Summary of Financial Articles

Tomáš Jakúbek[1], Roman Valovič[1], Jan Přichystal[1]

(Working paper, January 2022 version)

Abstract

The Summary of Financial Articles (SFA) application is intended for obtaining articles in the field of financial markets published by various online media and their subsequent analysis. The aim is to facilitate the work of financial analysts, who have to process a large amount of information on a daily basis, use it to create a picture of the state of financial markets and individual securities, and then issue recommendations to customers on their trading strategy. Gathering the articles and news from online sources is very demanding process due to the technologies used there and ever-changing structure of these web sites. Not only the web application but also the API for integration into customer applications is available.

Keywords: natural language processing, machine learning, sentiment analysis, text summarization, SFA application, web application,

JEL Code: L86

1. Introduction

Every day, financial analysts and advisors process a large amount of information from the financial markets and related areas, so that they can make timely and correct decisions, adjust their trading strategies or advise clients. Their daily activities include the analysis of published articles and reports. However, a large number of them are currently published on the Internet and it is practically impossible to read and analyze them all in a short time.

Summary of Financial Articles is a system designed to speed up the process of obtaining and processing information for these experts and thus facilitate their work. This system obtains articles, news or tweets from selected sources, especially in the field of financial markets, and analyzes their content. The aim is to offer a summarized view of individual documents in the form of several sentences summarizing the main ideas, to determine their sentiment and to provide aggregated data for specified time periods.

2. System design

The application creates (articles and short reports) abbreviated summaries describing the essence of the text from the obtained documents and analyzes their sentiment. It regularly downloads documents and displays them in the web application in the form of a clear list. Users are not presented with full articles, but only a summary. It is available in several forms. If the author of the article summarizes the topic in the introduction, this section is

[1] Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic

displayed (Author tab). It is similar with a possible conclusion at the end of the article, which also often contains important information (Conclusion tab). However, the most important is the summary generated by the SFA application (Generated tab). It is created whenever an article contains enough text to create a summary. For example, this is not always necessary for short messages. The individual summaries are available clearly on separate tabs, so the user has the opportunity to easily choose what interests him.

For each document, there is also information about the date of publication, keywords characterizing the area covered by the report, and a link to the full article available through its title. It is also indicated whether it is an article (Analysis) or a short report (News). Individual documents can be filtered in the application according to:

- the date they were published (Date range),
- ticker – the names of the securities to which they relate (Tickers),
- tags – the area to which they belong (Tags),
- category – article or news (Category),
- sentiment – positive, negative or neutral evaluation (Sentiment),
- domains – web resources from which it is downloaded (Domains)

and sort by date ascending or descending.

The application also displays an overview of the sentiment of the obtained documents for a certain period in the header. In the left graph, it is possible to analyze the ratio of positive and negative messages. The right graph contains information on the most discussed tickers for the selected period, or those for which there was the largest change in the number of comments. In the initial settings, the most discussed are displayed regardless of sentiment. Using the Change button, it is possible to specify a sentiment and an overview of the most discussed with the given sentiment is displayed. For each ticker, a value is also displayed indicating whether the number has increased or decreased (positive/negative value).

2.1. Frontend implementation

The frontend and backend of the system are two standalone applications that communicate via thin REST API on the web server. Heavy frontend is delivered to client browser just once and the content of the website is then partially updated, in places that require change, directly in the browser. This concept is called Single Page Application (SPA) and can be implemented in various JavaScript libraries or frameworks. In our case, to deal with view layer in MVC software architecture, we chose ReactJS library.

2.1.1. Used technologies

ReactJS is still a popular open-source library that is used for building web (and mobile) composable user interfaces. In ReactJS, the user interface is broken down in a large number of reusable components. These components manage their states to handle both the logic and the view itself. Their hierarchical structure clearly defines unidirectional flow of data in the application and their APIs and lifecycle hooks describe the behavior.

