News Search Engine

Hongchuan CAO, timothy Shen, Fanwei Meng, Jinman Xing, Ziang Tan, Mingyu Yue

March 17, 2022

Abstract

The news search engine is a web application which provides news search and news recommendations for users. Request a query, servers will return related news. When users login in, it will give recommended news for specified person according to history. The collection of news contains 102,067 news until the article finished, and spiders crawl news from BBC,CNN, and New York Times every day. The ranking algorithm is BM25 achieved by Spark, the recommend algorithms are Collaborative Filtering (CF), Content-based Recommendation and Hot News Recommendation, the abstract generation algorithm is Bart, which is based on transformers. Project also achieves spell correction and query suggestion. The web service can be accessed from the link: `http://mighty-beyond-16569.herokuapp.com/`. The instruction is at Appendix A And source code has been published on github `https://github.com/ttdsgroup7`, which has crawlnews, Search-engine-frontend and newsearchegine projects.

1 Introduction

With the advent of the information age, the speed of information changes is getting faster and faster, and news emerges endlessly every day. It has become even more important for people to get the information they want from the plethora of daily news. People are becoming more and more precise about the speed and accuracy of getting news. People may want to get news happening in a certain place at a certain moment, or they may want to find all relevant news through a certain keyword, or get the most popular news at a certain moment, and so on.

Therefore, based on this need, the project combines three relatively large news websites, BBC, CNN and New York Times as data sources, and designs and implements a news recommendation system according to the content of TTDS classroom. In order to improve the user experience, we have designed a good interface and provided data visualization to facilitate users to view news happening in different regions. The news system takes BM25 as the core algorithm, processes query strings according to news content and headlines, and provides users with relevant news. In addition, we also provide a series of additional functions, implementing Collaborative Filtering (CF), Content-based Recommendation and Hot News Recommendation, as well as spell checking, drop-down search and query suggestions, etc., to provide users with better experience.

2 System Design

We split the project into two parts, front-end and back-end. Front-end uses *VUE* framework to interface with users. Users can input query string to obtain news results and login to get recommended news for individuals. We also visualize news data into a global map, which makes it easy to find news in specified region. Back-end supplies APIs for front-end. First, we crawl news datasets from BBC,CNN,New York Times and store them in **MySQL**. The database tables relation is shown in figure 6. Then we apply BM25 algorithm to generate indexer and store them in **Redis**, where all servers can access to it. So it is convenient to deploy services among clusters. After that, we should parse query string and calculate news results then send top 100 news back to clients. Except above, we design recommend system for users to provide related news, provide spell check to avoid wrong input query, and give suggestion for users' query to make query much better.

Divide into three main parts according to function: **crawler**, **core NLP algorithm**, **web interface**. We will crawl data and update indexer every day. In order to speed up calculation, we use **Spark** framework to parallelly run algorithms. Besides, we store all the middle-results in database, which let us easy to extend projects among clusters. The whole structure is shown in figure 2.

3 Data Collection

3.1 News Crawler

We selected **BBC**, **CNN** and **The New York Times** as data sources. We implement the crawler by using the Forcrawler framework, and the Python-based requests library, Goose3, etc. By analyzing the website pages, we obtain the required information: each news, weget title, content, the url of the cover image, the url of the news, the release date and Location. Then we write all the information of each news into the Mysql database in turn.

We found a url in the networkpage of the browser's developer tools that contains all the metadata for a page of news. This is a json format page, which contains the meta information of all the news in the BBC, CNN news list page. In order to turn the page, we only need to modify the pageNumber in the url. To ensure that crawlers can get news without being blocked, we set a download delay for each page. So the crawler will pause for 0.2 seconds between crawling two pages. In order to ensure the continuous growth of news id, we do a SELECT before inserting a crawled news, and skip it if the news is already in the database (this is because we use the innodbas storage engine, the id is an incremental index, if unsuccessful Since the same url inserts news, the index will skip instead of being consecutive). After that we deploy the crwaler to the server and run it every 12 hours. At the same time, since we have a total of three crawlers, accessing the database at the same time may cause locks, and then loading the data in it fails. Therefore, for robustness, we use a timeout retransmission mechanism, and if the input fails, we try 5 times. Since the crawler itself may also be temporarily blocked by the website, we use the sleep function to wait between each request. If a request fails, the crawler will sleep for 5s and try again. Additionally, we checkpoint the crawler and when the data size reaches a threshold, we can write to the database. These additional features can help our crawler have better performance and incident handling.

In terms of website classification, the classification of different websites may have different divisions. We first use the BBC theme as the basis for classification. The New York Times then uses a dictionary to convert similar category names into BBC-style subject names. Because the subject classification of CNN is too detailed and has more types, the subject has been expanded to a certain extent.

3.2 Abstraction Auto Generator

The news abstraction can help people grab the main point and motivates them to go on reading if it catches them. Some of the news we collected have already provided it, while the majority lack it. Therefore, we manage to use an machine learning way to automatically generate the abstraction based on the content. The machine learning model requires us to train the dataset, test and fine-tune to get the best performance. The problem is that it requires large time and computing power to do, which requires using powerful GPU like Tesla T4 to continuously run for a long time. If we rent such devices on Google Cloud, the cost will go beyond $200/month if turned on all day long[1]. Therefore, we need to find the suitable model which provides pre-trained model to use.

At the beginning, we look at a newly proposed abstraction model called PEGASUS[2], which is proposed by Google Search. This is a model based on transformer and sounds suitable for this task, but we only find the xsum version of pre-trained model[3], which can only generate one sentence of summary and is too short to use. So, we look at another popular model, which is Bart[4], led by Facebook. This is also a transformer encoder-encoder(Sequence2Sequence) model, with a bidirectional encoder and an autoregressive decoder. It uses an arbitrary noising function to corrupt the text and train the model to reconstruct the original text. This model is popular and many pre-trained versions can be found. Finally, we choose the distilbart-cnn-12-6[5] version, a BART model training on the CNN news, having 12 layers of encoder and 6 layers of decoder. This is a fine-tuned version which has been 1.68 times faster than the baseline model. It is suitable for our project, since our search engine is also targeted at news. The pre-trained model still requires us to use GPU, but it costs much less and only requires to execute tasks daily after the news collected. It is tested that, on Tesla T4, we can generate abstractions at 170/min. We only need to turn on the GPU machine for 10 minutes every day, so the given Google Credits can suffice.

When we test our abstraction code, we use the Google Colab[6], which offers free GPU resources to use. Since the GPU RAM is limited, it is not possible to calculate all the abstraction simultaneously. We manage to solve it by dividing news into many small blocks to calculate. The Pegasus model is also implemented inside, and it may become useful in the future since it can nicely generate single sentence abstraction. Besides, we have found that Bart model has the additional feature to fill multi-token masks, which is helpful to fill out the missing words inside the text. We have implemented this function and may put into use in the future.

3.3 Nation Name Auto Generator

Although some of the data contains nation name, there is still a part which does not have. This information is important in our new search system as it shows the news by regions. Besides, the news data visualization is also dependent on this.

After searching for efficient ways to solve this problems, we have found a python library called "geograpy3". This is a powerful library that can extract the countries, regions and cities from the given text. Basically, the extraction process can be partitioned into two steps. The first step is to use Natural Language Processing to analyze texts and retrieve potential mentions of geographic locations. Then the words which represent locations are looked up using the Locator. This library uses a database file to match the nation name, which contains cities, countries, regions, their wikidata and so on. Since the NLP algorithm of this method needs to traverse all over the content to get the potential nation names, the generation time is a bit slower. However, we have managed to deploy it on the Google Cloud, so that it can generate daily after the new crawling is finished. The daily update routine helps to keep our data up-to-date and can then use country name to categorize and see the related news.

4 Preprocessing and Indexing

Okapi BM25 algorithm is the core section in this project. Use the content and title of news as source and then deal with query string according to results of Okapi BM25.

Term Frequency $tf(t,d)$ is the relative frequency of term **t** within document **d**.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1}$$

where $f_{t,d}$ is the raw count of term **t** in document d.

Inverse Document Frequency $idf(t,D)$ is a measure of how much information the word provides, which obtained by dividing the total number of documents by the number of documents containing the term.

$$idf(t,D) = ln\left\{\frac{N - |\{d \in D : t \in d\}| + 0.5}{|\{d \in D : t \in d\}| + 0.5} + 1\right\} \tag{2}$$

where $N = |D|$ is the total number of documents in corpus D. $|\{d \in D : t \in d\}|$ means the number of documents containing term t in corpus D. In order to avoid a division-by-zero, so here adds 0.5.

