DSCI 551 Final Report

A Tesla News Article and Stock Searching Engine

Linle Jiang (Psychology, Applied Data Science)
Yu Ren (Statistics, Applied Data Science)
Yucheng Zhang (Computer Science(BS & MS))

Project Description

Tesla, a renowned electric vehicle and clean energy company, is among the most heated topics in the stock market in recent years. With that, we built a search engine that allows users to conveniently query articles and stock data related to Tesla. Our main objective in this project is to build a web application, which enables stock price query, news query, and keyword searches on the interface. Additionally, our search engine also enables users to explore the data conveniently via our web interface. Most of the results are displayed in a table, except for the actual article content, which is listed in the Article web page separately. With all the information available, we intend to help our users understand, at least qualitatively, how the price of this particular stock changed over time, and how the market reacted to its performance on the trading market, as reflected in the news articles.

Datasets and Data Management

Tesla Stock Price Dataset

This dataset (170 KB) was obtained from Kaggle. The data was originally collected from Yahoo finance, containing Tesla's stock data, at the daily level, from 2010-2020. In particular, this dataset contains the date, opening price, highest price, lowest price, closing price, adjusted closing price, and trading volume on the corresponding date. This dataset is stored in the Cloud Firestore of the Firebase, **a cloud-hosted NoSQL database** (**Fig 1**). When constructing this dataset, we performed **data cleaning** in the step. The data was pre-processed to only retain data from 2012/05/23 to 2020/02/03, in order to match the same date range with data from the article dataset (more details in the following section). Also, the adjusted close price attribute was

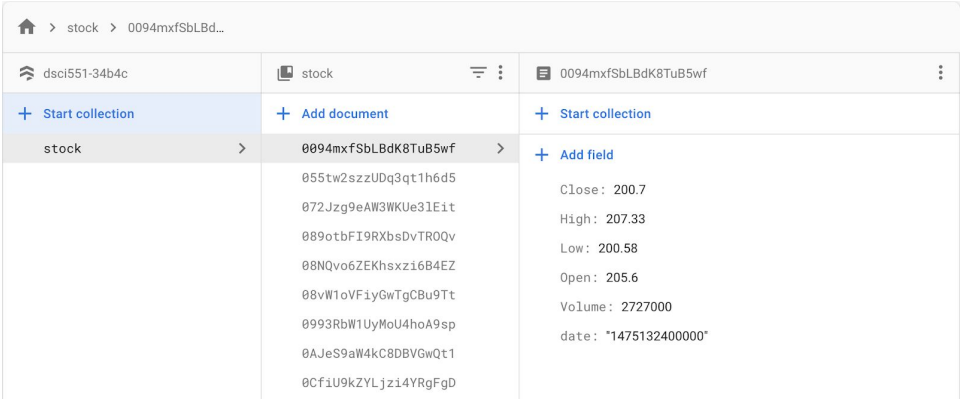


Fig 1. Tesla Stock Data in the Cloud Firestore

removed from the dataset since it yields identical values as with the close price attribute. Note that we also transformed the date format to timestamp when uploading to Firestore.

