2023

XRP Price Prediction using NLP and LSTM Network Model

WORD COUNT: 2992 (EXCLUDING CODE, FIGURES, TABLES & REFERENCES)

JAYDEN DZIERBICKI

Contents

Git-Repository

Documentation and code can be located in a Git-repository, refer for technical specifications:

https://github.com/jaydendzierbicki/webcrawl-long-short-term-memory-network

Introduction and Proposed Model

The emergence of cryptocurrency and blockchain technology has captured the attention of investors. Cryptocurrency is a digital currency which relies on robust cryptography to validate and secure financial transactions. Ripple is a fintech company which created the Ripple payment protocol and exchange network, focusing on crypto solutions for businesses and central banks through their XRP coin. However, like other cryptocurrencies, XRP experiences significant price fluctuations, presenting risks and uncertainties for investors at all levels. To help investors make informed decisions, market movement prediction systems have been developed previously (Lahmiri & Bekiros, 2019). Traditional supervised learning algorithms have previously been used to predict cryptocurrency changes based on historical price data, but predicting price fluctuations can be difficult due to the efficient market hypothesis, which suggests that markets always follow random patterns (Jaquart et al., 2021). Consequently, investors often actively monitor discussion boards to glean insights that may signal upcoming market movements; though task can be daunting, and data science can be utilised to create an automated tool to support investors. To address this challenge, we propose a model that uses natural language processing (NLP) feature extraction with word2vec to attempt to predict closing price within a specific timeframe, utilizing long short-term memory (LSTM) as well as utilizing lagged variables from the previous day such as high, low and volume. This approach has seen accuracy on validation sets in related studies exceed 50% for all analyzed stocks, which utilized word2vec to predict stock price movement through NLP on news Headlines (Chandola et al., 2022). The focus is on XRP, given its active community, substantial market presence, and the ongoing developments in the SEC vs. Ripple case in the United States, which has attracted extensive media coverage. The model aims to serve novice, intermediate, and experienced investors by gathering data from various sources, such as Yahoo Finance and Investing.com, using web crawling and scraping techniques. The data is preprocessed and transformed to prepare it for the LSTM network.

Model Workflow

In Figure 1, we present an overview of the data pipelines we propose to achieve our objective. These pipelines comprise three key phases: web crawling and initial data storage, NLP combined with data wrangling and explorative data analytics (EDA), and machine learning along with the output layer. Throughout the report, we will discuss each of these phases in detail. Through utilizing a LSTM network model we hypothesis that we will be able to predict XRP price close price with some level of accuracy similar to different neural network models citing an RMSE of 0.0499 for XRP (Lahmiri & Bekiros, 2019), this will allow us to contribute further to this field of research.

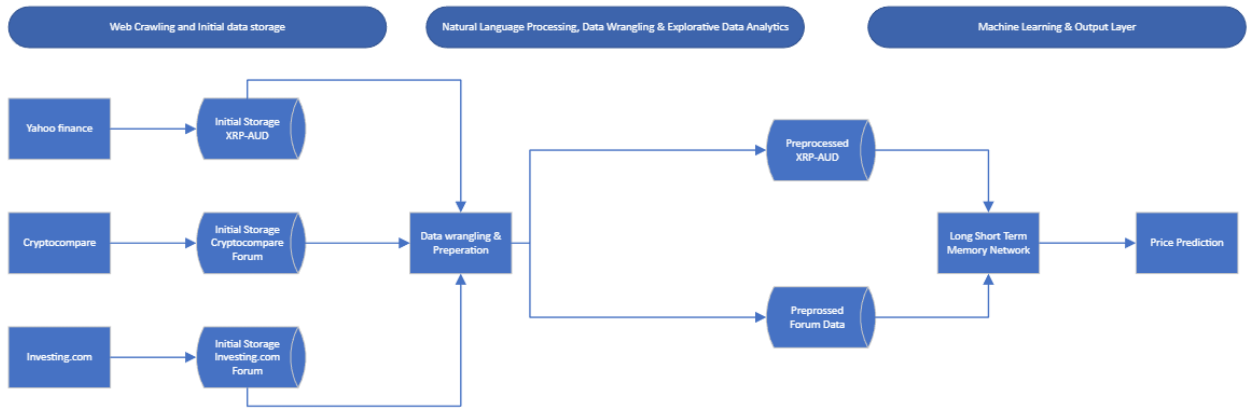

Figure 1: A schematic representation of the pipeline, from left to right: Data is scraped from various sources and saved into landing tables in MySQL. The data undergoes wrangling, preparation and EDA before being stored in preprocessed tables in

MySQL. The two input layers, consisting of preprocessed XRP-AUD data and preprocessed forum posts, are fed into a Long Short-Term Memory (LSTM) network for price prediction.

