Transformers (BERT). The BERT model result was compared with traditional Logistic Regression (LR), Support Vector Machines (SVM), Long Short-term Memory (LSTM). The BERT model result shows more accuracy 89%, compared to other models' LR 75%, SVM 74.75%, and LSTM 65%. This experiment classified sentiment into fear, sad, anger, and joy based on the keywords.

Since this study is attempting to gather data from tweets, sentiment analysis is crucial in helping us understand and efficiently structure unorganized information. The research can be expanded by obtaining more data and using deep learning techniques to determine any more improvements.

2.8.3 *Polarity Classification of Tweets Considering the Poster's Emotional Change by a Combination of Naive Bayes and LSTM*

A study[21] aimed to produce an improved approach to polarity analysis wherein the emotional states of humans were taken into account. By training a Naive Bayes classifier, the researchers were able to score categories considering the result of their dependency analysis. By doing so, a user's emotional state could be estimated from the calculated scores and a time series model that LSTM created. That being the case, the study resulted in a better classification accuracy than the baseline and other related studies. The authors' work further proves that Naive Bayes in conjunction with an LSTM model is definitive for sentiment analysis.

2.8.4 *Designing a Web Application using the Twitter streaming API as a Tracking Tool for Weather Updates*

According to the author of [17], tweets can be used as a reliable source of information in typhoon-related natural disasters. Using Tweepy, the proponent successfully gathered data for their dataset, aiming for tweets that contain interactions with the hashtag #nonaPH. After which the proponent realized that before proper analysis could occur, the tweet corpora had to undergo various pre-processing procedures: removing symbols, hyperlinks, user mentions, hashtag symbols, blank cells, duplicate tweets, retweeted tweets, and stop words. Tweet classification proceeded using Sklearn's Multinomial and Bernoulli Naïve Bayes methods. The author identified that Multinomial NB performed better with better accuracy than its counterpart, Bernoulli NB.

Using the Bayesian variant, Multinomial Naive Bayes, the results proved that tweet relevance is valuable data in typhoon-related events. The study's outcome also proved that Multinomial NB is a reliable method of classifying tweet data wherein this study will be conducting its dataset.

2.9 Naive Bayes for Document Classification

This study undergoes a machine-learning domain of document classification. Document Classification is the process in which each instance represents a document, and this instances' class is the document's topic. Documents are characterized by the words that appear in them, and a way to apply machine learning to these documents is by treating the presence or absence of a word as a boolean attribute. One of the most popular techniques for this job is Naive Bayes, which is relatively fast and accurate [12].

According to Ablazo [17, p. 6], the Naive Bayes classifier has proven to be a highly effective machine-learning algorithm used in sentiment analysis. Schneider [18] conducted a study about two popular Naive Bayes methods, Multinomial and Bernoulli, proving which was more effective in class labeling given the context of e-mail filtering. The findings stated that Multinomial NB proved to be less biased towards labeling one class and achieved slightly higher accuracy than its counterpart. Although both NB variants make assumptions about feature vector distribution of documents, Multinomial NB is often used to classify data based on the frequency of words in a string. In contrast, Bernoulli NB is used to classify data based on words or features.

Nevertheless, Naive Bayes classifiers are helpful in this study as these help in accurately classifying data. This is important as this will directly affect the outcome's validity.

2.10 Word2Vec

Word2Vec is well known and widely used in various NLP methodologies. Introduced by Mikolov et al. [27], it is often used for word embeddings from raw text. There are two primary models: the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model (SG). In nature, they are opposites, as the CBOW method uses the context of the sentence to predict the next word, while the SG method uses the word to predict the context. In the studies [29,30,31], Word2Vec is proven to be an effective representation in sentiment analysis tasks.

Pouransari and Saman [28] further add that Sentiment classification accuracy is improved by employing word embedding. High dimensional word vectors that learn contextual information of words are produced using Word2Vec. These studies establish the Word2Vec methodology as effective in sentiment analysis. This study will be utilizing Word2Vec for vectorizing the manual classified categories of tweet data to improve the accuracy of the research.

2.11 Splitting Dataset into Train and Test Sets

Observations must be used in the training phase to detect a machine learning model's behavior. Otherwise, the model's judgment would be skewed. The simplest method is to divide the whole dataset into two sets. This means we use 80% of the observations for training

and the rest for testing. With less training data, the parameter estimates have a high variance. On the other hand, less testing data leads to high variance in performance measures [26].

2.12 VADER Algorithm as benchmark

VADER or Valence Aware Dictionary and Sentiment Reasoner is a lexicon and rule-based sentiment analysis tool specifically attuned to sentiments expressed in social media [7]. It is advantageous when used in micro-blogging sites such as Twitter as text data, and another variety of text is prevailing. Pope [9] states that VADER is also helpful in labeling text data that has a massive dataset that would take a human too much time and effort. Moreover, in a similar study by Kim et al. [24], the VADER algorithm was also used to obtain accurate polarity scores from social media texts to predict cryptocurrency price fluctuations. Gilbert and Hutto [25] show that VADER can outperform both human annotators and most classifier benchmarks. In the case of this study, VADER will be utilized to compare our output as it is an effective and proven result benchmark when it comes to sentiment analysis.

Theoretical Framework

A study on gaining an understanding of the Filipino views towards different cryptocurrencies through the use of deep learning and sentiment analysis has the need to explain what theories or methods were used to produce a satisfactory output and new knowledge.

A. Naive Bayes and other Pre-processing

The dataset, composed of tweet corpora, will be passed through several steps to clean, filter, and prune the relevant and appropriate tweets about cryptocurrency. It is important for this study to have this step due to how much data is available in the data gathering. There will be unnecessary symbols, emojis, or images that are not relevant nor are covered by the scope of the study. By cleaning and filtering the data set, it is expected that the results will be more accurate and would provide substantially correct results.

a. Tweepy

To gather tweets, the use of Tweepy to communicate with the Twitter API will be necessary. Tweepy also provides ways to clean data, or in other words remove unnecessary objects like symbols and emojis from tweets [2].

b. Manual Classification of Tweets

Tweets about cryptocurrency are wide and varied, therefore it is necessary to classify these tweets for easier processing later on. Modeled after [17], the tweets will undergo both manual and automatic classification. The manual classification process will use CrowdFlower while each assessor will show a tweet, the crowd will then copy-paste a word or phrase from the tweet conveying the specific information. Derived from [17], the tweets will be classified thrice.

b.1 The initial classification of the tweets will filter tweets into three main classes: Personal, Informative, and Other. Personal tweets are of interest only to its author and to their immediate relationships. Informative tweets peak the interest of others outside of the author's immediate relationships. Other tweets are tweets not relevant to cryptocurrency or of the keywords searched, spam tweets are also included here.

b.2 Both Personal and Informative tweets are to be classified further into three classes: Positive, Negative, and Neutral. Positive tweets convey favorable emotion or view towards the subject of the tweet. Negative tweets convey the opposite, as such may be of dissatisfaction, disappointment, or distraught. Neutral tweets convey the message without any emotional undertone.

b.3 For each of the said types, a separate crowdsourcing task will be created where the crowd will be asked to further classify the tweet by sub-type and to extract a substring from the tweet containing a given piece of information.