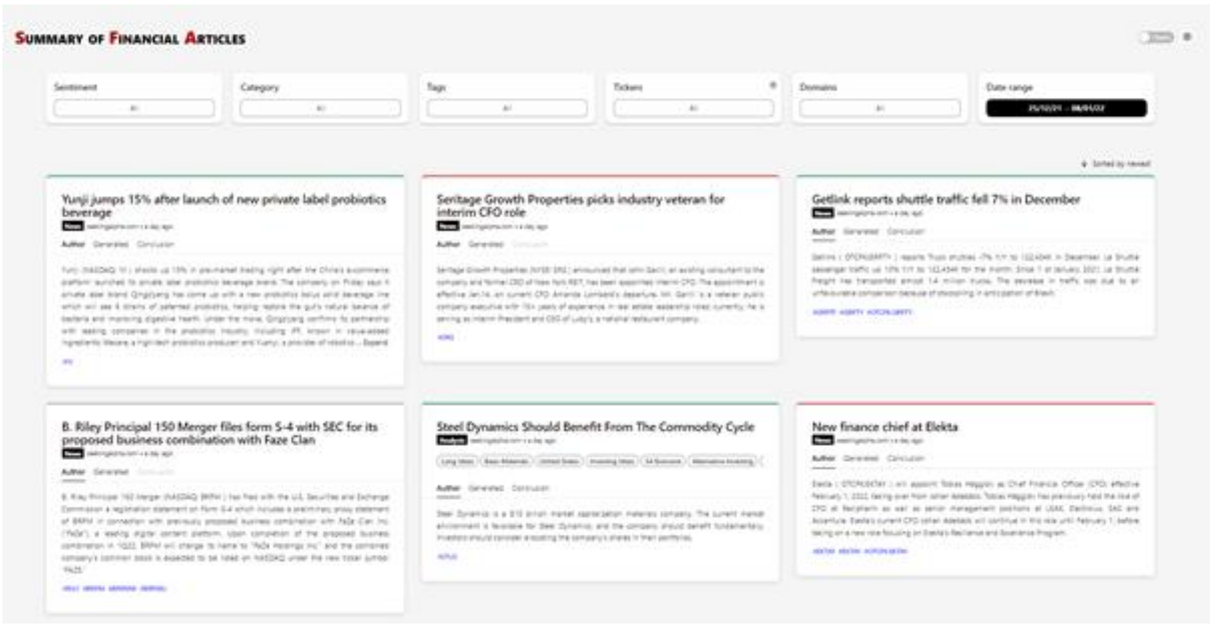
The state of the component is changed after data mutation or by executing a user action and this update triggers a smart and controlled rendering process. In the background, ReactJS engine tries to improve performance by minimizing expensive manipulation with real DOM as much as possible, so it first renders the changes to the virtual DOM represented in JavaScript. Then its job is to reconcile the recent changes against the old one and only the necessary ones applied to the real DOM.

To get the best of ReactJS world, we used JSX (syntax extensions to JavaScript), TypeScript, functional components and React hooks. To manage application-level state we injected Redux into application. Redux is a predictable state container which plays a helping hand while developing large scale JavaScript application. To keep a similar look and feel to our previous apps, we selected Ant Design as the UI framework. This framework is known mostly in the East and does not follow ubiquitous and shabby material design.

2.1.2. User interface

We decided to render the individual article records into easily expandable card components. The grid design seems to be the best choice for large amounts of purely textual data (without any graphics). Each article can have up to three types of summaries (author, generated and author's conclusion), which are displayed as quickly switchable tabs component. The coloring of the cards at first glance corresponds to the sentiment of the article.

To enhance the user experience, article feed is updated automatically by infinite vertical scrolling and a potentially huge number of tags and tickers associated with the article are rendered in a horizontal virtual scrollbar. The application is completely responsive, including specific elements rendered only on smaller devices, and user settings are remembered in the browser storage.


