NaratAI Sentiment Screener

Abstract

This project targets the problem of retail investors lacking tools that provide high-quality synthesized information to assist their investment decisions. The functionality of the software centres on giving investors an understanding about one of the most important soft data predictors of price movements – news sentiment. The system allows users to quickly gauge the prevailing sentiment for a given asset and also understand the topics that are driving it by extracting key phrases and sentiment-carrying words (polarity words). The functionality is enabled by a state-of-the art language architecture coupled with novel logic. News articles' sentiment is extracted using a Bidirectional Encoder Representations from Transformers (BERT) model and text tiling – an approach that has not previously been used in literature. The system is able to score 83% of news articles correctly using the described method. The model is fine-tuned on domain specific data and also used for key phrases extraction. Lastly, polarity words are extracted using sentiment lexicons and heuristic rules based on seminal research.

Contents

List of Figures

1 Introduction

1.1 Motivation

In his book 'Narrative Economics' Nobel prize winning economists Robert Schiller discussed the power of narratives and sentiment conveyed through news in determining stock market price movements (Schiller , 2019). Since then, numerous other research has confirmed the strong correlation between news sentiment and asset prices highlighting the crucial role these factors play in investment decisions (Garcia, 2013; Li, et al., 2014; Heston & Sinha, 2016). Large institutional investors have been harnessing the power of machine learning and natural language processing to extract such analytics from unstructured data in order to inform their actions. An interesting fact proving the use of unstructured news data in algorithmic trading is that every time actress Ann Hathaway's name appears in the news, there is an immediate response in the price of Warren Buffet's Berkshire Hathaway – two entities that have nothing in common but their name similarity.

More recently, with information becoming more accessible and proliferated, the importance of processing the semantic meaning news communicate has only grown. Partaking in markets has never been easier and the rapidly growing pool of private investors (interchangeably referred to as retail investors) are also seeking every tool available to enhance their chances of profitability. A limiting factor that still places them at a significant disadvantage compared to larger market players is the lack of accessible tools for deriving meaningful analytics from unstructured data. More specifically, quantifying sentiment and understanding its determinants remains an unsolved problem in the market for software solutions assisting investors.

1.2 Problem Statement

The goal of this projects centres around building a platform where retails investors can quickly gauge the current 'narrative' for a given stock or crypto currency (together referred to as assets) derived from the available daily news for it. The software aims to provide a practical tool for investors, the backbone of which is state-of-the art research and architectural advances in the field of natural language understanding. Users will be able to receive real time information about the sentiment of stocks and cryptocurrencies and also track historical values. Additionally, a deeper understanding of the topics driving the prevailing sentiment towards a given asset will be deduced by extracting sentiment-carrying words (interchangeably referred to as polarity words) and key phrases aggregated over large sets of news articles. In that manner, the user will be exposed to dense and meaningful insights without having to spend hours gathering and synthesizing that information from the myriad of sources available. As mentioned, the project will seek to provide the above-described practical benefits by relying on breakthrough innovation in the field of natural language understanding. A novel approach to analysing longer text sequences which remains a gap in current literature will be implemented from which further follows that the same methodology has not been used by other similar software solutions.

1.3 Report Structure

The remainder of this project is structured as follows:

Chapter 2 discusses the state of literature in the domains concerning the functionality of the application – sentiment analysis and key phrases extraction. The available solutions offering similar features to the software developed in this project are also described.

Chapter 3 is concerned with the specification of the solution developed. An overview of the software's features is presented and functional and non-functional requirements are laid out. The target audience and use cases of the software are also discussed.

Chapter 4 goes in more detail about the design of the software. The database design and high-level software architecture are discussed.

Chapter 5 discusses the backbone of this project, namely the natural language understanding architecture used. The chapter describes the details of fine-tuning and selecting the model that was used in the application.

Chapter 6 goes in detail about the actual implementation of the functionality stated in the specification. The scientific grounding for each decision made when implementing the software logic is discussed. Moreover, an overview of how the features were materialized with coding is explained.

Chapter 7 is meant to familiarize the reader with the design of the user interface and what principles were followed when implementing it.

Chapter 8 discusses the testing strategy employed to ensure the robust functioning of the software. An overview of the approach taken to testing is given and the types of tests taken are explained.

Chapter 9 presents a summary of what the project achieved from both a research and practical perspective. Proof of the behaviour of the software and its value for users is also laid out.

Chapter 10 scrutinizes the inherent limitations encountered. A perspective of what could have been done better and why that was not feasible within the resource constraints of this project is given. Where relevant, suggestions for future work in both the research and practical elements of this project is suggested.

Chapter 11 gives perspective on how the software development process was structured. The overall model followed is explained and an overview of the steps taken at each stage of development is given.

Lastly, chapter 12 provides a logical conclusion of the project report.

2. Literature Review

This section is intended to first familiarize the reader with the state of literature in the topics that concern the main functionality of the software and also explore what are the existent similar solutions.

2.1 Sentiment Analysis

Research on sentiment analysis typically adopts one of the two broadly established methods for defining the polarity of texts. Within these methodologies there are various choices that researchers need to make so as to maximize the robustness of their experiments.

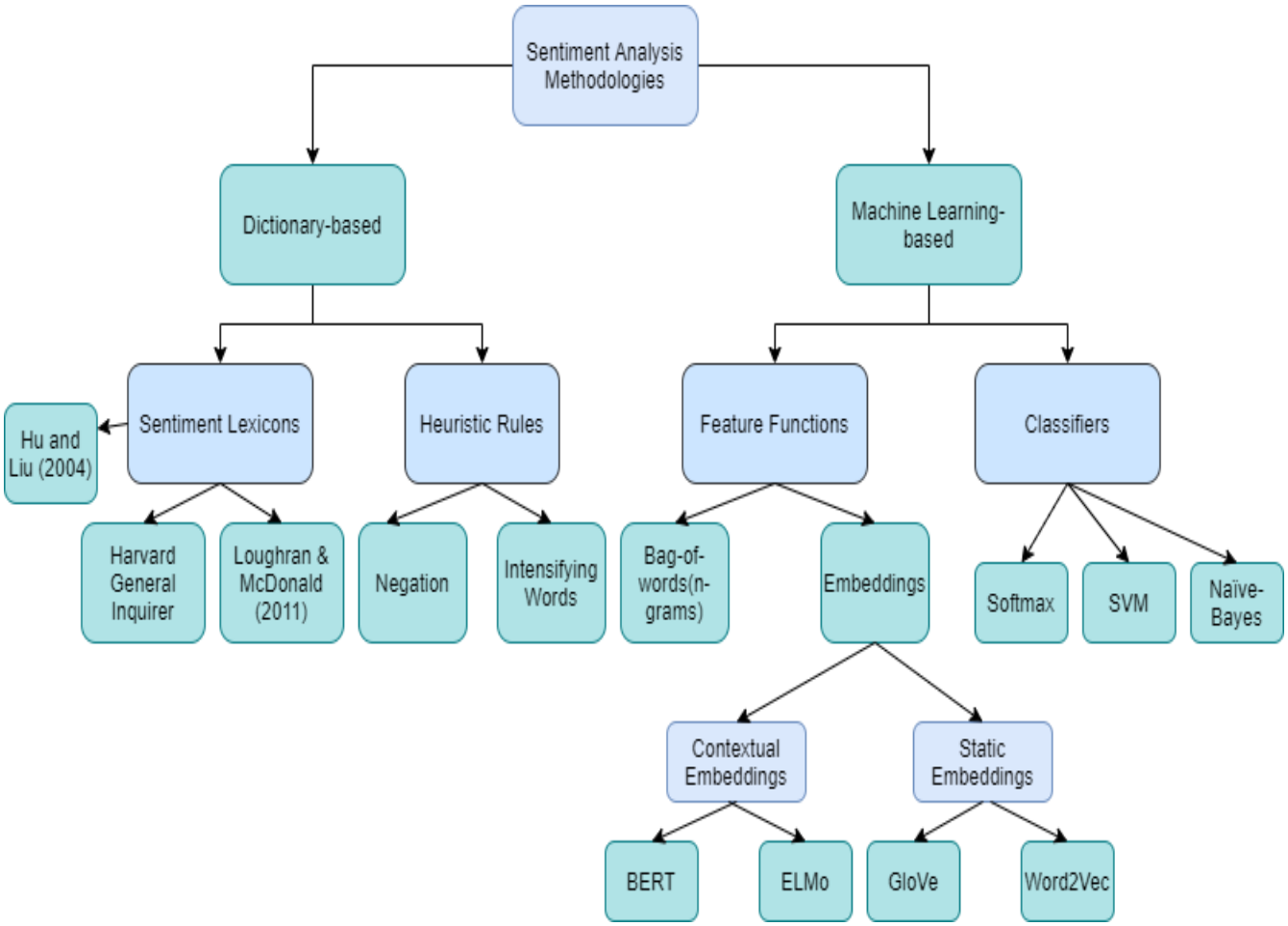


Figure 1: Overview of topics discussed in the sentiment analysis section of the literature review.

2.1.1 Dictionary-based Approach

The polarity words extraction functionality of the platform developed will have its scientific grounding derived from the literature that is discussed in this section.

Researchers using the dictionary-based methodology employ sentiment lexicons that contain opinion words with polarity scores to derive the sentiment of texts. Often, additional heuristic rules or semantic modifiers concerning negation, punctuation, intensifying words (highly, mostly, very etc.), and other text features are further included (Taboada, et al., 2011). For example, a simple rule that is often adopted is to invert the sentiment score of all words preceded by some form of negation until a punctuation mark (Das & Chen, 2007). Usually, the net score of the difference between positive and negative words considering semantic modifiers is used to arrive at the sentiment of a given text (Shapiro, et al., 2020). Some of the lexicons that are often used are Harvard's General Inquirer, Loughran and McDonald's (Loughran & McDonald, 2011) and Hu and Liu's (Hu & Liu, 2004). There are various techniques for generating these lists of opinion words. For example, Hu and Liu (2004) first use part-of-speech tagging to identify adjectives as they are often used for opinion expression and manually classify their sentiment. Subsequently, they adopt a commonly used lexicon-generating technique, namely using these adjectives as seed words and expanding their range by traversing through large lexical databases and finding their synonyms and antonyms – in this case WordNet (Miller, et al., 1990) was the database of choice. Loughran and McDonald (2011) build their lexicons manually with opinion words from the finance domain while Shapiro et al. (2020) combine words from different lexicons and also create their own domain specific dictionary. The work of Loughran and McDonald (2011) and Shapiro et. al. (2020) provides most value for the purposes of this project and will be employed (discussed in later parts).

There are a few common points of criticism concerning the dictionary-based approach. Firstly, words are treated in a bag-of-words manner, hence mostly ignoring their use in context, their position in a given sentence and other latent characteristics. Hutto and Gilbert (2014) attempt to address that issue by introducing VADER which is a text sentiment classifier based on a sentiment dictionary enhanced with comprehensive heuristic rules that they claim mitigate the problem of losing context. Despite that, the extent to which dictionary-based models are able to capture the full richness of language remains controversial. For example, SentiWordNet – an opinion lexicon derived from WordNet has completely opposite scores for given words (e.g *serious* as in appealing to the mind and *serious* as in causing fear or anxiety). Additionally, disambiguating negation has proven a complex task. Potts (2010) discusses all the complex idiosyncrasies of negation in texts (e.g double negation – "I do not not like it", neg. polarity terms – *any, ever, whatsoever* etc.) based on which one could argue that capturing negation using some of the earlier-mentioned rules could be an oversimplification introducing errors. The second point of criticism is the domain transferability of different lexicons. Loughran and McDonald (2011) found that three-fourths of the words classified as negative by the Harvard General Inquirer would not be considered as having the same sentiment in the finance domain (e.g crude is scored as negative but in financial texts crude(oil) would not carry the same semantic meaning). Notwithstanding some of the downfalls of this methodology, it remains transparent and easy to implement which is one of its major advantages over the methods discussed next.

2.1.2 Machine Learning-based Approach

For the relevancy of this project, this section will focus only on supervised machine learning techniques for sentiment classification which are also the most used in research. The simple part of such models relies on some type of supervised machine learning algorithm that uses labelled datasets to learn sentiment classification. The significantly more complex aspect that usually is the deciding factor for a model's performance is finding an appropriate function for extracting features from the text data. Below some of the common feature functions used in literature are discussed.

Bag-of-words (n-grams)

Pang et al. (2002) delivered a seminal article in the field of sentiment analysis where they compared the performance of machine learning techniques to a dictionary-based baseline that relies on sentiment word counts. They used Naïve-Bayase classification, support vector machines (SVMs), and maximum entropy (MaxEnt) classification and concluded that all three methods outperform the baseline with SVMs exhibiting the best results. Das and Chen (2007) employ a similar methodology and test the performance of other machine learning models like vector distance and discriminant-based classifiers. Boiy and Moens (2008) also explore various classifier and manage to achieve 83% accuracy in sentiment categorization on a ternary scale of negative, neutral and positive.

The above researchers use bag-of-words feature functions which in an essence resembles what was described in the previous section. Words are treated as n-grams, meaning that texts are tokenized into either single words (unigrams), or other combinations of words (bigrams, trigram etc.). In some cases, rules for negation, position in text and part-of-speech tagging are adopted. The input provided for the classifiers are vectors of either the frequency or presence (one-hot vectors) of given n-grams in an example.

Despite the fact that machine learning models seem to be able to capture more complex, non-linear relationships in data, there a few issues with bag-of-words feature functions. Firstly, training data is not plentiful and the existing datasets are often narrowly domain-focused which raises questions on the transferability of trained models to texts from other domains. For example, due to the ease of labelling, many of the existing datasets are based on movie or product reviews where language is used in a significantly different manner as compared to in, say, financial news. Secondly, longer sequences where contextual dependencies may span several sentences are often misrepresented when using bag-of words feature functions (Socher, et al., 2013). Illustrative of that would be a case of thwarted expectations: "Critics felt this movie was going to be his best masterpiece and were excited to see it. It turned out to be disappointing". Almost certainly, this sentence would be classified as positive despite the clear negative connotation. The software developed for this project requires entire news articles analysis with complex semantic relationships, rendering such approaches ineffective.

Distributed representations (embeddings)

The idea behind distributed representations used for feature functions involves generating continuous vector representations of words from texts. These learned representations are also known as embeddings and are obtained using different supervised or unsupervised models. A distinguishment between two types of embeddings is usually made in literature. The first type is referred to as static or word embeddings. They are generated using word co-occurrence matrices from which word vectors are obtained. Examples are Word2Vec (Mikolov, et al., 2014) which uses a neural network for processing co-occurrence matrices and Global Vectors (GloVe) (Pennington, et al., 2014) which relies on word co-occurrence probabilities. The second and more recent type is contextual embeddings. Two prominent models in this category are BERT (Devlin, et al., 2018) and ELMo (Peters, et al., 2018). The difference between word and contextual embedding is that contextual embeddings are generated by processing texts sequentially rather than using word co-occurrence matrices. Consequently, a richer understanding of a given word could be obtained as its vector representation is contingent on its neighbouring words. BERT will be further elaborated on later as it was the model selected for use in this project.

In sentiment analysis models using static embeddings, usually, after a given text is tokenized the vector representation of words are looked up in the pre-generated embedding space. Next, these vectors are either averaged or summed to arrive at a representation of the entire document (text). As described earlier, despite using richer word representations, such models still largely ignore the sequence in which words occur in a given text.

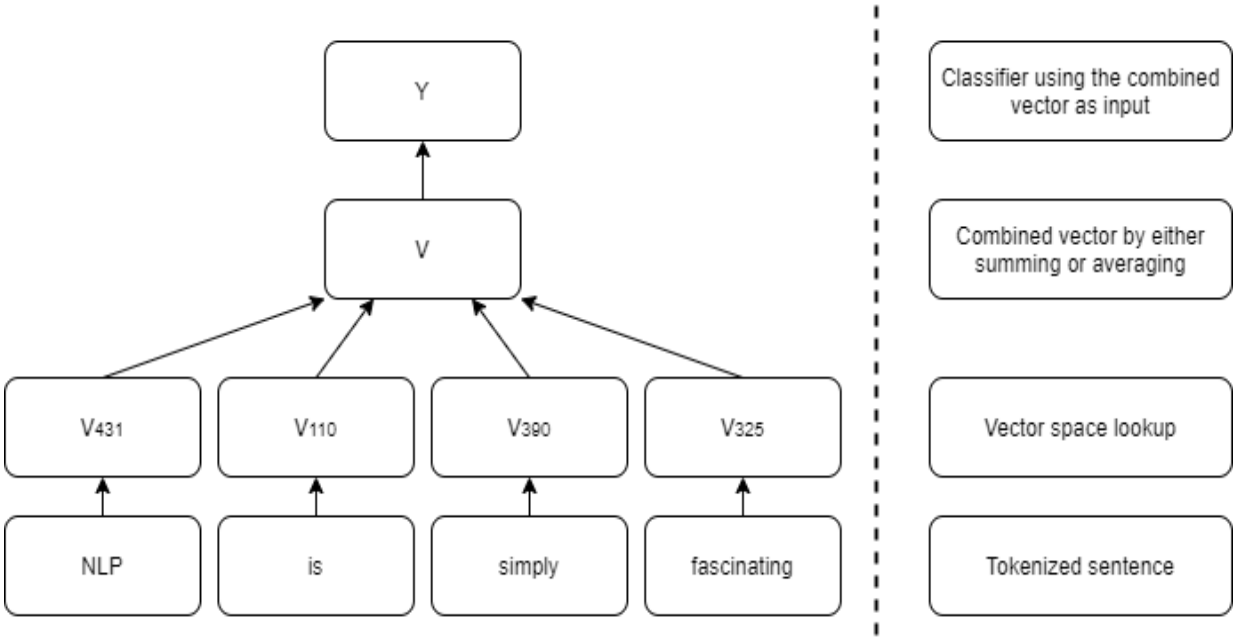


Figure 2: Sample sentiment analysis model using word embeddings

Socher et al. (2011) present a novel architecture based on recursive autoencoders that learns word vector representation that they use for sentiment analysis. The researchers address the problem of losing context and not capturing linguistic complexities of models like the one Figure 2 describes. Their model processes words in a given example sequentially, hence pertaining to the contextual embeddings category discussed above. Similarly, Tai et al. (2015) use a Long-Short Term Memory

(LSTM) network to address the vanishing gradient issue of simple recurrent neural networks that renders them unfit for processing longer text sequences (Hochreiter, 1998). They introduce a tree-structured LSTM and showcase its superiority in predicting sentiment over other LSTM variants.

A major breakthrough in the field of NLP came with the introduction of Bidirectional Encoder Representations from Transformers (BERT) by a team at Google AI Language (Devlin, et al., 2018). The architecture builds upon the seminal Transformers network model (Vaswani, et al., 2017) that uses attention mechanisms to generate embeddings. BERT considers text sequences in a bidirectional manner, hence allowing a richer contextual understanding of a given word derived from other words in its vicinity. Two mechanisms for learning word embeddings are used– masked language modelling (MLM) and next sentence prediction (NSP). MLM implies masking 15% of the tokens fed to the model and allowing it to predict them while NSP involves mixing 50% of sequential sentences and creating mechanisms for the model to distinguish between the correctly connected sentences and the shuffled ones. The first token of any sequence fed into BERT is a special token called [CLS] which is used for classification. More specifically, the final hidden state of that token is what is used as it contains the embedding representation of the entire sequence. The model is pre-trained on a large corpus of cross-domain data but it also allows for fine-tuning which often increases its precision for specific tasks.

2.1.3 News Sentiment Analysis in the Finance Domain

Research using dictionary-based methodologies

The existing research stock on financial news sentiment analysis predominantly focuses on establishing how does news sentiment reflect on given stock prices or indexes at the micro level and certain macroeconomic indicators like GDP at the macro level. Indeed, Li et al. (2014) find that mapping financial news to bag-of-word spaces has less predictive power for stock price movement than analysing them using sentiment models for establishing the same relationship. Tetlock (2007) analyses the immediate impact on stock prices of articles posted in the 'Abreast of the Market' column of the Wall Street Journal. The author uses the Harvard General Inquirer's semantic categories and by applying some dimensionality reduction techniques he generates what he refers to as a single media factor. His findings suggest that high media pessimism correlates with short-term downward pressure on stock prices. The short-term predictive power of daily news on stock prices is also confirmed by Heston & Sinha (2016) who find that daily news only have impact on stock prices for about 1 to 2 days. An interesting finding they arrive at is that the predictive power of news sentiment aggregated over the period of a week could last for up to 13 weeks. Relying on one of the main assumptions of financial economics, namely that stock prices react to unexpected information about a company's fundamentals, Boudoukh et al. (2012) conduct a textual analysis that classifies news into 14 different categories and evaluate their sentiment. They argue that the methodology used improves the accuracy of measuring the predictive power of news sentiment as only news describing events relevant to a company's fundamentals are evaluated. The researchers use Loughran & McDonald (2011) dictionary with some extra rules for negation, sentiment modifiers and phrases. Garcia (2013) uses financial news derived from the New York Times and analyses the relationship between sentiment and stock price movements using a similar lexicon-based approach. He considers the net fraction of positive and negative words as a proxy for the polarity of news and finds that news sentiment has an even higher impact on price fluctuations during economic downturns.

Despite being transparent and relatively simple to implement, the above-described methodologies perhaps suffer from a few common problems. Firstly, some of the dictionaries used like, for example, Harvard's General Inquirer are not specifically built for the finance and economics domain which could lead to distortions (Loughran & McDonald, 2011; Aue & Gamon, 2005; Glorot, et al., 2011). As mentioned previously, Loughran & McDonald (2011) state that almost three-fourths of the negative words in the Harvard dictionary have a different semantic polarity in the finance domain.