Web crawling

Web crawling and scraping are terms that are often used interchangeably, but they represent different concepts. Web scraping involves extracting data from a specific website, where the target site is known, and the data is obtained by inspecting the HTML elements of the webpage (Perez, 2023). However, web crawling refers to the automated process of navigating the internet to download or index content from multiple websites or URLs. Many sites implement technologies, such as Cloudflare to prevent unauthorized crawling (Gillis, 2022). It is essential to review a website's terms of service before engaging in web crawling or scraping activities, as some sites may have specific rules or restrictions in place which could hinder the automation.

Methodology

In each domain, we employed a similar extract-transform-load (ETL) methodology. Data was extracted from the target website and underwent minor transformations, such as adding an ID and splitting datetime into date and time components, before being loaded into a designated MySQL table. This method offers efficient data storage and management, particularly for large datasets. MySQL Workbench provides scalability, accommodating growing data volumes and Python integration. Through leveraging these storage benefits and scalability, our data infrastructure is both robust and ready for future expansion or the inclusion of new domains, acting as a central repository.

Domains and workflow

The goal of this section is to collect data from three sources for our LSTM network model's two primary input layers: one containing combined online forum posts, and the other featuring XRP-AUD financial time series data. We sought posts from two forums to minimize biases and include a diverse user base of novice, intermediate, and experienced investors. If time had permitted, we would have attempted to include social media data, like seen in past studies (Vo, 2019). Table 1 summarizes the explored domains, covering terms of service considerations, available data, and website limitations. For investing.com, we emailed the website owners, informing them of our intent to scrape data and ensuring compliance with their terms of service. All data was scraped according to the terms and was publicly available at the time of writing for anyone to access.

Table 1: This table provides an overview of the two websites/domains targeted by the investingCrawler and cryptocompareCrawler web crawlers. It outlines the specific terms of service for each website, the available data that can be scraped, and the limitations faced during the data extraction process.

Website/Domain	Terms of Service	Available Data	Limitations
https://www.cryptocompare.com/coins/xrp/forum	Not prohibited in terms and conditions	Date of post Post by user	Dynamic page which requires ability to scroll down, unsure how far data goes back
https://www.investing.com/crypto/xrp/chat	Prohibits scrapping usernames, and requires prior approval	Date of post Post by user	Need to navigate multiple pages automatically and adhere to terms of services such as seeking prior approval and not scaping user information

Forum Data Scrapping

The investingCrawler and cryptocompareCrawler are custom web crawlers designed to scrape comments and their respective dates from specific websites. Although they share several similarities

in terms of the technology components and libraries they utilize, they target different websites with distinct structures, and their methodologies for accessing the data differ. Table 2 highlights the similarities and differences between the two crawlers which is further described in detail.

Table 2: This table presents a comparison of the investingCrawler and cryptocompareCrawler web crawlers, highlighting the key differences and similarities in their technology components, target website structures, methodologies, and data storage.

Feature	Investing.comg Crawler	Cryptocompare Crawler
Libraries	Selenium, Chrome WebDriver, BeautifulSoup, pandas	Selenium, Chrome WebDriver, BeautifulSoup, pandas
Target Website Structure	"commentInnerWrapper" class	"post-content" class
Data Location	"span" tags with "js-date" and "js-text" classes	"div" tags with "content-body" and "item-ago ng-binding" classes
Methedology	Iterating through pages	Scrolling through pages for a set duration
Data Storage	Pandas DataFrame, CSV file and stored in mySQL	Pandas DataFrame, CSV file and stored in mySQL

Both crawlers utilise multiple Python libraries such as Selenium, Chrome WebDriver, BeautifulSoup, and pandas. Selenium and Chrome WebDriver are used to load and interact with dynamic web pages, making them ideal choices for handling JavaScript rendering. BeautifulSoup is employed to parse and extract data from the HTML content of a webpage, while pandas is utilized to store the scraped data in a DataFrame, allowing for easy manipulation.

The investingCrawler targets investing.com, where comments and dates are stored within elements of the "commentInnerWrapper" class. The data is located within specific elements ("span" tags) with corresponding class names "js-date" and "js-text." The crawler follows a sequence of methods, including initializing the crawler, iterating through the pages, extracting data from the page, saving the DataFrame to a CSV file, and closing the WebDriver session. It is designed to crawl all 1222 pages of the targeted website with a wait time of 15 seconds set using the implicitly_wait() method to ensure the content is loaded before extracting the data and prevent server strain.

On the other hand, the cryptocompareCrawler targets a website with a different structure, cryptocompare.com, where comments and dates are stored within elements of the "post-content" class. The data is located within specific elements ("div" tags) with corresponding class names "content-body" and "item-ago ng-binding." The crawler's methodology includes initializing the crawler, scrolling through the pages for a set duration, extracting data from the page, saving the DataFrame to a CSV file, and closing the WebDriver session. It is designed to scroll through the targeted website for a specified duration, loading and extracting data from the dynamically loaded content, with a scroll interval of 6 seconds set to ensure the content is loaded before extracting the data.