Tesla News Dataset

Similarly, this dataset (about 15 MB) is part of the Historical financial news archive downloaded from Kaggle. The original dataset contains news archives of more than 800 US companies for the last 12 years. In this project, we only retained the news headlines, sources, contents and dates variables, that are related to Tesla. Specifically, these articles range from 2012/05/23 to 2020/02/03. Additionally, to provide a more semantical way of organizing the articles and to enable scalable data processing, we used a parallel processing technique (i.e., Spark) to process the data, generating a topic attribute for each article in order to assign each into a cluster with similar topic. This dataset is stored in the **MySQL database** (**Fig 2**).

```
mysql> select id, ticker, title, category, release_date, provider, url, article_id from news limit 5;
+--------+--------+-----------------------------------------------------------------+----------+--------------+---------------------------+----------------
------------------------------------------------------------------------------------+------------+
| id     | ticker | title                                                           | category | release_date | provider                  | url
                                                                                    | article_id |
+--------+--------+-----------------------------------------------------------------+----------+--------------+---------------------------+----------------
------------------------------------------------------------------------------------+------------+
| 228706 | TSLA   | Tesla  TSLA  Expected To Beat Earnings Estimates  Should You Buy | opinion  | 2020-01-22   | Zacks Investment Research | https://www.inv
esting.com/analysis/tesla-tsla-expected-to-beat-earnings-estimates-should-you-buy-200500661 | 200500661  |
| 228707 | TSLA   | What s The Right Valuation Multiple For Tesla                   | opinion  | 2020-01-22   | Zacks Investment Research | https://www.inv
esting.com/analysis/what39s-the-right-valuation-multiple-for-tesla-200500769        | 200500769  |
| 228708 | TSLA   | Tesla  NIO And The Electric Boogie                              | opinion  | 2020-01-22   | Zacks Investment Research | https://www.inv
esting.com/analysis/tesla-nio-and-the-electric-boogie-200500783                     | 200500783  |
| 228709 | TSLA   | Toyota  TM  Recalls 3 4M Vehicles Worldwide Over Airbag Defect   | opinion  | 2020-01-23   | Zacks Investment Research | https://www.inv
esting.com/analysis/toyota-tm-recalls-34m-vehicles-worldwide-over-airbag-defect-200500917  | 200500917  |
| 228714 | TSLA   | Tesla  TSLA   Does This Rally Still Have Legs                   | opinion  | 2020-01-20   | Zacks Investment Research | https://www.inv
esting.com/analysis/tesla-tsla-does-this-rally-still-have-legs-200500132            | 200500132  |
+--------+--------+-----------------------------------------------------------------+----------+--------------+---------------------------+----------------
------------------------------------------------------------------------------------+------------+
5 rows in set (0.00 sec)
```

Fig 2. Tesla News Article Archive in MySQL

In terms of **data integration**, the two datasets are integrated by the date attribute.

System Architecture

Our architecture design is shown in **Fig 3**. First of all, we uploaded our datasets to the databases. As mentioned before, the Firestore database stores Tesla stock data, while the MySQL database stores Tesla News dataset. Next, we created a web interface that allows users to make queries through a graphical user interface. And this interface further sends the user queries to the corresponding database, and awaits for the results sent back from the database.

Web Interface

We designed a responsive web interface using Vue.js. Specifically, there are three kinds of web pages in this web app. The home page (**Fig. 4**), which allows users to query specific articles by applying filtering conditions. A stock page (**Fig. 5**), which enables users to query stock data. And the article page (**Fig. 6**), which is used to display the actual content of a news article. We also set up a navigation bar, following web design conventions, which allows users to conveniently navigate between pages.

In this project, we tackled the **data searching** problems in a number of ways. In the home page, we allow users to specify a date range, and input keywords for any one field with textual data in the article database to submit the search query. In the stock page, we use graphical tools (i.e., sliders) to enable users to specify searching range for date and one

selected stock attribute to compose the search query. In the end, users are able to receive the queries results in a fairly **short** amount of time.

In terms of the **data exploration**, we also addressed this problem with different techniques. In the home page, before users send out the search query, they could also use the group by feature to request the searching output grouped in a specific way when showing in the result table. To bring it to the next level, we also allow users to sort the results based on any of the attributes they are interested in. Same thing goes for the stock page, that is, users could sort the result table based on any of the attributes they want. Additionally, between pages explorations are also enabled. For instance, users could click on any of the articles they are