c. Word Embedding

A learned representation for text where words that have the same meaning have a similar representation are called word embeddings. Its use was considered in [19] as one of the key breakthroughs of deep learning on challenging natural language processing problems. In describing word embeddings they stated: *"One benefit of using dense and low-dimensional vectors is computational: the majority of neural network toolkits do not play well with very high-dimensional, sparse vectors. However, this is just a technical obstacle, which can be resolved with some engineering effort. The main benefit of the dense representations is in generalization power: if we believe some features may provide similar clues, it is worthwhile to provide a representation that is able to capture these similarities."* Word embedding is how words are represented as real-valued vectors in a vector space, often ten to hundred times multidimensional. Each word is mapped to one vector and the way the values of the vector are learned resemble a neural network, such is why it is associated with deep learning. The key in it is the idea of using a dense distributed representation for each word. This distributed representation is learned through the usage of words. Further explained in [20], words will have the same representation if these words have the same usage, naturally capturing the meaning. Dealing with tremendous amounts of tweets, the amount of words to process is simply too many, and some even have the same meaning. The use of word embeddings help in minimizing the processing needed in any machine learning methods to solve NLP related problems like this research.

d. Naive Bayes

Naive Bayes classifiers help in characterizing words inside a document (tweet) and also applying machine learning to these said documents through treating the presence or absence of words. It can be used to filter spam tweets from actual relevant tweets by checking the frequency of words and through calculation [12]. Additionally, it is a kind of supervised learning that classifies words based on how probable it is to belong to a category. Based on that probability, the score for each category is calculated for the word. Naive Bayes requires that words appear independently [21]. This serves to prove how much the Naive Bayes classifier is an effective machine-learning algorithm for sentiment analysis. In [22], a comparative study between Multinomial Naive Bayes and Multivariate Bernoulli Naive Bayes on which one is more effective in labeling in the context of anti-spam email filtering. The findings favored Multinomial NB in both accuracy and for being less biased in one label than Multivariate NB.

B. Deep Learning

Deep learning, quite simply, is the application of artificial neural networks to learning tasks using multilayer networks. It can be thought of as the next step from a base neural network that was only deemed practical with one or two layers and a small data set. An application of this is the Long Short-Term Memory neural network. It is a special type of Recurrent Neural Network, which is capable of learning long-term dependencies. LSTM utilizes four network layers interacting in a complicated manner as its repeating module with two states: hidden state and cell state. The latter with its many gates are at the heart of LSTMs [8]. The cell state serves as a transportation highway for relative information as it travels down the sequence chain. It can be thought about as the network's "memory." Additionally, the use of Convolutional Neural Network will help in capturing local features. CNN is particularly useful for classification of NLPs which in its findings suggest strong local clues regarding class membership. It has special spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. The decoder framework will be used to reproduce the input matrix, leading to a more intrinsic and effective feature learning in the convolution which consequently leads to a better output [23].

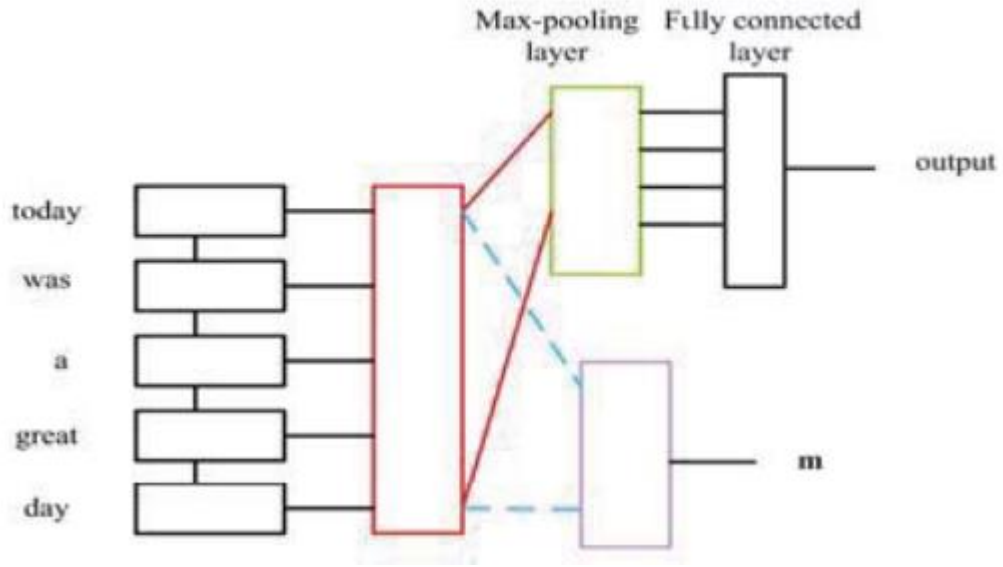


Fig. 4 LSTM-MCNN-decoder Model

This model was chosen over other potential neural network models since [23] result's show that other hybrids and even base LSTM and CNN are less accurate than the model. The combination of LSTM and MCNN with a decoder framework is also less researched upon, proving the need to explore its potential in the field of NLP problem solving.

Model	Average accuracy
CNN	75.18 %
Multilayer CNN	76.09 %
Multilayer CNN-decoder	78.18 %
CNN-LSTM	76.36 %
LSTM-CNN	76.91 %
LSTM-MCNN	77.09 %
LSTM-MCNN-decoder	78.6 %

Fig. 5 Average Accuracy result table comparing various CNN's.

C. Sentiment Analysis

To understand the views of the target demographics about cryptocurrencies, the use of sentiment analysis could prove effective in providing relevant and accurate interpretation of the perspectives in the tweet corpora. Sentiment Analysis is a subfield of NLP that attempts to identify and extract opinions or *sentiments* within a given text [15].