For query **Q** containing $q_1, q_2, \ldots, q_n$, the BM25 score is

$$S(D,Q) = \sum_{i=1}^{n} IDF(q_i) \frac{f(q_i,D)(k+1)}{f(q_i,D) + k(1 - b + b * \frac{|D|}{avgdl})} \tag{3}$$

where $f(q_i,D)$ is term frequency in document D, $|D|$ is the length of document D and avgdl is the average document length in corpus.

In order to speed up calculation, we use *Spark* framework to achieve TF-IDF algorithm. There are main five steps (process flow shown in figure 4):

1. Read all news from database(Mysql). [docid $\longrightarrow$ content]
2. Process news content and obtain word list for each news. [docid $\longrightarrow$ wordlist]
3. Calculate word frequency for each news. [docid $\longrightarrow$ (word, word frequency)]
4. Calculate **idf** for each word in corpus. [wordid $\longrightarrow$ idf]

These jobs are shown in figure 1, use spark cluster (contains three nodes, cluster details shown in figure 3) to balance workload and run at the same time.

5 Ranking Retrieval

The main function of the project is to deal with query. The web interface send request with query to backend servers. The query sent by web interface is an advanced search query, which contains keywords, **AND**, **OR**, **NOT**, **WINDOW[num, word1, word2]**. For example, a query is *(word1 OR word2) AND word3*, as we all known, it is a **Infix expression** so we have to transfer it into a **Suffix expression**. Therefore, we use stack data structure to parse query string. And then use **suffix expression** to calculate news results.

There are main 5 steps:

1. check spell error and correct them.
2. send the correct query into suffix expression generator to calculate suffix expression of the query.
3. scan suffix expression from start to end and calculate news results according to BM25

middle-indexer and return top 100 news to next step.

4. Query extension: receive the top 100 news and select top k to obtain their top n words and then add these words to previous query and apply the query string again.
5. response the top 100 news from the second query to clients.

In order to quickly find the top k news from all the datasets, we use PriorityQueue with k size to store the result and scan all news to obtain results, whose time complexity is $O(Nklogk)$.

We use BM25 algorithm to obtain news results, all the term frequency and inverse document frequency are stored in **Redis**, query servers can get them directly through network. And the calculation equation is [3].

6 Additional Features

6.1 Recommend System

This project mainly achieve **Collaborative Filtering (CF)**, **Content-based Recommendation** and **Hot News Recommendation**.

6.1.1 Collaborative Filtering (CF)

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It finds neighbors based on the user's preference for items, and then recommends what neighbors like to the current user. Use a user's preference for all items as a vector to calculate the similarity between users. After finding K neighbors, according to the similarity weight of the neighbors and their preference for the item, predict the current user's preference, and get a sorted list of items as recommendations.

In order to achieve this algorithm, we have to maintain a table, contains ratings of the items for each user. The preferences are stored in database table *newslogs*. To speed up calculation, here we use Apache Mahout in *Spark MLlib* to achieve **user-based CF**.

1. Define similarity function (eg. Euclidean Distance or Cosine Distance) for users. (we will use this function to calculate the top **n** nearest users to the current user.)
2. Then calculate the rating that current user would give to a certain news. (We will use the average rating of **n** similar users as its value.)

$$R_U^I = \frac{1}{n}\sum_{u=1}^{n} R_u^I \tag{4}$$

where R_u^i represent the rating of item i from user u.

3. Use the top k in R_U as recommendations.

6.1.2 Content-based Recommendation

Content-based recommendation systems recommend an item to a user based upon a description of the item and a profile of the user's interests. While a user profile maybe entered by the user, it is commonly learned from feedback the user provides on items. Each user's preference contains several themes with ranked words list according to TF-IDF value. This part is achieved from scratch using Java.

1. Decrease TF-IDF value each day to keep that newest recommendation for users.
2. Travel all the new-added news and use its word TF-IDF value to compare with word TF-IDF in current user's preferences list. Add these TF-IDF value of words appeared at both list as the score of the news.

$$R_u^i = \sum_{i=1}^{N}\sum_{word}(tfidf[word] + pre[word]) \tag{5}$$

where $word$ is in $pre_u \cap news^i$, and R_u^i is rating of news i for user u. N is the number of new-added news, pre_u means the word set from user u preferences. $news^i$ means the word set in news i.

3. Then sort R_u^i for each user, and select top k news as the recommendation.

6.1.3 Hot News Recommendation

Like its name, obtain the most popular news (according to views) and recommend them to each user if we can't calculate enough recommended news for the current user.

Each day, after spider crawls news, system will automatically preform the three recommendation algorithms above for all users and update them in database.

6.2 Spell Check

Spell Check is an important part in the informational retrieval system. The users may have type errors when searching. Basically, the spell check can be divided into two parts, which are spelling error detection and spelling error correction. To detect the typos, we develop a term dictionary to store all the terms appearing in the Reuters dataset and oxford common sentences dataset. The correction part will then use the Bayes Theorem, edit distances, uni-gram and bi-gram dictionaries to solve.

6.2.1 Spell Error Detection

To detect errors, we firstly need to construct our own dictionary. If the users' input words are not in the dictionary, we can mark it as a wrong word and use the correction operation. With the help of the python library NLTK, we easily get the Reuters dataset. However, since Reuters is a news website and may lack some common words used in life, we explore and find a mini dataset of Oxford dictionary[7], which contains nearly 90,000 example sentences. This is a Chinese-English version of Oxford dictionary dataset, and we managed to use Python to extract the English sentences from it. The effect of applying new dataset is satisfactory overall all. When we only use the Reuters dataset, we can extract about 31,000 terms and about 360,000 bi-gram words. After merging new dataset, we get about 77,000 terms and 540,000 bi-gram words. The bi-gram words will be helpful to calculate the relativity between the candidates of the wrong-spelling words and the neighbour words.

After the consideration of the possible user action, we decide not to remove the stopwords. This is because the removal of stopwords may break the sentence coherence, making two unrelated words adjacent. Such sentences may have grammar errors and lack subject, predicate or object, which will make the calculation of bi-gram words relativity wrong. The stemming method is also not applicable with the similar situation. For example, the phrase "like eating" after stemming becomes "like eat", which has the wrong probability to choose "eat" rather than "eating". Since we use the Java as the main service platform, the term and bi-gram data are stored as Json files and load into the MongoDB. The use of MongoDB can help to maintain the structure and can easily recover the data structure as HashMap in the Java.

6.2.2 Spell Error Correction

After detecting the wrong words, we need to generate the candidates of the possible right words. The algorithm used here is the Edit Distance Algorithm. This is a way of quantifying how dissimilar two words are to one another by counting the minimum number of operations required to transform. Basically, we define four ways to transform, which is to replace, insert, delete or transpose alphabet inside the words. According to statistics, 80% of misspellings have an edit distance of 1, and almost all misspellings have an edit distance of less than or equal to 2. Therefore, we use recursion to calculate the edit distance of 2, but gives a smaller weight than edit distance 1. We can then use the following Bayes' Theorem to know the calculation steps.

$$w = \operatorname*{argmax}_{x \in V} P(w|x) = \operatorname*{argmax}_{x \in V} \frac{P(x|w)P(w)}{P(x)} \tag{6}$$

$$w = \operatorname*{argmax}_{x \in V} P(x|w)P(w) \tag{7}$$

We use w to represent the candidate of true word, x to be the spelling error. Then $P(w)$ is the prior probability, $P(x|w)$ is the transition probability. Since $P(x)$ is a known value, we can safely ignore it in the equation and simplify. $P(w)$ can then be calculated by the common spelling error dataset[8], and $P(x|w)$ can be calculated by the count of words. Here, we use bi-gram to represent connection between adjacent words, which also use Bayes' Theorem. However, the multiplication may be 0, if the specific phrase is not in the bi-gram dictionary. So, we can smooth, and use log to replace. Therefore, if the possibility is 0, we can set it to $log(0.001)$ to identify as uncommon candidate. Although we use the bi-gram to predict, the uni-gram is still useful, especially when all the candidates' bi-gram scores and language model scores equal to $log(0.001)$. However, the weight of uni-gram should be lower, since it is a helper method. we use a coefficient to control its score percentage.

$$P(w) = P(w|former) * P(latter|w) \tag{8}$$

$$log(P(w)) = log(P(w|former)) + log(P(latter|w)) \tag{9}$$

By developing Spell Check feature, we can automatically correct some of the wrong spelling words and tell them about this by showing a "Did you mean xxx" notification. This feature still has the potential to improve, since we have not take other common mistakes into consideration. For example, when we calculate edit distances, we can give a higher weight of the alphabets which appear adjacent with the words on the keyboard. Given more time, we may improve and provide a better performance.

