

CECI1

SumNews
(Automatic News Summarization System)

FYP Final Report

by

CHAU Shun Wai, LEUNG Hang Kam,
LEUNG Hin Fung, TO Ming San

CECI1

Advised By

Dr. Cecilia CHAN

Submitted in partial fulfilment
of the requirements for COMP 4981
in the

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
2020-2021

Date of Submission: April 14, 2021

Table of Contents

1	Introduction	5
1.1	Overview	5
1.2	Objectives	6
1.3	Literature review	7
1.3.1	Google News.....	7
1.3.2	Feedly.....	7
1.3.3	Stand News	8
1.3.4	Nwsty	9
1.3.5	Conclusion from literature review	10
2	Methodology.....	12
2.1	Application Design	12
2.1.1	System Flow.....	12
2.1.2	User Interface.....	14
2.1.2.1	Reading news from landing page carousel	15
2.1.2.2	Reading news from the landing page latest news section.....	15
2.1.2.3	Reading news from the search page.....	16
2.1.2.4	Reading news from the saved articles page	16
2.1.2.5	Reading news from a list of related news from the news detail page.....	17
2.1.2.6	News detail page	17
2.1.3	Database.....	18
2.2	Implementation	19
2.2.1	User Interface.....	19
2.2.2	API Server.....	20
2.2.3	News Fetching And Processing	20
2.2.3.1	Web Scraper.....	21
2.2.3.2	Grouping related news articles.....	23
2.2.3.3	Multi-document extractive summarization	26
2.2.3.4	Finding the most unique sentences	28
2.2.3.5	Related news	29
2.2.4	Database.....	30
2.3	Features Implemented and Objectives	32
2.4	Testing.....	33
2.4.1	Testing the User Interface	33

2.4.2	Testing the API server	33
2.4.3	Testing of News Fetching and Processing	33
2.5	Evaluation	34
3	Discussion and Future Work	37
4	Conclusion	38
5	Project Evaluation.....	39
5.1	Division of Work	39
5.2	Gantt Chart.....	40
6	Hardware and Software Requirements	41
6.1	Hardware Requirements.....	41
6.2	Software Requirements	41
6.3	Cloud.....	41
7	References	42
8	Appendix A: Meeting Minutes	44
8.1	Minutes of the 6 th project meeting	44
9	Appendix B: Questionnaire for Evaluation	46

1 Introduction

1.1 Overview

In the information age, publishing wide-spread informative articles is no longer an exclusive privilege for traditional media enterprises like SCMP or Ming Pao. Small-to-medium-sized digital media such as Stand News are also gaining more and more popularity among younger generations. It is therefore not hard to picture a large stream of information is continuously flowing among citizens in this city.

Meanwhile, studies have shown that the production and promotion of misinformation and disinformation has never been easier for some untrustworthy media, because of the handiness provided by mobile application and the wide coverage of advertisement built on various social platforms. Together with a high degree of social differences and a tense business-oriented atmosphere, citizens apparently have been tending to be more biased to the news pieces that are delivered purposely to suit their own personal stands [1]. It hence calls for a more neutral, and convenient platform for fast information delivery in nowadays society.

In this final year project, our team implement a full-fledged news reading application that summarizes the articles according to their scopes and concerned area, in hope of giving the user a more complete view on the controversial matters without having to read from multiple news sources, and facilitating a more proper communication among stakeholders in the local community.

1.2 Objectives

The goal of this application is to help distinguishing and rating the degree of information truthfulness and trustworthiness for several popular global and local news sources, and at the same time giving users a more enjoyable experience in analysing daily news. Details are elaborated as below.