Historic Price Data

To acquire the historical price data of XRP-AUD, which is essential for our model, we utilized the yahoo_fin package in Python. By executing the command get_data("XRP-AUD", start_date="01/01/2017", end_date="15/04/2023", index_as_date = False, interval="1d") in Python, we successfully retrieved daily XRP-AUD data ranging from November 10, 2017 to April 14, 2023. This data contained information on date, opening price, highest price, lowest price, closing price, adjusted closing price, and volume. We then processed the data and imported it into MySQL. This serves as a vital component, as it represents the actual price for a specific day in our machine learning model. It is important to note that the specific time zone employed is not explicitly stated; however, we hypothesize that the data is reported in UTC, as suggested by the XRP price page (XRP AUD (XRP-AUD) price history; historical data 2023).

A possible limitation of this study is that we are unable to confirm the time zone of forum posts associated with our custom web crawler, and would suggest the need to explore this on future

iterations as there might be an overlap between time zones and dates. Due to time constraints we were unable to explore this option further and highlights the trade-off between efficiency and loss of possible accuracy in any machine learning model.

Data Wrangling

Corpus Data Wrangling Methods & Possible Limitations/biases

We loaded all raw data into Python and analysed the earliest and latest dates in each table to determine suitable filtering criteria for data processing and corpus handling. To ensure a comprehensive representation of users across various platforms, as previously mentioned. We decided to exclude any data prior to 2021-02-02 and up to 2023-04-14. This choice was based on investingcom_xrp containing data from 2018-03-16, while cryptocompare_xrp only had data since 2021-02-02. We also eliminated rows with empty comments, resulting in the removal of 43 rows from cryptocompare_xrp and 129 rows from investingcom_xrp. To address spam comments, we identified potential spam as instances where a user posted the same comment multiple times on the same day. This approach led to the removal of 375 duplicates from cryptocompare_xrp and 16 duplicates from investingcom_xrp, with examples of duplicate comments displayed in Figure 2.

Duplicate preview for cleaned_cryptocompare_xrp

	date	comment	count
124	2021-04-13	lol	6
96	2021-04-10	lol	5
38	2021-02-21	shut up	4
105	2021-04-11	My newbie opinion: if price break 1.3200, it m...	4
25	2021-02-14	Guys, how long do u suppose this place of resi...	4
..	...	...	...
118	2021-04-13	Im not selling a penny till we beat the SEC an...	2
117	2021-04-13	Here we go again	2
116	2021-04-13	HODL💞	2
115	2021-04-13	Bitso	2
337	2022-11-18	Exactly! Even 10% of swift business puts this ...	2

Duplicate preview for cleaned_investingcom_xrp

	date	comment	count
9	2022-01-22	#freeveyor	7
0	2021-04-02	I want to Post a sun downer Photo of my flat b...	3
1	2021-04-14	run..	2
2	2021-05-04	Not the best start to a tuesday.. I'm still re...	2
3	2021-05-08	The moment of 1000 truths	2
4	2021-05-16	Today's a good day?	2
5	2021-05-17	;-)	2
6	2021-06-29	Pump incoming\t\t\t\t\t\tView Edits	2
7	2021-10-27	.\t\t\t\t\t\tView Edits	2
8	2021-12-22	Support	2

Figure 2: Examples of duplicates from cryptocompare_xrp (left) and investingcom_xrp (right)

We combined our forum post data into a single dataframe using the concat function, retaining the date, source, and comment associated with each post. We found the average number of posts per day between 2021-02-02 and 2023-04-14 was 24.84, with a median of 8.50 posts per day. However, it is suggested that for robust timeseries models predicting stock price there should be data points available for every hour (Hilpisch, 2015) highlighting a possible limitation in our study which might comprise our ability to accurately predict XRP price. In Figure 3, we notice a significant spike in XRP related online discussions between 2021-01 and 2021-05, which coincides with major announcements regarding the SEC vs Ripple lawsuit (Cointelegraph, ND). We observe that cryptocompare.com generally has more daily posts than investing.com across most timeframes which could potentially result in bias in the model as it may disproportionately influence the overall analysis; if time persisted we would suggest web crawling more websites to capture different users. In addition, all our posts are associated with English speaking investors only. This highlights the complex challenge of addressing limitations in any machine learning application and that limitations are fundamentally part of any model.