Chapter 3

Research Model and Methodology

3.1 Conceptual Framework

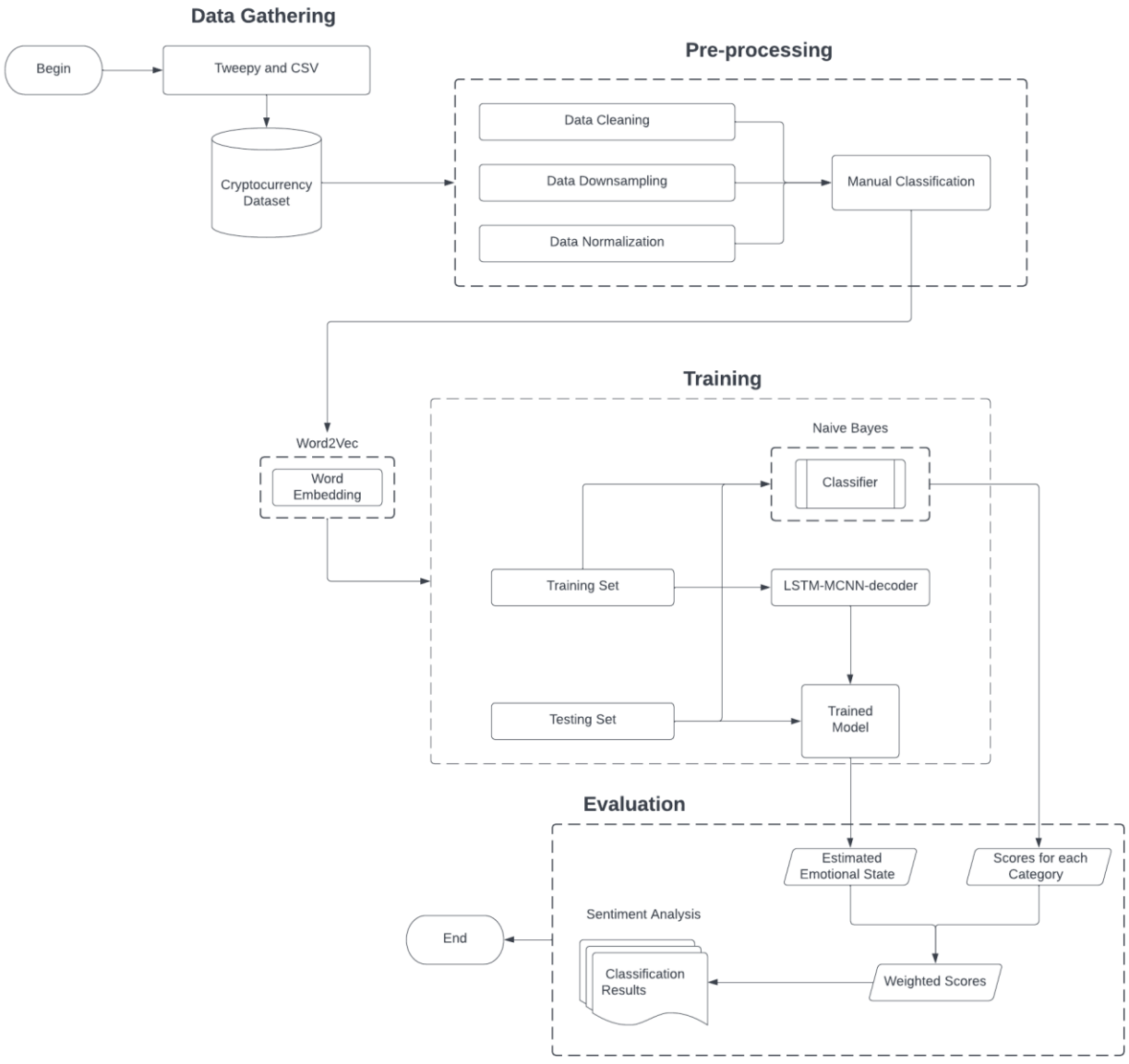


fig . 6 Conceptual Framework

Dealing with NLP problems through machine learning has been an endeavor for years now. Many researches were made in looking into the accuracy of different models with different dataset to see how efficient the networks are in understanding human emotions through

sentiment analysis. Cryptocurrency is relatively new but many have already made their opinions of it, varying opinions. The model presented above could prove useful in understanding these varying opinions specifically of the Filipino people.

The model shows the framework that will be used in this research. The data gathering starts through the Twitter API using Tweepy and CSV. Keywords are defined and searched to help in gathering cryptocurrency tweets. The dataset of tweets will then be cleaned, sampled, and normalized. The pre-processed dataset will then be manually classified through crowdsourcing into main types and subtypes. After manual classification, the dataset is converted into numbers. A vectorizer or word embedder will be used. In this study, Word2Vec will be used to convert each document (tweet) into vector matrices of numbers. The vectorized word embedded dataset is then split to train and test the models to be used. The training dataset will be 80% of the gathered data, while 20% will be for the testing set. This dataset split will be used for training and testing both Naive Bayes and the LSTM-MCNN-decoder models to produce emotional scores. The scores of both models will then be used to calculate the weighted score, which will be used further to determine the polarity of the tweets.

3.2 Research Questions

This research is conducted to perform Sentiment Analysis using a combination of LSTM-MCNN-decoder with Naive Bayes on cryptocurrency-related tweets. This study aims to answer the following questions:

- How can the combination of LSTM-MCNN-decoder with Naive Bayes determine the polarity of cryptocurrency-related tweets in the Philippines?
    - How effective and accurate is the proposed model compared to other models in dealing with the same dataset?
    - How were the tweets manually and automatically classified and processed?
    - What sentiments were determined after analyzing the data through the weighted scores?

3.3 Methodology

3.3.1 The Dataset

Using the Python library, Tweepy, tweets containing the following keywords were collected and the data gathered will be used as the dataset:

Bitcoin, XRP, Litecoin, Dogecoin, Ethereum, Cardano, Binance Coin, ShibaInu, Tether, Polkadot

BTC, ETH, LTC, DOGE, ETH, ADA, BNB, SHIB, USDT, DOT, Investment, Invest, Crypto, Cryptocurrency, Binance

3.3.2 Develop Tweepy bot for data gathering

A python script will be written to gather cryptocurrency-related tweets using the libraries: Tweepy and CSV. Tweepy is a Python library that will be used to gather tweets using Twitter API. This is possible through the OAuth object, which takes in the consumer API key, as well as a consumer API secret key that can be obtained from https://developer.twitter.com/en.

The CSV library allows creating, reading, updating, and deleting interactions with CSV files. Using this library, the tweets will be written into a CSV file.

```python
# script which gathers tweets about certain texts.
# Ogs Ablazo
import csv, tweepy, os

class Conn:
    '''
        twitter credentials class
    '''
    def __init__(self, *args):
        self.ck = ''    #consumer key
        self.cs = ''    #consumer secret
        self.at = ''    #access token
        self.ats = ''   #access token secret

    def tweep_creds(self):
        auth = tweepy.OAuthHandler(self.ck, self.cs)
        auth.set_access_token(self.at, self.ats)
        return tweepy.API(auth)

#initialize api w/ twitter api credentials object
api = Conn().tweep_creds()

def gather_tweets(query_word,twt_lmt):
    counter = 0
    for tweet in list(tweepy.Cursor(api.search, q=query_word).items(twt_lmt)):
        csvWriter.writerow([tweet.text, tweet.user.screen_name, tweet.created_at, tweet.user.location, tweet.retweet_count, tweet.favorite_count])
        counter = counter + 1
    return counter

# tweet filename
global csvWriter
namefile = "tweets.csv"
dir_name = os.path.dirname(os.path.abspath(__file__))
file_dir = open(dir_name+"/"+namefile,'r', encoding='ISO-8859-1')
csvFile = open(dir_name+"/"+namefile,'a')
csvreader = csv.reader(file_dir)
csvWriter = csv.writer(csvFile)
row_cnt = sum(1 for row in file_dir)


q_words = "" # enter query words here (separate by spaces)
q_words = q_words.split()

t_lmt = tweet_limit = 500 #adjust tweet limit accordingly
limit_num = tweet_limit * len(q_words)
new_rc = row_cnt

for q in q_words:
    new_rc += gather_tweets(q, t_lmt)
print("initial rows: %s\ngathered rows: %s"%(row_cnt,new_rc-row_cnt))
csvFile.close()
```

Fig. 7 Sample code for gathering tweets [40].

3.3.3 Cleaning of Tweet Dataset

Upon data collection, tweets will inevitably contain a lot of noise/irrelevant information. These come in the form of URLs, mentions (@username), retweets, and duplicate tweets. Empty/blank columns will be replaced with ‘none’ values as these may cause errors when fed in the automatic classifiers.