1. To aid the user on analysing news material from multiple web sources. As mentioned, in the information age, there is a need of having an efficient and effective news-summarisation app for users. To achieve this, our system is equipped with the ability of categorizing fetched news into various groups, and computing on multiple mentions of the same news event, in the target of generating concise, to-the-point and meaningful context, so that users can be presented with the complete view on each news event. In this way, the application serves as a handy tool for users to analyse news material without referring to multiple web resources.
2. To efficiently scrape and store news from multiple trustworthy news sites. As the app relies heavily on news data, it is trivial to ask for a proper way to fetch and store the raw texts of the collected news. To achieve this, our system is implemented with a fast information retrieval from news sites, for ensuring the textual computation can be done on the most updated source, and together with a proper storage mechanism of these data, for facilitating the calculation efficiency. In this way the application serves as a tool for efficient news scrapping.
3. To deliver a good user interaction experience on reading summarized news result. It is well-understood that a user-oriented application cannot gain its success without leaving room for users' interactions. To achieve this, the application is equipped with tools to allow users giving critical reviews on each of the computed news summary, and at the same time to have instant discussions and interaction with other users based on the news content. In this way a good user interaction experience can be delivered during news-reading on the application.
4. To employ basic security measure for securing application users' privacy. Anyone on a modern application site with a proper account should be able access and leave the page safely without damaging others' right on using it. To achieve this, the application is equipped with mechanisms like user registration and (re)-authentication, so as to delivering a secure and reliable experience on news reading and commenting.

1.3 Literature review

Research on apps which is related to news reading have been done, and we have selected Google News, Feedly, Stand News and Nwsty to make comparison of their advantages and disadvantages with our application.

1.3.1 Google News

Google News [2] is a platform designed for news-reading. It provides news articles for various categories, like local, world, business, sports, etc. As shown in Figure 1, it groups the articles from multiple sources belonging to the same topic and shows them as a card, so that the user need not determine if they already read similar news articles before. If the user is interested in a particular story, he/she can choose to check for more related topics like this, and vice versa. In addition, it has a fact-check section which includes articles with a deeper analysis on certain topics. Users can share and save news articles listed on the Google News page.

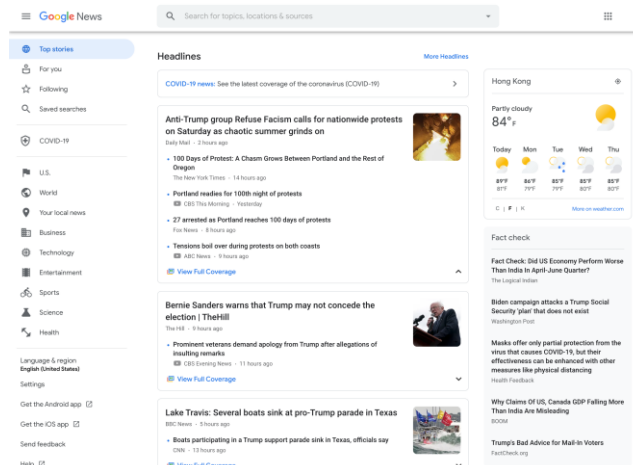


Figure 1. Google News

1.3.2 Feedly

Feedly [3] is one of the most well-known news aggregators which allows users to select the category and news sources of their choice. It offers a wide range of categories, and on each of those, an array of sources is provided. Users can subscribe from up to 100 publication sources for the free version, to possibly an unlimited number of sources for its charged premium version. It also uses RSS feeds for news fetching, so that the news can be listed on a well-designed interface, with the headline, popularity score and original source clearly shown to the application users, as illustrated in Figure 2. And from the UI, one can further tell how the application favours article sharing and bookmarking on the readers' side.

Apart from the basic functionality and well-constructed UI layout, the latest version of Feedly also equips itself with an artificial intelligent assistant called Leo for filtering duplicated information and prioritizing articles according to users' preferences, which in a way helps readers to avoid multiple site-visiting just to get some similar descriptions on the same news event.