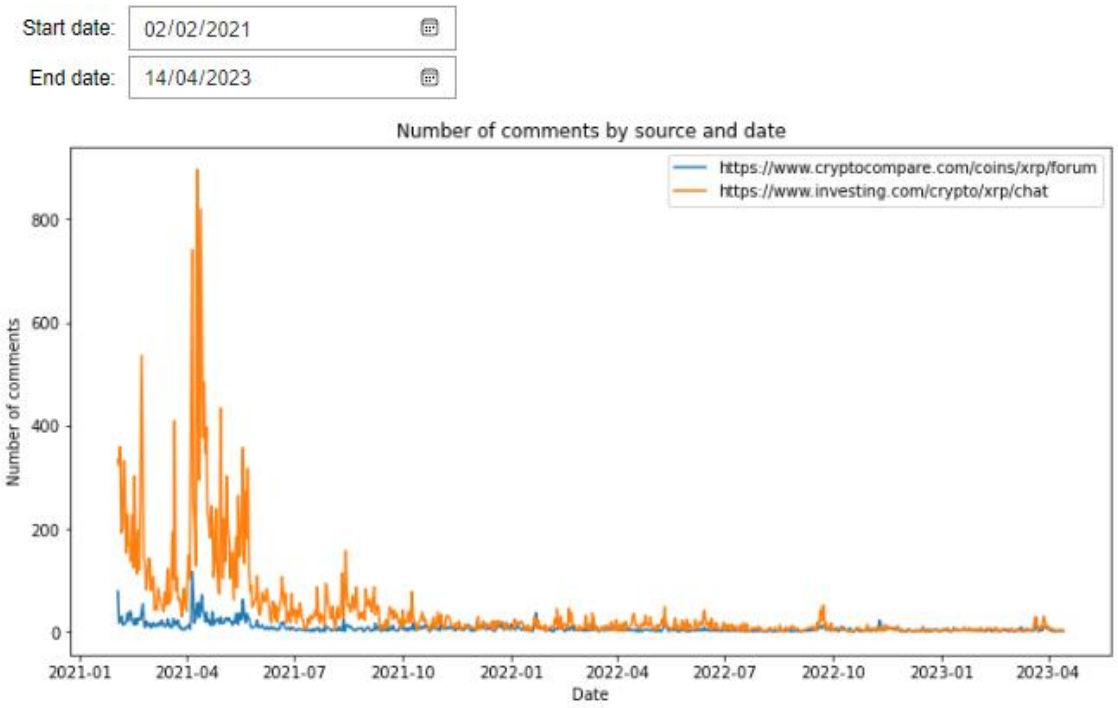

Figure 3: This graph displays daily forum posts on XRP from 2021-02-02 to 2023-04-14, with an average of 24.84 and a median of 8.50. It shows a spike in discussions between 2021-01 and 2021-05, coinciding with the SEC vs Ripple lawsuit developments. The graph also indicates that investing.com typically has more daily posts than cryptocompare.com which could suggest in a bias

It is widely acknowledged that models cannot directly utilize raw text data; therefore, it is essential to preprocess the text data. Following a similar approach to prior studies that aimed to predict stock and cryptocurrency price movements using the Natural Language Toolkit (NLTK) in Python (Saxena et al., 2023), we implemented the subsequent text cleaning steps summarised in table 3:

Table 3: summarizes the techniques used to preprocess the text data. These include removing punctuation and converting text to lowercase, tokenizing the text, lemmatizing the tokens, stemming the tokens, and removing stop words.

Method	Summary
Lowercase and Punctuation Removal	Text is converted to lowercase using lower(). Punctuation is removed using str.maketrans() to map punctuation characters to spaces, and translate() to apply the mapping. This approach ensures consistency by treating words with varying capitalization as identical and reduces noise in the data, simplifying the text for further processing.
Tokenization	Tokenization is a technique that allows a text to be expressed as a series of individual units. For example, by tokenizing the sentence "XRP to the moon", we obtain a list of tokens such as [XRP, to, the, moon]. Each token corresponds to a specific unit of meaning, and this approach makes it easier to analyse and manipulate the text. In this particular case, the word_tokenize() function from the nltk package was used to achieve the tokenization.
Lemmatization	The text was lemmatized using the WordNetLemmatizer from the Natural Language Toolkit (NLTK). The lemmatizer was instantiated, and then each token in the list of tokens was lemmatized by applying the lemmatize() function. This process transforms words to their base or dictionary form, reducing inflected or derived word forms to a common lemma, which helps in simplifying the text data and improving the efficiency of text analysis tasks.
Stemming	Stemming is a technique in natural language processing that reduces words to their base form, such as "boat", "boating", and "boater" being stemmed to "boat". This helps reduces the number of unique words in a text corpus, making it easier to analyze and compare different texts, and group together words with similar meanings. We used the PorterStemmer class from the nltk library to apply the Porter stemming algorithm to a list of tokens generated by the word_tokenize() function, using its stem() method to produce the list of stemmed tokens.

Stop word removal	Stop word removal is another technique used in natural language processing that involves removing commonly used words that do not contribute much to the meaning of a sentence, such as "the", "and", "in", etc. These words are known as stop words and removing them can help reduce noise in the data and improve the performance of text analysis algorithms. We used the NLTK library's list of English stop words to remove them from our text corpus. This was done after tokenizing and stemming the text data, using the list of stop words to filter out those tokens that were considered stop words.

Feature Extraction & Hyperparameters

In our research, we explored various word embedding methods, including word2vec, skip-gram model, bag of words, and sentiment analysis (Chandola et al., 2022; Al-Saqqa & Awajan, 2019). However, we opted for word2vec due to its widespread use in word embedding, particularly in stock and cryptocurrency price forecasting (Chandola et al., 2022; Al-Saqqa & Awajan, 2019; Lahmiri & Bekiros, 2019). Word2vec enables us to represent words as semantically related vectors (Al-Saqqa & Awajan, 2019) and with the abundance of online resources available for constructing this model it also highlights the considerable support it has within the field of NLP.