A script will be made to clean these unnecessary values. The Tweepy library has the needed functionality to access and remove certain characters from tweets that will be written into the CSV output file.

```
In [42]: df = df[~df.tweets.str.contains("RT")]

In [44]: df = df.reset_index(drop=True)

In [45]: df
Out[45]:
```

	tweets	likes	time
0	“Conservatives Call on State Legislators to Ap...	137162	2020-12-21 23:51:21
1	...Republicans in Wisconsin should take these ...	138257	2020-12-21 21:48:08
2	...WOW, he just voted against me in a Big Cour...	116169	2020-12-21 21:48:08
3	Two years ago, the great people of Wisconsin a...	152517	2020-12-21 21:48:08
4	....@sendavidperdue will not be able to win on...	104980	2020-12-21 15:30:09
...	...	...	...
122	Georgia, where is signature verification appro...	174021	2020-12-11 22:36:07
123	If the Supreme Court shows great Wisdom and Co...	292105	2020-12-11 20:28:34
124	If the two Senators from Georgia should lose, ...	185550	2020-12-11 20:28:16
125	“Donald Trump must get the credit for the vacc...	212406	2020-12-11 15:32:54
126	Now it turns out that the Democrats want the P...	192956	2020-12-11 14:15:50

127 rows × 3 columns

Fig. 8 Sample code for removing retweets or “RT” from tweet dataset.

All the retweeted tweets will contain the prefix “RT” as this is how Twitter labels retweets. These will be removed to avoid any inconsistencies on the classification accuracy.

```
#script to remove the following in tweets:
#http:// and https:// (urls), mentions and hashtags

import re
import sys
import csv

def data_clean(fn):

        fin = open(fn, 'rb')
        fout = open("cleaned.csv",'wb')
        results = []
        patterns = [ r"http:\/\/\w+\S+", r"@\w+", r"#\w+"]
        for text in csv.reader(fin):
                newrow = [''.join('' if c in conversion else c for c in entry) for entry in row]
                csv.writer(fout).writerow(newrow)


text_file = sys.argv[1]
data_clean(text_file)
```

Fig. 9 Sample code for removing URLs, mentions, and hashtags from tweet dataset.

This script is used to remove all instances of hyperlinks, user mentions (@username), and hashtags in tweets. This script utilizes the RE and CSV Python libraries. The RE library contains all Regular Expression operations and methods.

```
# Ogs Ablazo
# it is assumed that the data set has been cleaned of hashtags,
# symbols, and URLs before executing this script.
# this does the following:
# 1 - reads a 'clean' csv.
# 2 - reads each line from the csv.
# 3 - replaces all blank cells in the current line iteration,  with the string "NONE"
# 4 - writes each updated line into a new row in a new csv file
import sys
import csv
import fileinput
import string

in_file = sys.argv[1]
row_reader = csv.reader(open(in_file,"rb"))
row_writer = csv.writer(open("noRTnoBlankData.csv", "wb"))

first_row = row_reader.next()
row_writer.writerow(first_row)

for row in row_reader:
    new_row = [val if val else "NONE" for val in row] + (["NONE"] * (len(first_row)-len(row)))
    row_writer.writerow(new_row)
```

Fig. 10 Sample code for replacing blank cells with ‘none’ in tweet dataset.

This script is made to replace all blank cells with the string ‘NONE’ for each tweet instance. This is done because these tweets do not have any value to the analysis and keeping them will only lead to inaccuracy.

3.3.4 Manual Classification of Tweets

Using Google forms, crowdsourced respondents were tasked to do the manual classification of each tweet corpora to attain training data that were used to train Automatic classifiers. Two hundred fifty crowdsourced annotators labeled the cleaned tweet dataset three times. These annotators were tasked to label survey sets that contained 144 to 145 tweet texts. All annotators were briefed of the following tweet categories before further classification was done:

- Positive tweets convey favorable emotion or view towards the subject of the tweet.
- Negative tweets convey the opposite, as such may be of dissatisfaction, disappointment, or distraught.
- Neutral tweets convey the message without any emotional undertone.

However, if the tweet didn’t belong to the said tweet categories, the annotators would then opt to choose the “none of the choices” category. The none of the choices option – signifies that the tweet is not relevant, or not informative.

A script based on [17] was made to determine which tweets had two to three label agreements between three batches. The script contained the define_class() method which compared each tweet label. If there were two to three similar labels – then that would be the

defined classification for the said tweet row. However, if there were no similar labels, a ‘conflicting’ string would be returned. By the end of the script, all tweets that had a common class would have their labels saved in a CSV column named ‘defined_class’.

```
import csv, pandas as pd, numpy as np

df = pd.read_csv("consdata.csv")

def define_class(x,y,z):
    if (x == y) or (y == z) or (z == x):
        if x == y:
            def_class = x
        if y == z:
            def_class = y
        if z == x:
            def_class = z
        return def_class.lower().strip(' /')
    else:
        return 'conflicting'

df['defined_class'] = try_class = ""
c = d = 0

for i, row in df.iterrows():
    try_class = define_class(row['tweet_class1st'].lower().strip(' /'),
                             row['tweet_class2nd'].lower().strip(' /'),
                             row['tweet_class3rd'].lower().strip(' /'))
    if try_class == 'conflicting':
        d = d + 1
    else:
        c = c + 1
    df.at[i,'defined_class'] = try_class
print "defined classes %s" %c
print "conflicting classes: %s" %d
df = df[df.defined_class!='conflicting'] #remove conflicting classes
df.to_csv('common_classes.csv')
print len(df)
```

Fig. 11 Script to remove rows that didn’t have label agreements

3.3.4.1 Fleiss’ Kappa

The previous step mentioned that the tweets will be manually classified by many individuals, for that, a difficulty may arise in determining what rating or categorization a tweet may belong to. In cases as such, the use of Fleiss’ kappa may serve well. Fleiss’ kappa [36] is a measure of inter-rater agreement to determine the level of agreement between two or more raters in rating in categorical variables. The use of this concept is valid in this study since it follows the basic requirements and assumptions which are:

- The **response variable** that is being assessed by your two or more raters is a **categorical variable**.
- The **two or more categories** of the **response variable** that are being assessed by the raters must be **mutually exclusive**, which has two components.
- The **response variable** that is being assessed must have the **same number of categories for each rater**.
- The **two or more raters** are **non-unique**.
- The **two or more raters** are **independent**, which means that one rater's judgment does **not** affect another rater's judgment.
- The **targets** being rated are **randomly selected** from the **population** of interest rather than being specifically chosen.

In this study’s manually classified data, the annotators, which labeled the data three times, are the raters. In the interpretation of how good the kappa value is, the guideline below, an adaptation from Landis and Koch, by Altman will be used.

Value of κ	Strength of agreement
< 0.20	Poor
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Good
0.81-1.00	Very good

Table: Classification of Cohen's kappa.

Fig. 12 Classifications of Cohen's Kappa

3.3.5 Data Splitting

Before feeding the dataset to the classifier and the neural network model, the data must be split into training and testing data. The training data will be used to train and fit both the classifier and model. On the other hand, the testing data will be used to test the accuracy of the models on how well it predicts and classifies the tweets. Scikit learn's train_test_split module was used to split the dataset into random 80% training and 20% testing subsets.