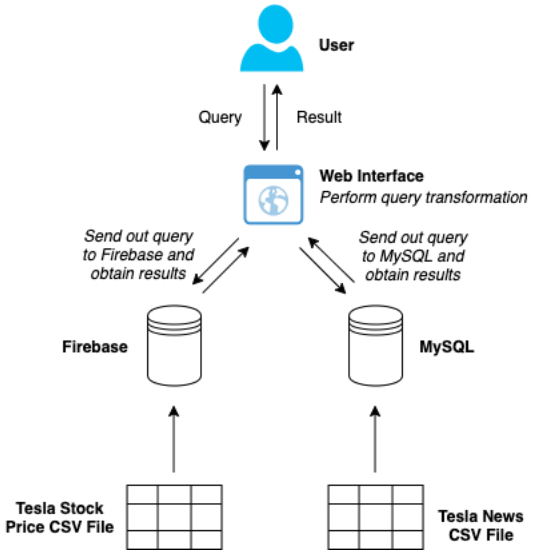


Fig 3. System Architecture

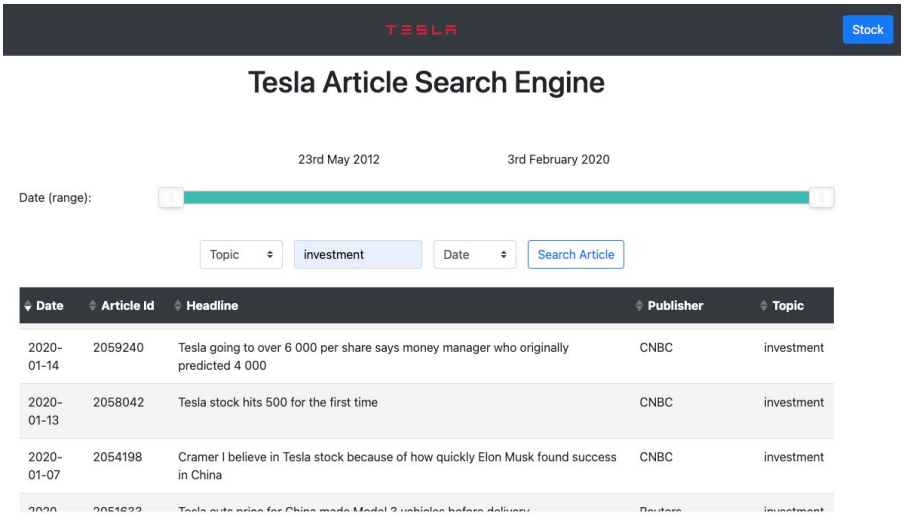


Fig 4. Home Page

interested in and be directed to the article page to browse the contents. One **user case** here might be that users may find any of the dates, publishers, or topics interesting, they could click

on any of them to be directed back to the home page, and be able to see all articles that match their interests. From the stock page, one typical **user case** is that after users receive the stock result, they might be interested to see how experts analyze the stock performance of a given date. In this case, users could click on the row they are interested in and be directed to the home page to see the news coverages for that date, if any. Notably, we optimized our code so that all queries are sent even before users are redirected to the corresponding page, this **increased the speed** for users to see the results.

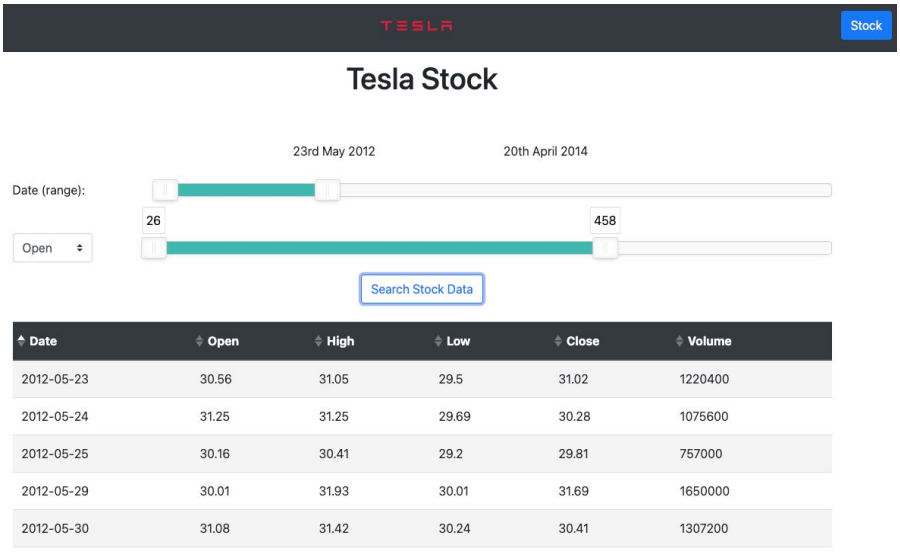


Fig 5. Stock Page

Fig. 6 Article Page

As to MySQL connection, we experienced a little twist at setup. We started after the Vue framework was settled, only to find there is no way to have MySQL engine connected to our Vue project directly. We performed some research and tried to use PHP as a connecting bridge.