6.3 Drop-down box search algorithm

To help users quickly make searches with fewer words, we implement the query suggestion to show the potential search queries that users may be interested in. We display it in the drop-down list shown below the search bar.

6.3.1 Database-based approach

Since no logs are stored in the database when the algorithm is implemented, the full log search is abandoned and a drop-down box search algorithm based on database news headlines is chosen. First, perform natural language processing on news headlines in the database, remove stop words and other content, and leave useful key information. At the same time, we added the function of identifying bigrams and trigrams in the title, in order not to remove the possibly related phrases in the title and provide recommendations to users. After that, we built a suggestion table to store the first hundred news headlines that were processed before. When the user enters input, the front-end transmits the real-time input content to the back-end and compares it with the suggestion table, and returns the top ten suggestions with the highest similarity to the front-end as suggestions in the drop-down box.

Since this recommendation method requires a very short system response time, and the user's input changes at any time. Therefore, when the algorithm is just completed, it does not meet the requirements at all because it takes seven to eight seconds for each connection to the database. So we put the comparison function in the background, and each time we only need to compare the user input with the suggestion table to get the return value. The natural language processing of the title and the update of the suggestion table are scheduled as daily tasks. This not only improves the running speed of the system, but also ensures that different suggestions are recommended according to the daily news.

6.3.2 Search based on user history

In addition, we can also provide suggestions through the history user search queries. Currently, we have found the real query dataset from Microsoft Bing Search Engine called MS MARCO (MicroSoft MAchine Reading COmprehension)[9], and implement the first part, which is to show the suggestion based on the history query data.

$$cosine\,similarity = \frac{S_1 \cdot S_2}{|S_1||S_2|} = \frac{\sum_{i=1}^{n} S_{1i} S_{2i}}{\sqrt{\sum_{i=1}^{n} S_{1i}^2}\sqrt{\sum_{i=1}^{n} S_{2i}^2}} \tag{10}$$

As shown above, the use of this equation is based on the Euclidean dot product, and we need to convert the query and dataset into vectors. Fortunately, the sklearn library provides functions to vectorize text. Since Tfidf has better performance than just counting occurrences, we use TfidfVectorizer and we can set many hyperparameters. After experimenting with performance and latency for each parameter, we chose to remove stop words, smooth with a sublinear function, and limit the maximum number of occurrences of words. The algorithm is simple, all we have to do is calculate the cosine similarity of the given query to the dataset, so that we can output 10 suggestions. However, due to the huge amount of computation overhead of this method, and lacking the support of very powerful machines, the full realization of this method is temporarily unrealistic. Therefore, we manage to use the random function and limit the size of the input dataset to speed up. This works well and the response time is 0.3s now. Given powerful computational resources, the full realization is applicable to finish.

6.4 L2R ways to rank news

We are also looking for machine learning ways to implement the search engine. Previously, the Bert model[10] is powerful in many NLP tasks, such as question answering[11], token classification[12], multiple choices[13] and so on. However, due to the high computational overhead, it is not suitable to apply to the real-time search engine. Recently, the Sentence-BERT architecture greatly reduce the computing overhead[14]. Drawing on the framework of the twin network model, different sentences are input to two BERT models (sharing parameters), and the sentence representation vector of each sentence is obtained. At last, the final obtained sentence representation vector can be used for semantic similarity calculation. This greatly reduces the time, from 65 hours to 5 seconds when doing the calculation of 10,000 sentences.

The developers implemente a Python library **sentence_transformers**, which contains a pre-trained model based on the MS Marco Passage Reranking dataset[9]. We can use the provided encoder to encode the query and news, and then use the cross-encoder to re-rank so that we can get the best results. The cosine similarity is used inside to calculate and rank the candidate match.

The basic process is as follows. We encode news with a bi-encoder at the beginning. If a new query is entered, it will be encoded by the same bi-encoder. Then it will do a cosine similarity search to calculate and rank the candidate match. Next, the candidates will be scored by a Cross-Encoder re-ranker and return some news with the high scores to the user.

We test our code on Google Colab[6], and the performance is satisfactory. The prediction time is short, which is less than 0.3 seconds. The start of service may cost some time since it needs to load news from database, but it will not affect speed of user search afterwards. Currently, we have about 100k news in the database, and the loading takes about 5 minutes for Tesla T4. This time cost is actually still shorter than the traditional BM25 or Tfidf, which may benefit from GPU parallel computing.

The combination of BM25 and BERT-Sentence may give a better result if we use BM25 to do a preliminary screening. We may test the different performance of different combination of methods in the future. Also, the L2R method of implementing news search engines does not deploy onto our current news, since it still requires us to rent an expensive GPU server and turn it on all day long. Besides, our main structure of back-end search engine is based on Java, since it can use Spark to get features like parallel computing and distributed structure. This makes it harder for us to migrate our Python codes since Java does not many machine learning libraries. We also think about building API for Java to call and retrieve the results generated by machine learning from Python. However, considering the network overhead and time requirement of search engine, we do not implement it. Given more GPU resources and time, we may manage to deploy it in the future.

7 GUI Design

7.1 Search APP interface

The front-end framework VUE.js is used to create the web application interface for this project. Using the VUE framework can significantly accelerate the development speed and provide greater scalability to the project. Two main reasons for using the VUE framework are component-based architecture and a two-way binding system. The component-based architecture ensures that the browser only renders components into the DOM when needed and improves the re-usability of the code. The bound data is updated re-actively with the two-binding system, as are DOM objects.

The main part of the interface includes the query search for news and a result page for displaying the search result. Furthermore, we implemented a user account system to generate the view records and related news suggestions to improve the user experience within the website. The advance search feature will allow the user to filter by theme, location, and time range.

The user can search news by entering the terms or go to the map page for geographic news. To view the related news listing we provide a test account username: trump2 and password: trump2 for testing. Otherwise user can create an account via the login page. After each click of news, a pop up will appear to ask user for rating. This pop up will only appear if user has already logged in.

7.2 Data Visualisation

To represent the search results in a form that non-technical audiences can understand, we interpreted them in two sideways. Apart from traditional search engine UI design, we choose interactive choropleth map to enhance the understanding of geospatial data by audiences and offered scaled colors to demonstrate a range for the amount of clustered news. When the user hovers over the region, the name of country will be noted. If user wants to see what happened in this country, it can be clicked.

8 Deployment

There are four servers. Details are shown in below table and in figure 7.

Domain	IP address
gc.caohongchuan.top	**34.105.253.192**
gc2.caohongchuan.top	**34.246.112.10**
gc3.caohongchuan.top	**34.89.114.242**

For newsearch service, We deploy **newsearch** in **gc.caohongchuan.top**, **gc2.caohongchuan.top**, **gc3.caohongchuan.top**. And install **Nginx** on **gc.caohongchuan.top** then when requests arrived, **Nginx** will select one server to deal with the request according to load-balance as in figure **??**. The newsearch service API can be accessed through `http://gc.caohongchuan.top/swagger-ui/index.html`. I have wrote details about each API and their parameters. These main APIs are listed in figure 5.

About database, this project uses **Redis** to store middle results such as term frequency, **MongoDB** to store JSON configuration files, **MySQL** to store the news, users, and browsing history. **Redis** has been installed on **gc.caohongchuan.top**, **MongoDB** is on **gc2.caohongchuan.top** and **MySQL** is on **gc3.caohongchuan.top**. The MySQL address is *jdbc:mysql://gc3.caohongchuan.top:3306/TTDS_group7*, user is *root*, password is *!ttds2021*.

For spark, we use **gc2.caohongchuan.top** as the master node and use **gc.caohongchuan.top**, **gc2.caohongchuan.top**, and **gc3.caohongchuan.top** as worker nodes. We can submit spark tasks through *spark://gc2.caohongchuan.top:7077*. Details are shown in figure 3.

For front-end, we use `http://mighty-beyond-16569.herokuapp.com/`, the Heroku cloud service as our front-end. The main reason for choosing the Heroku as the server is mainly because of the quality of the infrustructure and the automation deployment feature they provided.