3.3.6 Automatic Classification of Tweets

Before feeding the manually classified tweets into a mathematical model, we first need to prepare the data in a way the network understands. Word2Vec is a model for generating word embeddings that represents words or phrases in the vector space of real numbers. Word2Vec will use two architectures: CBOW (Continuous Bag of Words) and Skip Gram. CBOW model predicts the current word given context words within the window while Skip Gram predicts the surrounding context words within a specific window given word. The input layer contains the context words and the output layer contains the current word. The hidden layer contains the number of dimensions in which words are represented. For generating word vectors in Python, modules needed are *nltk* and *gensim*.

```
pip install nltk
pip install gensim
```

Fig. 13 Installation commands for the necessary libraries.

```
# Python program to generate word vectors using Word2Vec

# importing all necessary modules
from nltk.tokenize import sent_tokenize, word_tokenize
import warnings

warnings.filterwarnings(action = 'ignore')

import gensim
from gensim.models import Word2Vec

#  Reads 'alice.txt' file
sample = open("C:\\Users\\Admin\\Desktop\\alice.txt", "r")
s = sample.read()

# Replaces escape character with space
f = s.replace("\n", " ")

data = []

# iterate through each sentence in the file
for i in sent_tokenize(f):
    temp = []

    # tokenize the sentence into words
    for j in word_tokenize(i):
        temp.append(j.lower())

    data.append(temp)

# Create CBOW model
model1 = gensim.models.Word2Vec(data, min_count = 1,
                                size = 100, window = 5)

# Print results
print("Cosine similarity between 'alice' " +
               "and 'wonderland' - CBOW : ",
    model1.similarity('alice', 'wonderland'))

print("Cosine similarity between 'alice' " +
                 "and 'machines' - CBOW : ",
      model1.similarity('alice', 'machines'))

# Create Skip Gram model
model2 = gensim.models.Word2Vec(data, min_count = 1, size = 100
                                             window = 5, sg = 1

# Print results
print("Cosine similarity between 'alice' " +
          "and 'wonderland' - Skip Gram : ",
    model2.similarity('alice', 'wonderland'))

print("Cosine similarity between 'alice' " +
           "and 'machines' - Skip Gram : ",
      model2.similarity('alice', 'machines'))
```

Fig. 14 Implementation of Word2Vec Sample Code

3.3.6.1 LSTM Implementation

The real number vectors will then be fed to the neural network models, specifically LSTM first. LSTM (Long-Short Term Memory) Neural Network is a type of RNN (Recurrent Neural Network) which performs better than base RNN on tasks involving long time lags. LSTM has three gates: input gate, forget gate, and output gate. The input gate controls which new information to memorize. The forget gate determines the duration values will be held. The output gate controls how much value stored in memory affects the output activation of the block. Implementing LSTM involves importing *TensorFlow* and *rnn* class form *tensorflow.contrib* which will build the LSTM Network.

```
import tensorflow as tf
from tensorflow.contrib import rnn

class RNNGenerator:
    def create_LSTM(self, inputs, weights, biases, seq_size, num_units):
        # Reshape input to [1, sequence_size] and split it into sequences
        inputs = tf.reshape(inputs, [-1, seq_size])
        inputs = tf.split(inputs, seq_size, 1)

        # LSTM with 2 layers
        rnn_model = rnn.MultiRNNCell([rnn.BasicLSTMCell(num_units),rnn.BasicLSTMCell(num_units)])

        # Generate prediction
        outputs, states = rnn.static_rnn(rnn_model, inputs, dtype=tf.float32)

        return tf.matmul(outputs[-1], weights['out']) + biases['out']
```

Fig. 15 Sample code to generate a LSTM network.

After implementing the model, it must be run and evaluated. 5000 iterations are run by the model. Injecting a model, optimer, loss function, etc. in the constructor is necessary for the class to have information to process. Initially, the data will be sliced up in the provided dictionary and make encoded outputs. To avoid overfitting, there is a need to push random sequences in the model, handled specifically through the offset variable.

```
import tensorflow as tf
import random
import numpy as np

class SessionRunner():
    training_iters = 50000

    def __init__(self, optimizer, accuracy, cost, lstm, initilizer, writer):
        self.optimizer = optimizer
        self.accuracy = accuracy
        self.cost = cost
        self.lstm = lstm
        self.initilizer = initilizer
        self.writer = writer

    def run_session(self, x, y, n_input, dictionary, reverse_dictionary, training_data):

        with tf.Session() as session:
            session.run(self.initilizer)
            step = 0
            offset = random.randint(0, n_input + 1)
            acc_total = 0

            self.writer.add_graph(session.graph)

            while step < self.training_iters:
                if offset > (len(training_data) - n_input - 1):
                    offset = random.randint(0, n_input+1)

                sym_in_keys = [ [dictionary[ str(training_data[i])]] for i in range(offset, offset+n_i
                sym_in_keys = np.reshape(np.array(sym_in_keys), [-1, n_input, 1])

                sym_out_onehot = np.zeros([len(dictionary)], dtype=float)
                sym_out_onehot[dictionary[str(training_data[offset+n_input])]] = 1.0
                sym_out_onehot = np.reshape(sym_out_onehot,[1,-1])

                _, acc, loss, onehot_pred = session.run([self.optimizer, self.accuracy, self.cost, sel
                acc_total += acc

                if (step + 1) % 1000 == 0:
                    print("Iteration = " + str(step + 1) + ", Average Accuracy= " + "{:.2f}%".format(1
                    acc_total = 0
                step += 1
                offset += (n_input+1)
```

Fig. 16 Sample code that will run and evaluate the accuracy of the model.

3.3.6.2 Multilayer CNN Implementation

The output generated by the LSTM layer will then be the input of the CNN (Convolutional Neural Network) layer which can be good at capturing local features. All word vectors for the input matrix for the convolutional neural network are concatenated into two-dimensional matrices. Sentences S = { S1, S2, S3, …, Sn} with n word is represented as matrix x = { x1, x2, x3, …, xi, …, xn}, *x* is an element of real number matrix with dimension *n* by *d* where the *ith* element *xi* is a *d* dimension vector representing a word embedding of *ith* word *si* in S. The more layers of CNN there are, the better the features will be learned but it will be harder for parameter learning. This study will use four-layer CNN instead of a single layer model. Convolution is conducted by a convolution kernel. For a kernel with length 1:

$$c_i = f(\boldsymbol{\omega} \cdot \mathbf{x}_{i:i+l-1} + b) \qquad (1)$$

Fig. 17 Convolution is conducted by a convolution kernel.

Here, $\boldsymbol{\omega} \in R^{l \times d}$ is the weight matrix of the kernel, $\mathbf{x}_{i:i+l-1}$ is the word embedding matrix for the kernel. For the sentence with length n, then we get the feature vector:

$$\mathbf{c} = [c_1, c_2, \quad , c_i, \quad c_n] \qquad (2)$$

Fig. 18 Feature vector.

Here, $\mathbf{c} \in R^{n-l+1}$.

CNN's output will then be fed into a max-pooling and fully connected layer to extra the most important information of these feature and reduce the complexity of parameters, $\hat{\mathbf{c}} = \max\{\mathbf{c}\}$, for the window with m kernels, we get:

$$\hat{\mathbf{c}} = [\hat{c}_1, \hat{c}_2, \quad , \hat{c}_m] \qquad (3)$$

Fig. 19 Window with m kernels.

Here, $\hat{\mathbf{c}} \in R^m$ is the extracted features vector.

The utilize activation function to predict the emotional tendency:

$$\hat{y}(\mathbf{x}) = g(\hat{\mathbf{c}}) \qquad (4)$$

Fig. 20 Utilize activation function.

Here, g() is the activation function, $\hat{y}(\mathbf{x})$ is the predicted label (1 for positive, 0 for negative) of input sentence.

```
from keras.models import Sequential
from keras import layers

embedding_dim = 100

model = Sequential()
model.add(layers.Embedding(vocab_size, embedding_dim,
input_length=maxlen))
model.add(layers.Conv1D(128, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
history = model.fit(X_train, y_train,
                    epochs=10,
                    validation_data=(X_test, y_test),
                    batch_size=10)
```

Fig. 21 Sample code for CNN implementation [38].