Secondly, dictionary-based approaches fail to capture a lot of the contextual complexities of language and could very often lead to misrepresented classification of the tone of news texts. Longer texts where dependencies could span several sentences are particularly prone to erroneous interpretation. Lastly, the two most used lexicons – The General Inquirer and Loughran & McDonald's contain 3,626 and 2707 sentiment words which when applied to 238,685 financial and economic news articles by Shapiro et al. (2020) covered only 6.4% and 0.7% respectively of the unigrams in the news corpus. Consequently, one could argue that perhaps a lot of the sentiment expressed in these texts is not explicitly stated using polarity words but is rather more subtle and contained in context.

Research using embeddings and machine learning methodologies

A second group of researchers employ deep learning techniques to extract text features that are used as input to sentiment classicisation machine learning models. Kraus et al. (2017) clearly delineates the increased performance of classifiers using word embeddings generated by a long-short-term memory (LSTM) recurrent neural network compared to bag-of-words feature functions for predicting stock market gyrations. They further demonstrate the superior performance of the LSTM model compared to a simple recurrent neural network that is known to underperform when used for longer sequences due to the vanishing gradient problem (Hochreiter, 1998). Souma et al. (2019) use the global vectors word representation (GloVe) as input to a LSTM recurrent neural network. They use data from the Thompson Reuters News Archive and consider the returns for a given stock just prior and after the release of a news article mentioning it to determine the text's polarity. Their model performs with high accuracy but the method they use for labelling raises questions of potential confounders introducing noise or reverse-causality that is not explicitly addressed in the article. Sousa et al. (2019) explore the impact of news sentiment on stock returns using BERT. The researchers use financial news articles data to fine tune the model and also compare their results with other machine learning techniques like Naïve Bayes and support vector machines (SVM) which use bag-of-words feature functions. The results obtained indicate that the BERT model outperformed the rest. Despite the novel approach to analysing financial news polarity, the data released by the researchers seem to indicate that their results are obtained by testing on shorter text sequences, namely, just sentences from news articles. Therefore, the question of what methodology could be used to take advantage of BERT or other pre-trained deep learning models to deduce the sentiment of longer sequences like entire articles remains without a clear answer. In later parts this project will address that issue.

2.2 Keywords and Key Phrases Extraction

Research on keywords/phrases extraction can largely be separated in two different categories. One group of researchers consider keyword extraction a problem that is to be tackled using various statistical tools and a second group adopts more contemporary, machine learning techniques.

A very common technique on which researchers using statistical methods built upon is Term-Frequency Inverse-Document-Frequency (TF-IDF). The method implies normalizing the term frequency of each term in a given document (TF part) and then also considering the occurrence of that given term in all documents being evaluated (DF part). In order to mitigate the importance of words that occur frequently but do not convey important information (e.g stop words) the occurrence of terms in the considered documents is inverted (N/DF) (Robertson & Jones, 1976). Overall, TF-IDF succeeds in providing context about the importance of a given word not only within a given document but within an entire collection which makes it a powerful method. Matsuo & Ishizuka, (2004) address one of the problems with TF-IDF, namely that it relies on a corpus of documents rather than a single document. First, they derive the most frequent terms of a document using simple count frequency statistics and then create word co-occurrence matrices of each term and the extracted frequent terms. Subsequently, they judge whether a term is to be considered a keyword by analysing the co-occurrence probabilities deduced from the co-occurrence matrices. Wartena et al., (2010), employ a similar approach and expand upon TF-IDF by deriving measures for word similarity and context assumed from word probability distributions observed in co-occurrence matrices.

The second category of research often broadly follows a pattern of first extracting candidate keywords or phrases and then using either supervised or unsupervised machine learning to rank the probability that a given candidate is a keyword/key phrase (Meng, et al., 2017). A notable representative of this group is the research of Hulth, (2003). The author treats the problem of keyword extraction as a supervised machine learning problem. More precisely, a labelled dataset consisting of observations with features such as within-document frequency, collection frequency and relative position of the first occurrence were ran through a binary classifier. Different combinations of n-grams were also considered. The achieved F-scores were within the range of 25-34 which, normally, would not indicate good performance, however, in the domain of keyword extraction they are regarded as reasonable. Despite their performance, supervised learning suffers from the problem of being highly sensible to the choice of correct features. Often, these models do not include any contextual features of the candidate words hence reducing their effectiveness. Lastly, the need of manual labelling of large datasets to be used for training is another limiting factor. Meng et al. (2017) attempt to address some of the issues of supervised learning by leveraging deep learning. The authors claim that often features used in supervised learning fail to capture a lot of the semantics of given observations. They develop an encoder-decoder architecture implemented through RNNs for key phrase extraction and manage to obtain results comparable to these of Hulth, (2003) measured by the F-1 score achieved. More recently, Sharma & Li, (2019) make use of the exceptional performance of BERT (Devlin, et al., 2018), discussed in the previous section. They implement a novel methodology for keyword/key phrase extraction by using BERT's contextual word embeddings. The authors pool all word vectors of a given sequence to obtain a vector representation and then compare it using cosine similarity to each word embedding in the sequence. The result of that process is words being labelled as candidate keywords. Lastly, in order to arrive at a final set of keywords they feed the candidates to bidirectional

LSTM classifier. Overall, the researchers manage to tackle the problem of manual labelling and also take advantage of recent state of the art architectures that seem to produce good results.

2.3 Available Software Alternatives

Currently, the market offers a few available solutions that could be regarded as similar in some features to what this project seeks to achieve. The first one originates from one of the world's leaders in data analytics in the finance/economics domain – Bloomberg and is available for clients of their most popular product – the Bloomberg Terminal (Bloomberg Professional Services., 2021). Their news sentiment analysis tool provides a score between -1 and 1 for various stock tickers. It tracks news and social media sentiment but it does not appear that the user is able to get a more in-depth understanding of how is that score generated neither what texts were evaluated to arrive at it.

Another software offering stock sentiment analysis is provided by Ravenpack – a data analytics company targeting the financial sector (RavenPack., 2021). Their news analytics product further includes options for topic tagging and relevance scoring. Further information about their methodology was requested but no response was received. The only information one could derive from their website concerns some of the sources they use for news and that they are using machine learning to evaluate the texts extracted from these outlets.

Aylien is another data analytics company that offers various solutions using unstructured data (Aylien., 2021). Their news API includes features such as entity extraction, translation and sentiment analysis. They do not explicitly mention their methodology for conducting sentiment analysis but a paper released from their researchers uses a hierarchical bidirectional LSTM (Ruder, et al., 2016). The use of such architecture could lead to the assumption that they are able to extract sentiment from longer sequences unlike other solutions where more often smaller text sections or just headlines are employed. The next software discussed is an example of that group.

Stocksnips is a platform that offers stocks sentiment analysis (StockSnips., 2021). They also employ natural language processing but their method seems to rank only given small sections of news articles which could lead to inaccuracies. A positive feature of their offering is that they provide sentiment scores on the environmental, social and governance (ESG) criteria for a given stock. Placing news articles in one of these categories before analysing their sentiment appears to be done by using some form of topic tagging which could provide more fine-grained information.

Within the scope of crypto markets, Cryptomood provides sentiment analysis for various cryptocurrencies (CryptoMood., 2021). The users of their application are able to receive real time sentiment information based on news articles and social media posts from various sources. Their methodology seems to lean towards a dictionary-based approach (refer to section 2.1.1). The process they use to arrive at a sentiment score for a given article appears to be using n-grams and assigning sentiment scores to each word. Following that, these scores are aggregated to arrive at a final score for a given article and one could assume that that process is repeated over all articles for a given stock. The reader has been familiarized with the merits of similar approaches in section 2.1.3**.**

There appears to be a similar pattern of methodology issues with the majority of the above-discussed solutions. Firstly, from the information available, only Aylien's software seems to consider entire articles and the remainder seem to focus on snippets or headlines when deriving sentiment.

Additionally, it is once again only Aylien's solution that relies on more sophisticated natural processing techniques that are able to capture deeper contextual relationships in longer texts. A common point of criticism for many services using machine learning is that they are regarded as black boxes and the users rarely understand any intricacies about their functioning (Shapiro, et al., 2020). The fact that almost none of the discussed solutions provide information about their sources, the precise natural language processing technique used and the data they employ is problematic. Considering the claims that entire articles are evaluated, methodology becomes an imperative factor as sentiment analysis for longer sequences is still a problem that does not have a clear solution in natural language processing research. None of the above platforms uses any of the state-of-the-art architectures like BERT (Devlin, et al., 2018) , XLNet (Yang, et al., 2019) and others. Consequently, the additional logic that must have been implemented for assigning sentiment scores to news articles is crucial in understanding the viability of their product.

Lastly, none of the available alternatives provide any functionality that provides indication of what drives a given asset's sentiment. This project argues that observing sentiment in vacuum without context about the topics that determine it is highly limiting. Overall, all platforms, perhaps apart from Cryptomood, provide a broad range of other services and do not have an explicit focus on sentiment analysis. The hefty subscription pricing is also a very limiting factor.

3. Requirements Engineering

3.1 Specification

3.1.1 Overview of Purpose and Intended Audience

The software is called NaratAI inspired by Robert Schiller's 'Narrative Economics' (Schiller , 2019). The name hints at what the application seeks to do – communicate the prevailing narrative for any given asset investors might be interested in.

The NaratAI web application is developed to address the problem of access to condensed, high-quality information on news sentiment and the driving factors behind it, both being one of the most imperative factors when considering a given investment. The software enables users to not only get a rapid understanding about the spectrum of negativity or positivity expressed for an asset of their choice but also explore the context of what is driving that polarity. The intended target audience consists of retail investors of which there has been a large recent influx. That group rarely has access to sophisticated data analytics like institutional investors do and is also characterized by mainly using online media outlets as a guide for investment decisions. The application addresses the group's needs by providing high-quality dense information using the same technologies that market actors with significantly more dedicated resources also use. Additionally, considering the fact that a constraining resource for many retail investors is time, the software serves the purpose of saving users the need to collect and analyse that information themselves from the extremely large pool of sources available.

3.1.2 Overview of Software Functionality

The application's core functionality builds on the highest quality research in the field of natural language processing coupled with novel logic that is also informed by science.

Sentiment score – understanding the expressed news sentiment

Users are able to grasp the prevailing sentiment towards a given asset by observing its sentiment score on a scale -100 to 100 where 0 is considered neutral. The sentiment score is extracted by using custom built news scrapers and aggregating dozens of news articles from some of the most reputable sources in the field of finance. The users are also able to see charted the historical movement of the sentiment score for the past 10 days, a feature also grounded in research on the predictive power of sentiment based on its aggregation (Heston & Sinha, 2016)(refer to section 2.1.3).

Transparency is enhanced by listing all daily headlines for the selected asset and a visual indication of their predicted sentiment. Statistics about the number of assessed articles and their overall polarity are also present. Moreover, the user is provided with the current price of their asset of interest and the percentage change in price from the previous day's closing to the current moment, an important indication given the know correlation of these factors (Garcia, 2013; Li, et al., 2014; Heston & Sinha, 2016).

Polarity words and key phrases – understanding what determines sentiment

Furthermore, in order to provide more fine-grained information about the determinants of an asset's daily sentiment score, users are able to explore the sentiment-carrying words (interchangeably referred to as polarity words) and key phrases from the evaluated articles. Such functionality exposes users to the prevailing topics in the news and should they wish to explore any word or phrase that got their attention, they can easily navigate to the source of where it came from. The methodology applied to enable such functionality uses a special-built sentiment lexicon derived from high quality research in the field of linguistics and finance. Words' polarity is considered specifically within the context of finance and economics which improves the quality of information users are able to obtain. Key phrases are also derived using logic the foundation of which is recent research in the field of natural language processing but that also expands what has currently been done. The described content is rendered in a word cloud and the application allows for toggling between polarity words and key phrases and also the positive, negative or mixed group of those. Each word or phrase is styled according to its sentiment and frequency of occurrence to provide more context.

Choice of supported assets

The available stocks and crypto currencies users are able to pick from are displayed in a grid and conditionally styled based on the number of articles evaluated for them for the given day. That adds an extra layer of insight as to what assets are most discussed in the news. The supported stocks are for the most-traded listed companies and are all part of indexes such as the S&P 500, the Nasdaq composite and the Russel 2000. As for the cryptocurrencies, again the eight most traded names are included.

3.1.3 Assumptions and Dependencies

Concerning the correct functioning of the system, it is assumed that the sources which are used to retrieve news articles continue to exists and do not implement any logic blocking the functioning of the news scrapers. Additionally, it is assumed that the sources employed provide information that is a justifiable representation of reality (including opinion columns). Moreover, the external APIs used to retrieve stock and crypto currency prices are also assumed to remain useable. The persistent support for the libraries used in both the back and front-end parts of the application is also assumed.

3.2 User Story

The below user story is an example used for eliciting the functional and non-functional requirements of the system.

Jack is one of the retail investors who became interested in markets following the boom in popularity of crypto currencies. He uses one of the trading applications to buy different crypto currencies and also holds a few stocks. He does not have any academic background in finance and mostly relies on what is on the news to guide his decision. Being constrained by a full-time job like almost all other retail investors makes it hard for him to gather all information to make informed decisions.

Jack is looking for any software that would quickly allow him to see what is the prevailing 'narrative' for the assets he is interested in investing. He wants to know whether there are any sentiment swinging news that could impact the prices of the assets he holds and also the ones he is interested in

buying. The solutions he has found are usually both unattainable due to their price and provide superficial categorization that does not convey much. Ideally, he is also interested in understanding what are the prevailing topics that might be driving price swings so that he can capitalize on that.

3.3 Functional and Non-Functional Requirements

The functional and non-functional requirements of the system were developed considering the product specification. The importance of each requirement is also indicated on a ternary scale of low (L), medium (M) or high (H). The reader is referred to Appendix 14.1 for a list of the functional and non-functional requirements elicited for this project.

4 Software Design

This section is meant to familiarize the reader with the design decisions made when developing the application and argue their choice. First, a brief overview of the flow of tasks on which the functioning of the application is contingent is given. Next, the system database design is described. Lastly, the choice of software architecture for the system is argued.

4.1 Overview of Flow of Tasks

As described in section 3.1.2, the core functionality of the application relies on a dataset of aggregated news articles. In order to acquire news articles for each of the supported stocks or cryptocurrencies, two separate news scrapers were built – one for stock news and another for crypto currency news. Following the acquisition of texts, each one is pre-processed before extracting its sentiment, key phrases and polarity words using a state-of-the-art NLU architecture and logic described in the coming sections. Next, the application database is updated with the described data for each news article scraped. Lastly, the back-end part of the application provides API endpoints that serve the purpose of gathering and structuring the data in a format that can be consumed by the front-end. Figure 3 depicts the described flow of events graphically.

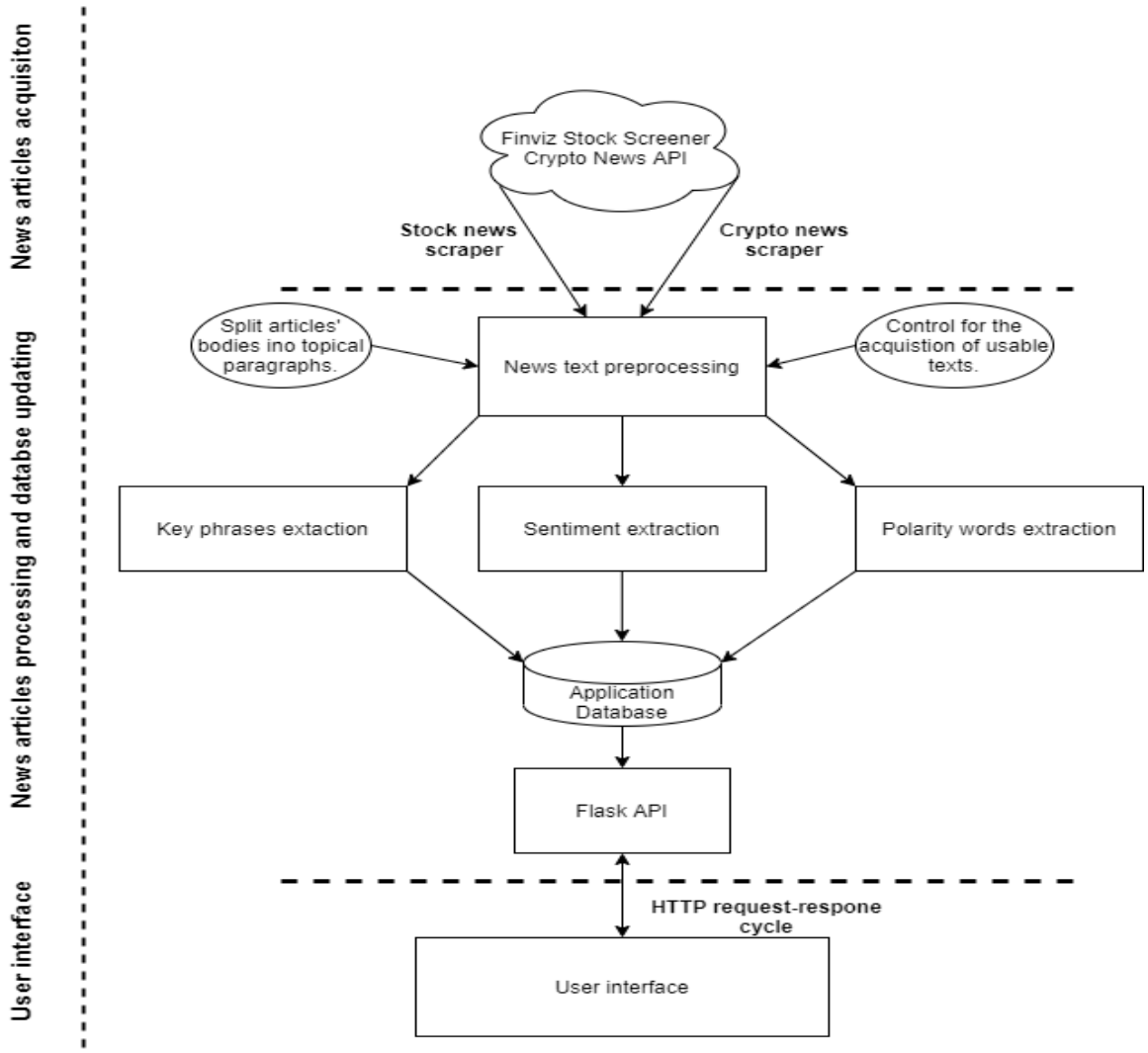


Figure 3: Overview of flow of events

4.2 Database design

4.2.1 Overview

The database is a crucial component in the functioning of the system. As mentioned, all core functionality the application provides relies on aggregated data consisting of news articles and the analytics extracted from them. For the purposes of this project the PostgreSQL relational database management system was used. The database consists of five relations that were designed to be in Boyce-Codd normal form. Three of the relations are directly related to the core functionality of the software and the other two contain information about each entry and information about the supported assets. All tables were built to implement constraints required for the robust functioning of the application and to ensure data integrity. Below the reader can observe the relationships between the different tables illustrated using entity relation diagrams (ERDs).

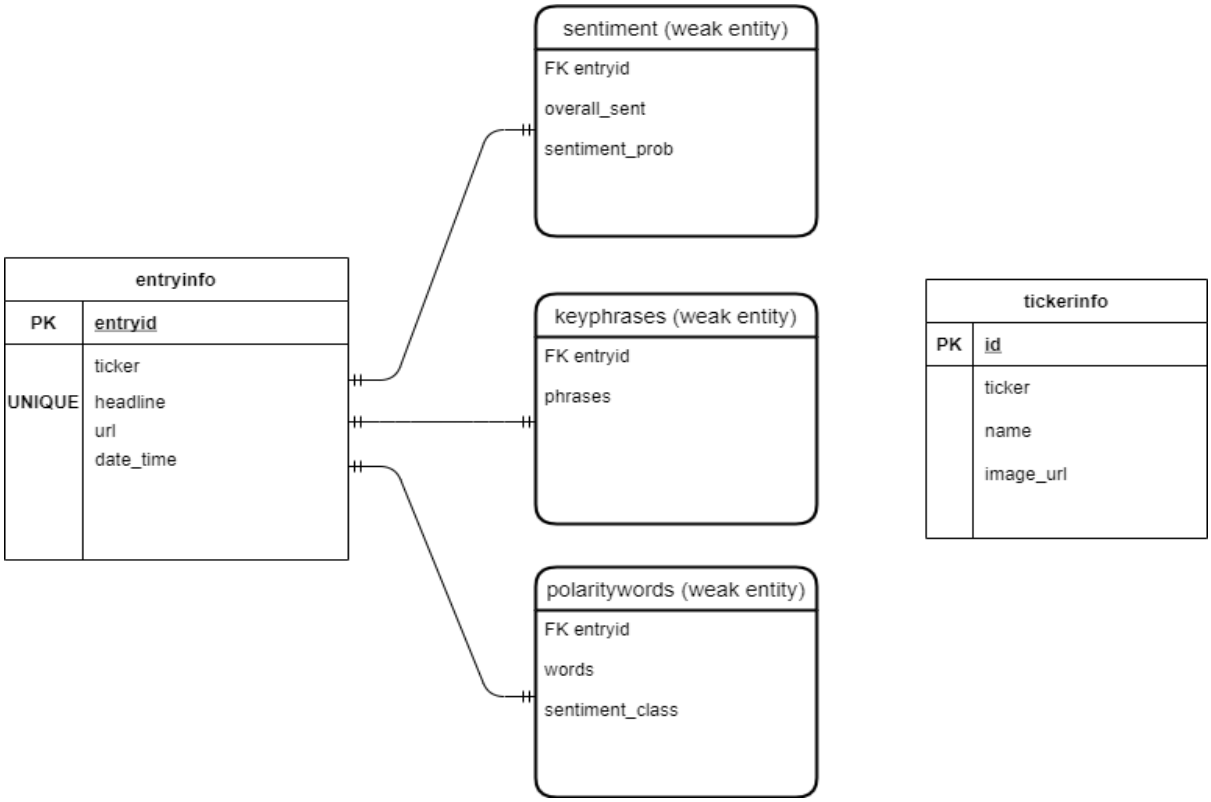


Figure 4: Database structure.