To daily collect the news and do the related abtract and nation generation, we deploy these service onto the Google Cloud with two VM instances, one for crawling news and nation generation, the other for abstract generation using Tesla T4. To reduce the cost, we use the scheduled start-up scripts to automatically turn on the instance and run the task, then it will automatically turn off when the time is off. Currently, the news crawler will start crawling at 00:00 am and last for 2 hours, and after it is finished, the nation generator will then be run. The GPU instance will start at 02:00 am, and turn off at 02:30 am since T4 is sufficient to handle about 4500 news in 30 minutes. We also record the logs on the disk so that we can track any potential errors happening during executing.

9 Evaluation and Future Work

The basic functions of news search work well in our services. Users can search related news using advanced search options. Our news search engine combines CNN, BBC and New York Times and use BM25 ranking algorithm to return news. Besides, we apply three recommend algorithms to recommend news for specified person according to his recording. What's more, our spiders crawl news each day to update into databases. So users can obtain the latest news from different publishers.

But our system also has some limitations, because we use only one Redis to store middle results, so when update BM25 value, the services will break down for several minutes. In order to reduce risks, we update it at 4AM each day. Besides, we don't have many users' information and their history, so at the beginning of system, the recommend system doesn't perform well as expect. When the number of users increases, the recommend results will be better.

In the future, we will try to combine the BM25 with BERT-Sentence to get a better result. With L2W ways to rank news, the system can be trained better when the dataset becomes larger. The use of GPU is a good way to do parallel computing, if we have more powerful resources, we will try to migrate the current computation of cosine similarity from CPU to GPU, which will definitely improve the response time. Java has provided good support for Web development, but it lacks machine learning related packages. This will be a concerning point that we need to decide on how to coordinate with two languages.

10 Individual Contributions

10.1 s2177384, Hongchuan CAO

I mainly focus on back-end development, such as parsing query, BM25 design, recommendation system and database design. I use **SpringBoot** framework to develop query servers, which provides API for front-end. In the query server, I apply suffix expression to parse advanced query string and use PriorityQueue to speed up. And send organized data to web interface. Besides, I use **Spark** framework to achieve **BM25** algorithm parallelly then store their middle-result in **Redis**. What's more, I write three recommendation systems **Collaborative Filtering (CF)**, **Content-based Recommendation** and **Hot News Recommendation**. Then store recommendations for each user in **MySQL**, making it easy for other services to obtain. I use **Quartz** to scheduler BM25 algorithm and three recommendation systems to run at 24:00 every day. Finally, I write these sections in the report that I am responsible for and deploy services on Goolge Cloud Servers. Except above, I organize my team, distribute tasks for members and coordinate requirements and data format between back-end and front-end.

10.2 s2221623, Ziang Tan

I am responsible for the back-end algorithms and help some work of the data collection. At the beginning, the news size is not large enough for search engines to show, so I attend and implement the New York Times Crawler to solve the problem. Later on, I focus on the abstraction generation and read some paper, then choose to use the pre-trained Bart model to implement it. I also involve in developing the nation name generator and searching for a good algorithm or library to get country names. After that, I put them onto the Google Cloud with 2 VM instances to run daily.

Apart from the data collection and handling work, I implement most of the additional features. I search for the accurate dataset for the spell error, and then determined to use edit distance and Bayes Theorem. Then, I turn to implement query suggestion, and I choose to use the history user queries based method to calculate the candidates of the suggestions. The implementation using history queries is inspired when I am searching for L2R ways to rank news, since the Sentence-BERT use the Microsoft dataset[9], which helps me find a large real user queries and can then further use the cosine similarity to get the final results back to users.

I am also involved in implementing the L2R way of ranking news. I find the Sentence-BERT model which has a short time delay and a python library, with good documentations. After loading our own dataset, I find the performance is satisfactory. Even if the dataset of pre-trained model is collected in 2018, the search result of ”covid vaccine” can still return a relevant result. Therefore, for the future work, I will focus on implementing L2R related features and deploy it onto the search engines.

10.3 s2230054, Timothy Shen

For this project, I initially set up the project structure on GitHub and VueJS framework to help team members organise and access a collaborative working environment. After that, I was responsible for most of the front-end features and the front-end's main pressure tester.

For the GUI part, the website's main features cover how we could assist in news searching and related news suggestions. To provide the view record, I have implemented a user login/register feature to allow the system to upload the view record to the back-end each time users submit the rating. In addition, I have applied UX knowledge to improve the user journey. For the UI part of the front-end, I have used the component library Vuetify. Using the component library can heavily reduce the development time and provide a better user experience.

10.4 s2191894, Mingyu Yue

I am mainly responsible for the backend algorithm and data collection. At the beginning of the project, I discussed and determined the data format with the classmates in the group, and established the relevant content of the database table, and constantly modified it during the testing process. At the same time, I wrote a crawler to crawl news from the CNN website. Because the news classification method of CNN website is different from that of BBC and other websites: CNN involves more categories, so compared with BBC classification, I have re-expanded some topics in CNN news and filled in the database, such as ” travel”, ”sports”, etc.

In the back-end part, I implemented the implementation of the drop-down box search algorithm based on news titles. After the news headlines are processed in natural language, all the useful information is added to the suggestion table of my newly established database, and the content input by the user is obtained in real time and searched in the suggestion table, and the top ten words or phrases with the best popularity are returned. So as to play the role of recommendation or prompt. In addition to this, at the end of the project, I completed the compilation and preparation of the entire report, and formed the final report.

10.5 s2133851, Jinman Xing

To prompt UX design and result display, I actively participated the implementation of data visualisation. After having a deep understanding of what our audiences expect and the objectives of our project, I found that news is strongly related to locations and countries and can improve the user's interaction by data visualisation to appeal general public's attention to our news dataset by introducing it through maps.

I communicated with front-end and back-end for a better API design planning and got to know what kind of statistics data I can explore and present to meet our expectations as a consistent and attractive search engine. A prototype was used before real practices so we can have a preliminary understanding of our collections. To achieve it, I used d3.js package to bring data to life by dynamic and interactive data visualisation.

10.6 s2220866, Fanwei Meng

I am mainly responsible for the data collection of the project. I built a BBC news crawler using the scrapy framework. At the beginning of the project, I provided a lot of test data for the debugging of the front and back ends of the project. Then I communicated and coordinated with the front-end and back-end managers, determined the data format and participated in the construction and modification of the database.

Due to the structure of the BBC News website, my first version of the crawler could not crawl all the news of the website. After analyzing the website structure and data flow, my second version of the crawler successfully obtained almost all the news of the website. Besides, after discussing and practicing with other members for many times, I standardized the structural information of getting news, and provided an example for the development of other crawlers in the future.

References

[1] Google, “google-pricing,” https://cloud.google.com/pricing Accessed March 6th, 2022.

[2] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, “Pegasus: Pre-training with extracted gap-sentences for abstractive summarization,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 328–11 339.

[3] sshleifer, “pegasus-xsum-version,” https://huggingface.co/google/pegasus-xsum Accessed March 6th, 2022.

[4] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019.

[5] sshleifer, “distilbart-cnn-12-6,” https://huggingface.co/sshleifer/distilbart-cnn-12-6 Accessed March 6th, 2022.

[6] Google, “Colab research,” https://colab.research.google.com/ Accessed March 6th, 2022.

[7] Mdict, “Oxford dataset,” https://mdict.org/post/oxford-9/.

[8] P. Norvig, “Spell error dataset,” http://norvig.com/ngrams/spell-errors.txt.

[9] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen *et al.*, “Ms marco: A human generated machine reading comprehension dataset,” *arXiv preprint arXiv:1611.09268*, 2016.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[11] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin, “End-to-end open-domain question answering with bertserini,” *arXiv preprint arXiv:1902.01718*, 2019.