3.3.6.3 Decoder Implementation

The input matrix will then be reproduced by utilizing the decoder layer.

$$\mathbf{m} = deconvolution(\mathbf{c}) \quad (5)$$

Fig. 22 Decoder layer.

Here, $\mathbf{m} \in R^{n \times d}$, is the reconstructed matrix of x.

```
input_sequence = Input(shape=(max_spanish_len,))
embedding = Embedding(input_dim=spanish_vocab, output_dim=128,)(input_sequence)
encoder = LSTM(64, return_sequences=False)(embedding)
r_vec = RepeatVector(max_english_len)(encoder)
decoder = LSTM(64, return_sequences=True, dropout=0.2)(r_vec)
logits = TimeDistributed(Dense(english_vocab))(decoder)
```

```
enc_dec_model = Model(input_sequence, Activation('softmax')(logits))
enc_dec_model.compile(loss=sparse_categorical_crossentropy,
              optimizer=Adam(1e-3),
              metrics=['accuracy'])
enc_dec_model.summary()
```

Fig. 23 Sample code on the implementation of the decoder model [39].

```
model_results = enc_dec_model.fit(spa_pad_sentence, eng_pad_sentence, batch_size=30, epochs=100)
```

Fig. 24 Sample code for training the decoder model.

3.3.6.4 Multinomial Naive Bayes Implementation

The Multinomial NB methods utilized the Python library SciKit Learn or Sklearn, known for its uses in data science. Sklearn's Pipeline function was used to simplify the feature extraction and classifier code further and contain the entirety of the classifier models. Sequentially, the Pipeline function applies a list of transformers and vectorizers and a final estimator or classifier. The Transformers and Vectorizers are tasked to classify the data, meaning they convert the text data into numeric data for the estimator to work. The Estimators are tasked to calculate an estimate of a given quantity based on observed data, features, or labels.

The Pipeline's parameters will contain a pythonic list with the following tuple pairs: vect and CountVectorizer(), tfidf and TfidfTransformer(), and clf and MultinomialNB(). Each tuple pair that observes a key implies a value relationship, such that each value would then be referenced later on by calling its respective key.

For this study, the ngram_ranges of (1, 3) to (4, 6) were the features tested on by the Pipeline, which means that a minimum of 1 to a maximum of 3 n-grams ranges to a minimum of 4 to a maximum of 6 n-gram ranges were the considered features for each NB estimator. These ranges were identified to test how well each classifier fared with little n-gram range to bigger n-gram range combinations. The "stop_words" parameter accepts a list of Tagalog and English words to be filtered from the text corpora. These can be found in the appendices.

```
i, j = 1, 4
for j in range(4,7):
    for i in range(1,4):
        text_clf = Pipeline([('vect', CountVectorizer(ngram_range=(i,j),
                                                      stop_words=tl_enSW)),
                             ('tfidf', TfidfTransformer()),
                             ('clf', MultinomialNB()),])
```

Fig. 25 Sample code to initialize the Pipeline with varying n-gram features

```
text_clf.fit(x_train,y_train)
y_pred = text_clf.predict(x_test)
```

Fig. 26 Sample code to train the classifier

3.3.7 Evaluation

After the automatic classifiers classified tweets, these were evaluated according to the accuracy_score and the following classification metrics from the sklearn library: Precision, Recall and F-measure (or F1-score).

The accuracy score method computes subset accuracy. If the entire dataset of predicted labels for a sample exactly matches with the ground truth of labels, then the accuracy would be 1.0 (or 100%). The accuracy score is computed using the following formula:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Fig. 27 Accuracy score formula.

The accuracy_score method accepts two parameters which are y_test and y_pred and returns a floating point. Sets of accuracy scores and classification results were derived from the n-gram ranges of (1,3) to (4,6).

```
text_clf.fit(x_train,y_train)
y_pred = text_clf.predict(x_test)
print("accuracy score %s"%accuracy_score(y_test,y_pred))
print(classification_report(y_test, y_pred))
```

Fig. 28 Script to determine accuracy score and classification report [17].

The classification_report method accepts two parameters. Like accuracy_score, it also accepts y_test and y_pred, from which it computes for the precision, recall, and f1-score of the classification model. The classification_report method then returns a table containing the computed precision, recall, f1-score and the support involved behind each class in the dataset. Precision is the ratio of the correctly predicted labels and the total predicted labels. In other studies, this is also known as the hit ratio. This is calculated by dividing the total number of True Positives by the sum of True and False Positives. The recall is the ratio of correctly labeled text data and the actual number of correct labels, in other studies this is also known as the detection rate. This is calculated by dividing the total number of True Positives by the sum of True Positives and False Negatives. The F1-score or the F-measure is the harmonic mean of both precision and recall scores, which is derived by using the following formula: Support merely identifies how many instances were sampled and labeled for testing.

References:

[1]P. Bandoy, "Sneaky hacks to grow your blockchain community on Twitter", *Meetbunch.com*, 2020. [Online]. Available: https://www.meetbunch.com/the-twitter-guide. [Accessed: 23- Mar- 2022].

[2]B. Dean, "How Many People Use Twitter in 2022? [New Twitter Stats]", *Backlinko*, 2022. [Online]. Available: https://backlinko.com/twitter-users. [Accessed: 23- Mar- 2022].

[3]M. Garcia, "How to Make a Twitter Bot in Python With Tweepy – Real Python", *Realpython.com*, 2021. [Online]. Available: https://realpython.com/twitter-bot-python-tweepy/#what-is-tweepy. [Accessed: 07- Mar- 2022].

[4]A. Go, R. Bhayani and L. Huang, "Twitter Sentiment Classification using Distant Supervision", *Www-cs.stanford.edu*, 2022. [Online]. Available: https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf. [Accessed: 23- Mar- 2022].

[5]S. Kumar and M. Nezhurina, "Sentiment Analysis on Tweets for Trains Using Machine Learning", *Reseachgate*, 2020. [Online]. Available: https://www.researchgate.net/publication/332330960_Sentiment_Analysis_on_Tweets_for_Trains_Using_Machine_Learning. [Accessed: 23- Mar- 2022].

[6]N. Yadav, H. Gupta, S. Pande, N. Karale, A. Khamparia and V. Bhagat, "Twitter Sentiment Analysis Using Deep Learning", *Reseachgate*, 2021. [Online]. Available: https://www.researchgate.net/publication/348590973_Twitter_Sentiment_Analysis_Using_Deep_Learning_Twitter_Sentiment_Analysis_Using_Deep_Learning. [Accessed: 23- Mar- 2022].

[7]P. Pandey, "Simplifying Sentiment Analysis using VADER in Python (on Social Media Text)", *Medium*, 2018. [Online]. Available: https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f. [Accessed: 23- Mar- 2022].

[8]M. Phi, "Illustrated Guide to LSTM's and GRU's: A step by step explanation", *Medium*, 2018. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. [Accessed: 23- Mar- 2022].

[9]L. Pope, "Comparing VADER and Text Blob to Human Sentiment", *Medium*, 2020. [Online]. Available: https://towardsdatascience.com/comparing-vader-and-text-blob-to-human-sentiment-77068cf73982. [Accessed: 23- Mar- 2022].

[10]S. Dan, "Social media statistics in the Philippines", *Talkwalker*, 2019. [Online]. Available: https://www.talkwalker.com/blog/social-media-statistics-philippines. [Accessed: 23- Mar- 2022].