SUMMARY OF FINANCIAL ARTICLES

Sentiment | Category | Tags | Tickers | Domains | Date range

Yunji jumps 15% after launch of new private label probiotics beverage
Author Generated Conclusion

Seritage Growth Properties picks industry veteran for interim CFO role
Author Generated Conclusion

Getlink reports shuttle traffic fell 7% in December
Author Generated Conclusion

B. Riley Principal 150 Merger files form S-4 with SEC for its proposed business combination with Faze Clan
Author Generated Conclusion

Steel Dynamics Should Benefit From The Commodity Cycle
Author Generated Conclusion

New finance chief at Elekta
Author Generated Conclusion

Fig. 1. Article feed with filters

2.2. Data scraping

Other parts of the system are web scrapers. Their task is to automatically retrieve documents from various internet sources. Specifically, these are newspaper articles and short reports from the financial markets.

The articles are fetched at regular intervals, which are appropriately set according to the frequency of publication. Around these scraping modules are therefore created complex dev-ops mechanisms in the form of various pipeline jobs and schedulers that combine the practices of continuous integration and continuous delivery. Scrapers store data in a well-defined structured format in our databases, from where it is consumed by text analysis in the following step.

Scrapers were created in two different fashions. One group of crawlers is implemented in the Apify framework and interpreted in the NodeJS runtime environment, the other makes use of the Scrapy library and the Python programming language. It is also possible to identify two different web scraping approaches prevalent in our solution. In the first one, the scraper starts on a certain web page and, in addition to data scraping, simultaneously collects new links to other documents or subpages. These are recursively visited later and the whole process is repeated. The second approach assumes that a list of URLs to be crawled is available in advance. In this case, the scraper does not look for subsequent links but just crawls the list. This approach was preferred when suitable sitemaps were available. Its significant advantage is the possibility of parallelism and concurrency. To provide incremental crawling functionality and ensure that the already crawled page won't be visited again we implemented middleware for URL filtering.

One of the most efficient ways of preventing access to a public website is IP address blocking. It is very common to encounter during web scraping. To solve this problem, we implemented a custom module to rotate the proxies. Proxy manager runs in parallel with scrapers, searches the internet for freely available proxies, automatically tests them and filters out dead ones. After the crawler detects that the page is blocking access or loading significantly slower, it automatically replaces the proxy with fresh one and repeats the action.

2.2. Application backend

Server-side applications provide several services such as web-scrape (see above), text summarization (SFA Summarization), sentiment classification (SFA Sentiment) and, most importantly for the user, provide a communication interface to the results of the data analysis (SFA Rest API).

The technological complexity and computing power requirements vary dramatically between applications; for example, the SFA Rest API, which responds to a client request with a preprocessed result stored in a database, has only minimal performance requirements on the machine on which the application runs. SFA Sentiment Analysis, on the other hand, provides outputs based on the computations of a complex BERT neural network.

However, the workload of each application varies throughout the day. While the Rest API is queried virtually constantly during normal business hours (8AM–4PM), Sentiment Analysis classifies a new batch of articles every hour. Not only from the point of view of application sustainability and robustness, but also from the point of view of cost-effectiveness, we did not implement the SFA system as a single monolith but opted for a "micro services" architecture. Although there is no strict definition of the term micro service, but we can talk about the framework that this architecture should fulfill:

- services communicate via HTTP (Fowler, 2015),
- services are independent of each other (Nadareishvili, 2016),
- services can be implemented using different programming languages, databases, environments and technologies, depending on the task they are supposed to solve (Chen, 2018),
- services are small, communicate only in a bounded context, autonomous, built and deployed by an automated process (Nadareishvili, 2016).

We implemented the fulfillment of these requirements using Docker technology. In Docker jargon, each service represents a so-called container. The container is a small simple standalone software package and as a monolith contains everything needed to run the application – runtime environment code, system libraries and configuration. Containers isolate software and the environment, ensuring platform independence. Containers can be shut down, restarted and changed independently without directly affecting the running of any other service.