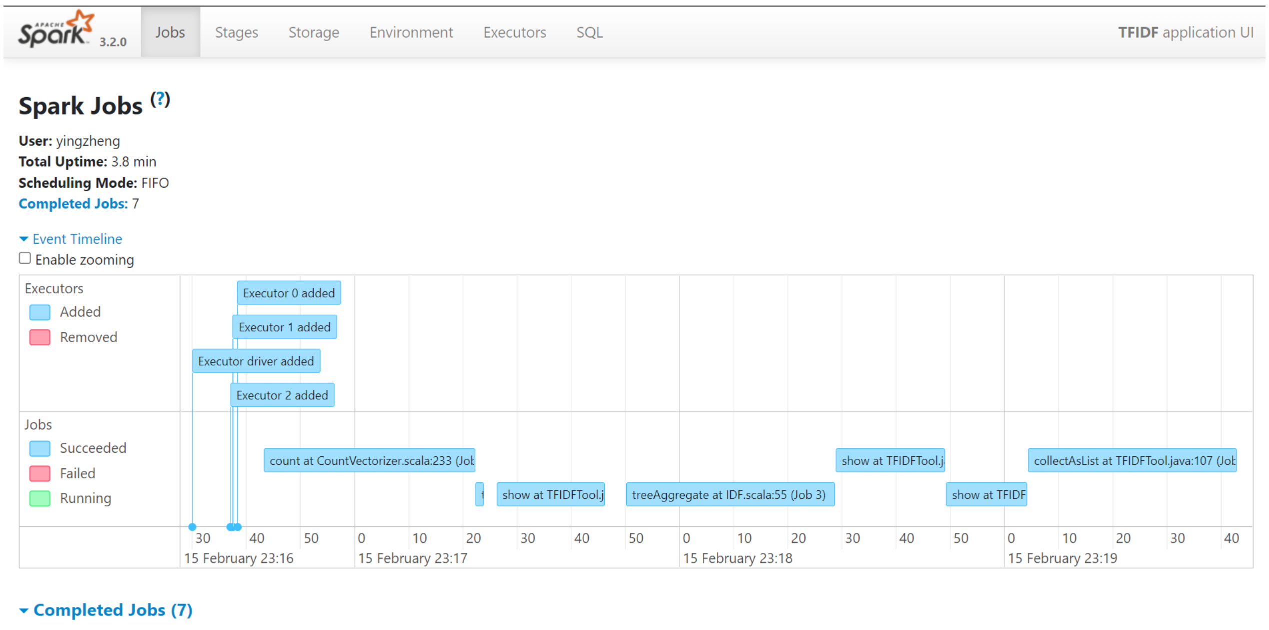
[12] S. Garg and G. Ramakrishnan, “Bae: Bert-based adversarial examples for text classification,” *arXiv preprint arXiv:2004.01970*, 2020.

[13] Z. Li, X. Ding, and T. Liu, “Story ending prediction by transferable bert,” *arXiv preprint arXiv:1905.07504*, 2019.

[14] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” *arXiv preprint arXiv:1908.10084*, 2019.

Appendices

A TF-IDF Spark



Spark Jobs (?)