However, with this Vue + PHP + MySql tech stack, we figured we could only have a Vue component in a PHP framed project, where MySQL is connected and configured using PHP, yet no vice versa. Since we have done lots of work on Vue, it was not reasonable to redo everything again. Therefore, we decided to build our own Restful API to connect our MySQL engine and Vue project using NodeJS. We then had connected and configured MySQL at our new API, having it run at a certain localhost port, then set this url be the base url of our Vue project, thus MySQL and Vue finally working together smoothly as charm.

The framework for MySQL queries was built on node.js's mysql2, express, and cors APIs. Our MySQL framework will take user selected parameters such as what to query on, group by, timeframe, etc. from the Vue web interface and then send the proper sql query to the connected MySQL database (local or ec2). Once query results are received, our MySQL framework will send the results back to our Vue web interface to be displayed.

The framework for Firestore queries was built in the client side (i.e., frontend). Our framework will take the date range and range of any one stock attribute as a query and send that to the Firestore. Once query results are received, they will be displayed on our Vue web interface.

Descriptive Analyses and Evaluation

We performed some **data analyses** to understand more about the datasets.

Tesla news dataset

There were 4,283 articles collected from Investing.com, starting from 2012/05/23 to 2020/01/30. No missing data in this dataset. Among these articles, 59.61% of them were news, while the remaining of them are opinion articles. On average, there were two articles released per day. Below is the bar chart for the top 20 providers of the articles (**Fig 7**).

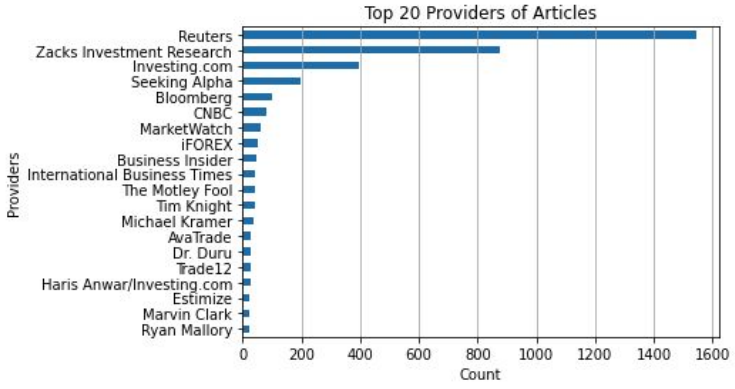


Fig 7. Top 20 Provides of Tesla Articles

To investigate which dates Tesla became a hotter topic, we also included the bar chart for the top 20 article release dates (**Fig 8**).

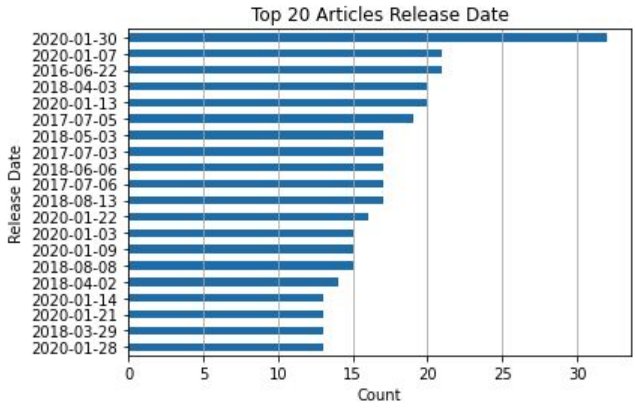


Fig 8. Top 20 Articles Release Date

To understand the release date information of our dataset, we also looked at the frequency distribution regarding the amount of articles released per day (**Fig 9**).