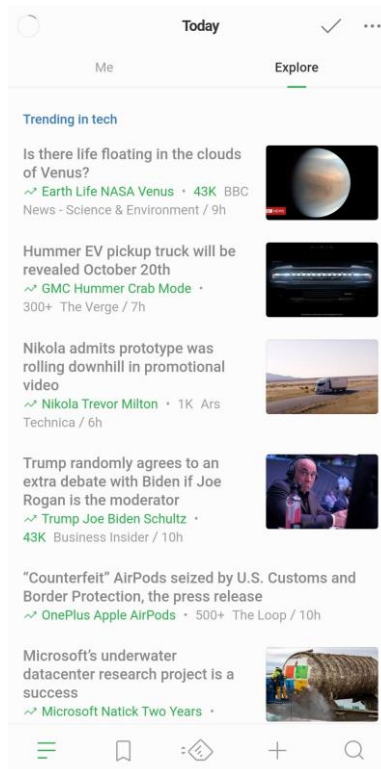


Figure 2. Feedly

1.3.3 Stand News

Stand News(Figure 2a) is one of the popular news reading applications developed by a local media going by the same name. It provides a good user experience, in a way that with the clean user interface, application user can notice which actions they can perform just from the UI layout. The user can either click into the news to read from its recommendation system, or select the category of news that the user wants and then read the related articles, or else they can go to the any bookmarked or saved page to check for the saved articles. With little actions needed, the application does bring an enjoyable reading experience to their users.

However, like other news media, the Stand News unavoidably involves its own stands and biases into its news reporting in order to exclusively serves for its target audiences, which makes it undesirable to be a direct example of a neutral news-reading application. In addition, in terms of the UI design, some appealing features like keyword-search also do not present in this application, probably due to the lack of consideration in application development, causing inconveniences to the application users.



Figure 2a. Stand News

1.3.4 Nwsty

Nwsty [4] (Figure 3) is a news digest application targeted at people who seldom spend time on news-reading. Its artificial intelligent system picks 6 to 10 out of the most interesting articles each day, among all articles scraped from predefined sources, and summarizes each in few sentences using extractive summarization. Users hence can quickly know what is happening around the globe by reading the headline and summary, and such news summary can be shared with others as well with just two clicks. In addition, users can check the original article from source to learn more about the issue if they want.

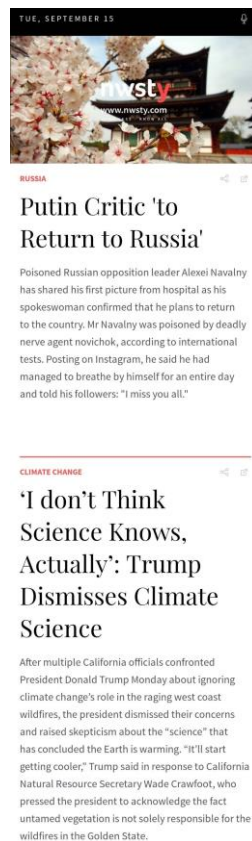


Figure 3. Nwsty

1.3.5 Conclusion from literature review

After reviewing the above applications, it is found that having a well-designed user interface is one of the key factors that could make an application stand out from its opponents. Like for the above news platforms mentioned, to inform users on features like news-source embedding and article-sharing, the application owners offer an understandable and usable layout with clear buttons and page-flow. However, these are far from enough, in a way that these popular applications mentioned are not fulfilling the four objectives we desire to achieve in this project.

Apparently popular news-reading applications like to incorporate artificial intelligence techniques for helping users to read similar news under multiple sources. For example, Google News groups articles under the same event, and Feedly rearranges the feeds' order based on users' preference. With such an approach, while arguably reducing one's time-load on intaking information, there could still be a considerable amount of time cost for normal citizens to spend on actual reading and analysing when no convenient summarizing tool is provided. In short, some news-reading applications might not be doing their best on relieving users from analysing news material on multiple web sources.

On the other hand, although there are few applications providing a summarization feature, the way they are implementing it could be unreliable. Like in Nwsty's case, it only extracts the first two sentences on each news to form a partial summary, which is clearly inadequate and in fact puts the accuracy of its summarization feature into question. In addition, having relied on a sole source on each produced summary could also be a problem, In Nwsty's model, without

the aid of proper rating and filtering mechanisms, readers could still suffer from the risk of consuming biased information. These examples unfortunately show that similar applications on the current market might not have their summarization feature implemented in the most promising way.

Having gathered the benefits and limitations on each application, we find even the most successful business products can only achieve a small part in our objectives in each of their own, hence suggesting that the idea of an efficient news-summarization application is worth to be proposed and implemented in this project.