Each database entry in tables *entryinfo*, *sentiment*, *keyphrases* and *polaritywords* corresponds to one of the articles acquired using the news scrapers. Tables *sentiment*, *keyphrases* and *polaritywords* are considered weak entities as they do not have primary keys. All of them have a one-to-one relationship with table *entryinfo*, meaning that each entry in table *entryinfo* corresponds to exactly one entry in any of the weak entities. The foreign key in all weak entities is relation *entryinfo's* primary key – *entryid*. Table *tickerinfo* does not have any relationships with any of the other four. Next, all relations' attributes are described and the constraints imposed on the database to adhere to the cardinality illustrated are laid out.

4.2.2 Relations' attributes description

A tabular description of the attributes of each relation is provided below.

- Relation ***entryinfo*** contains information about each article that was acquired by the news scrapers

Table 1: Relation entryinfo Attributes Description

entryinfo	
entryid	A unique id given to each entry.
ticker	The ticker symbol of the asset the article concerns.
headline	The article's headline.
url	The source url.
date_time	A time stamp of the entry.

- Relation ***sentiment*** contains the corresponding categorical classification the given article and the sentiment probabilities acquired by the model for each of the three classes used – negative, neutral and positive.

Table 2: Relation sentiment Attributes Description

sentiment	
entryid	A unique id given to each entry.
overall_sent	The article's sentiment class – negative, neutral or positive.
sentiment_prob	The probabilities for each of the three sentiment classes.

- Relation ***keyphrases*** contains all extracted key phrases for any given article.

Table 3: Relation keyphrases Attributes Description

keyphrases	
entryid	A unique id given to each entry.
phrases	The key phrases extracted from the article.

- Relation ***polaritywords*** contains all polarity words and their corresponding sentiment.

Table 4: Relation polaritywords Attributes Description

polaritywords	
entryid	A unique id given to each entry.
words	The polarity words extracted from the given article.
Sentiment	The corresponding sentiment of each polarity word extracted.

- Relation ***tickerinfo*** contains information about all stocks and cryptocurrencies supported by the platform.

Table 5: Relation tickerinfo Attributes Description

tickerinfo	
id	A unique id given to each entry.
ticker	The ticker symbol of the given asset.
name	The full name of the company or crypto currency.
Image_url	A url to an image of the company's or crypto currency's logo.

4.2.3 Constraints

In order to ensure the robust and reliable functioning of the application given constraints are imposed on certain attributes in the relations.

Firstly, any article acquired should be evaluated once and only once. Therefore, the one-to-one cardinality between strong entity *entryinfo* and the 3 weak entities – *sentiment*, *keyphrases* and *polaritywords* is ensured by placing a foreign key constraint on attribute *entryid* in each of the weak entities. Since *entryid* is a primary key in table *entryinfo* uniqueness is an inherent constraint which further implies that no more than one entry with the same *entryid* can be placed in any of the relations described as having a one-to-one cardinality in the ERDs.

Next, an additional data integrity issue is presented by potential duplicate entries. Since the database is updated daily, the chances of duplicate insertions provided no constraints are imposed are very high. Evaluating the same articles could skew the reliability of all features of the system. Therefore, a 'unique' constraint was placed on the *url* attribute in table *entryinfo* to ensure the robustness of the database.

4.3 Software architecture

4.3.1 Layered architecture

The application was developed following the layered architecture pattern. Such structure implies that the software functionality is split in layers that have clearly delineated responsibilities. Each layer is meant to provide a service to the layers adjacent to it. The choice of architecture was guided by the fact that such modularity allows for adjustments in one layer to occur without impacting others. Additionally, due to the large size of some of the dependencies and static files used in the software, such modularity is the only way to deploy the application to production (discussed later).

The developed system could be split in five separate layers each communicating with the layers below and above it. These include a presentation, application, business, persistence and database layers. Normally, the flow of events between layers takes place in a top-down manner. Namely, a user triggers an event via an action in the top-most layer – the presentation layer and that trickles down to the last layer in the application. Figure 5 depicts the layers used for the purposes of this project. Next, each layer's responsibilities are described.

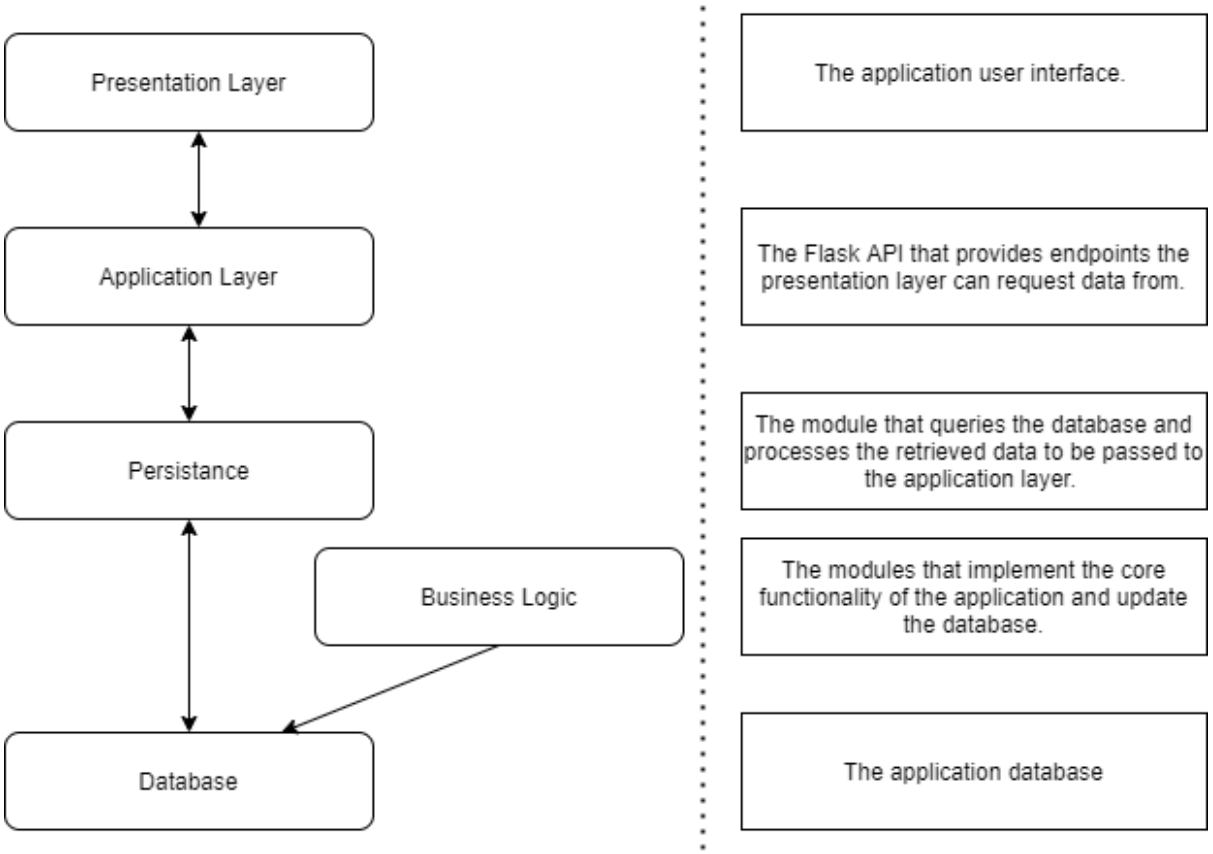


Figure 5: System architecture

Presentation layer

The presentation layer constitutes the application's graphical interface and is the only point of contact between the user and the system. Additionally, any data that is not used for the core logic of the application and is not sourced from the database is also retrieved directly in the presentation layer. In the case of the NaratAI application, such would be the prices of stocks and cryptocurrencies which are fetched using two external APIs – the Yahoo Finance API (2021) and the CryptoCompare API (2021). As mentioned, upon triggering a certain event in the user interface, the presentation layer requests the required data from the application layer sitting below it. For the purposes of this system the communication between these two layers is carried out via HTTP requests and responses. For example, upon selecting a favourite stock, the presentation layer sends a request to the application layer to retrieve the historical sentiment score, polarity words and key phrases for the given asset. The presentation layer was practically enacted using the React JavaScript library (discussed later). Next, the application layer is described.

Application layer

The application layer serves the purpose of handling requests from the presentation layer by providing URL endpoints used to receive HTTP requests. Depending on the endpoint used and the parameters passed to the HTTP GET request (where applicable) the application layer is responsible for sending the correct HTTP response containing the requested data in a JSON format. Handling of multiple threads allowing for simultaneous use of the application by various clients is also handled in that layer. Moreover, the application layer serves as an abstraction between the presentation and persistence/business logic layers and provides useful modularity. Should the front-end implementation of the software need any overhaul, that could be done without any considerations about how the application logic is enacted on the back-end. For the implementation of this layer a Flask API was used (Grinberg, M., 2018). In order to obtain the data requested by the presentation layer, the application layer communicates with the persistence layer discussed next.

Persistence layer

The persistence layer is responsible for gathering the data from the application database and adding any minor logic to process it. It communicates with the database layer which in the structure of this project is first updated using the software's business logic layer which is described next.

Business layer

The business layer is responsible for carrying out the core logic that determines the application's functionality. In the case of the NaratAI application, it is implemented through the scripts that are responsible for evaluating news articles' sentiment and extracting polarity words and key phrases from them. Crucially, since the application requires the acquisition of these articles the two news scrapers built are also part of that layer. Lastly, in order to provide access to that information to the persistence layer via the database, the script that is used to update the database by implementing the core logic is also positioned in that layer.

Database layer

The database layer consists of a PostgreSQL relational database that is used to store the data required for the functioning of the application. The communication between the database layer and the persistence layer is mediated through the psycopg2 PostgreSQL adapter (Psycopg2 Documentation., 2021) which converts Python objects into string representations suitable PostgreSQL.

4.4 Software Deployment

When deploying to production the layered architecture described in the previous section was considered. As mentioned, the large size of some of the dependencies and static files used in the application dictates the modular separation of the different layers when deploying.

The lowest layer in the system – the database layer, consisting of the PostgreSQL database was deployed using Amazon Web Services (AWS) – Relational Database Service (RDS). The service provides database hosting and ensures continuation given outages occur. Issues like security are also handled by Amazon's RDS which makes it an appropriate choice for hosting.

The next layer in the hierarchy – the business layer was kept locally. An administrator is tasked with running database updates at least three times a day. Additionally, the behaviour of the scripts responsible for with updating the database is tracked easier when hosted on a local machine. This layer contains the large-sized dependencies and static files too so hosting it is hardly achievable given the resource constraints of this project.

The persistence, application and presentation layers were deployed using the Heroku cloud application platform (Heroku., 2021). Heroku uses Linux containers to run applications in the cloud and provides a reliable alternative for hosting an application. Two separate applications were created for the persistence and application layers and for the presentation layer. Observing the below figure and referring to Figure 5 the reader can understand the enactment of each layer when deploying.

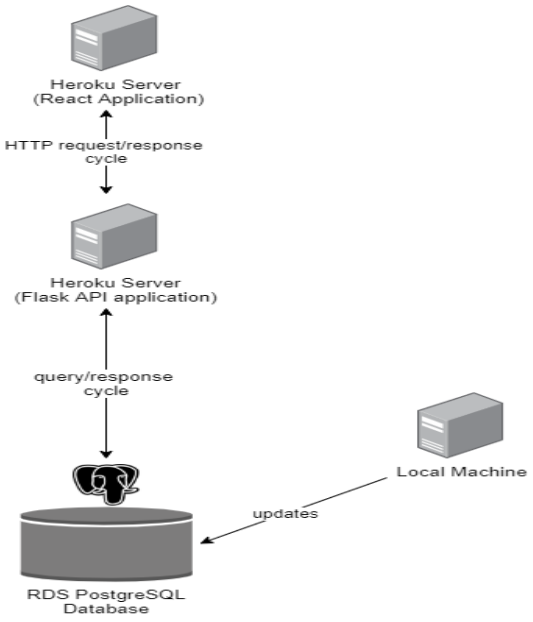


Figure 6: Deployment Architecture.

5 Machine Learning Architecture Overview

This part is meant to familiarize the reader with the choice of natural language understanding architecture and the process of creating the model used in the application. Firstly, the selected architecture is discussed. Secondly, the implementation used is described. Lastly, the process of training, evaluating and selecting the model used in production is depicted.

5.1 Bidirectional Encoder Representations from Transformers (BERT)

The backbone of the application relies on a state-of-the-art language processing architecture. The model chosen for implementing the core functionality of the software relating to sentiment analysis and key phrase extraction is a Bidirectional Encoder Representations from Transformers (BERT) developed by Devlin, et al., (2018). The model architecture builds upon the seminal Transformer network model (Vaswani, et al.,2017) and consists of 12 transformer blocks, 768 hidden layers, 12 attention heads, and 110 million parameters. The authors of the original paper further propose an architecture that has twice as many of each of the above-described features but it will not be considered due to its extremely high resource requirements.

Each word in any sequence is represented by a 768-dimensional vector and BERT also includes what is referred to as special tokens that are meant for usage in downstream tasks. The final hidden state of the [CLS] special token is the one employed for classification and it essentially serves as a summary representation of an entire sequence. As mentioned in section 2.1.2, BERT excels by considering text sequences in bidirectional manner which allows it to learn deep contextual representations of words and sequences. The model was trained on a large corpus of cross-domain texts from Wikipedia and the BooksCorpus. The two mechanisms used for error signalling are masked languge modelling (Masked LM) and next sentence prediction (NSP). The former involves replacing 15% of the words in any sequence by a [MASK] token and letting the model predict it. The latter technique implies mixing sequential sentences 50% of the time and allowing the model to predict whether the second sentence follows from the previous one or if it is a random replacement.

As described, one of the advantages of the architecture developed by Devlin, et al., (2018) is that it allow for fine-tuning on given downstream tasks using domain-specific data. Employing the model for sentiment classification requires the addition of an extra fully connected linear layer that will be trained for the task. The extra layer uses as input the [CLS] token embeddings. During the process of fine-tuning the model is first initialized with its pre-trained parameters. Subsequently, the embedding weights are adjusted through BERT's established learning mechansims - masked LM and NSP. The weights of the ouput layer for the given task, in this case sentiment classification, are learned using the labelled data. In that manner the model is trained end-to-end for the given task.

5.2 BERT Implementation

The Hugging Face library provides various implementations of the BERT model (Hugging Face., 2021). For the purposes of this project the BERT base model Pytorch implementation was used. The architecture consists of 12 transformer blocks (768 hidden layers), 12 attention heads, and 110 million parameters. The library also offers many options for pre-trained weights, and bert-base-cased (casing is considered) was the one chosen. Additionally, since for this project BERT will be used for sentiment classification, the Hugging Face library provides a model that includes an extra fully

connected linear layer that can be trained for the task of sentiment classification - BertForSequenceClassification (Hugging Face., 2021). This classification head is built on top of the [CLS] pooled output mentioned above. The Hugging Face library also provides text tokenizers for each of the pre-trained models they offer (Hugging Face., 2021).

5.3 Model selection

5.3.1 Overview

In order to select a model for use in the application the following process was adopted. Firstly, the original BERT model was fine-tuned using a domain-specific dataset for the task of sentiment classification. Stratified 10-fold cross-validation was used and the process of fine-tuning was repeated 5 times using different choices of hyperparameters and with slight adjustments in data pre-processing. 4 models that exhibited highest macro-averaged f1-scores were exported for testing on the test portion of the dataset and lastly were further tested on a separate dataset built for this project that contains entire news articles. In that manner, the performance of the models coupled with the logic applied for classifying long sequences is more robustly evaluated. The model that performed best on the dataset of articles was selected for use in production. Overall, the described process seeks to argue the selection of the model based on three aspects. First, the evaluation metrics during fine-tuning are assumed to provide an indication about the correctness of choice of hyperparameters. Second, once a model is saved during fine-tuning, in order to prove its ability to generalize on unseen data, it is further tested on the test portion of the dataset. Third, in an effort to validate the capabilities of the model to produce viable results coupled with the logic adopted for classifying long texts (discussed later), the models were also tested on a dataset of news articles built for this project.

5.3.2 Dataset (the reader is referred to the 'Model Data Generation and Exploration' notebook as a supplement to this section)

Two distinct datasets were employed to create the final dataset. The datasets contain news headlines and sentences extracted from financial news and their respective sentiment label on a ternary scale of negative, neutral or positive. The first set was generated by Malo et al. (2014) and the second one by Sousa et al. (2019).

The dataset built by Malo et al. (2014) consists of 4,840 financial news headlines relating to companies present in the OMX Helsinki - an index of companies listed in the Helsinki stock exchange. The headlines have been labelled by 5-8 annotators, all experts in the finance domain. The second dataset by Sousa et al. (2019) consists of 1109 observations, each being a sentence or two sentences extracted from one of 582 different financial news articles. Some of the sources used include: CNBC, Washington Post, New York Times, Forbes, Business Insider, and others.

After the two datasets were merged in a final dataset, the label distribution of the 5,946 observations was the following: neutral – 3,418, negative- 954, positive – 1,574.

5.3.3 Fine-tuning Procedure (the reader is referred to the 'BERT Fine-tuning' notebook as a supplement to this section)

Train/test split

Firstly, a train/test split of proportion 80/20 was generated using the scikit-learn Python library (Pedregosa, et al., 2011). The train split is to be used during the fine-tuning process and the test portion was kept for further evaluation of the best performing models according to the evaluation metric chosen (discussed later).

Choice of input length

A limitation of BERT and other models that build on the Transformers architecture developed by Vaswani, et al., (2017) is that there is a limit on the sequence length they can process. BERT uses attention mechanisms, the complexity of which scales quadratically as sequence input grows (Devlin, et al., 2018). Furthermore, when generating the contextual representations, BERT uses positional embedding implying that the model will not be able to generalize provided that the length of the sequence it needs to process is longer than what it has seen in training (Devlin, et al., 2018, Pappagari, et al., 2019). Therefore, the model can take only a maximum of 512 tokens at a time. Given that constraint, a maximum length of 300 for each observation was chosen as a balancing compromise between allowing the model to learn longer dependencies while maintaining computational costs at a reasonable level. Input that is shorter than the selected length will be padded. The tokenizer the model uses includes an attention mask for each example encoded that aids it in distinguishing between padding tokens and tokens it needs to consider.

Training/validation loops

As mentioned in section 5.1, BERT's original embedding weights are adjusted using Masked LM and NSP for error signalling and the fully-connected linear layer learns using the labelled data. Cross-entropy loss is the loss function used as multi-label classification is done. Due to the rather small dataset, stratified 10-fold cross-validation was employed in order to reduce bias when evaluating the performance of the models. The choice of using stratified rather than normal k-folds stems from the unbalanced label distribution of the dataset. The stratified k-fold implementation used was provided by the scikit-learn library and it ensures a balanced distribution between labels in the different samples in each fold. Impliedly, evaluating the performance of the model after each fold should contain less bias that would normally be injected due to the randomness of observations picked for each fold's sample.

Each epoch consists of a training and validation loops and after each validation loop, the macro averaged f1-score was evaluated and if its value exceeded the previous highest, the model was exported for further testing.

Hyperparameters choice

Devlin et al. (2018) suggest different choices of hyperparameters could be attempted during fine-tuning. Based on the values they propose, the above-described process was repeated using different combinations of hyperparameters. The adjustments made concerned the data loaders' batch size, the learning rate and the number of epochs. Additionally, for one of the models some extra data pre-processing was applied. Punctuation irregularities like leaving spaces between apostrophes, full stops and other punctuation marks were adjusted. That argument for using such an approach stems from the fact that BERT was trained on text corpuses without any pre-processing. Therefore, in order to recreate the format of the data used in the original method, the described data pre-processing was conducted. Four separate models exhibiting an overall macro-averaged-f1 score of around 0.8 during cross validation were exported for further testing on the test portion of the dataset.

Choice of evaluation metrics

The models' performance during fine-tuning and when testing on the test portion of the dataset is evaluated using the macro-averaged f1-score. The reason for deeming other evaluation metrics like accuracy, precision, recall or simple f1-score not appropriate stems from the label imbalance of the dataset used. Additionally, equal importance of each class should be attributed hence avoiding using just the f1-score. The macro-averaged f1-score expands on the simple f1-score by averaging the f1-score for each class. Impliedly, it should communicate both information about the true positive rate (recall) and incorrect predictions (precision) while avoiding the pitfall of using these metrics in a stand-alone fashion which could be deceiving due to the mentioned characteristics of the dataset. One potential weakness of the macro-averaged f1-score is that it could overestimate the model's performance in cases where it performs really well in classifying a given smaller class but not as good in others. As mentioned after the highest-scoring models according to the evaluation metric chosen were seeded out, they were further tested on the dataset containing news articles. The evaluation metric used in for that stage was accuracy.

5.4 Final Model Selection (the reader is referred to the 'Selecting Model for Production' notebook)

Having obtained the 4 models that scored highest macro-averaged f1-score during cross-validation, their performance was further evaluated on the test portion of the dataset. Three of the models scored a comparative macro-averaged f1-score to what was observed during cross validation. Values within the range of 0.8-0.87 were obtained for three of the models and one performed worse, scoring 0.65.

Lastly all models were evaluated on the dataset built for this project consisting of labelled news articles. The dataset contains news articles headlines, their sources, a URL that is used to acquire the articles' bodies and a sentiment label on a ternary scale of negative, neutral and positive. It was annotated by five people with expertise in the field of finance, including the author of this project. 69 news articles from reputable sources were gathered but unfortunately, due to some news websites blocking web scrapers, only 42 could be used for evaluation.