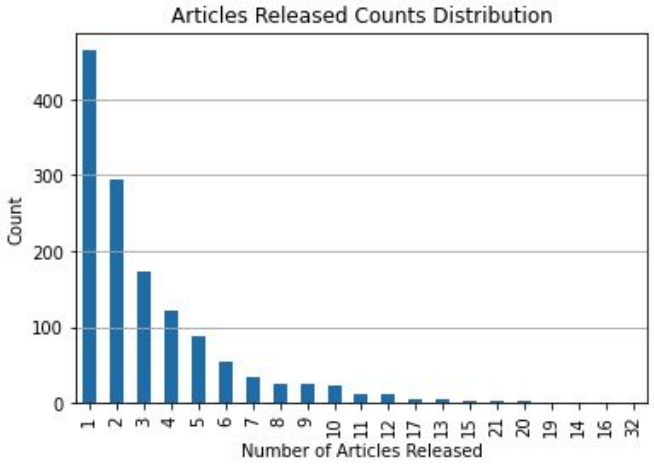


Fig 9. Articles Released Counts Distribution

Tesla stock price dataset

This dataset contains Tesla stock prices from 2012/05/23 to 2020/02/03. There was no missing data in this dataset. Below is a table showing the descriptive statistics for all numeric variables (**Table 1**).

Table 1. Descriptive statistics of Tesla Stock Attributes

	Open	High	Low	Close	Adj Close	Volume
count	1936	1936	1936	1936	1936	1936
mean	225.76	229.74	221.75	225.94	225.94	6617931.35
std	98.68	100.55	97.19	99.18	99.18	4990635.63
min	26.84	26.85	25.52	26.1	26.1	375800
25%	189.97	193.49	187.03	190.61	190.61	3565275
50%	230.7	234.76	227.07	230.54	230.54	5477550
75%	295.2	300.32	288.83	294.79	294.79	8163200
max	673.69	786.14	673.52	780	780	47065000

Topic Modeling and Evaluation

Overview:

The goal of topic modeling is to allow users to query on articles that share similar topics. With each news article assigned with a topic, users can search articles directly based on the topic name. As one of the unique features of this app is continuous exploration, if users queried news articles based on other attributes, users can still query again on topic through clicking on the article's topic name in the article page. This is also part of the **data analysis** problem we addressed in the project.

Methodology:

Two different approaches were used for topic modeling: parallel processing method and non-parallel processing method.

Non-parallel processing method:

Text Preprocessing: Text data are processed through: punctuations & numbers removal, tokenization, lemmatization, and stop words removal. Processed text data are then used to generate bigrams which would be typically more beneficial to topic modeling compared to unigrams. These were done with standard python code.

Topic modeling is done through LDA(Latent Dirichlet Allocation) from gensim library.

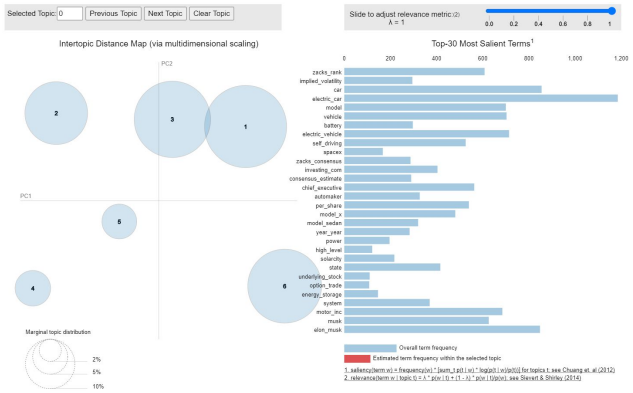
Parallel processing method:

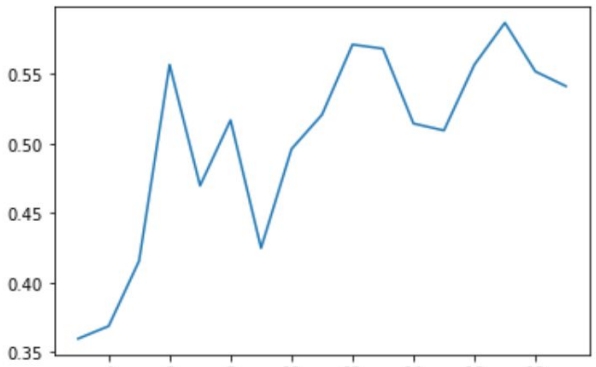
Pyspark is used for text loading, pre-processing, and topic model building. Data was first loaded into a spark dataframe from a csv file. News article content part of the dataframe is turned into rdd for further pre-processing (lower case, remove stopwords, etc.). Then rdd is turned into a dataframe with index to be fed into the pyspark.ml.clustering LDA model.

Result and Evaluation:

Hyperparameters:

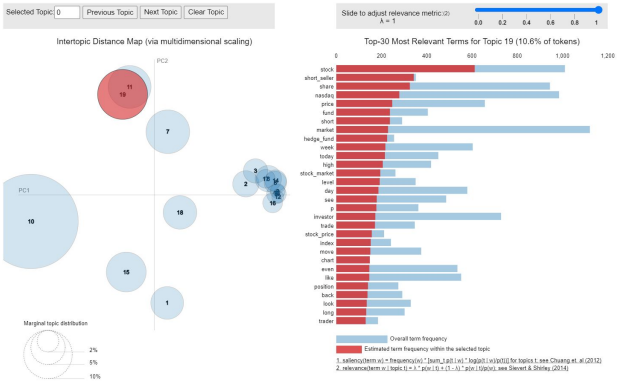
There are several hyperparameters for the LDA model: text n-gram, number of topics, number of iterations, and max vocabulary size that'll be used for each topic. After hyperparameter tuning process, we found that bigrams had the best performance on our dataset. As for the number of topics, we ended up with two values to choose from: 6 & 19. When the number of topics is set to 6, topics in the visualization of the lda model are mostly separate from each other. However, if we solely judge the model performance based on coherence score, then 19 topics is better. We decided to pick 19 topics for our project.





Num of topics: 6

Coherence Score wrt. number of topics



Num of topics: 19

Topic Manuel Labeling:

Although articles can be linked to each other through topic number alone, for users to query on topics explicitly, we need a topic name for all our topics. Below is a summary of all our 19 topics (**Table 2**).

Table 2 The 19 Article Topics

Topic ID	Name	Topic ID	Name	Topic ID	Name	Topic ID	Name
0	vehicle	5	research	10	finance	15	tax
1	technology	6	growth	11	accident	16	trading
2	ai	7	analysis	12	projection	17	product
3	production	8	expense	13	investment	18	stock
4	hyperloop	9	plan	14	environment		

Conclusion

Taking into account practical user cases, we built a responsive web interface for the Tesla News and Stock searching engine that enabled users to make queries conveniently to our

two databases, to receive the results quickly, and to explore the data efficiently. In this development process, we mimicked real-world app development processes, incorporating web design processes, such as identifying target users, ideation, and prototyping to determine the visual and functional aspect of our artifact; and following a mature software development branch workflow (i.e., Git). In the future, we would like to extend this project by connecting our interface to real-time databases, allowing users to retrieve up-to-date information. In addition, we would perform more advanced analyses and modeling to include an extra feature that shows users ranked news based on their impacts on the stock price of a given date.

Teamwork

Linle Jiang: project planning, design prototype, data processing, data analysis (descriptive analyses), implement web interface, setup Firebase connection, implement Firebase queries, report writing.
Ren Yu: project planning, data processing, setup MySQL connection, implement MySQL queries, report writing.
Yucheng Zhang: project planning, data processing, data analysis (topic modeling), setup MySQL connection, implement MySQL queries, report writing.

Video Link: https://youtu.be/Ws7XhgcVcOY