2 Methodology

2.1 Application Design

2.1.1 System Flow

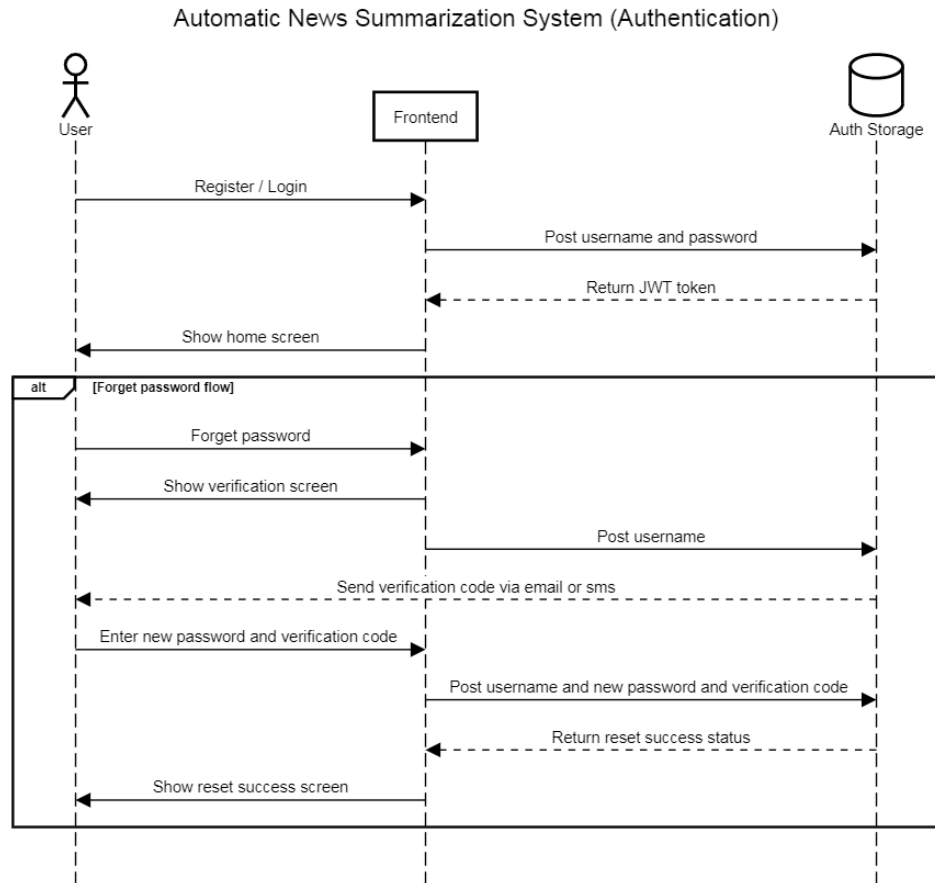


Figure 4. System Flow for User Authentication

For the purpose of user authentication, a proper mechanism as illustrated in Figure 4 is enforced. Site visitors are strictly divided into two classes,

- i. those who are going to register or have already registered and able to login with correct username and password, and
- ii. those who have already registered but forgotten password of their accounts.

For (i), we require the visitors to enter the username and password in a simple form and illustrate their login / registration intention by clicking on certain control buttons. Then their requests would be posted to the auth-storage for verification. If there is a match on the storage, in the case where visitors have registered before; or if there are none matched in the database, in the case where visitors are going through the registration process, their requests would be taken as valid and a proper json-web-token (JWT) would be issued to frontend site. And once a valid JWT is detected on the frontend side, the page would automatically bring users to the main page of the application.

And for (ii), we require visitors to indicate their intention on changing password by clicking some other control buttons so that a proper verification screen could be displayed. Then we assume the visitors can enter their username or email correctly, so that the auth-storage could locate the email account or the phone number on the intended account, and give certain one-time-passcode (OTP) by using the stored contact details. And on the visitor's side, they will then be redirected to a change-password screen, in which they would have to enter both the correct OTP and their new passwords to complete the password-changing process. Then visitors would be redirected back to the login page and be able to perform the actions mentioned in (i) for accessing the main page of the application.