Interestingly, the model that performed best in scoring the articles was not the one that also achieved the highest macro-averaged f1-score. It managed to guess 35 out of the 42 articles correctly, hence

achieving an accuracy of 0.83. The selection process was completed by choosing that model for use in the application.

6 Core functionality implementation description

This section is meant to describe the process of implementing the practical software solutions for the application's core functionality. The scientific background of each decision is described and where novel logic was used, careful justification is provided. Firstly, the two news scrapers created for the purpose of acquiring news articles are discussed. Second, the process of evaluating the sentiment of each article and generating a sentiment score for a given asset is described. Third, the logic and tools used for extracting polarity words from news articles is discussed. Lastly, the logic for acquiring key phrases from the evaluated news texts is illustrated.

6.1 News scrapers

As mentioned, the acquisition of news articles for the supported stocks and crypto currencies was made possible by building two news scrapers for the two classes of assets supported.

Stock news scraper

The stock news scraper relies on one of the most popular stock screening platforms – Finviz (Finviz., 2021). The platform offers a variety of different information concerning stocks, including news articles links from many different reputable sources. The Beautiful Soup 4 HTML parser (Richardson, L., 2020) is used to scrape the news articles URLs for any given stock only for the current day. Since often times articles that are irrelevant for the given stock will end up in the news section of the Finviz dashboard, additional logic is implemented that only retains URLs for articles that are relevant for the given stock. The logic is enacted by checking whether the article headline contains certain keywords relevant for the given stock.

Obtaining the article body is done by first sending HTTP requests using the URLs scraped and the Python requests library. Subsequently, the HTTP response payload is passed to the Newspaper3k article scraping library (Ou-Yang,L., 2013) to extract the article text. Some sources prevent web scrapers from acquiring articles' bodies and even if the HTTP response is OK (200), the scraped text contains a warning message. Therefore, in order to avoid such noise reaching the database extra logic is applied that disregards texts, the length of which appears to be too short.

Crypto currency news scraper

The scraper used for crypto currency news functions in the same manner with the only difference being the acquisition of article URLs. More specifically, the Crypto News API was the source used (Crypto News API., 2021). The API provides endpoints that respond with daily news articles headlines and source URLs for any given crypto currency (passed as a parameter to the GET request). As mentioned, the remaining process of acquiring the article bodies remains the same as for the stock news scraper.

6.2 Sentiment analysis and sentiment score generation

Arriving at a final daily sentiment score for an asset necessitates two consecutive steps:

1. The evaluation of sentiment of each article for the given asset.
2. Aggregating the obtained results from all articles for the given asset to arrive at a final sentiment score.

This section describes both steps in detail and provides research grounding for each decision taken.

6.2.1 News Article Sentiment Evaluation

Paragraph segmentation

Having obtained the news texts using the news scrapers, each text's sentiment is analysed. As described in section 5.3.3 BERT's input is limited to 512 tokens which renders evaluation of entire news articles at once not possible. Just recently, during the held in September Intelligent System conference, Lin et al. (2021) proposed a solution, namely expanding BERT's positional embedding to support longer sequences. Their method might be worth exploring in the future but it could not be considered for the purposes of this project due to its release being too recent. Pappagari, et al., (2019) approach the problem from a different perspective by suggesting that longer texts can be split into sequences of fixed length with some overlap. These segments are then stacked and fed into an additional LSTM layer to generate a final document representation that can be used for downstream tasks like sentiment classification.

Despite having merit, the semantic structure of news aricles renders such approach a questionable strategy. News articles usually have delineated topical paragraphs that might convey different semantic meaning and simply splitting according to length could lead to distorions in discourse representation. Consequently, for the purposes of this project, a novel approach to proccessing longer sequences was employed, namely by using text tiling. Text tiling is an algorithm developed by Hearst, (1997) that uses shifts in lexical patterns to split a given text into subtopical paragraphs. The algorithm identifies breakpoints in text by relying on different similarity measures between blocks of k tokens. Vector space models of these aritfical blocks are created using term vectors and correltion coefficients between them are calculated to determine similairty. Additionally, other factors like the number of new terms introduced ouside of the artificial gaps created are considered. The implementation used in this project was provided by the NLTK Python library (Bird, et al., 2009).

There are two edge cases in which text tiling would not be used for splitting the text. Firstly, rarely, the text tiling algorithm is not able to detect any breakpoints in a given text. Usually, that tends to occur if the article is too short or uses any peculiar punctuation (e.g bullet points). Secondly, it could happen that even after a text was split using text tiling, one or more of the paragaphs exceed the discussed maximum sequence length of 300 tokens (refer to section 5.3.3). In both situations defensive programming is used and the article or paragraph is recursively split into paragaphs. The length of the newly created paragraphs is determined by establishing a midpoint based on the number of sentences in the text block and splitting them into two new paragraphs. The process is recurisevely repeated until the maximum sequence length condition is met.

Extracting overall article sentiment

Having obtained the topical paragraphs of the article, each is pre-processed to prepare it to be inputted through the fine-tuned BERT model. That involves tokenizing the paragraphs using the special BERT tokenizer provided by the Hugging Face library (Hugging Face., 2021). The input generated by the tokenizer is a dictionary containing two tensors. The first one includes BERT's input ids for each token and the second one contains the attention mask for the entire sequence. As discussed, the attention mask is used to flag which tokens BERT needs to consider during evaluation and which are padding tokens (refer to section 5.3.3). The model outputs a tensor containing the unnormalized probability distributions for each of the three classes considered. The final probabilities for each of the three classes are obtained using a SoftMax function. Subsequently, these probabilities are pooled to arrive at the final class probabilities for the entire article. The class scoring the maximum probability would be considered as the article's overall sentiment.

After observing the model's performance and the semantic structure of many of the news articles in the specific domain, a decision was made to reduce the pooled neutral probability for the final article by 50%. The argument for that choice is that very often an article would have one or two highly negative or positive paragraphs and then an additional two or three that are more informative and are evaluated as neutral. Despite the fact that the given article clearly tries to convey a negative/positive connotation, it would be evaluated as neutral. Through reducing the neutral probabilities, articles that indeed contain neutral discourse are not falsely evaluated because their negative and positive probabilities would still remain low and will not have an impact on the final evaluation. On the other hand, in cases where the neutral probability just slightly exceeds the positive or negative one due to the described characteristics of news texts, the given article will not be misclassified as neutral. The described reduction in neutral probabilities led to a 10% increase in accuracy based on the news articles dataset (refer to the 'Selecting Model for Production Notebook'). Figure 7 depicts the process of article sentiment evaluation graphically.

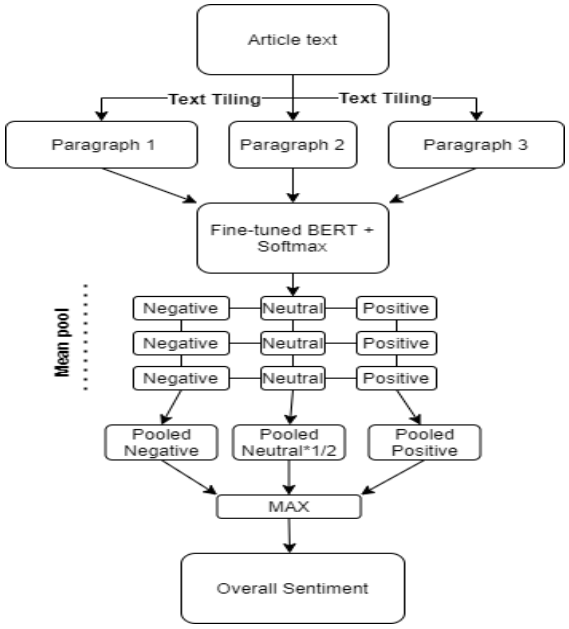


Figure 7: Overview of logic used to derive a news article's sentiment

6.2.2 Asset Daily Sentiment Score

After the overall sentiment classes and class probabilities for each article are acquired using the above-described logic, the prerequisites for generating the final daily scalar sentiment score for any given asset are in place. The score is meant to allow users a deeper understanding in the expressed positivity or negativity in news articles for the asset of their interest that goes beyond simple categorization.

Use of class probabilities

The constituencies used to derive the score are the positive and negative probabilities of all articles for the given asset that were evaluated as positive or negative. Maas, et al. (2011) depict the importance of considering nuance and spectrum of sentiment and describe it as a complex and multi-dimensional concept. As previously described, BERT learns deep contextual language representations that contain rich meaning of the attitudinal, social and other semantic contingencies that arguably are able to capture a lot of that complexity. The obtained article class probabilities are highly contingent on such semantic dependencies BERT's representations embed. Impliedly, an article that uses language that is strongly polarizing will have significantly higher probabilities for the given class that it was categorized to belong to (based on the maximum class probability). Conversely, one where sentiment is conveyed in a less expressive manner will exhibit a lower probability for its overall categorization.

Considering the aforementioned, a simple binary or ternary labelling of any given article will miss a lot of the contextual meaning a scalar representation can communicate. The user would be abstracted away from a lot of meaning a given article could convey as articles that might vary vastly in their degree of negativity or positivity would be indistinguishable. Based on the described arguments, the next-described methodology was employed for calculating a scalar sentiment score for a given asset.

Weighting scheme and sentiment score calculation

First, the class probabilities for all articles for the given asset that were evaluated as negative or positive were mean pooled to generate the final asset sentiment class probabilities. Next, a weight was attached to the positive and negative probabilities based on the number of articles that were evaluated as overall positive or overall negative respectively. The weighting serves to balance the importance of both classes and also assures that both the probability and number of articles are equally significant determinants of the final score. For example, if a given asset has two highly negative articles and three or four moderately positive according to the class probabilities, the sentiment score would still reflect the fact that the negative articles were more polarizing compared to the positive ones. It would although not be highly negatively skewed as after all there were three positive articles that were also evaluated. Table 6 describes that hypothetical example. Lastly, to generate the sentiment score the two weighted pooled probabilities are summed to arrive at a number between -1 and 1 where 0 would be considered as neutral. The value is scaled times 100 for a better visual representation in the user interface. Below the reader can observe the summarized formula for deriving the sentiment score and Figure 8 depicts the process graphically.

$$\text{Sentiment score} = \mathbf{((-w_0 * \sum^{n} P_{negative}) + (w_1 * \sum^{m} P_{positive})) * 100}$$

Where:

n=total evaluated negative articles

m=total evaluated positive articles

P= class probablity

$w_0 = -n/n+m;$

$w_1 = m/n+m$

	Article 1	Article 2	Article 3	Article 4	Article 5
Overall sentiment	Negative	Negative	Positive	Positive	Positive
Sentiment class probability	0.9 Negative	0.95 Negative	0.25 Positive	0.3 Positive	0.35
Sentiment score = ((-(2/5) *(0.9+0.95)) + ((3/5) *(0.25+0.3+0.35))) *100 = -20					

Table 6: Hypothetical example of sentiment score generation

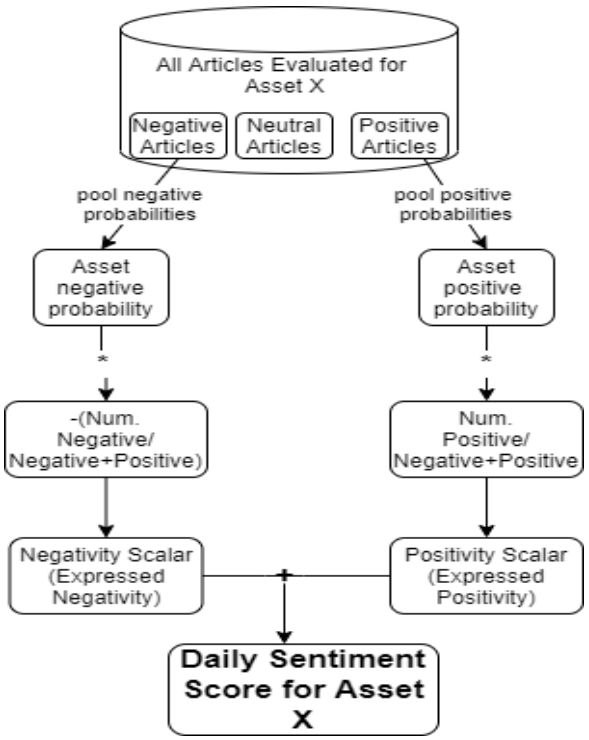


Figure 8: Overview of logic used for deriving the daily sentiment score of a given asset.

Preparing response for front-end consumption

As described, the database is queried for the sentiment probabilities for a given asset and the sentiment score is generated based on the acquired data. The endpoint serving the above-described functionality responds with a JSON literal consisting of key-value pairs- the asset ticker passed as a parameter in the URL and its corresponding sentiment score.

6.3 Polarity Words Extraction

Overview

The first layer of information that would enable users an understanding about the drivers of the daily sentiment towards a given asset consists of polarity words extracted from the evaluated news articles. In order to enable such functionality a sentiment lexicon consisting of words and their sentiment evaluated on a binary scale of negative or positive was built (the reader is referred to the 'Sentiment Lexicon Generation' notebook for detailed information). Additionally, a rule for negation was applied in order to avoid out-of-context interpretation of given words' sentiment. Below the process of lexicon generation, the negation rule logic and actual steps taken to extract polarity words from news articles are described.

Lexicon generation (the reader is referred to the 'Sentiment Lexicon Generation' notebook as a supplement to this section)

Two existing lexicons were merged to create the sentiment lexicon used for this project. The selection of the given two datasets was based on the work of Shapiro, et al., (2020). They evaluate the performance of different sentiment lexicons in classifying financial news texts which deems their findings particularly suitable for the domain of this project. The authors find that combining the lexicons of Loughran & McDonald, (2011) and Hu & Liu, (2004) showcased best results. Loughran & McDonald's lexicon has particularly high merit as it tackles the problem of domain-specificity when it comes to word sentiment (refer to section 2.1.3). It was built specifically for the finance and economics domain using companies' 10-K reports. The lexicon has since received regular updates and its latest version from 2020 was used. It contains 2,709 words of which 2,355 are negative and 354 are positive.

The second lexicon used – Hu & Liu's, expands the rather small vocabulary of Loughran & McDonald's lexicon. It contains 6,800 words and was built using movie reviews. Again, the distribution of word polarity is skewed towards the negative class and of a total of 6,789 words, Hu and Liu's lexicon contains 4,783 negative words and 2,006 positive. When merging the two lexicons all duplicates were removed and for any given duplicated word precedence was given to the entry in Loughran & McDondal's dictionary due to the domain-specificity value it provides. As described in section 2.1.3, almost three-fourths of negative words found in cross-domain sentiment lexicons have opposing polarity in the finance domain. The final lexicon used contains 8,453 words of which 6,294 are negative and 2,159 positive.

Heuristic rule for negation

In order to avoid any misrepresentation when extracting polarity words, the context in which they occurred was also considered. More specifically, a negation rule was applied that inverts the sentiment of a word if it was preceded within three words by any form of negation and no punctuation separates the word and the given negation term. In the interest of correctly representing such words, apart from inverting their sentiment a 'not' term is added before them.

Let us consider the sentence *'Analysts are not bullish on Tesla's long-term potential, however, many remain confident in the company's value*.'. *Bullish* would not be wrongly represented as positive, instead the extracted word would be '*not bullish*' with a negative sentiment. Additionally, the word

confident would still be correctly represented as positive despite the fact that there is a negation term preceding it in the sentence. The applied negation rule was adapted from the research of Das & Chen, (2007) and was also used by Pang, et al., (2002) and Potts, (2010). The negation terms that were considered as polarity inverting were derived from the VADER rule-based model for sentiment analysis developed by Hutto & Gilbert, (2014). The actual list consists 59 terms and was obtained from the NLTK Python library (Bird, et al., 2009) that offers an implementation of VADER.

Preparing response for front-end consumption

Using the generated sentiment lexicon and the negation described, extracting polarity words from a given article that involves the following steps. Firstly, the article is tokenized using NLTK's TweetTokenizer (Bird, et al., 2009). Other tokenizers were also probed but the TweetTokenizer implementation seemed to perform best at detecting punctuation correctly which is crucial for the robust functioning of the negation rule applied next. The negation rule is practically applied by marking words that require sentiment conversion with a '_NEG' tag.

Subsequently, the pandas DataFrame (McKinney, 2010) containing the sentiment lexicon is queried for each token and the ones present in the lexicon are appended to a Python list with words having a '_NEG' tag replaced by 'not' preceding the word. An additional list with corresponding indexes is appended with the words' sentiment.

The described process is repeated for all articles for any given asset and the results are saved in the application database. In order to acquire all polarity words from the evaluated articles for a given asset, the database is queried. The obtained data is stored in a pandas DataFrame that includes columns for the polarity words, their corresponding sentiment and the source they were derived from. The data contained in the source column will be used to satisfy requirement 10.a.iii stating that the source of each word should be displayed upon hovering over the given word in the user interface (discussed later).

Finally, the ten most-occurring polarity words and their corresponding sentiment are extracted by operating on the data frame. Since according to the specification, users should be able to toggle between seeing all words, just the positive or just the negative words in the word cloud interface, the ten most-occurring positive and ten most-occurring negative words are also acquired from the data frame. For example, the word 'brilliant' might be the 5th most-occurring polarity word overall but also the most-frequent positive one.

The endpoint serving the above-described functionality responds with a JSON literal containing key-value pairs with keys corresponding to the three categories of 10 most-occurring polarity words (all, positive, negative) and values – the arrays of words.

6.4 Key phrases extraction

Overview

The second layer of functionality allowing users an insight into the context of what determines an asset's sentiment consists of key phrases obtained from the evaluated articles. The method used involves firstly splitting an article text into candidate key phrases (n-grams), acquiring an embedding representation of each candidate and then comparing it to an embedding representation of the entire news article. Lastly the candidates with highest similarity to the article embedding are split into positive and negative using the sentiment lexicon built for this project. Next, each step is discussed in detail.

Generating candidate key phrases embeddings

Firstly, splitting the article into candidate n-grams was done using the scikit-learn Python library's Count Vectorizer (Pedregosa, et al., 2011). The implementation further includes various pre-processing options and stop word removal was applied.

Secondly, an embedding representation for each of the candidate n-grams must be obtained. As described in section 5.1 during fine-tuning the BERT model, both the classification head is trained using error signalling from the labelled data and the word embedding weights are tweaked through BERT's learning mechanisms – Masked LM and NSP (refer to section 5.3.3). Impliedly, the contextual word and sequence representations of the fine-tuned model should contain even richer semantic meaning for the domain of the data used for fine-tuning. Normally, BERT is used for downstream tasks like sentiment classification or next sentence prediction but recent research advanced the notion that the contextual embeddings generated by the model could be used in a static manner for tasks such as word similarity and keyword extraction.

Indeed, Bommasani et al. (2020) describe the superior performance BERT's embeddings compared to other static embeddings like Word2Vec or GloVe in such tasks and propose certain strategies on how to use BERT's embeddings to represent words. The two proposed strategies are referred to as the decontextualized approach and the aggregated approach. In the former, word embeddings are directly obtained from the pre-trained BERT model and since often one word is represented by more than one token by BERT, some form of pooling is applied to arrive at a final word representation. The researchers conclude that mean pooling produced best results. The latter strategy involves firstly processing large corpuses of text containing the words of interest in various contexts and then pooling their different representations.

The decontextualized strategy was considered more appropriate for the purposes of this project since the words semantic meaning only necessitates consideration in a single domain. Moreover, as argued, the fine-tuned model weights ought to provide an even deeper contextual richness to the word embeddings in the specific domain. Acquiring the word vectors was done by indexing into the last hidden state of the model output. The decision to do so was argued by Ethayarajh (2019), who found that word representations in upper layers of contextualizing models like BERT embed more context-specific semantic meaning.

Next, since key phrases rather than keywords are considered, an additional step involving pooling must be applied in order to gather the representation of a given candidate key phrase. Again, in line with the suggested pooling method by Bommasani et al. (2020), mean pooling was used and the words' 768-dimensional vectors were averaged to produce one vector representation of the entire candidate key phrase.

Generating a news article embedding

Having obtained the candidates representation, next, an embedding for the entire article must be acquired in order to compare their similarity. Due to the discussed limit of input tokens BERT can take, as done for sentiment classification, the news article is first split into topical paragraphs using text tiling (Hearst, 1997). As previously mentioned, apart from single word embeddings, BERT also provides a special [CLS] token that has the same dimensionality of the word embeddings. Its final

hidden state is meant as an aggregated representation of the entire sequence, in this case of the entire paragraph. Considering that, in order to obtain an embedding representation for any given news article, the [CLS] tokens of each of its paragraphs were pooled.

Evaluating candidate key phrases semantic importance

Finally, in line with what Sharma & Li (2019) and Ethayarajh (2019) propose, the semantic importance of each candidate key phrase was ranked by deriving its cosine similarity with the entire article embedding. According to the cited research, such an approach is particularly effective when contextual representations are used. Despite the fact that the researchers do not explicitly discuss the choice of distance metric, cosine similarity is perhaps more appropriate as it uses L2 length normalization when considering the vector dot product. That implies that, any impact of vector magnitudes purely generated by factors like word frequency are diminished.

$$\text{cosine similarity (k,v)} = \frac{k \,.\, v}{||k|| \,.\, ||v||}$$

Figure 9 depicts the process of extracting key phrases from a given news article graphically.