In our study, we adopted a thorough approach for feature extraction and model training, using specific hyperparameters with these steps outlined and summarized in Table 4.

Table 4 summarizes the steps involved in training a Word2Vec model on the tokenized data, including creating a callback to monitor training losses, building the vocabulary, training the model, determining optimal epochs, and defining a function to calculate average word vectors. The resulting average word vectors are saved as new columns in the DataFrame.

Step	Description
1	Import necessary libraries and modules, such as Word2Vec and CallbackAny2Vec from Gensim, and other modules for data preprocessing and visualization.
2	Create a callback class to monitor and plot training losses over time, including on_epoch_end and plot_losses methods.
3	Instantiate a Word2Vec model with specific hyperparameters: vector_size=200, window=15, min_count=2, workers=1, sg=1, negative=5, sample=1e-5, seed=42. (Seed for reproducibility)
4	Build the vocabulary for the Word2Vec model using tokenized_data.
5	Create a callback class instance to track training losses.
6	Train the Word2Vec model for 1501 epochs using w2v_model.train() and employ the callback to compute and display training loss over time.
7	Plot the training losses and calculate the total elapsed time for the training process.
8	Determine the optimal number of epochs (400) for model performance and training time balance, and retrain the Word2Vec model.
9	Define a function average_word_vectors to calculate the average word vector for a list of tokens.
10	Set vector_size to 200, matching the vector size of the Word2Vec model.
11	Iterate through the vector size range, applying the average_word_vectors function to the 'cleaned_comment_stop' column in the DataFrame to create new columns for each dimension of the average word vectors.

Historical Price

To prepare for our LSTM model, we combined XRP's closing and opening prices with our 'df' using the merge() function, in addition we also combined high, low and volume and lagged it by one time period. We used the closing price as our primary reference and included the opening price and lagged values as a feature since it is essential information that traders have on hand and is considered vital for predicting the closing price more accurately (Lahmiri & Bekiros, 2019). The data was then stored in MySQL and would later be used for the intended machine learning task.

Explorative Data Analytics

We undertook EDA on our corpus of text after the preprocessing was completed. In addition to the EDA, we assessed the quality of our optimized Word2Vec model by examining word similarity. The results aligned with our domain knowledge, indicating that the model learned meaningful semantic relationships. For example, the term 'XRP' was associated with 'ripple' and 'coin', as expected, as

illustrated in Figure 4 which was true for all testes terms. This suggests that the Word2Vec model has effectively captured the underlying relationships between words in the given context.

```
Top 3 words similar to xrp: [('rippl', 0.7138864398002625), ('show', 0.6888630986213684), ('coin', 0.6730276346206665)]
Top 3 words similar to sec: [('rippl', 0.8073084950447083), ('case', 0.804876983165741), ('lawsuit', 0.7215989828109741)]
Top 3 words similar to meme: [('unaccept', 0.5286688804626465), ('rap', 0.5117706656455994), ('boomer', 0.4783424735069275)]
Top 3 words similar to gensler: [('gari', 0.7387188673019409), ('nomine', 0.5521851778030396), ('faction', 0.4737000465393066
4)]
Top 3 words similar to hodl: [('winer', 0.5292115211486816), ('att', 0.44229626655578613), ('✈️❤️', 0.4250231981277466)]
Top 3 words similar to lab: [('courtsouthern', 0.5044182538986206), ('✴️✴️✴️', 0.48758387565612793), ('gape', 0.46044936776161
194)]
```

Figure 4: Within the Word2Vec model, we noted associations such as 'sec' with 'ripple', 'case', and 'lawsuit'; 'meme' with 'boomer'; and 'gensler' with 'Gary'. These connections showcase the model's effectiveness in capturing meaningful relationships between words, particularly in the context of our domain knowledge regarding the XRP crypto space.

We examined the top 40-word frequency distribution and observed the expected results, with the most frequently occurring word in the corpus being associated with XRP and mention of the SEC case (figure 5). Furthermore, we also noticed evidence of Zipf's law, which states that the frequency of a word in a corpus is inversely proportional to its rank. This is a common characteristic of natural language corpus, and our analysis confirms its presence in our corpus, this approach is much more informative than a word cloud in our opinion which we explored but excluded from the report as it is targeted to more non-technical users.