[11]S. Srivastava, "Machine Translation(Encoder-Decoder Model) !!", *Medium*, 2019. [Online]. Available: https://medium.com/analytics-vidhya/machine-translation-encoder-decoder-model-7e4867377161. [Accessed: 23- Mar- 2022].

[12]I. Witten, E. Frank, M. Hall and C. Pal, "Algorithms: The basic methods: Data Mining (Fourth Edition)", *ScienceDirect*, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128042915000040. [Accessed: 23- Mar- 2022].

[13]R. Kurban, "CNN Sentiment Analysis", *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/cnn-sentiment-analysis-9b1771e7cdd6. [Accessed: 23- Mar- 2022].

[14]J. Degenhard, "Twitter users in the Philippines 2025 | Statista", *Statista*, 2021. [Online]. Available: https://www.statista.com/forecasts/1144992/twitter-users-in-the-philippines. [Accessed: 23- Mar- 2022].

[15]Ö. Genç, "The basics of NLP and real time sentiment analysis with open source tools", *Medium*, 2019. [Online]. Available: https://towardsdatascience.com/real-time-sentiment-analysis-on-social-media-with-open-source-tools-f864ca239afe. [Accessed: 23- Mar- 2022].

[16]G. Preda, "GitHub - gabrielpreda/covid-19-tweets: Covid-19 tweets", GitHub, 2020. [Online]. Available: https://github.com/gabrielpreda/CoViD-19-tweets. [Accessed: 23- Mar- 2022].

[17]A. Ablazo, Designing a Web Application using the Twitter streaming API as a Tracking Tool for Weather Updates.. 2019.

[18]K. Schneider, "Techniques for improving the performance of naive Bayes for text classification", Citeseerx.ist.psu.edu, 2005. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.2085&rep=rep1&type=pdf. [Accessed: 23- Mar- 2022].

[19]Y. Goldberg, *Neural network methods for Natural Language Processing*. San Rafael, CA: Morgan & Claypool, 2017.

[20]J. Brownlee, "What are word embeddings for text?," *Machine Learning Mastery*, 07-Aug-2019. [Online]. Available: https://machinelearningmastery.com/what-are-word-embeddings/. [Accessed: 24-Mar-2022].

[21]K. Tago, K. Takagi, and Q. Jin, "Polarity classification of tweets considering the poster's emotional change by a combination of naive Bayes and LSTM," *Computational Science and Its Applications – ICCSA 2019*, pp. 579–588, 2019.

[22]K.-M. S. U. of Passau, K.-M. Schneider, U. of Passau, U. Kingdom, C. Republic, and O. M. V. A. Metrics, "A comparison of event models for naive Bayes anti-spam e-mail filtering: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - volume 1," *DL Hosted proceedings*, 01-Apr-2003. [Online]. Available: https://dl.acm.org/doi/10.3115/1067807.1067848. [Accessed: 24-Mar-2022].

[23]N. Chen and P. Wang, "Advanced Combined LSTM-CNN Model for Twitter Sentiment Analysis", Researchgate, 2018. [Online]. Available: https://www.researchgate.net/publication/332432775_Advanced_Combined_LSTM-CNN_Model_for_Twitter_Sentiment_Analysis. [Accessed: 24- Mar- 2022].

[24]Y. Kim, J. Kim, W. Kim, J. Im, T. Kim and S. Kang, "Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies", Plos One, 2016. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0161197. [Accessed: 24- Mar- 2022].

[25]C. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text", Proceedings of the International AAAI Conference on Web and Social Media, 2014. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14550. [Accessed: 24- Mar- 2022].

[26]A. Tokuç, "Splitting a Dataset into Train and Test Sets | Baeldung on Computer Science", Baeldung on Computer Science, 2021. [Online]. Available: https://www.baeldung.com/cs/train-test-datasets-ratio. [Accessed: 24- Mar- 2022].

[27]Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean."Efficient estimation of word representations in vector space." arXiv preprint arXiv: 1301.3781 (2013).

[28]Pouransari, Hadi, and Saman Ghili. Deep learning for sentiment analysis of movie reviews. Technical report, Stanford University, 2014.

[29]D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment specific word embedding for twitter sentiment classification," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, vol. 1, 2014, pp. 1555–1565.

[30]B. Xue, C. Fu, and Z. Shaobin, “A study on sentiment computing and classification of sina weibo with word2vec,” in Big Data (BigData Congress), 2014 IEEE International Congress on. IEEE, 2014, pp. 358–363.

[31]D. Zhang, H. Xu, Z. Su, and Y. Xu, “Chinese comments sentiment classification based on word2vec and svm perf,” Expert Systems with Applications, vol. 42, no. 4, pp. 1857–1863, 2015.

[32]Chintalapudi N, Battineni G, Amenta F. Sentimental Analysis of COVID-19 Tweets Using Deep Learning Models. Infect Dis Rep. 2021;13(2):329-339. Published 2021 Apr 1. doi:10.3390/idr13020032

[33]"Frequently Asked Questions (FAQs) on Virtual Currencies", Bsp.gov.ph. [Online]. Available: https://www.bsp.gov.ph/Media_and_Research/Primers%20Faqs/VC.pdf. [Accessed: 25- Mar- 2022].

[34]R. Francisco, N. Rodelas and J. Ubaldo, "The Perception of Filipinos on the Advent of Cryptocurrency and Non-Fungible Token (NFT) Games", Arxiv.org, 2022. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/2202/2202.07467.pdf. [Accessed: 25- Mar- 2022].

[35]J. England, "Cryptocurrency to become mainstream in the Philippines", fintechmagazine.com, 2021. [Online]. Available: https://fintechmagazine.com/digital-payments/cryptocurrency-become-mainstream-philippines. [Accessed: 25- Mar- 2022].

[36]“Fleiss' kappa in SPSS statistics,” *Fleiss' kappa in SPSS Statistics | Laerd Statistics*. [Online]. Available: https://statistics.laerd.com/spss-tutorials/fleiss-kappa-in-spss-statistics.php. [Accessed: 28-Mar-2022].