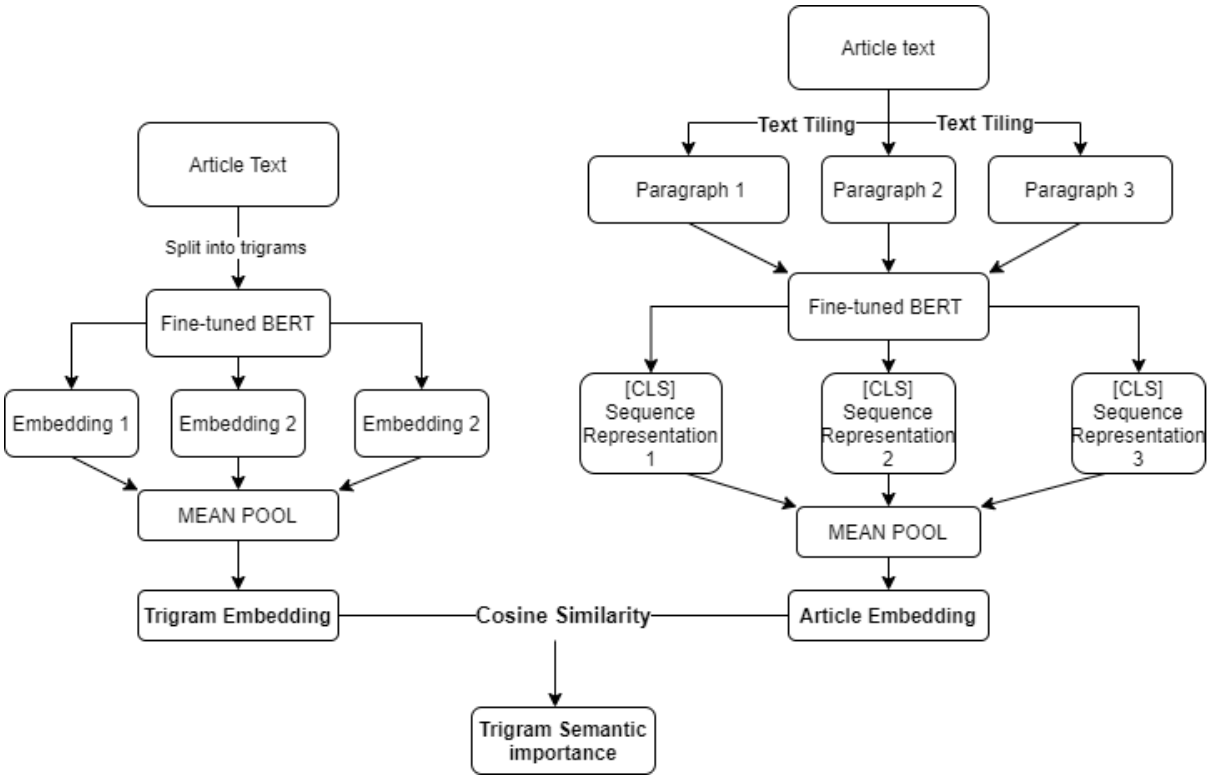


Figure 9: Process of Key phrase Extraction.

N-gram length

The above-described process was attempted using n-grams of different length. Trigrams appeared to produce best results and since these phrases are to be served to the user in a word cloud, length was a limiting factor rendering longer n-grams not viable. A potential explanation for why using bigrams seemed to generate more noise is that perhaps pooling more embeddings inevitably increases the semantic richness embedded in a representation and brings it closer to the overall article embedding.

Categorizing key phrases as positive or negative

After obtaining the ranked candidates, in order to classify each phrase as either positive or negative, a lookup of their constituent words was done in the sentiment lexicon. Employing the lexicon not only allows the option to consider key phrases according to their sentiment but also ensures that only phrases that contain relevant semantic information are kept.

Preparing response for front-end consumption

For each article evaluated seven key phrases that showed highest cosine similarity are stored in the database. The exact number was chosen considering limitations of the front-end interface where they will be rendered. The database is queried for the key phrases and they are first organized in a pandas DataFrame together with the headlines of the articles they were extracted from. As for polarity words, the source is needed to satisfy requirement 10.b.ii stating that the source of a given phrase should be displayed upon hovering over it in the user interface (discussed later). In order to adhere to the requirement that users should be able to toggle between positive and negative key phrases (requirements 4.a, 10.d), phrases' sentiment is evaluated. In doing so, an extra column for the key phrase sentiment is appended and the sentiment lexicon data frame is queried to obtain the corresponding polarity of each phrase. The endpoint serving the described functionality responds with a JSON literal containing three arrays consisting of the positive and negative, only positive and only negative key phrases.

7. User Interface

Before proceeding with this section, the reader can refer to Appendix 14.3 to observe the different component of the application design.

7.1 Design

Designing the user interface was done with thought to provide a pleasurable user experience that is intuitive and frustration-free. Norman's six principles were considered when designing the components of the application (Norman, 1990). The front end of the application was implemented using HTML, CSS and JavaScript and more specifically the React JavaScript library. The basic layout of some elements in the selection page were adopted after following the '*React Data Visualization'* course. Next, each design principle is discussed together with the efforts made to uphold it.

Visibility

The visibility principle states that meaningful information that has the highest importance for the user should be easy to find and access. In the selection page the user is directly exposed to the grid of assets, together with their logos for even easier recognition. A search bar is also provided should the user fail to spot the asset that interests him. The conditional darkening of the colour of the tiles containing the asset also allows for quickly grasping information about the assets with most evaluated articles.

The dashboard interface provides the most important information, namely the sentiment score and polarity words and key phrases in a format that is intuitive and easy to understand. The conditional colouring of the words, phrases and headlines according to their sentiment also enhances the effectiveness of consuming the information.

Feedback

The feedback principle dictates that users should be provided with feedback for the effect of their actions. Hovering over or selecting any of the elements provides graphical feedback. Usually, that is achieved by certain colour changes or shadows being conditionally applied. Examples are: dimming the already selected assets from the selection grid, highlighting the asset currently placed in the spotlight, displaying a red shadow and a cross to indicate clicking a tile in the favourites bar removes it and others.

Moreover, when the application fetches the required data '*Loading'* indicators are provided.

Affordance

The affordance principle is concerned with designing elements that provide clues to what their intendent use is and most importantly, do not provide false such. An example of the application of that principle is present in the selection page where clicking an asset tile can do two separate functions depending on where it is. If it is in the selection grid, clicking on the tile adds it to the favourites bar whereas once added there, clicking it removes it from favourites. Such polymorphism is achieved by conditionally applying styles to the tile upon hovering over it that direct the user about what a clicking event would do.

Mapping

The mapping principle relates to how are given controls connected to the function they do. Examples would be the positive and negative toggler buttons in the headlines panel and the polarity words and key phrases toggler buttons controlling the content rendered in the word cloud. Their mapping is made explicit and the user does not have to guess their functionality.

Consistency

The consistency principle advises on using the similarly designed elements to achieve similar tasks allowing for a shorter learning curve when understanding how to interact with an interface. An example for that would again be the design of the positive/negative and polarity words/key phrases toggler buttons. They serve the same logic and are therefore designed to look the same.

Constraints

Constraints are meant to limit the actions that could be performed on the system. They are meant to prevent incorrect actions taken by users. An example in the NaratAI application would be the disabling of any asset tiles that are already placed in favourites, prompting users to first select their favourites before navigating to the dashboard page and others.

7.2 Use case scenario

The below use case scenario seeks to emulate how a user would interact with the application user interface. The hypothetical example also aims to highlight how the software could provide value to its users.

Informing a decision to invest in AMC, Ethereum, Dogecoin or Google

Initial assumptions:

- The user has previous investing experience
- The NaratAI database has been correctly updated with the most current information
- Heroku servers are running and are able to serve the back and front-end parts of the application
- Amazon servers where the database is hosted are running and the database is reachable

Normal:

The user is a day trader interested in short-term investments that could be profitable when price volatility is predicted. The user navigates to the NaratAI website and is prompted to choose up to five assets from the asset grid. The grid is conditionally styled and the assets with most news present for them on the given day have a more pronounced colour. The user observes that and sees that Google has no potential price swinging news and decides investing in it is not worth it. However, AMC, Ethereum and Dogecoin have been pretty active and he selects all of them and clicks the confirm favourites button which then navigates him to the dashboard interface. He quickly spots that Ethereum and Dogecoin have a sentiment score of 60 and 20 respectively, whereas AMC scores at -30. The user places Ethereum in the spotlight and sees that its price has risen almost 15% since yesterday's closing and that its sentiment score has been consistently rising for the past 4 days.

He scrolls down and sees that of 20 evaluated articles 15 were positive, only 2 negative and 3 were evaluated as neutral. The user navigates to the headlines panel and quickly gauges what were the evaluated news. On the right-hand side, he scans through the polarity words and key phrases and amongst all positive words like *bullish*, *rallying*, *breakthrough* he sees a few negative words like *resistance*, *bearish* and *risk*. The user decides to check the key phrases from the sentiment driving articles and sees the key phrases *all-time high* and *key resistance*. He hovers over the *key resistance* phrase and navigates to the source from which it was extracted in the headlines panel. It is an article

from Bloomberg and after reading it the user is convinced that a correction of the rising price is imminent. He opens the trading platform he uses and takes a short position in Ethereum.

What could go wrong:

- The application is not rendering the correct data fetched from the database and the user had made a decision based on stale information.
- There is an inconsistency between the asset placed in the spotlight and the data provided in the sentiment chart and word cloud.

System state on completion

- The system stores the favourites the user has selected in browser memory to allow for quick access should he wish to revisit.

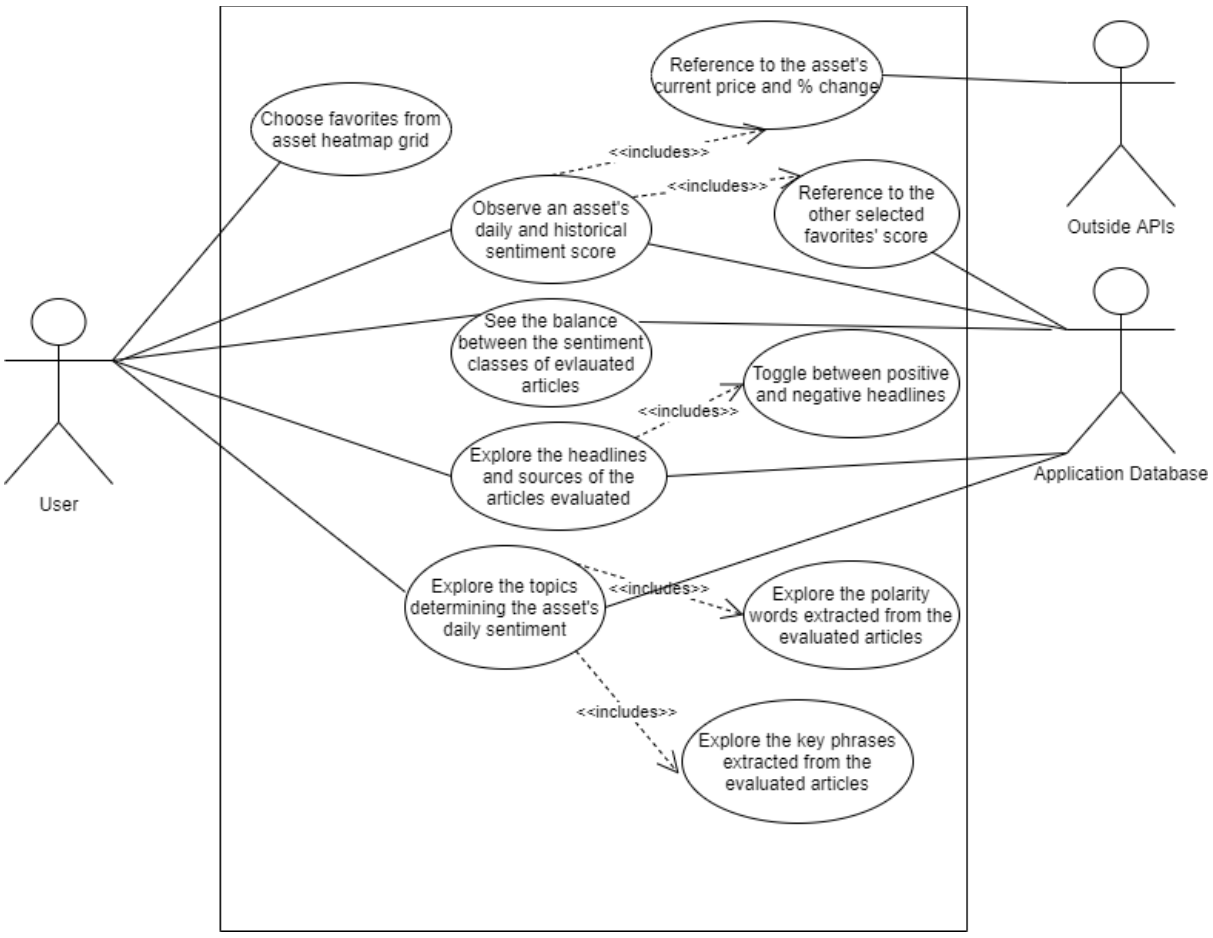


Figure 10: Use Case Scenario UML

8. Testing

8.1 Approach

In order to ensure the reliable and robust functioning of the system according to its specification, a comprehensive testing strategy was designed.

Tests were conducted on different levels of the system adopting the black and white-box strategies. The former includes functional and non-functional tests, namely testing some of the functional and non-functional requirements elicited during the requirements engineering stage of the project. The latter was implemented via unit tests aimed at probing the reliability of certain functions used for the core logic of the application.

Moreover, integration testing was conducted throughout the development of the system. A test-fix approach was taken with regard to integration testing, meaning that after a unit test on newly added functionality failed and the problem was fixed, all involved parts of the software were retested. Upon finalizing the entire software, system tests were carried out to ensure the correct synchronized functioning of all components. Finally, as a form of usability/acceptance testing, five users falling within the target audience of the application were recruited to interact with the software and provide feedback.

8.2 White-box testing

Unit tests were conducted on the functions responsible for the core logic of the application. Additionally, the database integrity was probed and the scripts responsible for updating the database were systematically evaluated. The reader can see the exact tests carried out in the *tests* directory of the project. Appendix 14.2 lists the functions tested and the outcome of the tests.

8.3 Black-box testing

As mentioned, functional and non-functional tests were conducted in an effort to ensure that the user requirements described in Appendix 14.1 are adhered to. Furthermore, the system tests conducted after the system development was finalized were also done at this stage. Appendix 14.2 describes the conducted tests and results tabularly.

9. Discussion

The goal of this project was to provide a useful tool for investors grounding its development in recent research and novel logic. Based on that, the final result could be considered as successful and exceeding what was considered achievable when it was still in a conceptual phase. Novelty was implemented in each stage of the process of development and yet it was carefully argued based on seminal or recent promising research in the field of natural language understanding. The process was made possible by gaining a very deep understanding about the available research and the detailed functioning of the state-of-the-art language architectures. The final product severs its purpose and satisfies everything that was laid out during the requirements elicitation stage.

9.1 Research aspect

Firstly, analysing longer text sequences using models that learn deep contextual representations like BERT is still largely considered a gap in literature. This project addressed that, by implementing logic to derive sentiment of long texts using an approach never previously used, namely by adopting text tiling. That strategy was carefully argued based on the context of the semantic structure of news texts in the given domain. Additionally, considering the sentiment extraction logic used and the structure of the texts being analysed, the decision to reduce neutral probabilities by 50% proved its correctness. An increase of 10% in accuracy when evaluating articles was observed following the adjustment. To the knowledge of this project, such an approach has not been used before.

Using BERT for analysing news articles in any domain let alone finance has not been previously done. There is research that considers only headlines which would be vastly limiting if it was to be used as a proxy for the actual sentiment of the article. Section 9.2 will provide an illustrative example demonstrating that.

Judging by the achieved macro-averaged f1-score of almost .85, the choice of hyperparameters for fine-tuning was also good and the procedure was executed successfully. The merit of the interaction between the logic developed and the fine-tuned BERT model used in production is also visible based on its accuracy in scoring the articles dataset built for this project (refer to notebook ‘Selecting Model for Production’). Almost 83% of them were classified correctly which despite the dataset’s small size is a good indicator. Lastly, having observed the behaviour of the system using numerous articles for almost a month, this project can confidently argue that the perceived accuracy in classifying articles correctly and deriving a sensible sentiment score for assets is very high.

Secondly, extracting key phrases using a fine-tuned BERT model has also not been previously done. Using the contextual representations of the model with weights adjusted for the specific domain is assumed to provide even better results. Every decision made in the logic adopted, starting from which hidden layer embeddings to use to how to pool them to arrive at one representation for a phrase, was backed by research. Furthermore, the use of the sentiment lexicon built for this project to provide a further layer of insight by splitting phrases into positive and negative classes is a method not seen before. It also achieves the purpose of reducing noise which is an inevitable issue in the domain of keyword/phrase extraction. Deriving an embedding representation for entire articles is also an aspect that was novel in the domain as once again text tiling was used before paragraph vectors were pooled.

Lastly, the sentiment lexicon built for this project coupled with the heuristic rule created were also inspired by some of the highest-quality research in the field of natural language understanding and linguistics.

9.2 Practical aspect

The fact that the methodology used for the core functionality of the application described above is novel inherently implies that no existing software provides the same features. To the knowledge of this project, existing solutions only support sentiment analysis and it is often unclear what methodology they are using. The ones that do mention their approach, mostly use just article headlines or net scores of positive and negative words, both methods that could lead to highly inaccurate results. An example of how a headline might not communicate the exact sentiment of an article is provided in the below user story based on actual events occurring on 2021-09-06 and 2021-09-07. The value the application could provide is highlighted and visual references are provided in Appendix 14.4.

On 2021-0-06 Ethereum's price is rising and there is a lot of overall enthusiasms in the daily news. The user is undecided whether to invest as he fears that he might enter when the asset price plateaus or worse, corrects downwards. He opens the NaratAI application and immediately spots that Ethereum is one of the assets most discussed in the news based on its darker colour on the grid (screen 1). He confirms ETH as his favourite and on the dashboard page he sees that there are 11 positive articles and just one evaluated as negative (screen 2). The sentiment score is around 50 though which means that the expressed positivity in the positive articles was not particularly strong. The word cloud has positive polarity words like bullish, optimistic, gains but also negative ones like correction, fall and difficulty. The key phrases show a pattern that tells a similar story (screen 3).

The user decides the explore further and hovers over the word 'resistance' in the word cloud which shows that it was extracted from an article with headline 'Ethereum Price Eyeing $4k! On-chain Analysis Hints at New All-time Highs!'. The headline is seemingly very positive but in fact that is the only article evaluated as negative. The user directs to its source using the headlines panel and sees that the article discusses the potential of ETH hitting an all-time high but also stresses the fact that the coin is in a correction zone near resistance and that even if the price rises there will be a lot of selling pressure on it leading to potential downslide. Seeing that, the user decides not to risk buying.

The following day Ethereum is down almost 10%. The topics one can distinguish in the word cloud are clearly indicative that what the user was able to spot with the help of the software yesterday was indeed correct. (screen 4, screen 6)

The above story depicts how the software would provide invaluable insight that would have likely never been seen understood if a software just provides sentiment based on analysing headlines. Even if that was not the case, if it was not for the other functionality of NaratAI allowing the user to explore the topics, the fine-grained details would have been missed too. Appendix 14.5 further showcases the ability of the software to detect sentiment and the assumed expressed polarity in news contained in the daily sentiment score. If the percentage drop in price of these assets is considered as an artefact of how negative the 'narrative' towards them is, one could see that assets with higher price drop seem to also have a more negative sentiment score. The difference in articles evaluated also did not skew the score indicating the weighting logic is working correctly. Once again, the words and key phrases provide an accurate summary of the topics that caused the decline.

Overall, no piece of software provides users the ability to have sentiment information, indication of the topics driving that sentiment and the ability to explore each in detail in one place. In other words, no application is able to tell the narrative towards a given asset like the NaratAI platform is.

This project was built by someone with a true passion for the domain so eliciting requirements that would be appreciated by users was not hard. Despite that, numerous discussions were held with people from the target audience and their feedback was considered for this project. The software was developed in a manner that would allow everyone to use it intuitively and derive value from its functionality.

10 Limitations and Future Work

Despite the perceived success of the project, it did not come without certain compromises. Firstly, a major limitation of the project is the lack of sentiment classification datasets consisting of news articles' bodies. Even if they existed, they would likely not be useful for training because of computational cost constraints but they would have been very beneficial in evaluating the model coupled with the logic. The small size of the dataset built for this project renders it insufficient for deriving highly reliable conclusions about performance. Furthermore, future work could consider named-entity-recognition (NER) when deriving sentiment from a given article for a given asset. For example, it could happen that an article for Tesla might have one paragraph that is highly negative towards it 2 or more that are highly positive towards its competitors. In such cases, the article would likely be evaluated as overall positive for Tesla which would be a misrepresentation that could be addressed using NER. Notebook 'Selecting Model for Production' discussed one such example. Moreover, even though a weighting scheme was specifically designed for the sentiment score calculation, there are times when it might over or underrepresent the expressed positivity or negativity in the articles for a given asset.

Next, when extracting key phrases, despite the logic used to reduce noise, there are still phrases that do not convey much semantic information that end up being scored as key phrases. As for the polarity words functionality, it could have been enhanced even more by adding extra heuristic rules for contextual features like intensifying words.

Lastly, in the user interface part, extra features could improve the user experience and perceived value. For example, providing the opportunity of more detailed hard data comparisons using some of the financials of the companies would be appreciated by some users. Being able to present the price and sentiment score in the same chart would also be an effective way of exploring their relationship rather than just showing the price and percentage change as the software does now. Allowing users the ability to create profiles could also be a useful addition. For example, they can receive notifications for significant changes in sentiment of assets of their interest, which could help with identifying investment opportunities or avoiding losses. That in terms would necessitate a larger database and more frequent updates. This project will be further developed in the future to hopefully address some of the laid-out limitations.

11 Project management

The overall approach to managing the development of the software followed a waterfall model coupled with evolutionary development. In other words, a clearly delineated structure of tasks that needed to be done was followed. Throughout that process the specification of the project was allowed to evolve.