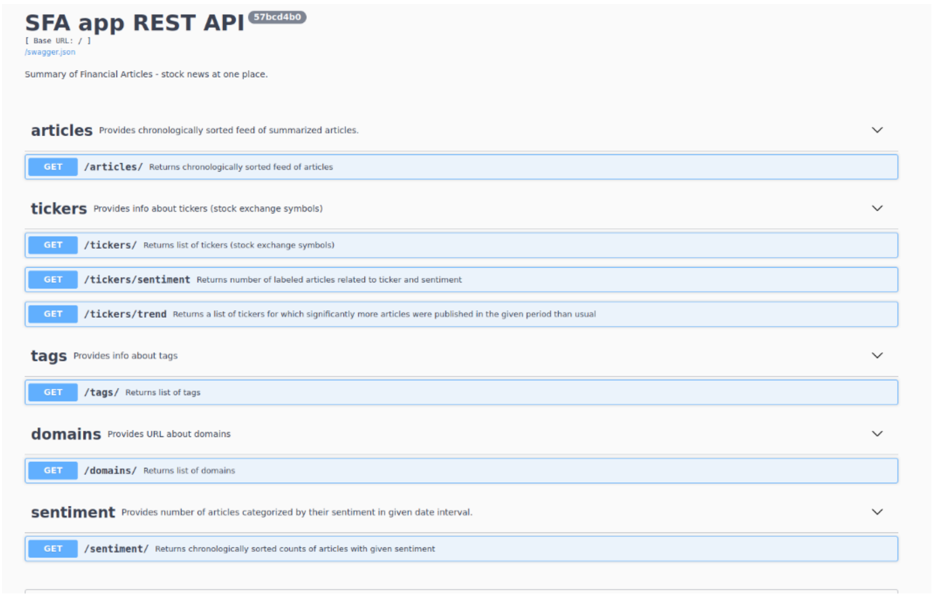
It also allows us to run each service individually and thus efficiently optimize costs – e.g. by scaling performance or shutting down servers completely when the service is not in use. This would not be possible with a monolith architecture.

We provided the automated build requirement for the deployment using Github CI/CD, which provides tools to automate this process in addition to versioning the code. Thus, the developer's only concern is to upload code changes to the main repository. To minimize errors in the code, automated tests are run before each deployment to prevent changes from being introduced to the code if they fail.

2.3. Communication interface

The Flask framework[2] was used for development, which implements the usual best-practices in the field of web application development. It also enables fast deployment thanks to the WSGI communication interface, which is supported by an Apache web server. The framework works as a REST-API, i.e. that, based on the URI and HTTP method of the request performs action and returns the appropriate response, usually structured in JSON format. As a result, a single backend implementation can handle both web and mobile client requests.

During the development we put emphasis on the creation of documentation, which is often neglected resulting in a mismatch between the actual and documented functionality. To avoid this, we used tools that document individual API endpoints automatically. The result is the Swagger framework user interface (Fig. 2), which clearly specifies how one can interact with our REST API.


SFA app REST API 57bcd4b0
[Base URL: /]
/swagger.json

Summary of Financial Articles - stock news at one place.

articles Provides chronologically sorted feed of summarized articles.
GET /articles/ Returns chronologically sorted feed of articles

tickers Provides info about tickers (stock exchange symbols)
GET /tickers/ Returns list of tickers (stock exchange symbols)
GET /tickers/sentiment Returns number of labeled articles related to ticker and sentiment
GET /tickers/trend Returns a list of tickers for which significantly more articles were published in the given period than usual

tags Provides info about tags
GET /tags/ Returns list of tags

domains Provides URL about domains
GET /domains/ Returns list of domains

sentiment Provides number of articles categorized by their sentiment in given date interval.
GET /sentiment/ Returns chronologically sorted counts of articles with given sentiment

Fig. 2. Auto-generated documentation

[2] https://flask.palletsprojects.com/

2.4. Data storage

Two different repositories are used for the storage of persistent data. Articles are automatically retrieved by scraping from various websites. Unfortunately, each site has a different structure, which also changes over time, so we cannot rely on the consistency of the data obtained. Therefore, in the first phase, the data is stored in the non-relational database MongoDB. Non-relational databases do not store data in tables, but as so-called documents. Each document then stores the data in key-value form, but the internal structure of the documents may not be uniform (i.e., each document may contain different keys).

In the second phase, the data is normalized and stored in the traditional PostgreSQL relational database. In this database, we store structured information about each article, such as the title, date of publication, author abstract and, of course, the textual content of the article.