User: yingzheng
Total Uptime: 3.8 min
Scheduling Mode: FIFO
Completed Jobs: 7

Figure 1: Process of TFIDF using Spark

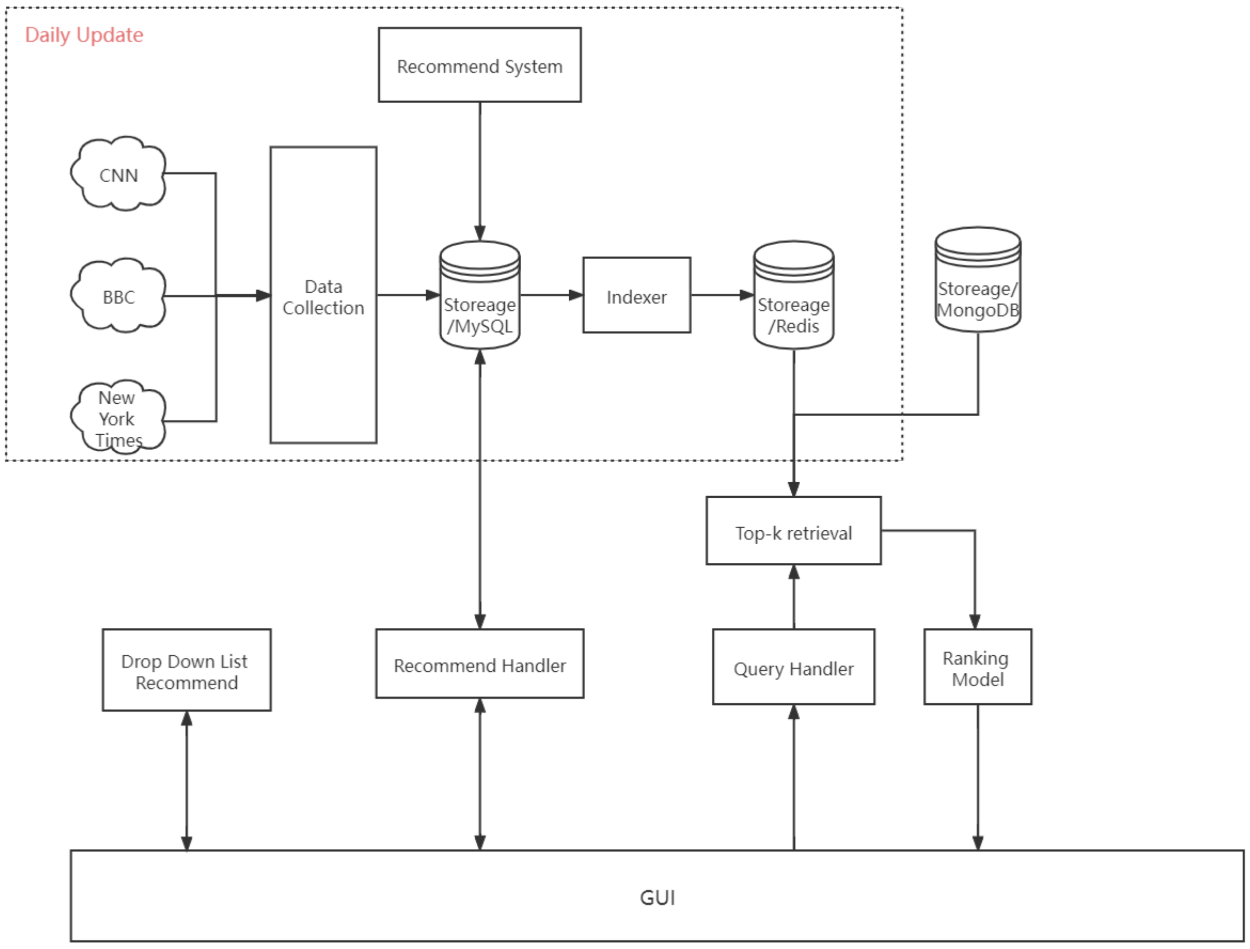


Figure 2: Project Structure

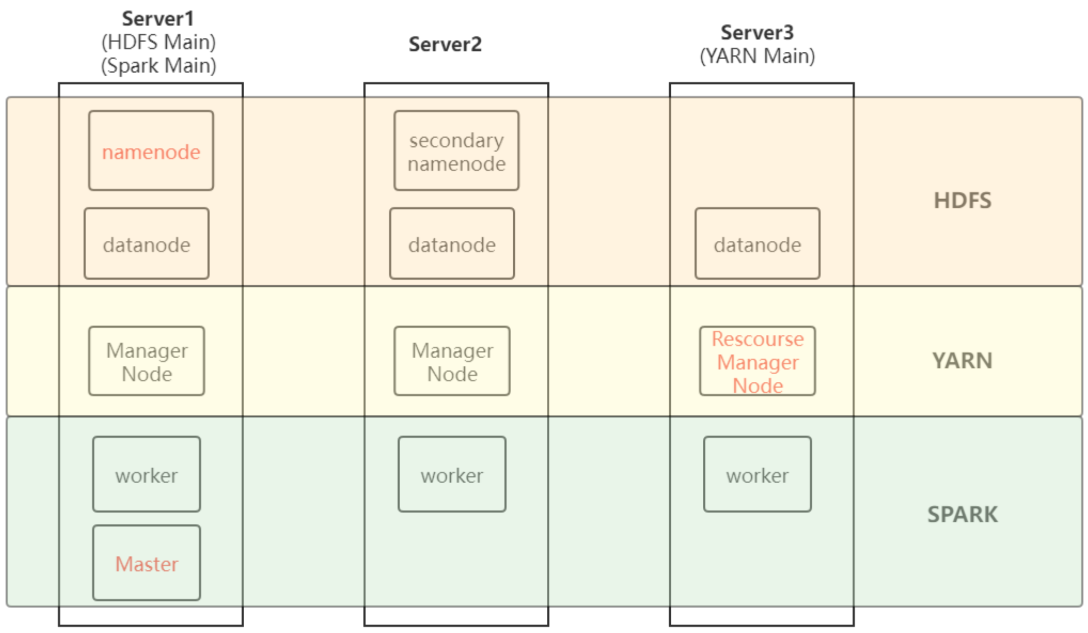
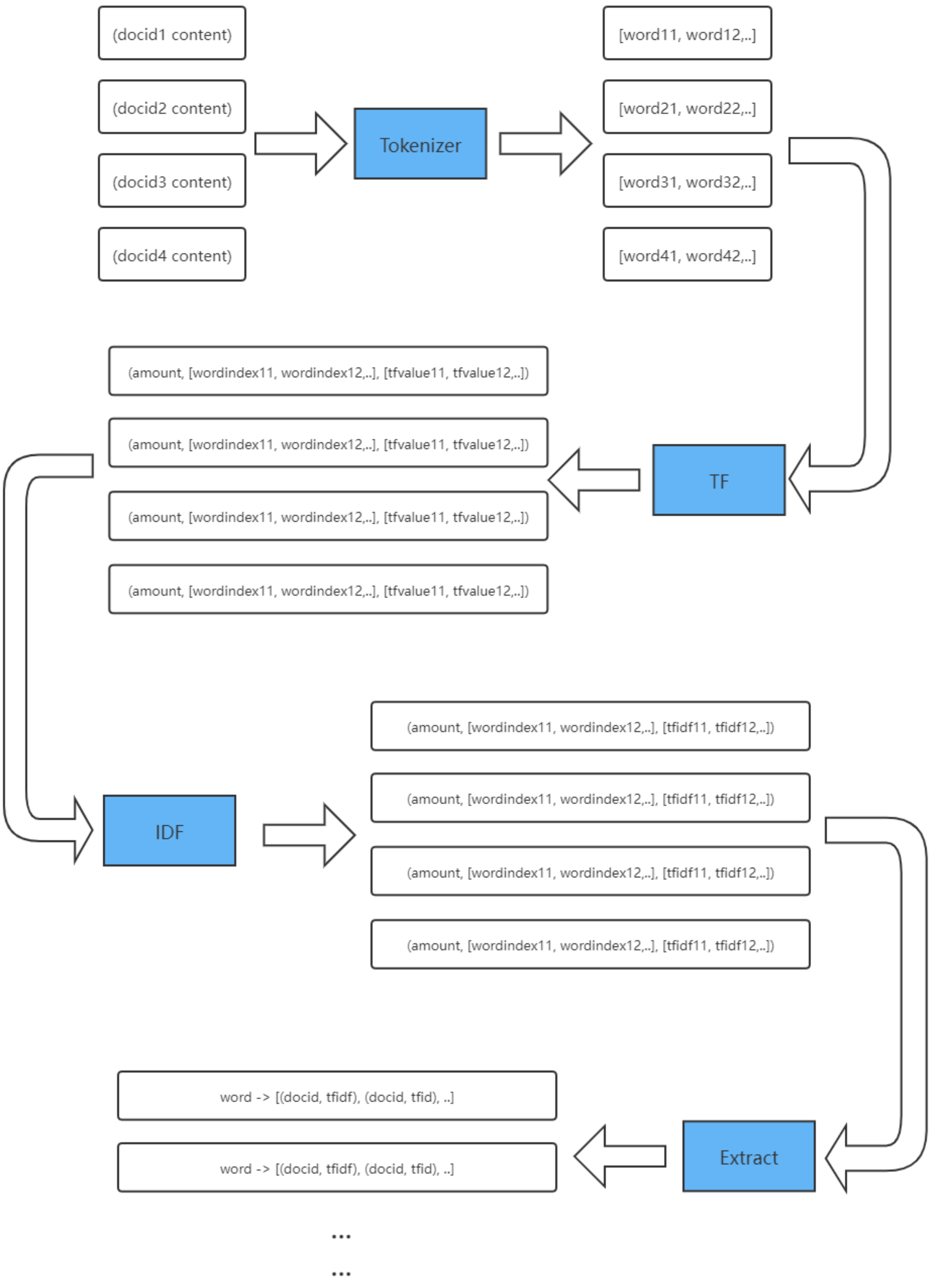
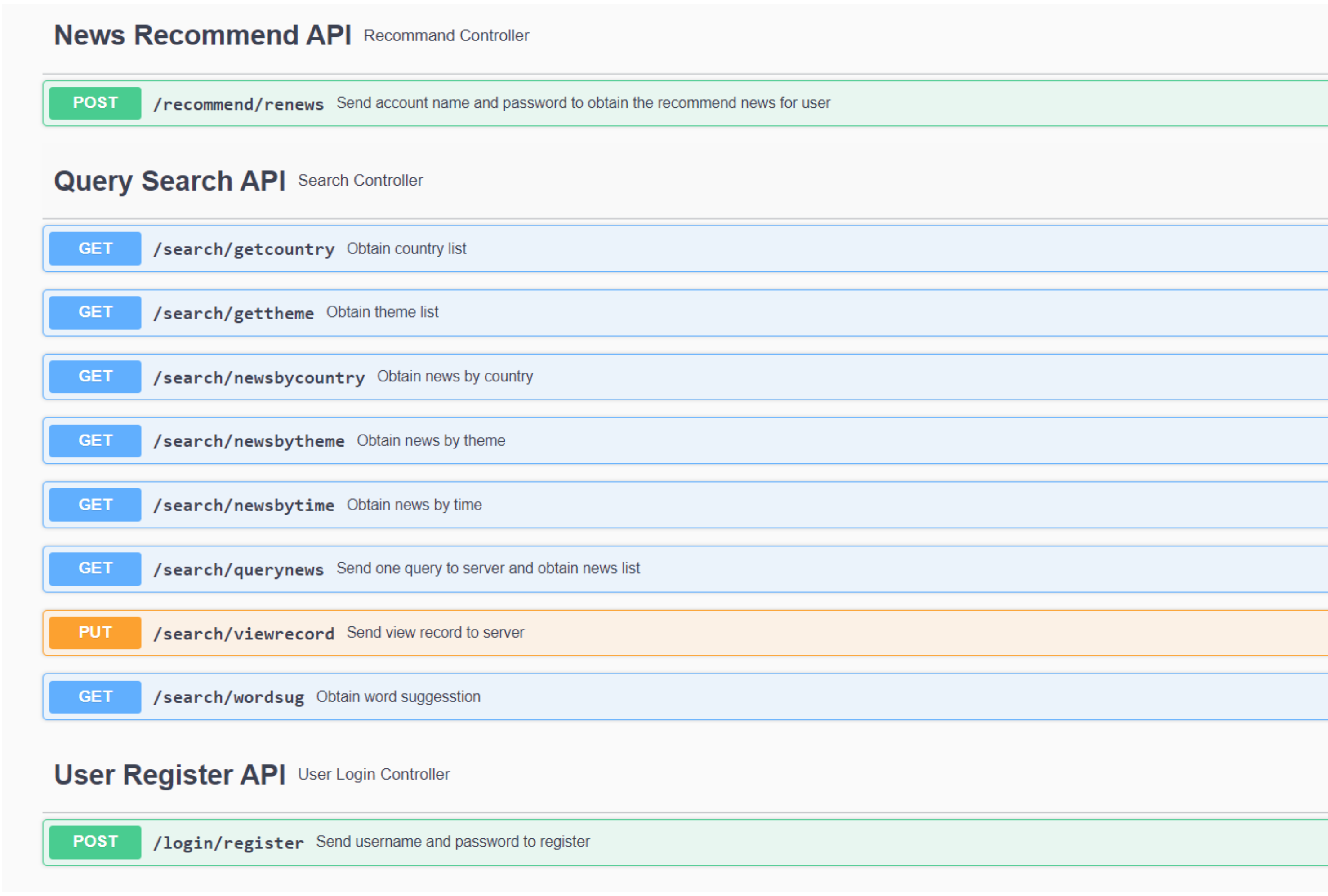


Figure 3: Spark Structure



(docid1 content)
(docid2 content)
(docid3 content)
(docid4 content)
Tokenizer
[word11, word12,..]
[word21, word22,..]
[word31, word32,..]
[word41, word42,..]
TF
(amount, [wordindex11, wordindex12,..], [tfvalue11, tfvalue12,..])
(amount, [wordindex11, wordindex12,..], [tfvalue11, tfvalue12,..])
(amount, [wordindex11, wordindex12,..], [tfvalue11, tfvalue12,..])
(amount, [wordindex11, wordindex12,..], [tfvalue11, tfvalue12,..])
IDF
(amount, [wordindex11, wordindex12,..], [tfidf11, tfidf12,..])
(amount, [wordindex11, wordindex12,..], [tfidf11, tfidf12,..])
(amount, [wordindex11, wordindex12,..], [tfidf11, tfidf12,..])
(amount, [wordindex11, wordindex12,..], [tfidf11, tfidf12,..])
Extract
word -> [(docid, tfidf), (docid, tfid), ..]
word -> [(docid, tfidf), (docid, tfid), ..]
...
...