Requirements analysis and definition

At this stage consideration was given to what the system should be able to do and what similar solutions already exists. Well-defined boundaries on what was achievable given the time and resource constraints were placed. Gaining a deep perspective on what functionality could be implemented was done by doing an elaborate review of the current state of research in the field of natural language understanding. Having done that, a requirement specification was elicited and functional and non-functional requirements were put in place. These were evolved throughout the development based on new understanding obtained from research or any feedback received.

System and software design

During the duration of this stage an architecture of how the elicited specification could be transformed into a viable software solution was developed. The logic for each layer of the application and the integration of all layers into a cohesive system was laid out. The interface that will be used to serve the solution to users was also thought through and basic prototyping was done.

Implementation and testing

The deliverables from the system and software design phase were used as a roadmap for this stage. First, a significant part of this stage included the data gathering and fine-tuning of the language model used. When the selected model was robustly tested, the project proceeded to implementing all functionality that would be provided by the software. Numerous unit tests were carried out on the individual components and integration tests were ran to ensure the correct integration between separate modules.

Once the system was developed system tests were ran multiple times to ensure its reliability. Furthermore, acceptance testing was done with potential users from the target audience of the software.

Evaluating the process

Overall, following above-depicted process ensured that the project was delivered in a timely manner despite the large volume of work required. A key takeaway is to avoid any assumptions about how long a given stage will take as that could be problematic. For example, the implementation of the BERT model took a very significant portion of time due to the downtime during which the model was being trained. Had this been predicted, the sunk cost of waiting could have been minimized by using that time to concurrently work on other functionality.

The evolution of requirements also proved an indispensable part of what was eventually achieved. During the meetings held with the project supervisor and the first project demonstration invaluable feedback was provided which helped guide the scope of this project. Ultimately, that resulted in a clear specification that allowed for a well-defined structural and temporal implementation.

Project plan

#	Activity	Start	End	Week 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Background research	0	4															
2	Literature review	1	5															
3	Requirements specification	4	5															
4	Software Architecture overview	5	7															
5	Database design	5	7															
6	Data gathering and pre-processing	7	9															
7	Model fine-tuning and testing	8	10															
8	Implementing of core back-end functionality	10	12															
9	Implementing front end functionality	11	12															
10	Report delivery	12	14															
11	Testing	6	14															
12	Reiterating requirement specification	3	14															

Figure 11: Project Plan.

12 Conclusion

The goal of this project was to deliver a software solution investors can use to assist their investment decision process. The underlying principle behind every decision made was that it must be grounded in high-quality research and where novelty was introduced – it must be systematically justified. Enabled by a state-of-the-art language architecture, a different perspective on topics like sentiment analysis of long texts and key phrase extraction was adopted. The results obtained indicate that the logic used has merit and it serves its intended purpose well. The practical aspect of the solution allows users to quickly grasp the prevailing 'narrative' towards a given stock or crypto currency by seeing how the sentiment towards it fluctuates. Being exposed to the polarity words and key phrases extracted from the news for a given asset allows users to gain a fast understanding about what topics are driving the sentiment towards it. Ultimately, as research has proven, understanding such analytics is crucial for making informed investment decisions and NaratAI serves exactly that purpose. The reception the project has received from people in its intended target audience has been very good and its future development will continue.

13 References

Aue, A. & Gamon, M., 2005. Customizing Sentiment Classifiers to New Domains: a Case Study. s.l., Microsoft Research.

Aylien., (2021) NLP-enriched News Intelligence Platform. Available from: https://aylien.com/product/news-api [Accessed 20 July 2021]

Bird, S., Loper, E. & Klein , E., 2009. Natural Language Processing with Python. s.l.:O'Reilly Media Inc..

Boiy, E. & Moens, M.-F., 2008. A Machine Learning Approach to Sentiment Analysis in Multilingual Web Texts. Information Retrieval, Volume 12, pp. 526-558.

Bommasani, R., Davis, K. & Cardie , C., 2020. Interpreting pretrained contextualized representations via reductions to static embeddings.. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 4758-4781.

Bloomberg Professional Services., (2021) Embedded Value in Bloomberg News and Social Sentiment Data. Available from: https://www.bloomberg.com/professional/sentiment-analysis-white-papers/ [Accessed 20 July 2021]

Boudoukh, J., Feldman, R., Kogan, S. & Richardson, M., 2012. Which News Move Stock Prices? A Textual Analysis, Cambridge, Massachusetts: National Bureau of Economic Research.

CryptoMood., (2021) Sentiment Analysis Tools for Superior Trading Decisions. Available from: https://cryptomood.com/ [Accessed 20 July 2021]

CryptoNews API., (2021) Available from: https://cryptonews-api.com/ [Accessed 1 August]

Das, S. & Chen, M., 2007. Yahoo! for Amazon: Opinion Extraction from Small Talk on the Web. Management Science.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Google AI Language.

Ethayarajh, K., 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. s.l.: arXiv preprint arXiv:1909.00512.

FinViz., (2021) Financial Visualizations Available from: https://finviz.com/ [Accessed 1 August]

Garcia, D., 2013. Sentiment During Recessions. The Journal of Finance, 68(3), pp. 1267-1300.

Glorot, X., Border, A. & Bengio, Y., 2011. Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. s.l., International Conference on Machine Learning.

Hearst, M., 1997. Text Tiling: Segmenting text into multi-paragraph subtopic passages.. Computational Linguistics, 23(1), pp. 33-64.

Heston, S. L. & Sinha, N. R., 2016. News versus Sentiment: Predicting Stock Returns from News Stories. Finance and Economics Discussion Series, Volume 048.

Hochreiter, S., 1998. The vanishing gradient problem during learning recurrent nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(2), pp. 107-116.

Hugging Face., (2021) BERT. Available from: https://huggingface.co/transformers/model_doc/bert.html [Accessed 10th July 2021]

Hugging Face., (2021) BertForSequenceClassification. Available from: https://https://huggingface.co/transformers/model_doc/bert.html#bertforsequenceclassification [Accessed 10th July 2021]

Hugging Face., (2021) Tokenizer. Available from: https://huggingface.co/transformers/main_classes/tokenizer.html [Accessed 11th July 2021]

Hulth, A., 2003. Improved automatic keyword extraction given more linguistic knowledge. Proceedings of the 2003 conference on Empirical methods in natural language processing, pp. 216-223.

Hu, M. & Liu, B., 2004. Mining and Summarizing Customer Reviews. s.l., Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.

Hutto, C. & Gilbert, E., 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Texts. s.l., Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media.

Kraus, M. & Feuerriegel, S., 2017. Decision support from financial disclosures with deep neural networks and transfer learning. Decision Support Systems.

Lin, Y., Lei, H. & Deng, Y., 2021. Research on Text Classification Modelling Strategy Based on Pre-trained Language Model. Proceedings of SAI Intelligent System Conference, Volume 294, pp. 248-260.

Li, X. et al., 2014. News impact on stock price return via sentiment analysis. Knowledge-Based Systems, Volume 69, pp. 14-23.

Loughran, T. & McDonald, B., 2011. When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. The Journal of Finance, 66(1), pp. 35-65.

Maas, A., Ng, A. & Potts, C., 2011. Learning word vectors for sentiment analysis. Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies, pp. 142-150.

Malo, P., Sinha, A. & Takala, P., 2014. Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts.. Journal of the American Society for Information Sciences and Technology.

Matsuo, Y. & Ishizuka, m., 2004. Keyword Extraction from a Single Document Using Word Co-occurrence Statistical Information. International Journal on Artificial Intelligence Tools, 13(1), pp. 157-169.

McKinney, 2010. Data structures for statistical computing in python. Proceedings of the 9th Python in Science Conference,, Volume 445.

Meng, R., Zhao, S. & Han, S., 2017. Deep Keyphrase Generation. arXiv preprint arXiv:1704.06879..

Mikolov, T., 2012. Statistical Language Models Based on Neural Networks, Mountain View: Presentation at Google.

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2014. Efficient Estimation of Word Representations in Vector Space. s.l., s.n.

Miller, G. et al., 1990. Introduction to WordNet: An on-line lexical database. International Journal of Lexicography, 3(4), pp. 235-244.

Norman, D., 1990. The Design of Everyday Things. New York: Doubleday.

Pang, B., Lee, L. & Vaithyanatha, S., 2002. Thumbs up?: sentiment classification using machine learning techniques. s.l., Proceedings of the ACL-02 conference on Empirical methods in natural language processing.

Pappagari, R., Zelasko, P. & Villalba , J., 2019. Hierarchical transformers for long document classification.. IEEE Automatic Speech Recognition and Understanding Workshop, pp. 838-844.

Pedregosa, F., Varoquaux, G. & Gramfort, A., 2011. Scikit-learn: Machine Learning in Python. JMLR, 12(85), p. 2825−2830.

Pennington, J., Socher, R. & Manning, C., 2014. GloVe: Global Vectors for Word Representation. s.l., ACL.

Peters, M., Neuman, M., Mohit, I. & Gardner, M., 2018. Deep Contextualized Word Representations. arXiv.

Potts, C., 2010. On the Negativity of Negation. Semantics and Linguistic Theory .

Pytorch., (2021) Writing Custom Datasets, Dataloaders and Transforms. Available from: https://pytorch.org/tutorials/beginner/data_loading_tutorial.html [Accessed 15th July 2021]

Ramos, J., 2003. Using tf-idf to determine word relevance in document queries. Proceedings of the first instructional conference on machine learning, 242(1), pp. 29-48.

RavenPack., (2021) News Analytics. Available from: https://www.ravenpack.com/products?news-analytics-tab [Accessed 20 July 2021]

Robertson, S. & Jones, K., 1976. Relevance weighting of search terms. Journal of the American Society for Information Science, 27(3).

Ruder, S., Ghaffari, P. & Breslin, J., 2016. A Hierarchical Model of Reviews for Aspect-based Sentiment Analysis. arXiv preprint arXiv:1609.02745.

Schiller , R., 2019. How stories go viral and drive major economic events. s.l.:Princeton University Press.

Shapiro, A., Sudhof, M. & Wilson, D., 2020. Measuring News Sentiment. Journal of Econometrics.

Sharma, P. & Li, Y., 2019. Self-Supervised Contextual Keyword and Keyphrase Retrieval with Self-Labelling.

Socher, R., Pennington, J., Ng, A. & Manning, C., 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. s.l., EMNLP.

Socher, R. et al., 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. s.l., Proceedings of the 2013 Conference of Emperical Methods in Natural Language Processing .

Souma, W., Vodenska, I. & Aoyama, H., 2019. Enhanced News Sentiment Analysis Using Deep Learning Methods. Journal of Computational Social Science, Volume 2, pp. 33-46.

Sousa, M., Sakiyama, K., Rodrigues, L. & de Moraes, P., 2019. BERT for Stock Market Sentiment Analysis. s.l., IEEE.

StockSnips., (2021) Financial Sentiment Available from: https://stocksnips.net/financial-sentiment/analytics-platform/ [Accessed 20 July 2021]

Taboada, M. et al., 2011. Lexicon-Based Methods for Sentiment Analysis. Computational Linguistics, 37(2).

Tai, K. S., Socher, R. & Manning, C., 2015. Improved Semantic Representations From Tree-Structured Long-Short Term Memory Networks. ACL.

Tetlock, P., 2007. Giving Content to Investor Sentiment: The Role of Media in the Stock Market. The Journal of Finance, Volume 62, pp. 1139-1168.

Vaswani, A., Shazeer, N., Parmar, N. & Uszkoreit, J., 2017. Attention Is All You Need. Advances in neural information processing systems, pp. 5998-6008.

Wartena, C., Brussee, R. & Slakhorst, W., 2010. Keyword extraction using word co-occurrence. Workshops on Database and Expert Systems Applications IEEE, pp. 54-58.

Yang, Z., Dai, Z., Yang, Y. & Carbonell, J., 2019. Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, Volume 32.

14 Appendix

14.1 Functional and Non-functional Requirements

Requirements importance is tagged on a ternary scale of Low (L), Medium (M) and High (H).

Functional

1. The system must support functionality to acquire news articles for all supported stocks and crypto currencies. -H
2. The system must support functionality to analyse the sentiment of the collected news articles. -H
3. The system must support logic for generating an overall scalar sentiment score for each of the supported stocks and cryptocurrencies. -H
4. The system must allow for the extraction of sentiment-carrying words from the gathered news articles for each stock or crypto currency. -H
    a. The system must allow for the separation of sentiment-carrying words in two classes- positive and negative. -H
5. The system must allow for the extraction of key phrases for each of the selected stocks or cryptocurrencies. -H
    a. The system must allow for the separation of key phrases in two classes- positive and negative. -H
6. The system must provide functionality allowing users to search and select stocks and cryptocurrencies from a page displaying the supported ones. – H
    a. The system must provide the user with the ability to select up to five stocks or crypto currencies from a grid panel to be added to a favourites bar for quick access. -H
    b. The system must allow the user to search the grid panel based on a ticker identifier or company/crypto currency name. -H
    c. Each tile on the grid must contain the ticker identifier, name and logo of the given stock/crypto currency. -H
    d. Each tile must be conditionally coloured based on the news articles gathered and evaluated for it on the current day. -M
7. The system must provide a dashboard interface where the user can toggle between the selected favourites from the grid. -H
    a. The tiles displaying the selected favourites must include their sentiment score.
8. The system must display in a spotlight panel one of the favourites the user has selected. The features described in points (8-11) should contain the data for the favourite currently displayed in the spotlight panel. -H
    a. The spotlight panel must contain the following information for the currently selected favourite: -H
        i. The name and logo. -H
        ii. The current price and percentage change in price, where the percentage change implies a timeframe between the previous day’s closing price and the current price. The price data must be acquired in real time. -H
        iii. The number of articles evaluated split between the categories of neutral, negative and positive. -H

9. The dashboard must include a chart representation showing the historical values of the sentiment score for the favourite currently displayed in the spotlight. -H
    a. The chart should include time series data for at least 10 days including the current day. -H
        i. The chart should display the score for the given day when the user hovers over it. -M
10. The dashboard should include a headlines panel where all articles used in the calculation of the sentiment score are listed. -H
    a. The headline panel should include the article headlines as links to their sources. -H
    b. The headline texts should be conditionally coloured based on their sentiment. -M
    c. The user should have an option to toggle between seeing all, just the positive or just the negative articles and they should be dynamically displayed based on the selection. -M
    d. The headline panel should only contain sentiment carrying articles (articles evaluated as positive or negative). -M
        i. In cases where there are less than 5 sentiment-carrying articles, articles evaluated as neutral can also be displayed. -M
11. The dashboard should include a word cloud interface for rendering polarity words or key phrases for the currently selected favourite in the spotlight. -H
    a. The word cloud should display the polarity words from the articles present in the headlines panel. -H
        i. The words should be conditionally coloured based on their sentiment. -H
        ii. The size of the words in the word cloud should be determined by the frequency of occurrence of the given word in the news articles. -M
        iii. The source from which a given polarity word was extracted should be displayed upon hovering over it. -H
    b. The word cloud should display the key phrases extracted from the articles present in the headlines panel. -H
        i. The key phrases should be conditionally coloured based on their sentiment. -H
        ii. The source from which a given key phrase was extracted should be displayed upon hovering over it. -H
    c. The word cloud should allow for toggling between polarity words and key phrases with content dynamically changing based on the selection. -H
    d. The word cloud should dynamically render all, just positive or just negative polarity words or key phrase based on the selection the user has made in the headlines panel. -H
12. The system must support a relational database to store and retrieve data. -H
    a. The database must be updated with all data required by requirements 7-10 at least three times a day. -H
13. The system interface should be responsive to all screen sizes. – M
14. The system should maintain memory of previous state. – M
    a. The system should render different content based on whether the user is visiting for the first time. -M

        i. The user will be redirected to the dashboard if she has previously visited -M
        ii. The user will be redirected to the selection page if she is visiting for the first time and message prompt to choose favourites will be displayed. -M
    b. The system should maintain its state if a user refreshes the page or toggles between pages. -M
15. The system must be accessible through all major web browsers including Chrome, Safari, Firefox and IE. -H
16. The code in the user interface part must be written in a manner that supports screen readers and other accessibility software. – M

Non-functional

Performance

1. The application should respond to all user prompts within 2 second to ensure a consistent flow during the user interaction experience. -H
    a. The chart and word cloud components where content is dynamically rendered must not remain unresponsive for more than 1 second. -M
    b. The data retrieved from the system database or external APIs (stock and crypto currency prices) should not take more than 2 seconds to be rendered. -H

Scalability and Reliability

1. The system's performance must not downgrade based on the number of users currently interacting with it. -M
2. The system must support at least 50,000 concurrent users -L
3. The system should support at least 20 stocks and crypto currencies and have the ability to scale to a larger number. -H
4. The system database should not allow for any duplicate entries where a duplicate entry is defined as an article that has the same URL as an existing entry in the database.
5. **The system database must adhere to the robustness and consistency principles.**

Privacy and Regulatory Safety

1. The system must not store any information about its users. -H
2. The system must not gather any news articles containing private information about individuals. -H
3. The system must only gather news articles from websites that do not explicitly prohibit web scraping. -H
4. The system must adhere to legislative regulations concerning the sharing of articles from different sources. -H
    a. The system must always use a given article only if it is linked to its original source. -H

Implementation

1. The system should be implemented using the Python, JavaScript, HTML and CSS programming languages and any libraries and packages they support. -M

2. The system should use a relational database for data storage. -M

14.2 Testing

Unit and Integration Tests

Script/s tested	Description (more detailed in the Test class)	Layer/s tested	Pass/Fail
sentiment_prediction keyphrases	System splits texts into paragraphs according to specification.	Business logic	Pass
sentiment_prediction	System produces model output probabilities that are statistically meaningful and follow the logic implemented.	Business logic	Pass
keyphrases	System applied the logic used for pooling token embeddings correctly.	Business logic	Pass
sentiment_prediction keyphrases	System is able to split texts when they have unorthodox punctuation or recursive paragraph splitting is required.	Business logic	Pass
sentiment_prediction db_updates	System database entries' overall sentiment corresponds to the highest-class probability.	Business logic Persistance Database	Pass
polaritywords db_updates	System database entries' polarity words have a corresponding sentiment.	Business logic Persistance Database	Pass
sentiment_prediction keyphrases	System does not lose the entirety of texts after splitting into paragraphs.	Business logic	Pass
polarity_words	System applies the heuristic negation rule correctly.	Business logic	Pass
db_updates	System restricts warning messages generated from websites that block web scraping entering the database.	Business logic Persistance Database	Pass
db_updates	System logic prevents the insertion of duplicate entries into the database.	Business logic Persistance Database	Pass
db_updates	System database prevents the violation of UNIQUE and PRIMARY KEY constraints that could result in duplicate insertions.	Business logic Persistance Database	Pass

news_scraper	System prevents the insertion of articles with irrelevant context for the given asset.	Business logic Persistance Database	Pass
db_updates sentiment_prediction keyphrases polarity_words	System layers integrate correctly to generate viable entries in all tables	Business logic Persistance Database	Pass

Functional and Non-functional Tests

Requirement tested. Non-functional (NF)/ Functional(F)	**Test description**	**Test steps**	**Expected result**	**Pass/Fail**
F 6.a, 6.b, 6.c, 6. d	Verify the selection grid functionality.	1. Check whether all supported assets are displayed. 2. Check whether styles are correctly applied to assets with more. evaluated articles. 3. Check that the user is able to select up to five favourites.	1. All supported assets are to be correctly displayed with their respective identifiers. 2. Assets with more evaluated articles must have a darker coloured tile. 3. The user should be able to select up to and no more than five favourites.	**Pass**
F 7.a	Verify that upon confirming favourites, the user is redirected to the dashboard page.	1. Select 5 favourites from the selection grid. 2. Click confirm favourites	1. Upon clicking the confirm favourites button, the user is redirected to the dashboard page. 2. The assets available for quick access are the ones he has selected as favourites and not others.	**Pass**
F 8.a	Verify that the asset currently placed in the spotlight has the	1. Select 5 favourites from the selection grid. 2. Toggle between all of them and check the	1. The asset's name and logo are correct. 2. The asset's current price and % change are	**Pass**

	correct data displayed for it.	correctness of data displayed.	correct. 3. The evaluated articles statistics matche the numbers in the database.	
F 9.	Verify that the sentiment score chart is responsive and displays the correct time series data.	1. Select 5 favourites from the selection gird. 2. Toggle between all of them and check whether the chart loads the correct sentiment score data for the favourite currently placed in the spotlight.	1. The chart is responsive and loads the correct data for each asset placed in the spotlight panel.	**Pass**
F 10.a,10. b	Verify that the headlines panel lists the positive and negative articles evaluated.	1. Select 5 favourites from the selection grid. 2. Toggle between all and check whether the correct headlines are displayed in the headlines panel. 3. Check whether styles are correctly applied depending on sentiment. 3. Check whether the user is redirected to the article source upon clicking the headline.	1. Upon switching between favourites the correct article headlines are rendered. 2. Positive articles are coloured green, negative ones red and neutral ones grey. 3. The browser opens the article source in a separate tab upon clicking on the headline.	**Pass**
F 10.d	Verify that only sentiment-carrying articles (positive or negative) are displayed in the headlines panel. Verify that in cases where there are less than 5 sentiment-carrying articles for a given asset, neutral articles are also rendered.	1. Choose 5 favourites of which one or two have less than 5 sentiment-carrying articles evaluated. 2. For the ones having more than 5 sentiment-carrying articles, check whether only they are displayed. 3. For the ones having less than 5 sentiment-carrying articles, check whether neutral articles are also rendered.	1. Articles having more than 5 sentiment-carrying articles do not have any of the neutral articles evaluated for them displayed in the headlines panel. 2. Articles having less than 5 sentiment-carrying articles also have neutral articles rendered in the headlines panel.	**Pass**