To avoid having to modify the scraper every time the site structure changes, we implemented an intermediate step where we transform data from a non-relational MongoDB database into a relational PostgreSQL database. In addition to changing the structure, this intermediate step allows us to keep track of changes to the article content itself that may have occurred over time.

2.5. Text processing

For each article, the system performs 2 operations:

- Summarization: the original long text is shortened by the extractive summarization technique to a paragraph containing a few sentences. The technique is based on identifying and selecting the most important parts from the original text.
- Sentiment classification: for the summarized content of the article, the sentiment is evaluated into one of 3 classes: positive, negative or neutral. The sentiment of each news can be evaluated from different perspectives, for example, information about the decline in electric car sales will not please Tesla shareholders, but on the other hand, oil company shares will probably rise.

Our model classifies sentiment from the perspective of the investor who owns the stock, so the example above would be evaluated as negative.

2.5.1. Text summarization

We analyzed several different approaches from the point of view of both abstractive and extractive summary creation. We compared outputs of our own solutions and commonly available libraries (e.g. Sumy[3], Python Text Summarizer[4]) in Chochula (2021).

The summary, which contains essential information from the whole article, is created in an extractive way. The Gensim library from the Python programming language is used to create the summary. The extraction is based on the TextRank algorithm (Mihalcea and Tarau, 2004), which is a variant of the PageRank (Brin and Page, 1998) algorithm used by Google to evaluate the relevance of websites.

Algorithm process:

- preprocessing (stop words elimination, stemming),
- creating a graph, sentences forming vertices,

[3] https://pypi.org/project/sumy/

[4] https://www.geeksforgeeks.org/python-text-summarizer/

- edges indicating the similarity between two sentences in vertices,
- run the PageRank algorithm on this weighted graph,
- the highest score is selected and added to the summary,
- the number of vertices to be selected is determined based on the ratio or number of words.

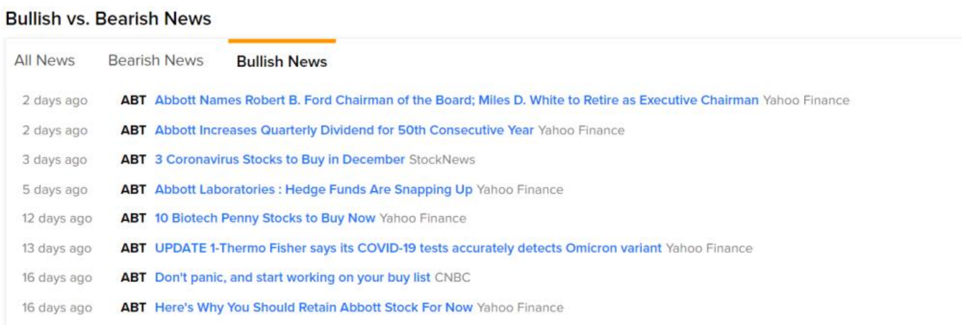
Although we identified more successful algorithms in our research, unfortunately it is not possible to use them in the application. Their deployment prevents technical issues related to the outdated processor architecture on the dedicated computer.

We are currently working on our own solution that combines both approaches (abstractive and extractive) and takes advantage of both. The summary includes unchanged key information, which is very important for financial analysts and at the same time the sentences follow each other smoothly. We will offer this solution in the web application as an alternative summary.

2.5.2. Sentiment analysis

For sentiment analysis, we used several deep learning algorithms according to the state of the art in this field, such as BERT (and its derivatives) or LSTM (Šťastná, 2022). In general, the drawback of machine learning is the need for a large amount of labeled data to train and build the model. Sentiment analysis is a domain-oriented task; a model trained with data from a certain domain (e.g., movie reviews) usually performs poorly in another domain (e.g., financial news). On the other hand, data from the financial domain is limited and manual labeling is expensive and time-consuming.

Therefore, we collect data from available sources on the Internet. Specifically, we use the web scraper tool (described in Section 2.2) to obtain data from seekingalpha.com The articles on this site are categorized into "Bearish" (market declines) and "Bullish" (market rising) (see Fig. 3). In this terminology, we consider bullish news to be positive and bearish news to be negative. In total, we have collected more than 8.5 thousand articles that we can use to train the model.