Figure 4: TF-IDF flow



News Recommend API Recommand Controller

POST `/recommend/renews` Send account name and password to obtain the recommend news for user

Query Search API Search Controller

GET `/search/getcountry` Obtain country list

GET `/search/gettheme` Obtain theme list

GET `/search/newsbycountry` Obtain news by country

GET `/search/newsbytheme` Obtain news by theme

GET `/search/newsbytime` Obtain news by time

GET `/search/querynews` Send one query to server and obtain news list

PUT `/search/viewrecord` Send view record to server

GET `/search/wordsug` Obtain word suggestion

User Register API User Login Controller

POST `/login/register` Send username and password to register

Figure 5: Back-end API

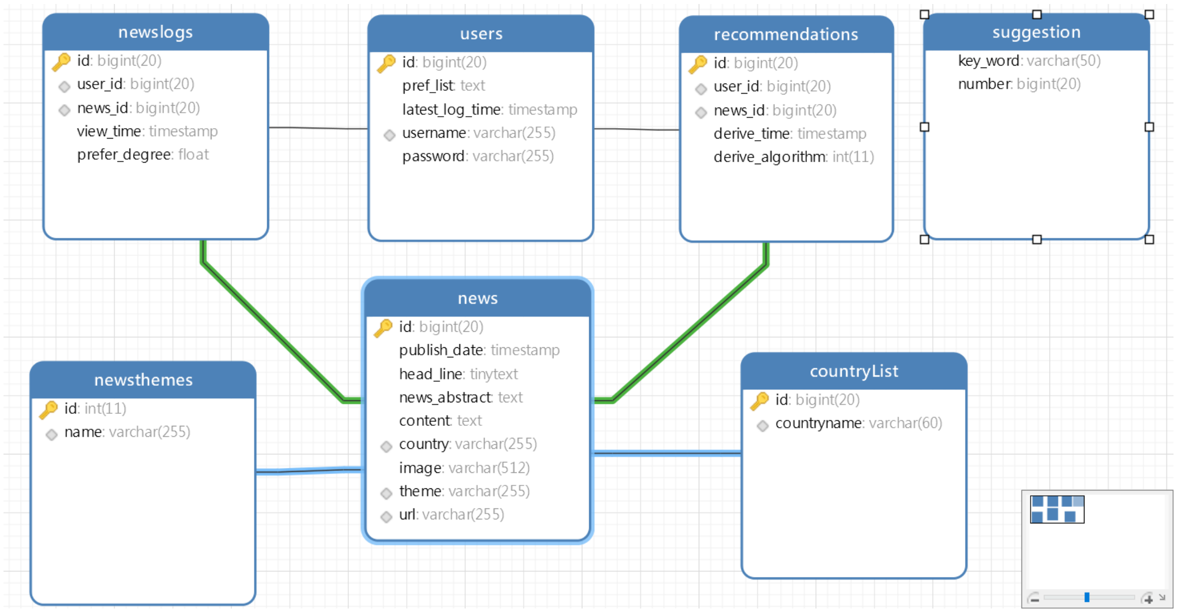


Figure 6: MySQL Relation Map

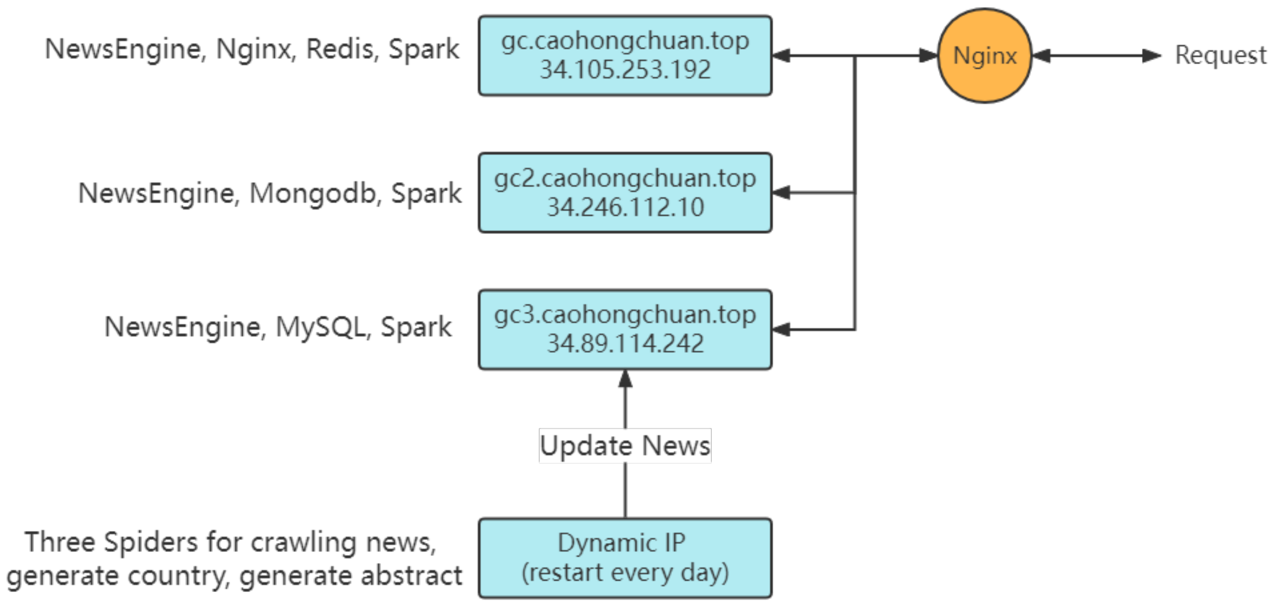
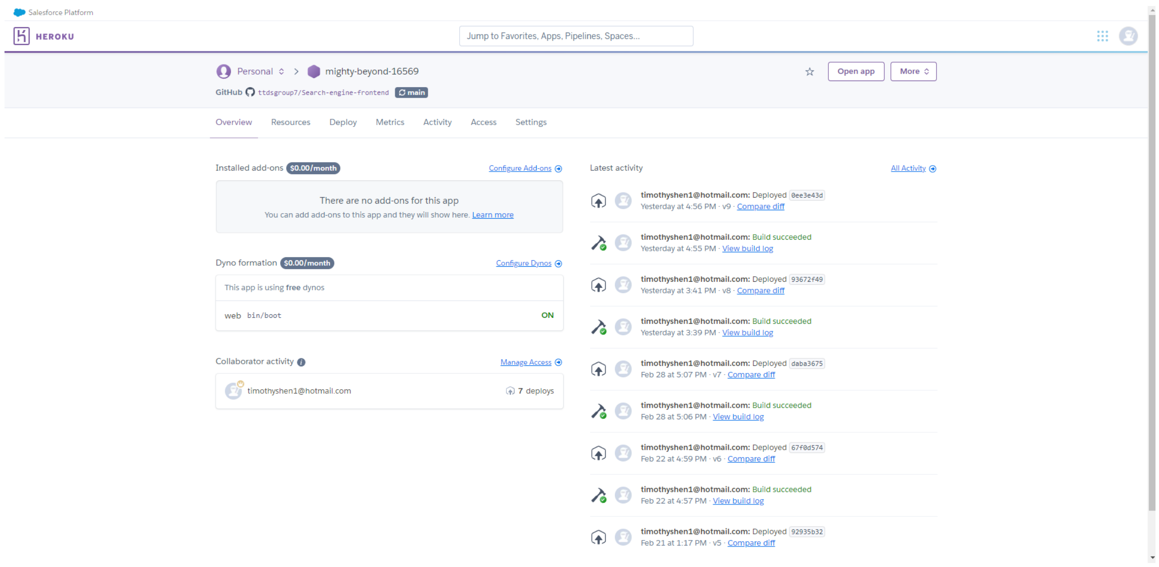