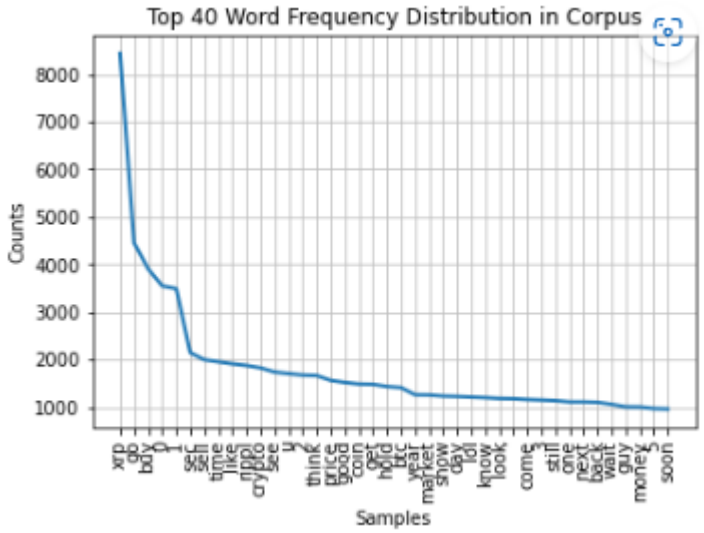

Figure 5: Shows the top 40-word frequency distribution in the corpus, indicating that the most commonly used word is related to XRP on two forum discussion boards between 02/02/2021 and 14/04/2021.

In analysing a word2vec model, it is crucial to comprehend the underlying distribution of the word vectors as this can shed light on the model's quality and facilitate optimization for targeted NLP tasks. To visualize the distribution of a word2vec model, the plot() function from the gensim library can be utilised. The resulting histogram displays the frequency of word vector values as seen in figure 6, with a normal distribution being preferable since it indicates that the model has effectively captured the language corpus's inherent structure (Mu, ND). By examining the distribution of word vectors in a word2vec model, we can ascertain its suitability for the intended NLP task.

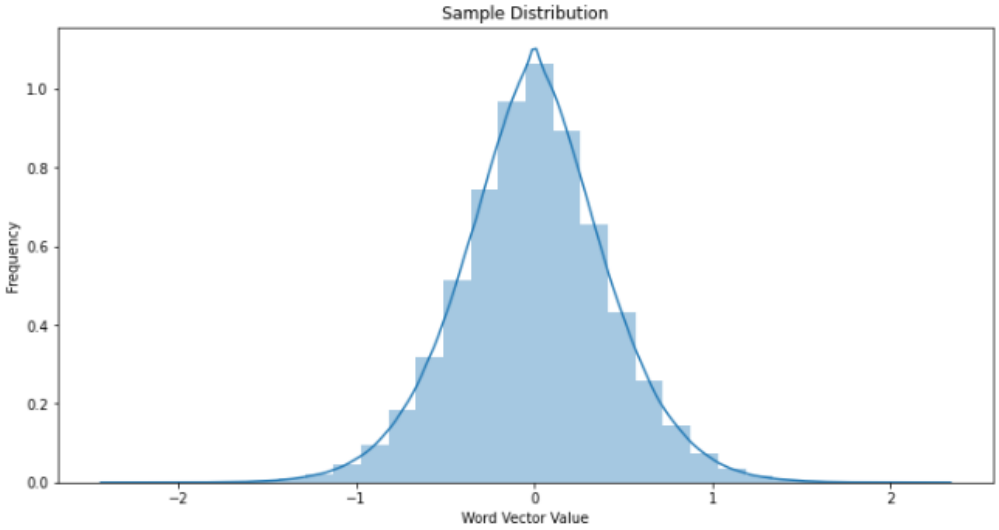

Figure 6: Demonstrates the distribution of word vectors in a Word2Vec model, which is important in understanding the quality of the model and optimizing it for specific NLP tasks.

Machine Learning

In this study, we aimed to predict XRP closing prices based on online forum posts throughout the day, which can indirectly help investors make informed decisions about price movement. Various approaches have been used in the past, such as predicting the price itself or the direction of price movement (up or down) relative to the opening price (Chandola et al., 2022; Vo, 2019). We imported our pre-processed data directly from MySQL and followed the steps outlined in Table 5 to create and train our LSTM model. The data was split into a testing and validation data set with 204 features each; and 35,015 observations for training and 3,891 for validation, with the split resulting on the newest 20% being used for validation in order of date.

Table 5: Outlines the steps taken for training and evaluating the LSTM model, including data preprocessing, hyperparameter optimization using grid search, model training and evaluation, and result visualization.

Step	Description
1	Set seed for reproducibility
2	Separate features and target variables
3	Split data into training and testing sets, were 20% testing was used on the latest data
4	Scale features and target variables
5	Reshape input data
6	Define create_model function
7	Perfomre grid search for hyperparameter optimisation
8	Train and evaluate the model with best hyperparameters
9	Plot training and validation losses
10	Make predictions on test data and inverse-transform prediction
11	Compute RMSE
12	Plot the true values and predictions

To optimize the model at step 7, we employed RandomizedSearchCV with time series cross-validation, exploring the following hyperparameter space which is described in table 6 with the best hypartmers also found and extracted from the RandomizedSearchCV function.

Table 6: Provides a summary of the hyperparameters and their values used in the LSTM model, along with their descriptions and best hyperparameter values.