Bullish vs. Bearish News

All News Bearish News **Bullish News**

2 days ago	ABT	Abbott Names Robert B. Ford Chairman of the Board; Miles D. White to Retire as Executive Chairman Yahoo Finance
2 days ago	ABT	Abbott Increases Quarterly Dividend for 50th Consecutive Year Yahoo Finance
3 days ago	ABT	3 Coronavirus Stocks to Buy in December StockNews
5 days ago	ABT	Abbott Laboratories : Hedge Funds Are Snapping Up Yahoo Finance
12 days ago	ABT	10 Biotech Penny Stocks to Buy Now Yahoo Finance
13 days ago	ABT	UPDATE 1-Thermo Fisher says its COVID-19 tests accurately detects Omicron variant Yahoo Finance
16 days ago	ABT	Don't panic, and start working on your buy list CNBC
16 days ago	ABT	Here's Why You Should Retain Abbott Stock For Now Yahoo Finance

Fig. 3. Bearish and bullish news on Seeking alpha website

We compared several algorithms and inputs for this task and tried to uncover how article length affects the accuracy of sentiment prediction. We attempted to classify sentiment for three types of inputs – article title only, article summary, and full article text. As shown in Fig. 4, all methods achieved the highest F1 score for article summaries, which suggests that the article title does not contain enough information; on the other hand, the raw article text is likely to contain unusable features that are not relevant for sentiment analysis.

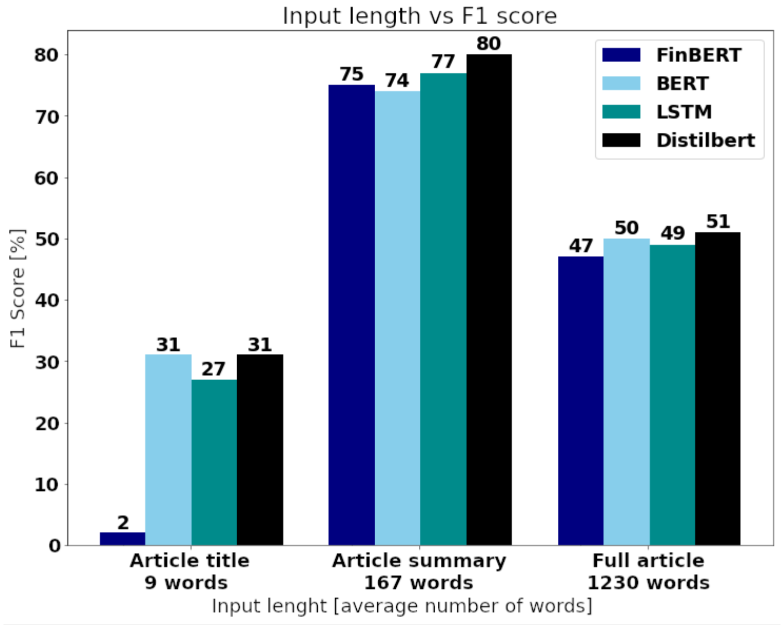


Fig. 4. Performance of methods depending on input length

2.5.3. Process pipeline

The whole process can be summarized as follows. Scrapers get a new batch of articles from selected sources. These are then stored in a non-relational MongoDB database since we cannot guarantee a fixed data structure at the time of download. Some aspects of the article are downloaded afterwards (list of related assets, tags etc.). Once the article is complete, the data is transformed into a relational Postgres database with a fixed structure.

The new records from the PostgreSQL database are taken as input by the summarization method, then the sentiment is classified. The results of both the processes are again stored in the Postgres database. At the last level, a presentation application figures, which clearly displays the analysis results in a user-friendly web browser environment.

Graphically, the pipeline is illustrated in Fig. 5.