Figure 7: Deployment Information



Personal > mighty-beyond-16569

GitHub `ttdsgroup7/Search-engine-frontend` main

Installed add-ons $0.00/month — Configure Add-ons

There are no add-ons for this app

You can add add-ons to this app and they will show here. Learn more

Dyno formation $0.00/month — Configure Dynos

This app is using **free** dynos

web `bin/boot` ON

Collaborator activity — Manage Access

timothyshen1@hotmail.com 7 deploys

Latest activity — All Activity

timothyshen1@hotmail.com: Deployed `0ee3e43d`
Yesterday at 4:56 PM · v9 · Compare diff

timothyshen1@hotmail.com: Build succeeded
Yesterday at 4:55 PM · View build log

timothyshen1@hotmail.com: Deployed `93672f49`
Yesterday at 3:41 PM · v8 · Compare diff

timothyshen1@hotmail.com: Build succeeded
Yesterday at 3:39 PM · View build log

timothyshen1@hotmail.com: Deployed `daba3675`
Feb 28 at 5:07 PM · v7 · Compare diff

timothyshen1@hotmail.com: Build succeeded
Feb 28 at 5:06 PM · View build log

timothyshen1@hotmail.com: Deployed `67f0d574`
Feb 22 at 4:59 PM · v6 · Compare diff

timothyshen1@hotmail.com: Build succeeded
Feb 22 at 4:57 PM · View build log

timothyshen1@hotmail.com: Deployed `92935b32`
Feb 21 at 1:17 PM · v5 · Compare diff

Figure 8: Heroku deployment information

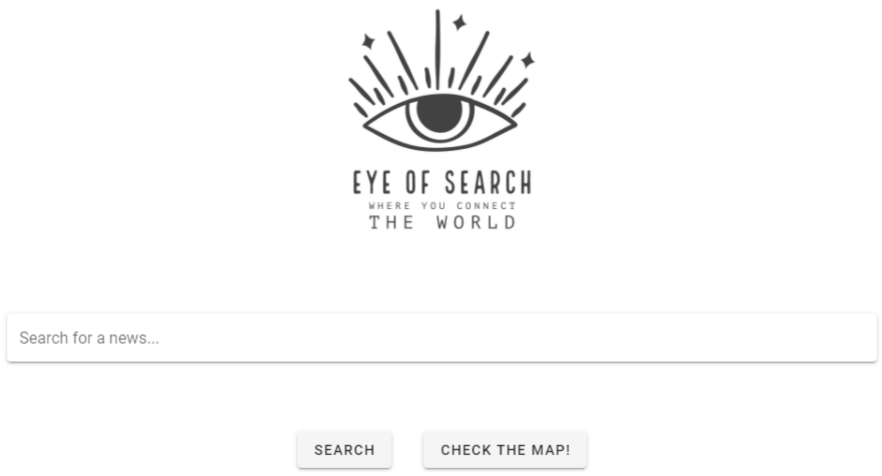
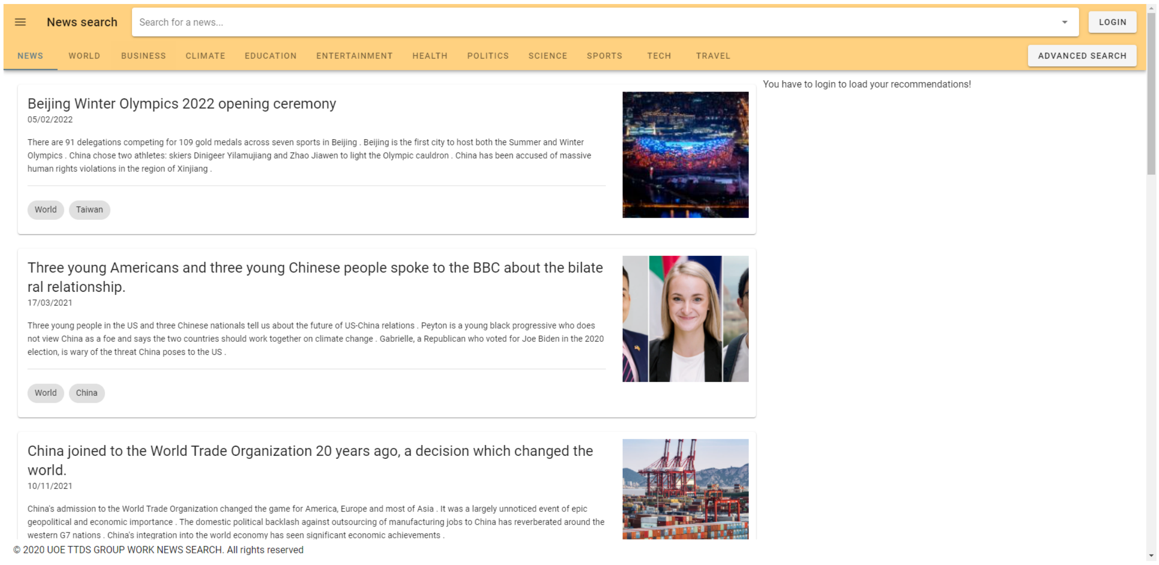


Figure 9: Search page



You have to login to load your recommendations!

Beijing Winter Olympics 2022 opening ceremony
05/02/2022

There are 91 delegations competing for 109 gold medals across seven sports in Beijing . Beijing is the first city to host both the Summer and Winter Olympics . China chose two athletes: skiers Dinigeer Yilamujiang and Zhao Jiawen to light the Olympic cauldron . China has been accused of massive human rights violations in the region of Xinjiang .

World Taiwan

Three young Americans and three young Chinese people spoke to the BBC about the bilate ral relationship.
17/03/2021

Three young people in the US and three Chinese nationals tell us about the future of US-China relations . Peyton is a young black progressive who does not view China as a foe and says the two countries should work together on climate change . Gabrielle, a Republican who voted for Joe Biden in the 2020 election, is wary of the threat China poses to the US .

World China

China joined to the World Trade Organization 20 years ago, a decision which changed the world.
10/11/2021

China's admission to the World Trade Organization changed the game for America, Europe and most of Asia . It was a largely unnoticed event of epic geopolitical and economic importance . The domestic political backlash against outsourcing of manufacturing jobs to China has reverberated around the western G7 nations . China's integration into the world economy has seen significant economic achievements .

© 2020 UOE TTDS GROUP WORK NEWS SEARCH. All rights reserved

Figure 10: Search result page

On this result page , you can press the login button on the top right corner to login. Here is a test account "trump1" & "trump1". On the left corner is the menu button where you can redirect to the map page shown below. By selecting the theme tab, user can see the related news. By pressing the advance search button, three functions will show up, country search, theme search, and date search. The user can use either date search, theme+country+OR/AND or theme/country individual search. All search supports date sort.

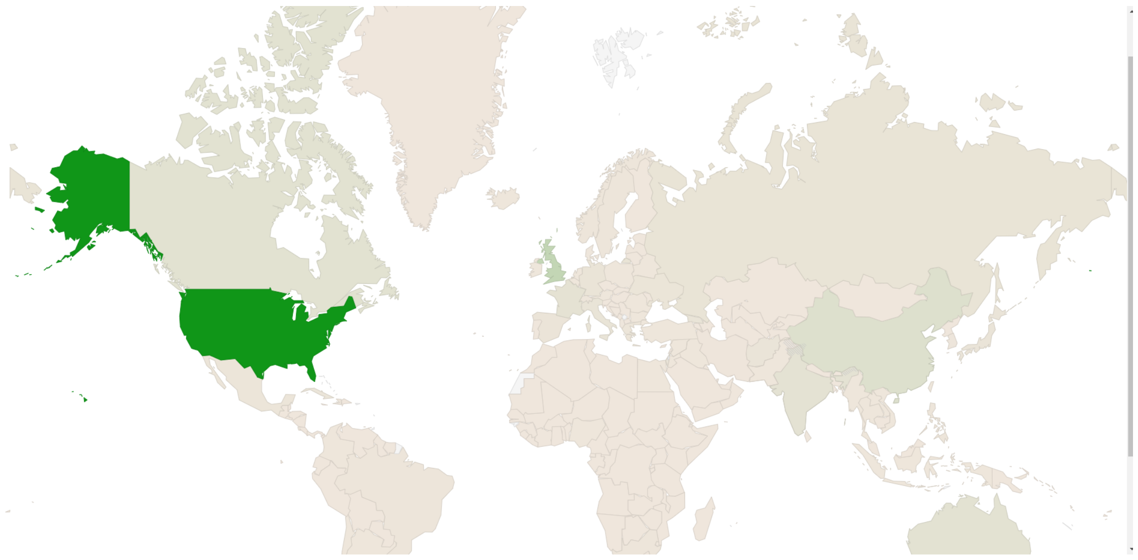


Figure 11: Visualisation map