Hyperparameter	Values	Description	Best hyperparameter
Layers	[1,2,3,4,5]	Number of layers in the LSTM model, determines the depth of the model. More layers can help capture complex patterns but may also increase the risk of overfitting.	3
Neurons	[15, 25, 50, 85, 100, 150, 200, 250, 270, 285, 290, 300, 350, 385, 400]	Number of neurons per layer, controls the model's capacity to learn features. A higher number of neurons can improve the model's ability to learn complex patterns but may increase the risk of overfitting and computational cost.	25
Dropout Rate	[0.2, 0.3, 0.4]	Dropout rate applied after each LSTM layer, used to reduce overfitting by randomly dropping a fraction of neurons during training.	0.25
learning_rate	[0.001, 0.01, 0.1]	Learning rate for the Adam optimizer, determines the step size during gradient descent optimization. A smaller learning rate may yield better convergence but slower training, while a larger learning rate may speed up training but risk overshooting the optimal solution.	0.001
Activation	['relu', 'sigmoid', 'tanh']	Activation function for the LSTM layers and Dense layer. Different activation functions can affect the model's ability to capture non-linear relationships in the data.	tanh

We configured RandomizedSearchCV to perform 48 iterations (n_iter=48) with 3 time series cross-validation splits (TimeSeriesSplit(n_splits=3)). The scoring function used was negative mean squared error, and the search was performed in parallel using all available CPU cores (n_jobs=-1). After completing the grid search, we selected the best set of hyperparameters and trained and evaluated the model using them. To prevent overfitting and improve generalization, we incorporated early stopping with a patience of 10 epochs (early_stopping = EarlyStopping(patience=10, restore_best_weights=True)). We trained the best model with the selected hyperparameters using the early stopping callback and a batch size of 32 for up to 1000 epochs (history = best_model.fit(X_train_reshaped, y_train_scaled, validation_data=(X_test_reshaped, y_test_scaled), epochs=1000, batch_size=32, verbose=1, callbacks=[early_stopping])). As seen in Figure 7, the training loss converged around epoch 2, and we observed oscillation of the validation loss between 0.005 and 0.000. Early stopping occurred at 14 epochs, which helped prevent overfitting and ensured that the model generalizes well on unseen data.

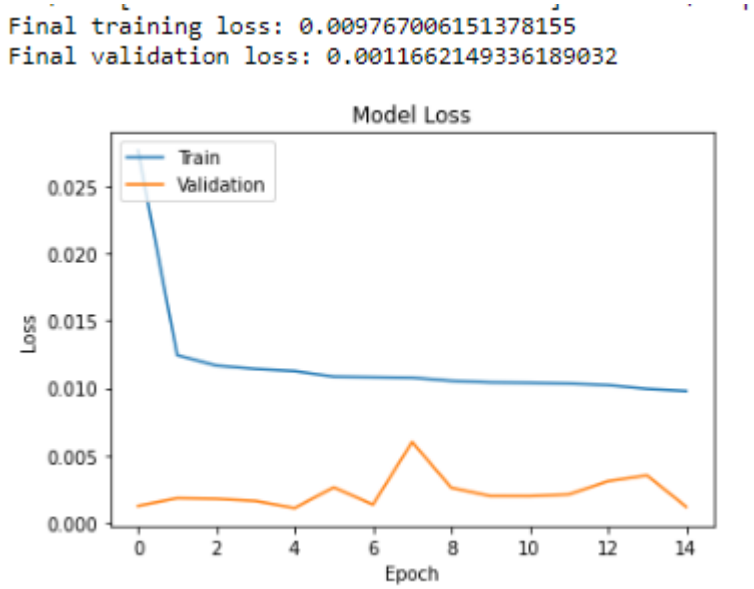

Figure 7: Displays the training and validation loss over the course of the LSTM model training. The plot shows that the training loss converged after about two epochs, while the validation loss oscillated between 0.005 and 0.000. Early stopping was applied at the 14th epoch to prevent overfitting and improve the model's generalization performance on unseen data. This figure highlights the importance of monitoring the training and validation loss during the model training to prevent overfitting and ensure good generalization.

We evaluated the performance of our LSTM model by comparing it to a naïve persistence model, which serves as a baseline, and additionally compared our LSTM against another study that predicted XRP price using a Deep Learning Neural Network (DLNN) and General Regression Neural Network (GRNN) models, which obtained an RMSE of 0.0499 and 0.3115, respectively (Lahmiri & Bekiros, 2019). The naïve persistence model assumes that today's closing price (t) is equal to yesterday's closing price (t-1). Comparing the performance of our LSTM model to this simple baseline allows us to evaluate the performance of our model in predicting XRP prices (Brownlee, 2019), and by comparing it to previous studies, we can assess how LSTM compares and the use of online forums for predicting XRP price. To implement the naïve persistence model, we shifted the closing prices by one time step and calculated the RMSE between the predicted values and the true values. We then plotted the true closing prices, LSTM predicted prices, and naïve model predicted prices to visualize the performance of both models as seen in figure 8.