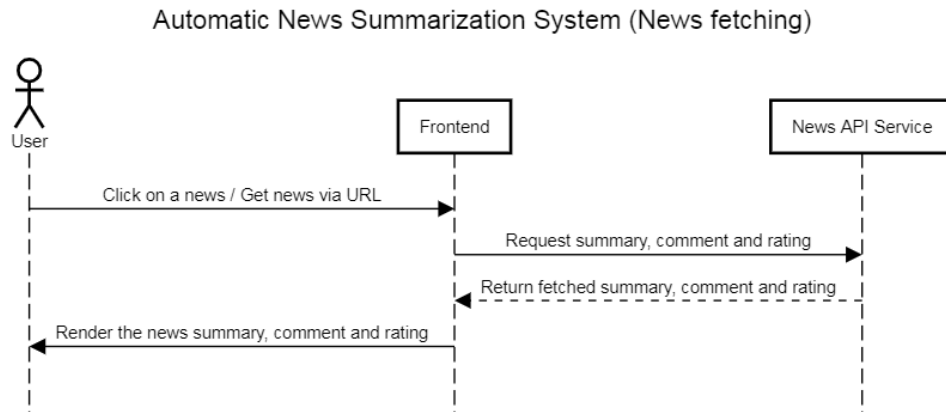


Figure 5. System Flow for News Fetching

And on the main page, users can view the summarised news via various means (with details explained in the “User Interface Design” section) through a news-fetching mechanism shown in Figure 5. After users submitting a news-access request on frontend, frontend would request the news application programming interface (news API) service for things like news summary, users’ comment and overall rating. Upon successful retrieval (in which the below “News Fetching And Processing” section would elaborate more on), those pieces of information would be rendered on frontend side and be displayed to the application users.

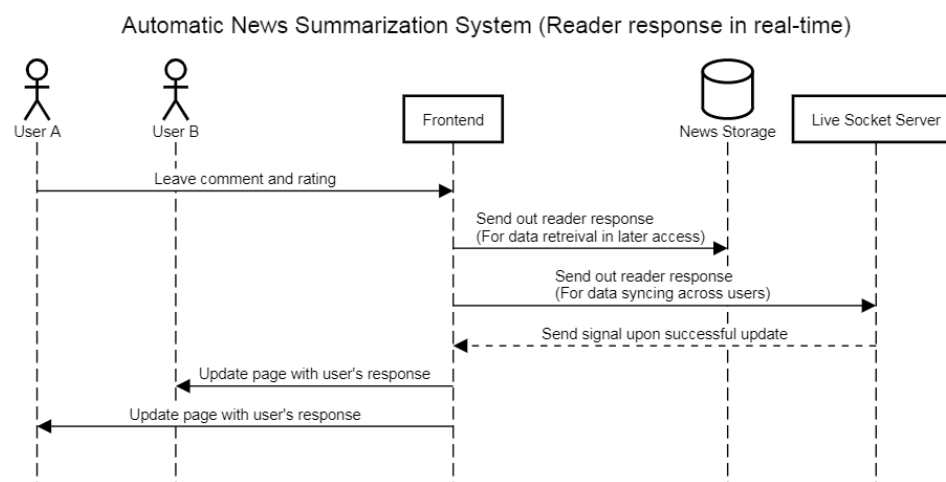


Figure 6. System Flow for Real-time User Response and Interaction

To achieve the goal of instant discussions and interaction among users on the summarised news, a mechanism as shown in Figure 6 is to be established. When one user, say user A, leave his / her review, comment and rating under certain news, that response is going to be taken to two

places. The first one is the news database, the one the behind the news-fetching API service, so that such reader's response can be retrieved upon next successful fetching on news summary. And the second one is a live socket server, in which it would be responsible to send out a specific signal about such comment changes to other users B, C, D, ..., so that the frontend applications on different devices could be informed and perform certain re-fetching or re-rendering to achieve the goal of real-time discussions and interaction among application users.