F 11.a, 11. b	Verify that the word cloud renders the correct polarity words or key phrases for the asset currently placed in the spotlight.	1. Select 5 favourites from the selection grid. 2. Toggle between all and check whether the word cloud renders the correct content for the favourite currently placed in the spotlight. 2. Check whether conditional styles based on sentiment are correctly applied. 3. Check whether polarity word/ key phrase source is displayed when hovered over.	1. The polarity words/ key phrases extracted from the articles in the headlines panel are the correct ones for the asset currently placed in the spotlight. 2. Positive words/phrases are coloured green and negative ones red. 3. The news article headline is displayed upon hovering over the word/phrase in the word cloud.	**Pass**
F 11.d	Verify that upon toggling between all, just positive or just negative headlines, only polarity words and key phrases corresponding to the given category are displayed.	1. Toggle between the *see all, positive and negative* options in the headlines panel and check whether the words/key phrases only for the corresponding class are rendered.	1. If the *see all* option is selected both positive and negative polarity words/ key phrases are rendered. 2. If negative or positive is the option selected only polarity words/ key phrases for the corresponding class are rendered.	**Pass**
F 11.c	Verify that the upon toggling between options *polarity words* and *key phrases,* the word cloud renders the correct content.	1. Toggle between the *polarity words* and *key phrases* options displayed above the word cloud and check whether the respective content for each class is rendered.	1. If *polarity words* is the option selected, only polarity words are rendered in the word cloud. 2. If *key phrases* is the option selected, only key phrases are rendered in the word cloud.	**Pass**
F 14.	Verify that the system maintains memory of previous state.	1. Check whether the page displays a greeting and prompt to confirm favourites if the application is opened for the first time.	1. Upon entering for the first time a greeting message and prompt to confirm favourites is displayed. 2. After given favourites are confirmed the	**Pass**

		2. Select and confirm 5 favourites from the selection page. 2. Refresh the page and check whether they are still maintained and the dashboard rather than selection page is rendered. 3. Close the browser and check whether they are still maintained.	greeting message is no longer rendered and the dashboard page is directly rendered instead of the selection page. 3. Refreshing the page does not lead to a loss of confirmed favourites. 4. Closing the browser does not lead to a loss of confirmed favourites.	
NF 1.a	Verify that the system responds within 2 seconds to changes triggered by user events.	1. Check responsiveness when asset tiles are being loaded. 2. Select 5 favourites and toggle between them. 3. Check responsiveness of the chart when switching between assets. 4. Check responsiveness of the spotlight, headlines and word cloud panels when switching between favourites.	1. Rendering new content or loading new data should not take more than 2 seconds for any of the described operations.	**Pass**

14.3 User Interface Components

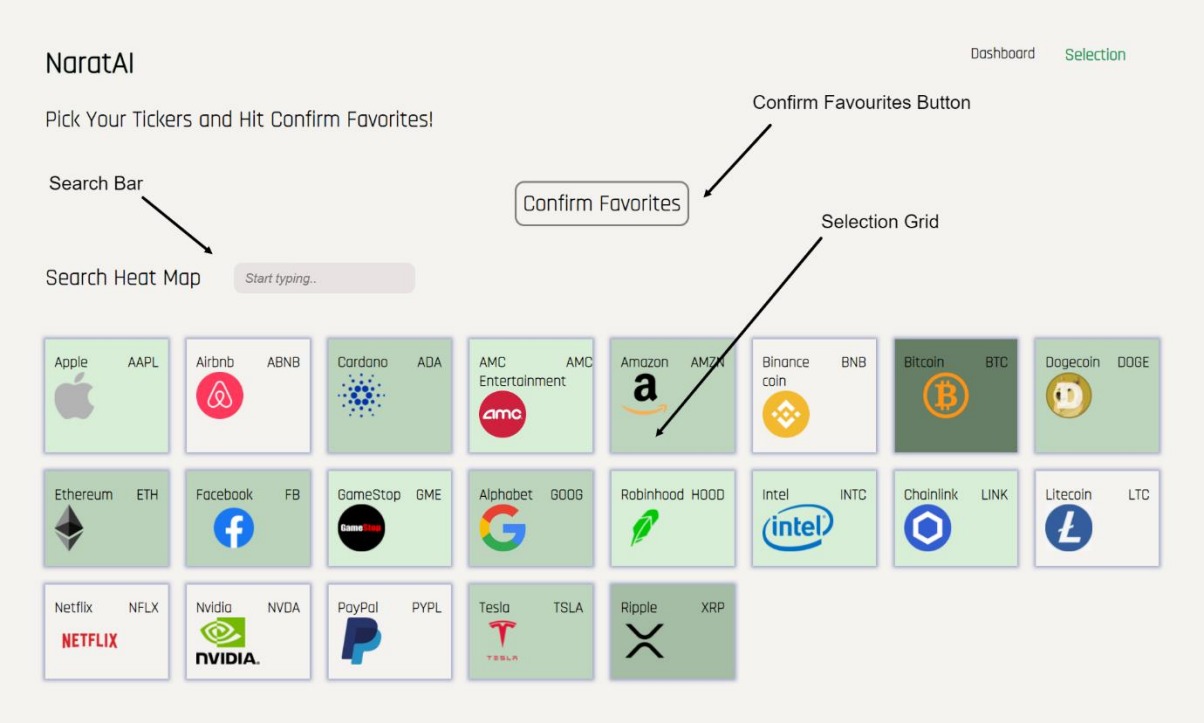


Image 1: Selection page

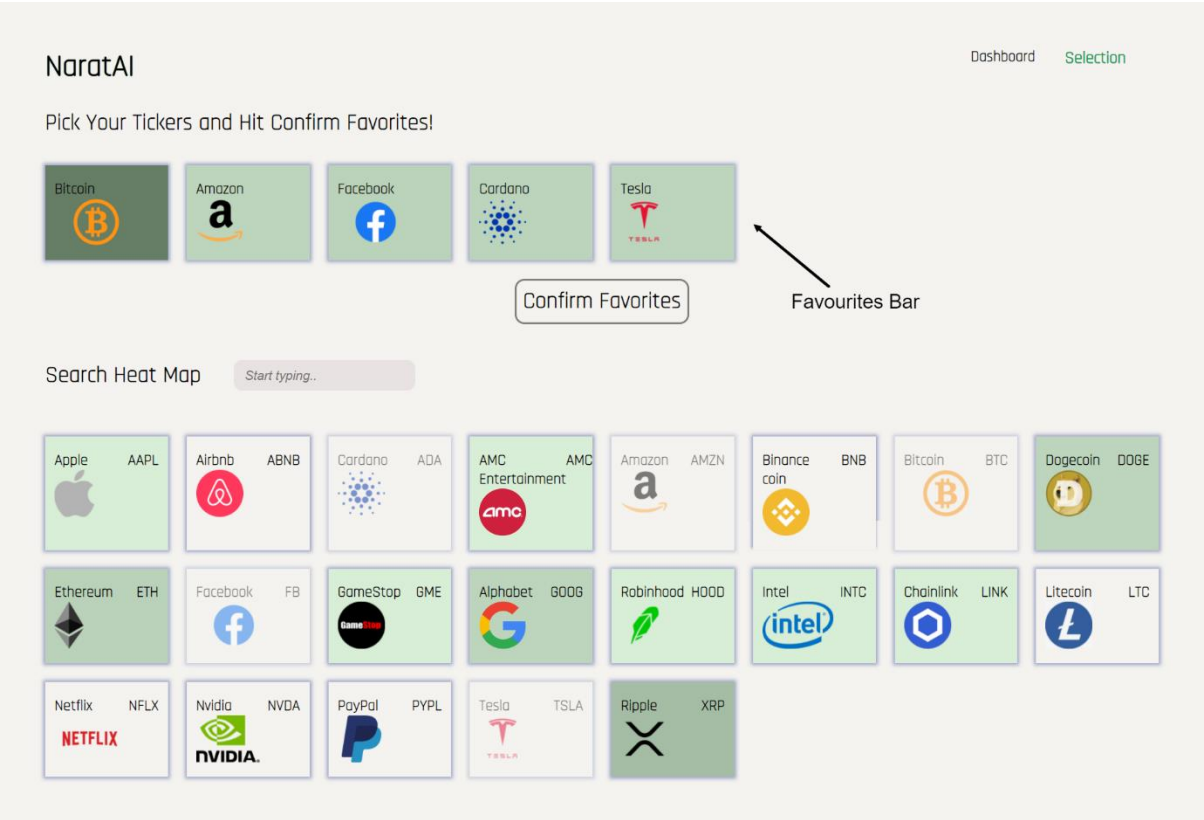


Image 2: Selection page

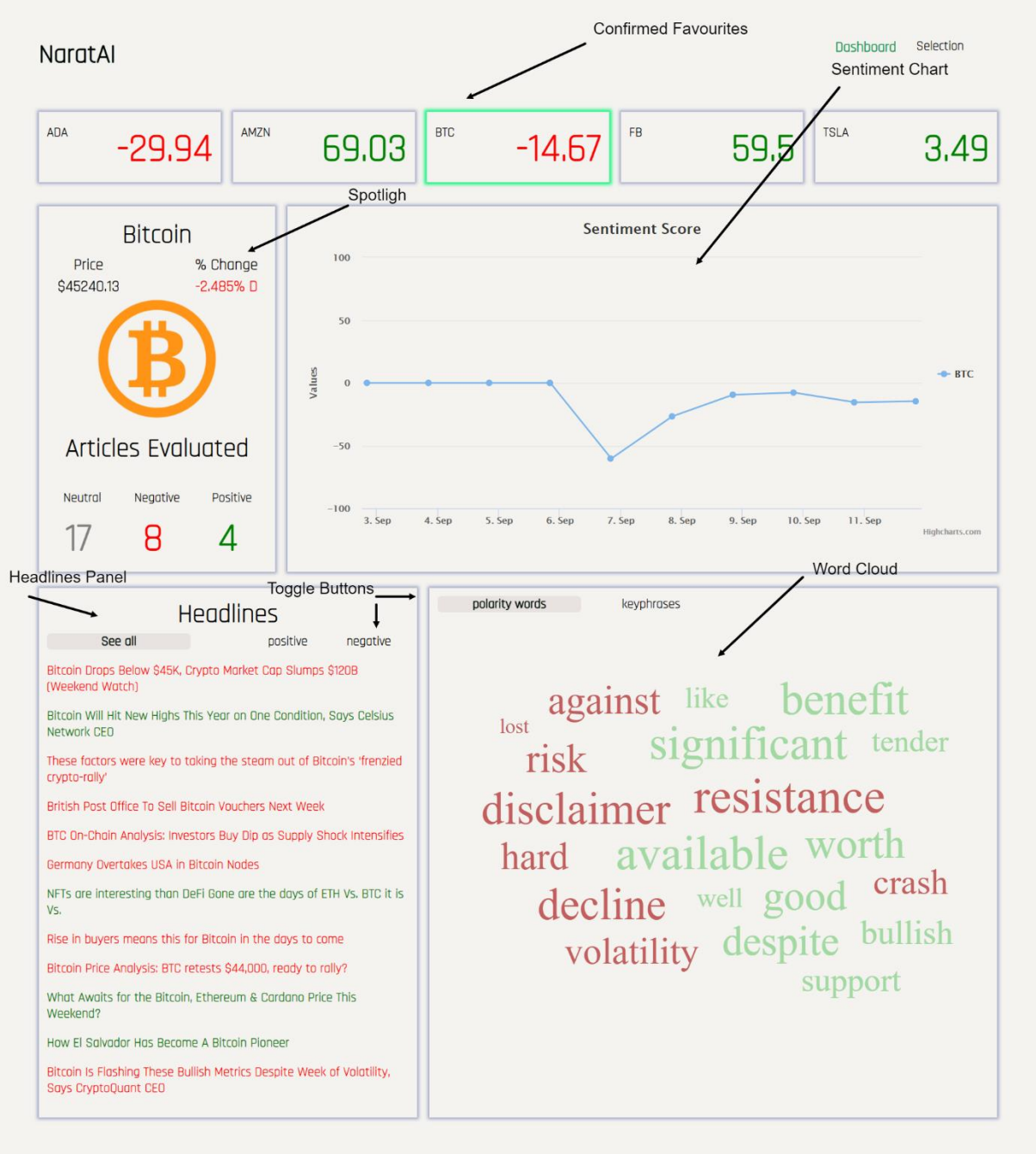
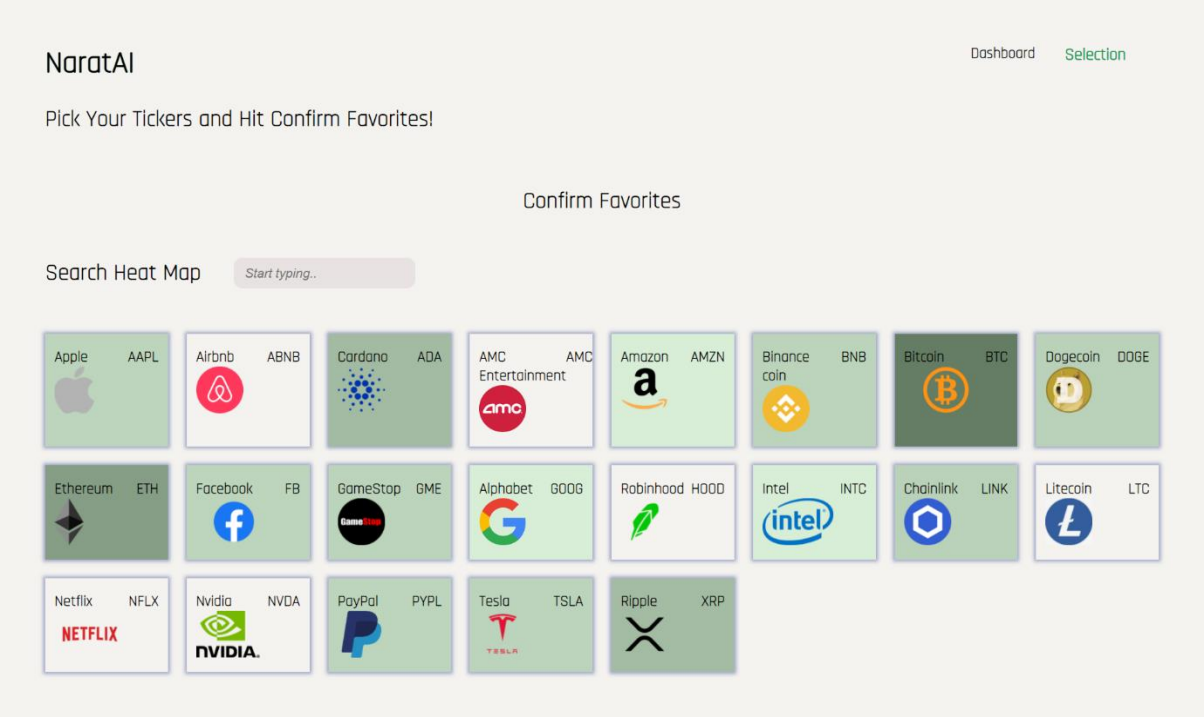
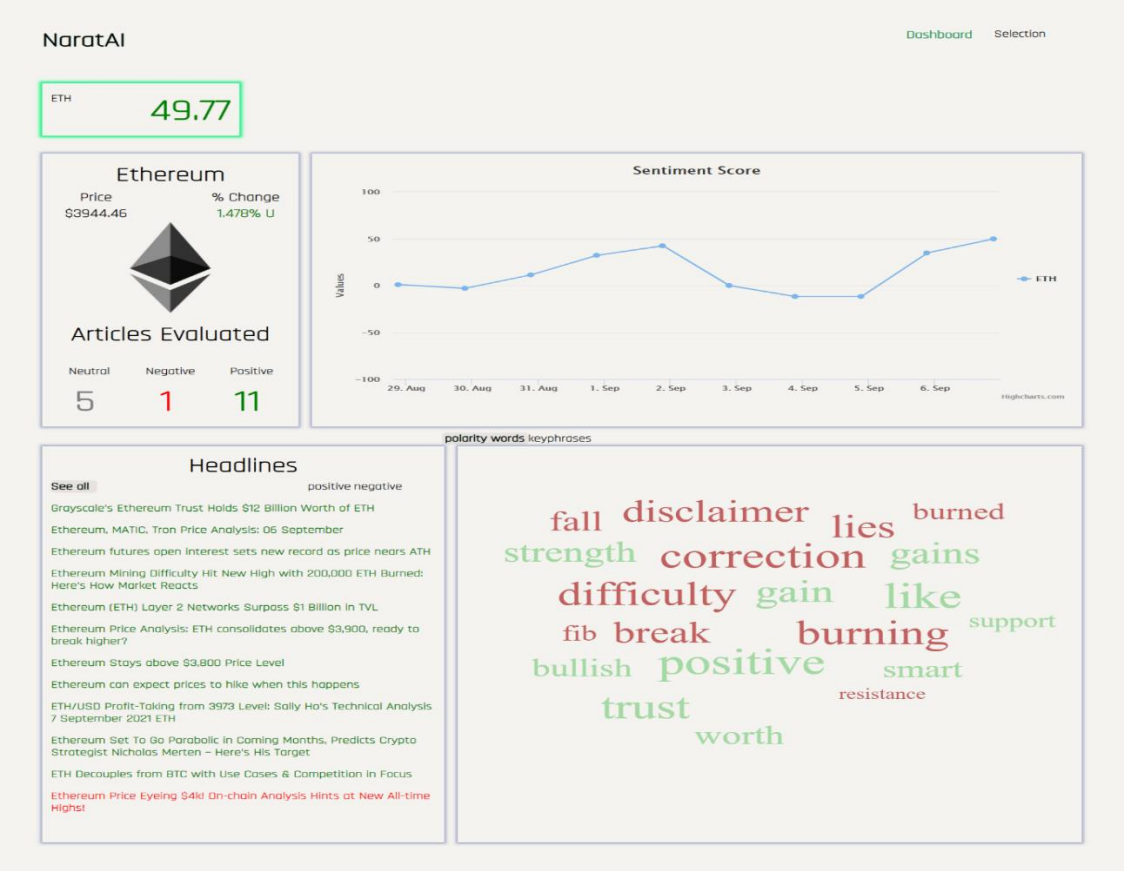


Image 3: Dashboard page

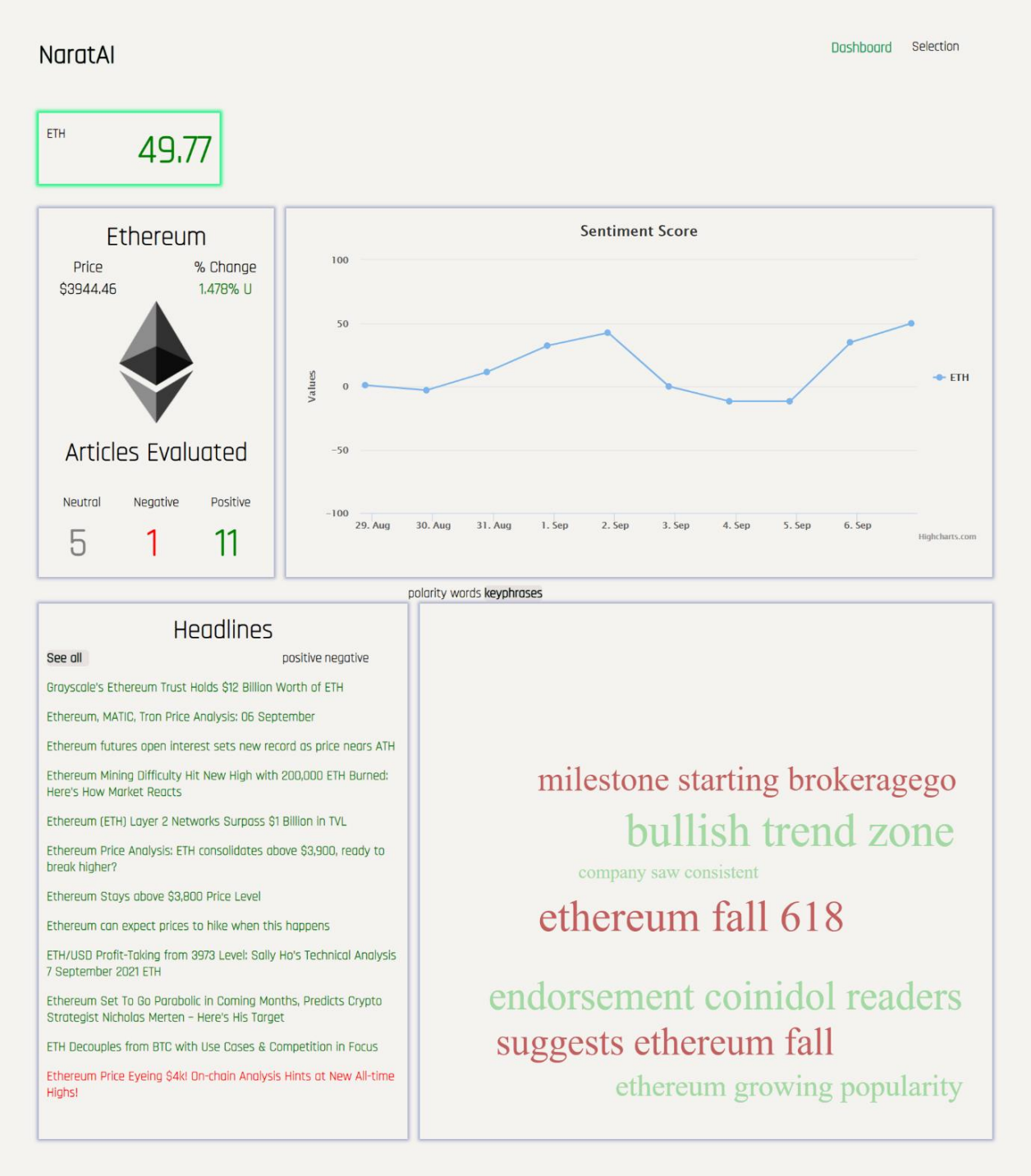
14.4 User Story Screens



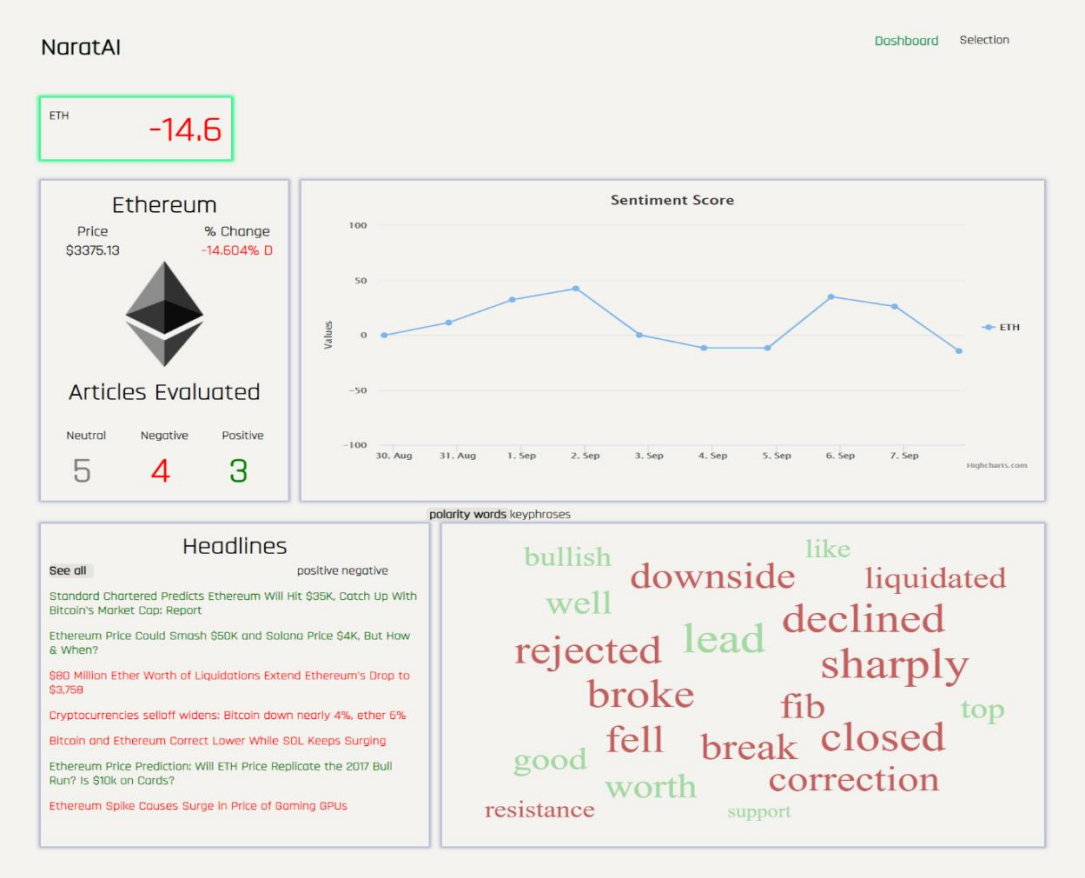
*Screen 1 *Note that small design changes have since been implemented*