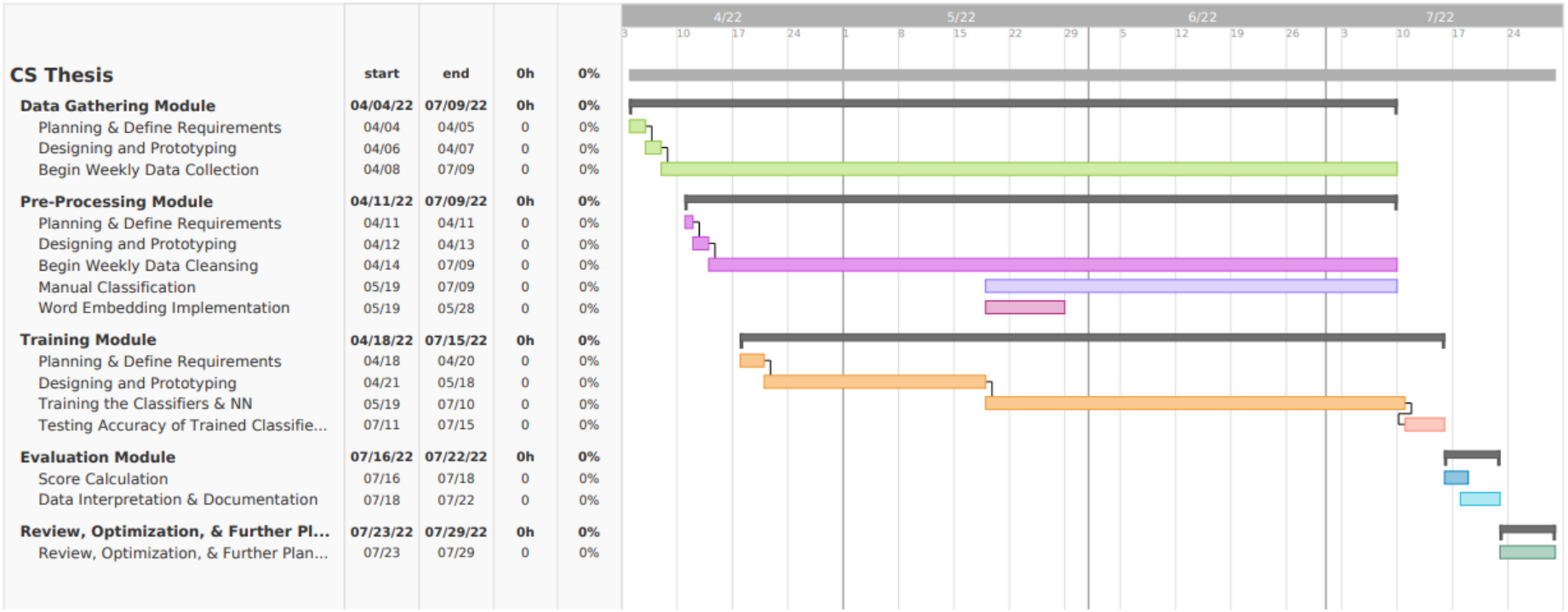
[37]Rahuljha, “Word2Vec implementation,” *Medium*, 29-Jun-2020. [Online]. Available: https://towardsdatascience.com/a-word2vec-implementation-using-numpy-and-python-d256cf0e5f28. [Accessed: 28-Mar-2022].

[38]R. Newatia, “How to implement CNN for NLP tasks like sentence classification,” *Medium*, 30-May-2019. [Online]. Available: https://medium.com/saarthi-ai/sentence-classification-using-convolutional-neural-networks-ddad72c7048c. [Accessed: 28-Mar-2022].

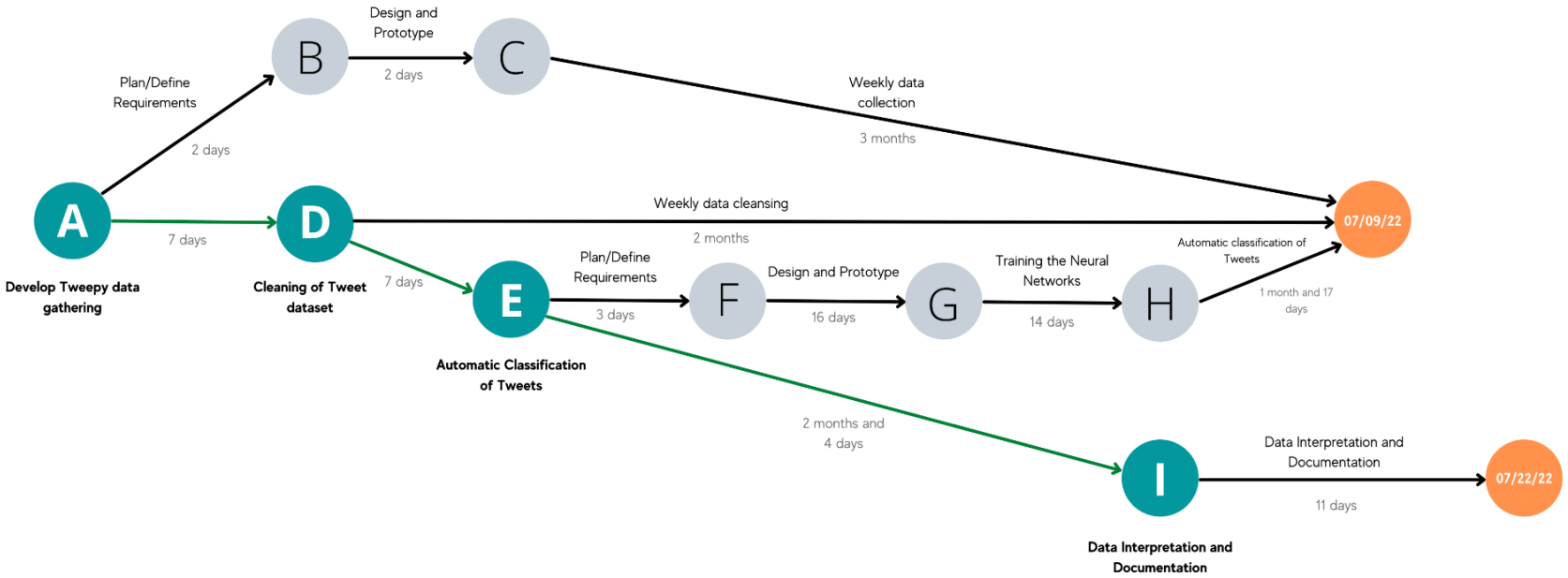
[39]“Rajat Newatia,” *Medium*. [Online]. Available: https://medium.com/@rajatnewatia. [Accessed: 28-Mar-2022].

[40]Ogspeace, “Twitter-scraper/tweetscraper.py at master · ogspeace/twitter-scraper,” GitHub, 05-Mar-2020. [Online]. Available: https://github.com/ogspeace/twitter-scraper/blob/master/tweetscraper.py. [Accessed: 28-Mar-2022].

Table of Thesis Completion



CS Thesis	start	end	0h	0%
Data Gathering Module	**04/04/22**	**07/09/22**	**0h**	**0%**
Planning & Define Requirements	04/04	04/05	0	0%
Designing and Prototyping	04/06	04/07	0	0%
Begin Weekly Data Collection	04/08	07/09	0	0%
Pre-Processing Module	**04/11/22**	**07/09/22**	**0h**	**0%**
Planning & Define Requirements	04/11	04/11	0	0%
Designing and Prototyping	04/12	04/13	0	0%
Begin Weekly Data Cleansing	04/14	07/09	0	0%
Manual Classification	05/19	07/09	0	0%
Word Embedding Implementation	05/19	05/28	0	0%
Training Module	**04/18/22**	**07/15/22**	**0h**	**0%**
Planning & Define Requirements	04/18	04/20	0	0%
Designing and Prototyping	04/21	05/18	0	0%
Training the Classifiers & NN	05/19	07/10	0	0%
Testing Accuracy of Trained Classifie...	07/11	07/15	0	0%
Evaluation Module	**07/16/22**	**07/22/22**	**0h**	**0%**
Score Calculation	07/16	07/18	0	0%
Data Interpretation & Documentation	07/18	07/22	0	0%
Review, Optimization, & Further Pl...	**07/23/22**	**07/29/22**	**0h**	**0%**
Review, Optimization, & Further Plan...	07/23	07/29	0	0%


A
Develop Tweepy data gathering
7 days
B
Plan/Define Requirements
2 days
Design and Prototype
2 days
C
Weekly data collection
3 months
D
Cleaning of Tweet dataset
7 days
Weekly data cleansing
2 months
07/09/22
E
Automatic Classification of Tweets
Plan/Define Requirements
3 days
F
Design and Prototype
16 days
G
Training the Neural Networks
14 days
H
Automatic classification of Tweets
1 month and 17 days
2 months and 4 days
I
Data Interpretation and Documentation
11 days
07/22/22
Data Interpretation and Documentation