2.1.2 User Interface

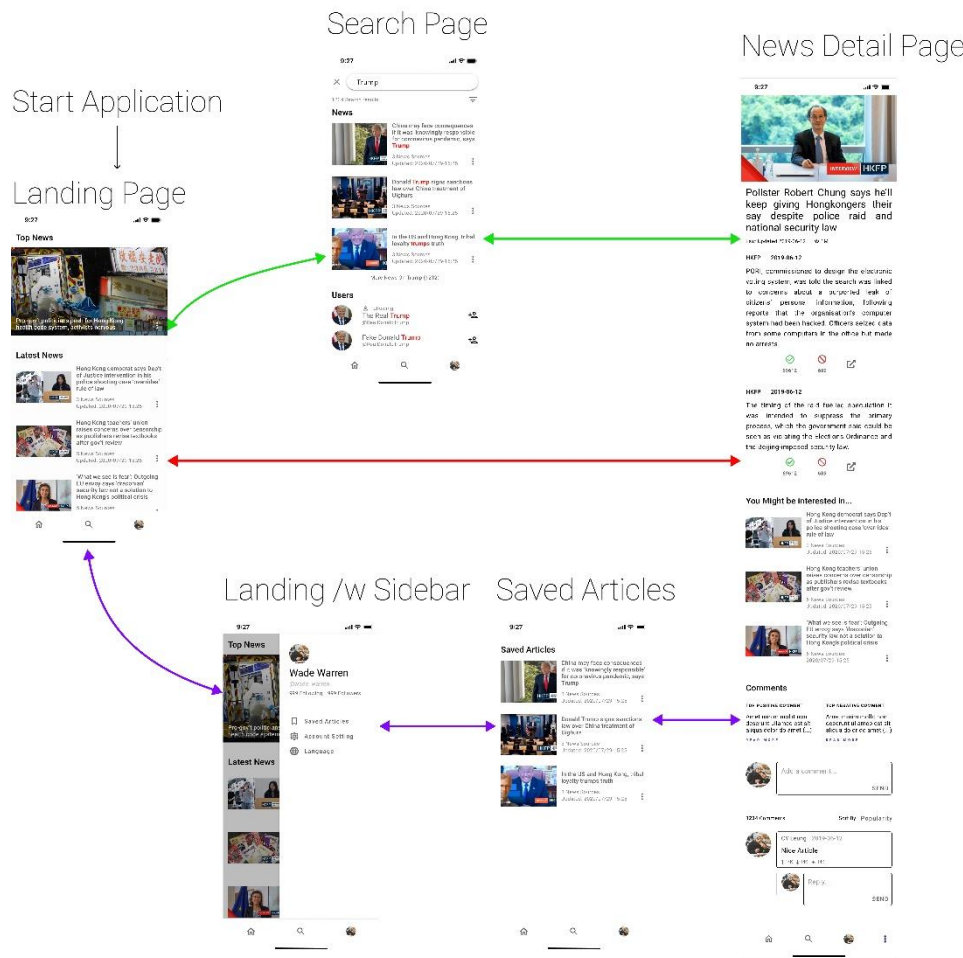


Figure 7. The Layout of the application

To achieve the aforementioned system flow, a brief user interface, which is shown in Figure 7, has been designed in Figma [5]. Figma is used because its free version has provided all of the functions we need, including collaborative usage, event-based prototypes, and a variety of plugins.

From a user perspective, there are 5 ways for users to read the articles: from landing page carousel, landing page latest news section, the search page, the saved articles page, and the list of related news from the news detail page. These 5 ways will be explained in detail in the following sections.