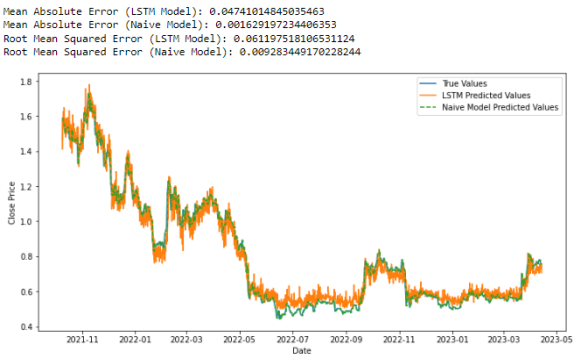

Figure 8: Displays a comparison between the true values, LSTM predicted values, and Naive model predicted values. The Naive model, which simply uses the previous day's closing price as a prediction, is almost identical to the true values. In contrast, the LSTM model's predictions exhibit more noise, but overall are relatively close to the true values. Notably, we observe a period of overestimation between July and September 2022.

This comparison enables us to determine the extent to which our LSTM model improves upon the baseline naïve persistence model, as well as comparing our LSTM model to other studies. The findings are summarised in table 7, where RMSE values are reported for each model. A smaller RMSE value indicates better performance. Our LSTM model achieved an RMSE of 0.0611, which indicates that it outperformed the GRNN model (RMSE of 0.3155) but not the DLNN model (RMSE of 0.0499), both of which were used in previous studies (Lahmiri & Bekiros, 2019). It is important to note that the naïve persistence model achieved an RMSE of only 0.009, indicating that it outperformed all other models reported. Overall, while our LSTM model did not achieve the lowest RMSE value, it still performed significantly better than the GRNN model in previous studies and can be considered a viable option for predicting XRP prices using online forum data. The comparison also highlights the importance of choosing an appropriate baseline model for comparison, as the naïve persistence model outperformed all other models despite its simplicity and highlights the complexity of attempting to predict XRP price.

Table 7: The LSTM model developed in this study showed relatively low RMSE (0.0611) compared to a GRNN model (0.3155) on XRP price prediction using online forum data. However, a DLNN model (0.0499) in a previous study outperformed our LSTM model. The naive model, which predicts the next day's price as the current day's price, had the lowest RMSE of 0.009.

Model	**LTSM Model**	**Naïve Model**	**DLNN (Lahmiri & Bekiros, 2019)**	**GRNN (Lahmiri & Bekiros, 2019)**
RMSE	0.0611	0.009	0.0499	0.3155

In summary, our study shows that web crawling and extracting forum posts from investors can provide valuable data for developing an LSTM model that performs well in comparison to previous models that do not utilize language data. The results suggest that further research could explore the potential of using NLP techniques to enhance the LSTM model further. We also discovered that combining LSTM with other neural network models like DLNN shows potential, but despite our model's success, a naïve persistence model outperformed all other models we examined including our LTSM model. This indicates that more research is needed in this area to improve the effectiveness of LSTM and other models for forecasting XRP prices and the possibility of exploring other features or online forums, though still highlights the potential of this as a tool for investors. We highlight the direct comparison between our LTSM with the DLNN and GRNN is made with caution as they utilised different input layers across a different period.

Reference

cointelegraph (ed.) (no date) The SEC vs. Ripple lawsuit: Everything you need to know, Cointelegraph. Cointelegraph. Available at: https://cointelegraph.com/learn/the-sec-vs-ripple-lawsuit-everything-you-need-to-know (Accessed: April 20, 2023).

Chandola, D. et al. (2022) “Forecasting directional movement of stock prices using Deep Learning,” Annals of Data Science [Preprint]. Available at: https://doi.org/10.1007/s40745-022-00432-6.

Gillis, A.S. (2022) What is a web crawler? everything you need to know from techtarget.com, WhatIs.com. TechTarget. Available at: https://www.techtarget.com/whatis/definition/crawler (Accessed: April 18, 2023).

Mu, J. (no date) Word2vec: What and Why - University of Illinois Urbana-Champaign. Available at: http://pramodv.ece.illinois.edu/qual.pdf (Accessed: April 23, 2023).

Hilpisch, Y. (2015) Python for Finance Analyze Big Financial Data. Beijing: O'Reilly.

Lahmiri, S. and Bekiros, S. (2019) “Cryptocurrency forecasting with deep learning chaotic neural networks,” Chaos, Solitons & Fractals, 118, pp. 35–40. Available at: https://doi.org/10.1016/j.chaos.2018.11.014.

Perez, M. (2023) What is web scraping and what is it used for?, ParseHub. Web Scraping Blog (Tips, Guides + Tutorials) | ParseHub. Available at: https://www.parsehub.com/blog/what-is-web-scraping/ (Accessed: April 18, 2023).

Saxena, A. et al. (2023) “Sentiment analysis of stocks based on news headlines using NLP,” Atlantis Highlights in Intelligent Systems, pp. 124–135. Available at: https://doi.org/10.2991/978-94-6463-074-9_12.

XRP AUD (XRP-AUD) price history & historical data (2023) Yahoo! Finance. Yahoo! Available at: https://finance.yahoo.com/quote/XRP-AUD/history?p=XRP-AUD (Accessed: April 18, 2023).