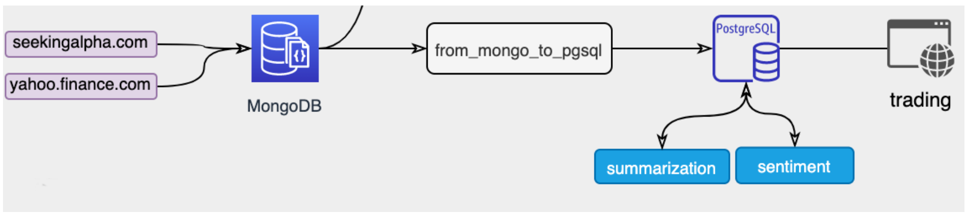


Fig. 5. Illustration of process pipeline

3. Discussion and conclusions

The presented system is currently used by Cyrrus Corporate Finance, a.s. for their daily usage. Based on client requirements, additional information sources are regularly added and the portfolio of analyzed articles is still expanding. The list of long-term analyzes presented in the form of graphs is also under development. The goal is also to integrate the created API into a complex application. This will allow regular users to create their own business strategies, where one of the key attributes may be the sentiment of published articles

There are a number of similar projects available on the Internet that offer analysis of articles in the area of financial markets and securities trading. Sentimentrader provides daily sentiment report and ad hoc reports if there is anything especially timely or unusual. The Daily Sentiment Report includes an overview of where short- and intermediate-term sentiment is each day, along with updates on indicator extremes or studies focused primarily on sentiment, breadth and price action (Sentimentrader, 2021).

Sentdex (2021) is an application providing the sentiment analysis in Finance, Politics and Geographical area. The Financial analysis shows the sentiment for selected companies and long-term analysis od S&P 500 Index. The analysis is available for 7 days, 30 days, 6 months and 1 year interval. The stock price is available in presented chart too.

Stockgeist is an interactive platform for monitoring the current popularity of 2200+ publicly traded companies. It provides ranking based on the number of published messages about companies, stock overview, aggregated news from various websites, stock sentiment analysis based upon social media messages, wordcloud with the most popular topics the crowd is talking about, fundamentals such as market capitalization and P/E ratio, short story on the selected company to give a general idea of what they do and charts that display historical data about social media messages and positivity. Stockgeist (2021)

Stocktwits is a platform focusing on the same problematic. It shows what actual investors and traders are saying in real time about the stocks, crypto, futures and forex. (Stocktwits, 2021)

We can see from this overview the sentiment analysis is very important task in natural language processing and the financial analysts are really interested in information published in articles and news. We know from a discussion with our customer that even a summarized view of published texts is very important. However, other applications do not provide this functionality, which we consider an important advantage of our solution. The prototype of our application is available at https://mta.pef.mendelu.cz/trading/#/feed.

Acknowledgements

This article is supported by the project EG17_176/0015722 – Data processing software (2020–2023, MPO/EG).

References

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems, 30(1-7), 107–117.

Chen, L. (2018, April). Microservices: architecting for continuous delivery and DevOps. In 2018 IEEE International conference on software architecture (ICSA) (pp. 39–397). IEEE.

Chochula, P. (2021) Sumarizace dokumentů z oblasti finančních trhů, Final thesis, Mendel University in Brno, available at: https://is.mendelu.cz/lide/clovek.pl?id=71901;zalozka=7;zp=69217;studium=107183.

Fowler, M., & Lewis, J. (2015). Microservices: Nur ein weiteres Konzept in der Softwarearchitektur oder mehr. Objektspektrum, 1(2015), 14–20.

Mihalcea, R., & Tarau, P. (2004, July). Textrank: Bringing order into text. In Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404–411).

Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). Microservice architecture: aligning principles, practices, and culture. O'Reilly Media, Inc.

Sentdex. (2021). available at: http://sentdex.com/.

Sentimentrader. (2021). available at: https://sentimentrader.com/.

Stockgeist. (2021). available at: https://www.stockgeist.ai/.

Stocktwits. (2021). available at: https://stocktwits.com/.

Šťastná, P. (2020) Analýza sentimentu v oblasti financí a obchodování s cennými papíry. Final thesis, Mendel University in Brno, available at https://is.mendelu.cz/auth/lide/clovek.pl?id=68274;zalozka=7;zp=71400;studium=107655.