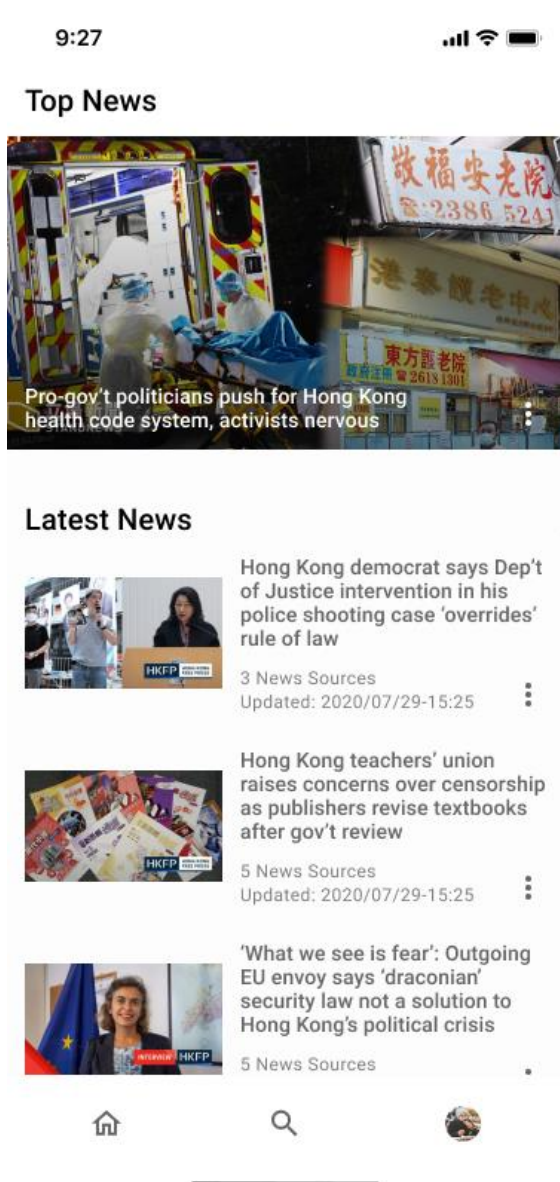


Figure 8. Landing page carousel (top) and list of latest news (bottom)

minute timeframe.

Near the news title, there is a list button with 3 dots placed vertically that the users can click and a menu will expand afterward. The menu contains a 'Save article' button that the users can click to save the article for later read and a share button that allows users to share the article in the form of URLs through social media, such as Telegram, WhatsApp, Instagram direct message, and so on. More buttons relating to articles might be added in the future.

2.1.2.1 Reading news from landing page carousel

When the user launches the application on the smartphone, the landing page as shown in figure 8 will be shown.

On the landing page, there is a list of the most trending news fit in a carousel that allows the users to slide to the left or to the right with finger gestures to switch to the previous or the next most trending news. The application will redirect users to the news detail page when users have clicked on the carousel container. The algorithm of trending news will have to be decided later.

2.1.2.2 Reading news from the landing page latest news section

There is also a list of the latest news underneath the trending news carousel (under the Latest News heading from figure 8). Users can read the articles by clicking on the news title or the thumbnail.

To enhance the performance, the news fetched are cached in the application front-end for 1 minute or until the users forcefully refresh the application. That is, even after the users have been redirected to other pages, for example, the search page, and come back, the news shown on the latest news list should still be the same as the front-end will not re-fetch the news in the 1-

2.1.2.3 Reading news from the search page

Next, at the bottom of every page, there is a bottom navigation bar. Once the users click the magnifier icon, they will be redirected to the search page as shown in figure 9.

Users can type in the keywords in the search box on the top of the search page to search for either the matching news articles or users. If the users click the title of the searched news, the application will once again redirect the users to the news details page.

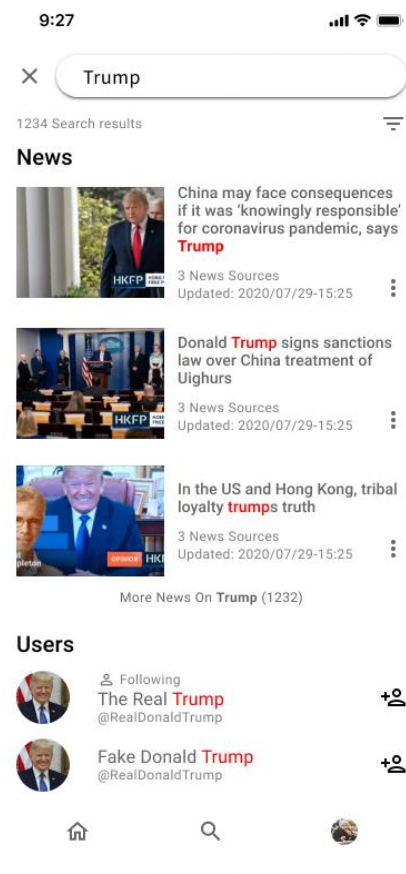
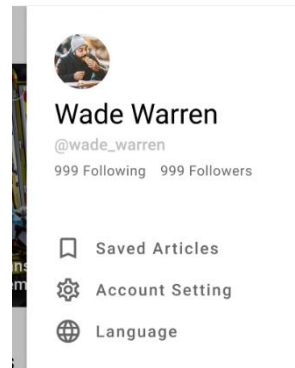