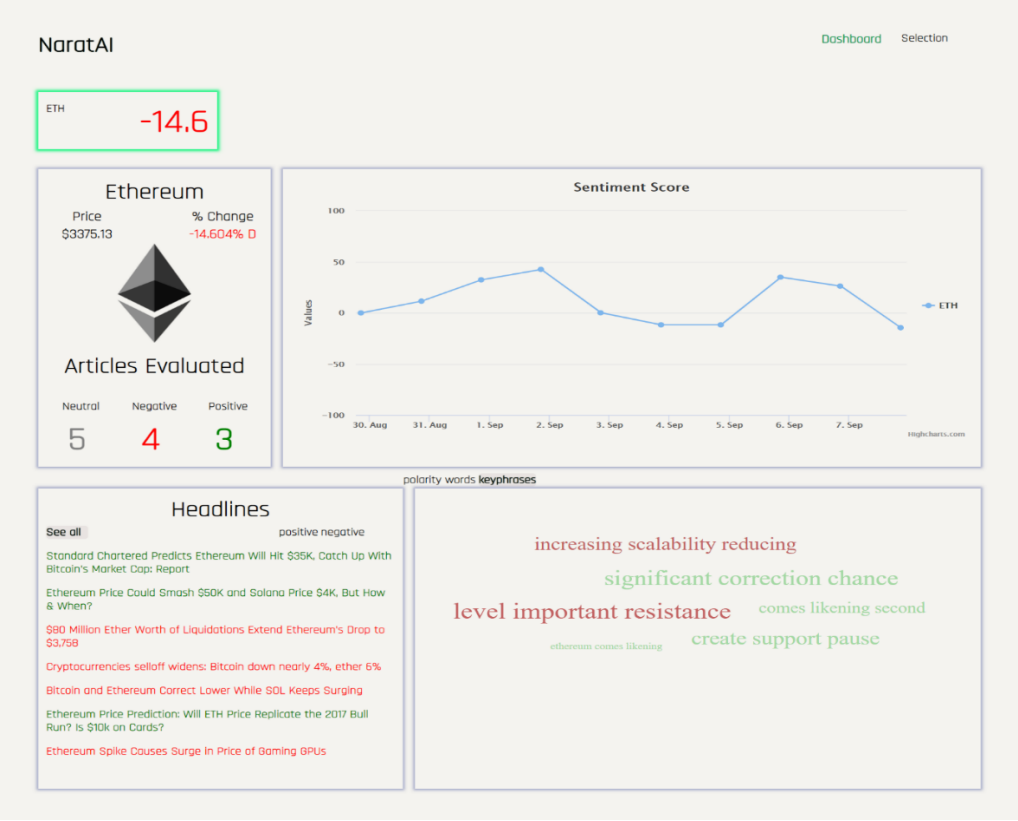
*Screen 2. *Note that small design changes have since been implemented.*



*Screen 3: *Note that small design changes have since been implemented*

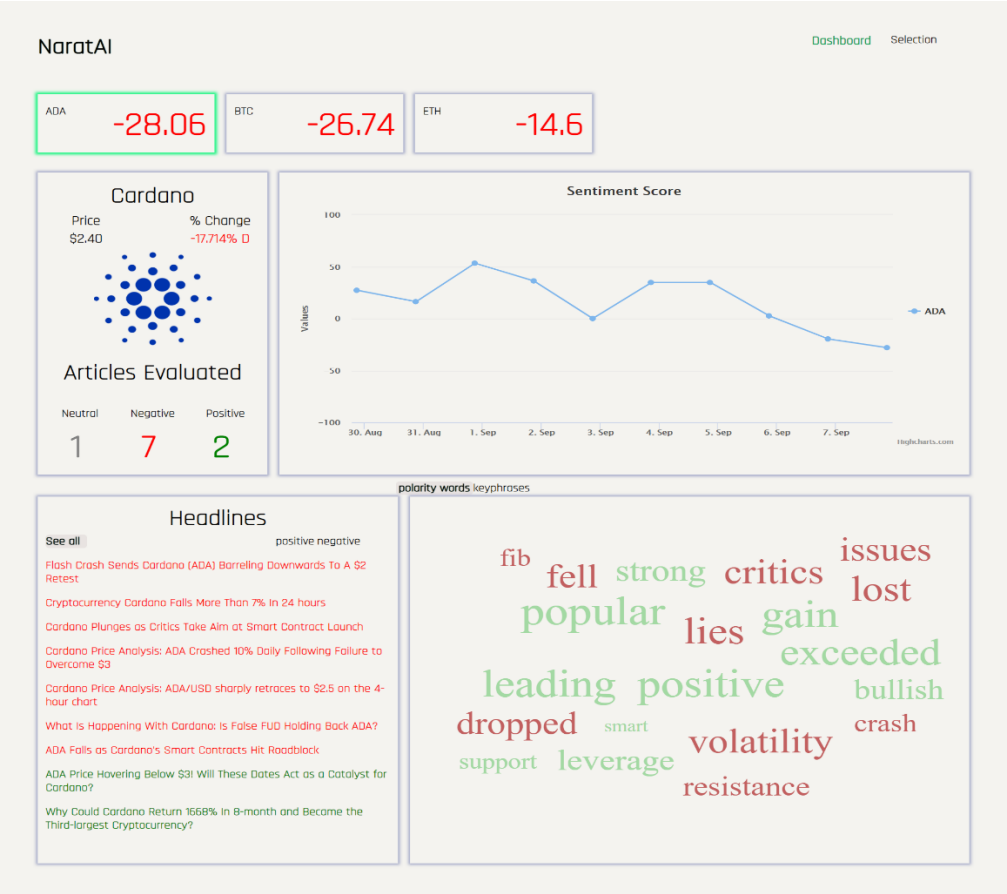


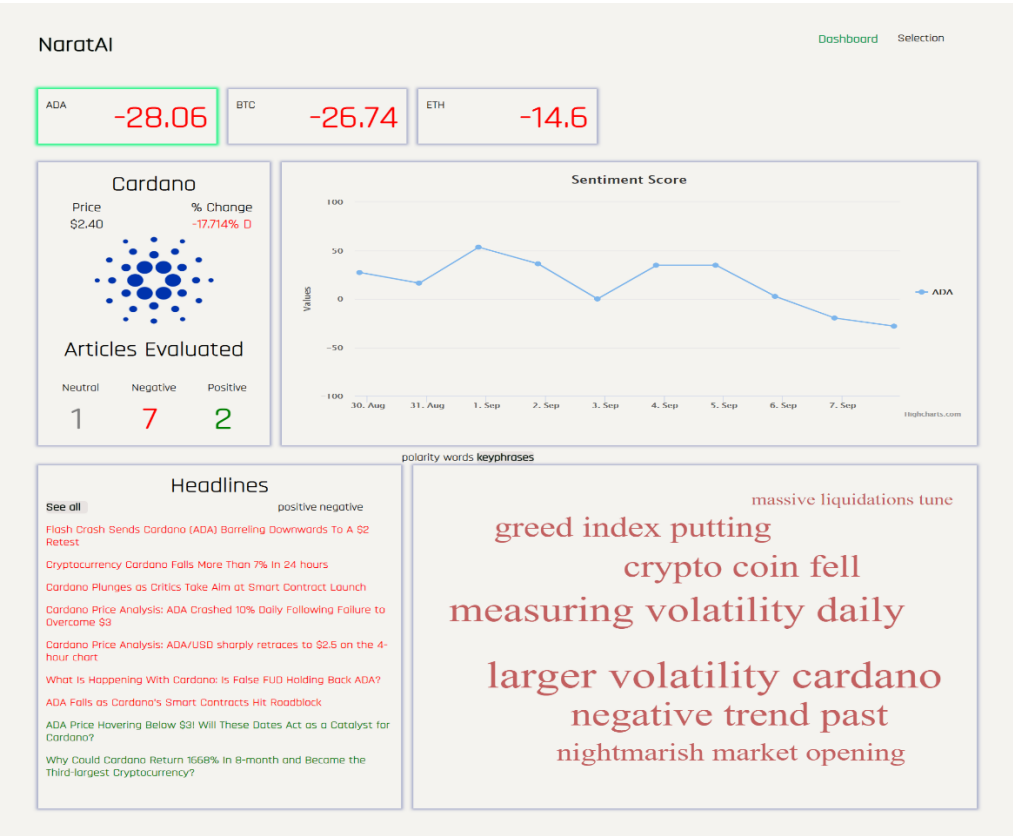
*Screen 4: *Note that certain design changes have since been implemented*

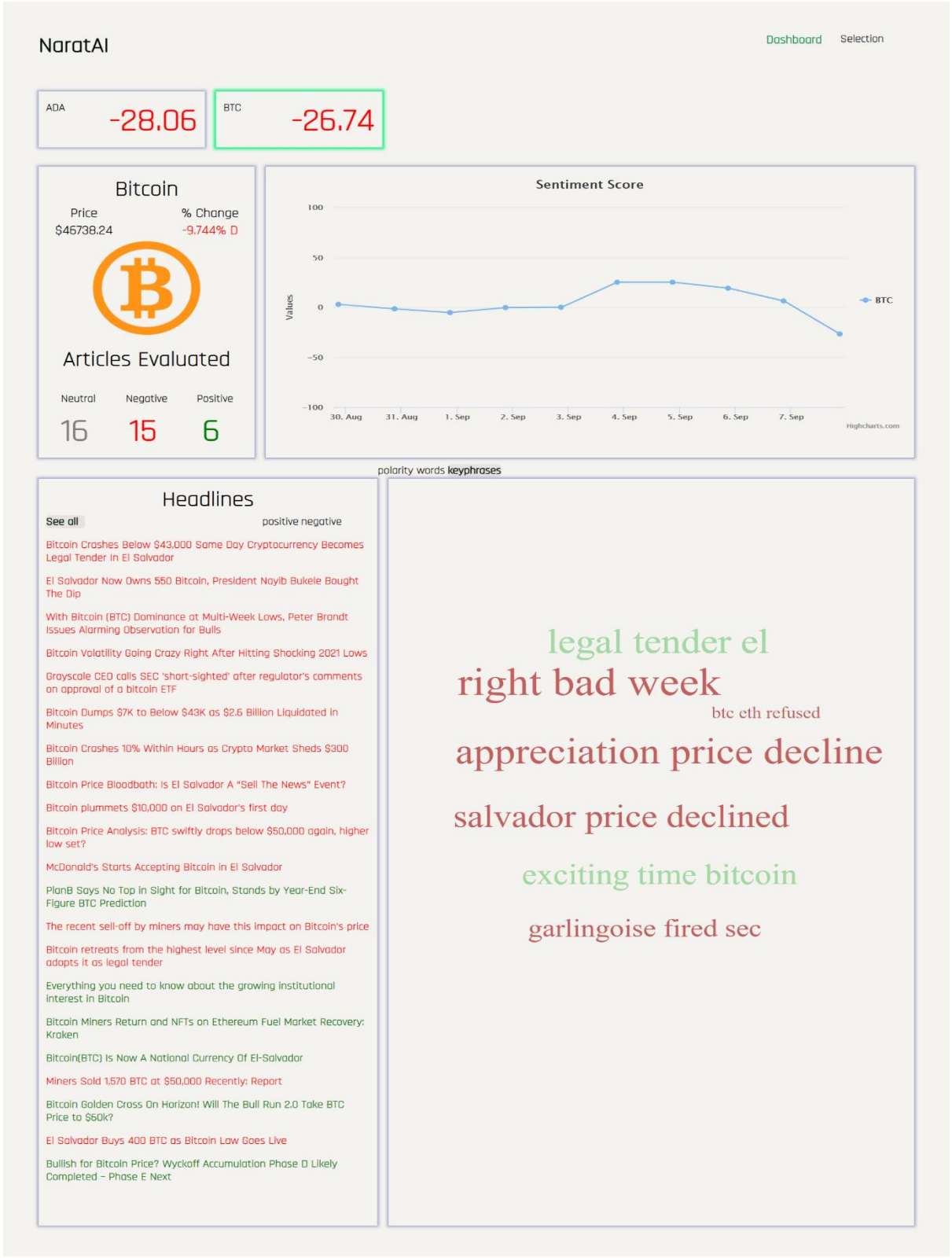


*Screen 5: *Note that certain design changes have since been implemented*

14.5 Software Usability Example (*Note that minor design changes have since been introduced)






NaratAI
Dashboard
Selection
ADA
-28.06
BTC
-26.74
Bitcoin
Price
$46738.24
% Change
-9.744% D
Articles Evaluated
Neutral
16
Negative
15
Positive
6
Sentiment Score
100
50
0
-50
-100
Values
30. Aug
31. Aug
1. Sep
2. Sep
3. Sep
4. Sep
5. Sep
6. Sep
7. Sep
BTC
Highcharts.com
polarity words keyphrases
Headlines
See all
positive negative
Bitcoin Crashes Below $43,000 Same Day Cryptocurrency Becomes Legal Tender in El Salvador
El Salvador Now Owns 550 Bitcoin, President Nayib Bukele Bought The Dip
With Bitcoin (BTC) Dominance at Multi-Week Lows, Peter Brandt Issues Alarming Observation for Bulls
Bitcoin Volatility Going Crazy Right After Hitting Shocking 2021 Lows
Grayscale CEO calls SEC 'short-sighted' after regulator's comments on approval of a bitcoin ETF
Bitcoin Dumps $7K to Below $43K as $2.6 Billion Liquidated in Minutes
Bitcoin Crashes 10% Within Hours as Crypto Market Sheds $300 Billion
Bitcoin Price Bloodbath: Is El Salvador A "Sell The News" Event?
Bitcoin plummets $10,000 on El Salvador's first day
Bitcoin Price Analysis: BTC swiftly drops below $50,000 again, higher low set?
McDonald's Starts Accepting Bitcoin in El Salvador
PlanB Says No Top in Sight for Bitcoin, Stands by Year-End Six-Figure BTC Prediction
The recent sell-off by miners may have this impact on Bitcoin's price
Bitcoin retreats from the highest level since May as El Salvador adopts it as legal tender
Everything you need to know about the growing institutional interest in Bitcoin
Bitcoin Miners Return and NFTs on Ethereum Fuel Market Recovery: Kraken
Bitcoin(BTC) Is Now A National Currency Of El-Salvador
Miners Sold 1,570 BTC at $50,000 Recently: Report
Bitcoin Golden Cross On Horizon! Will The Bull Run 2.0 Take BTC Price to $60k?
El Salvador Buys 400 BTC as Bitcoin Law Goes Live
Bullish for Bitcoin Price? Wyckoff Accumulation Phase D Likely Completed – Phase E Next
legal tender el
right bad week
btc eth refused
appreciation price decline
salvador price declined
exciting time bitcoin
garlingoise fired sec

14.6 Project Repository Structure

Repository address: https://github.com/martin2903/NaratAI-Sentiment-Screener

- app.py: Flask application file.
- update_database.py: Script for updating the database.
- requirements.txt: Dependencies.
- db_operations: directory containing the scripts for updating and querying the database. The sql file for recreating the database relations is also present in that directory.
- external_datasets: directory containing all external datasets that were used to generate the datasets used in this project.
- project_datasets: directory containing all datasets that were used in this project including:
    - bert_dataset.csv - the dataset used for fine-tuning the BERT model.
    - sentiment_lexicon.csv – the sentiment lexicon created for the purposes of this project.
    - articles_data.csv – the dataset of news articles created for this project used to pick the best performing model for use in the application.
- front-end: directory containing the React application.
    - src: directory containing all directories for the different React components.
        - App: directory containing the final React components build and component used in the Selection page of the application.
        - dashboard: directory containing React components used in the Dashboard page of the application.
        - layoutWrappers: directory containing React components used as layout wrappers in various parts of the application.
        - selection: directory containing React components used in the Selection page of the application.
- project_notebooks: directory containing all project supplementary notebooks including:
    - Bert Fine-tuning – notebook explaining and showing the procedure of fine-tuning the BERT model.
    - Model Data Generation and Exploration – notebook where the process of building the dataset for fine-tuning is explained and data exploration conducted.
    - Sentiment Lexicon Generation – notebook where the process of building the sentiment lexicon used in this project is explained and data exploration is conducted.
    - Selecting Model for Production – notebook where the procedure of selecting a final model is carried out by evaluating exported models' performance on the news articles dataset. Information about the dataset is also present.
- tests: directory containing the unit and integration tests carried out.
- utilities: directory containing all scripts responsible for the core functionality of the software and the exported BERT model used in the application.

14.7 Running the Final Software

Production Version

- The web application was deployed and it is accessible via the following link: http://naratai.herokuapp.com/ . Please, be aware that since a free version of Heroku was used for hosting, the containers holding the application are placed in a 'sleep' state when the application has not been active for some time. Therefore, the first time it is being loaded, it could take up to 10-15 seconds to for content to be rendered.

Development Version

Should the reader wish to run the application on a development server, the following must be done:

- For the back-end of the application:
    - The user must install all dependencies specified in the requirements.txt document.
    - Python 3.8.8 was the version used for development
    - The user must install PostgreSQL 12.5 and run the following command in a psql shell in order to rebuilt the database relations:
        - \i project_directory//db_operations//relations_def.sql
    - An update on the database must be done. The user should note that using a development version will imply that historical scores will not be accessible since the database will not contain any historical data.
        - Before updating the database, the user must change the credentials needed to connect to the PostgreSQL db. That could be done in function connect_to_db() in script db_operations.db_updates
        - In order to update the database, the following commands should be executred:
            - cd into the project directory
            - run: python update_database.py (updating can take up to 25 minutes depending on the amount of news at the given time of the day)
    - The user must run the Flask application as follows:
        - On Windows execute the following commands in a command prompt:
            - cd into the project's directory
            - run: set FLASK_APP = app.py
            - run: set FLASK_ENV = development
            - run: set FLASK_DEBUG =0
            - run: flask run
        - On Mac execute the same commands replacing 'set' with 'export'.

- For the front-end of the application:
    - The user must install Node.js 14.17
    - Before running a development server, useEffects that are used to fetch data from the Flask API endpoints must be adjusted given that different ports are used.

        - The user should change the endpoints in all usEffects that fetch data in scripts src/App/app-state.js and src/dashboard/headlines-state.js
    - All dependencies found in package.json must be installed by running the following commands:
        - cd into project_directory/front-end
        - run: npm install *dependency_name (for each dependency)*
    - In order to run a development Node server, execute the following commands form a command line terminal:
        - cd into project_directory/front-end
        - run: npm start

Recreating the model fine-tuning procedure

- All code and instructions on how to recreate the fine-tuning procedure is present in the 'BERT Fine-tuning' notebook in the project_notebooks directory.

Generating any of the datasets used

- All external datasets are present in the external_datasets directory. The user can recreate any of the datasets used for this project visible in the project_dataset directory by following the instructions in the respective notebook in project_notebooks.