Figure 9. Search page



2.1.2.4 Reading news from the saved articles page

Furthermore, if the users click their profile picture from the right of the bottom navigation bar, a sidebar drawer will pop up as indicated in figure 10.

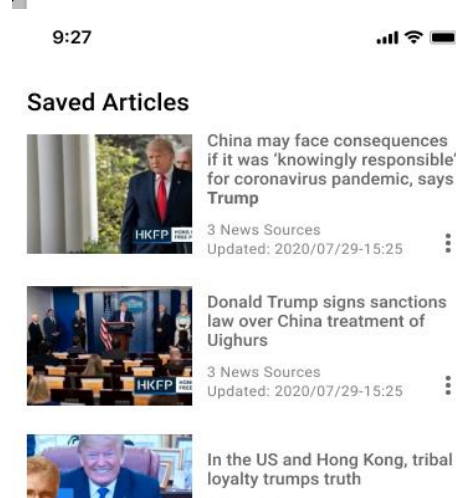


Figure 10. Sidebar drawer (top) and saved articles page (bottom)

The sidebar consists of a list of menu items, including a saved articles button, a language setting button, and an account setting button. Users can click the saved articles button to go to the saved articles page as illustrated in figure 10 and retrieved articles they have saved. Once the users clicked the title of the saved article, they arrive at the news detail page.

2.1.2.5 Reading news from a list of related news from the news detail page

On each of the news detail page, there will be a list of news the users might be interested in as shown in figure 11. By clicking the news titles from the ‘You might be interested in...’ list, users will be redirected to the dedicated news detailed page.

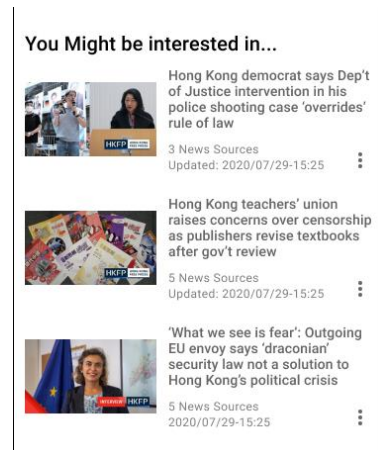


Figure 11. List of related news from the news detail page

2.1.2.6 News detail page

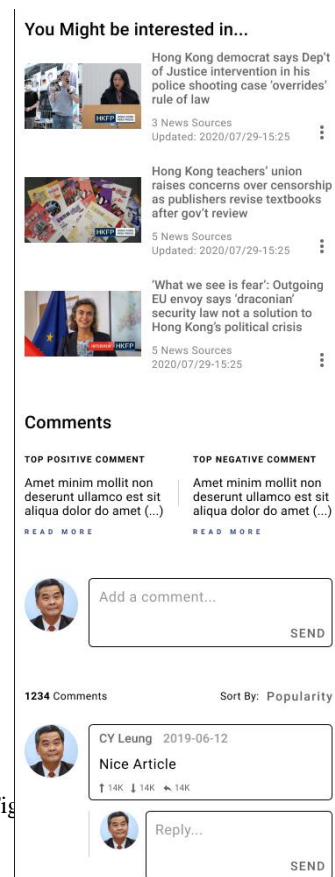


Figure 12 Top of the news detail page

By following any one of the methods mentioned above, users should be able to arrive at the news detail page as shown in figure 12 and figure 13. The news detail page is the heart of this application for it is the place where users read the summaries of one incident or one event across multiple news sources.

Starting from the top to the bottom of the page, there are the feature images from different articles, the summarized title, some metadata including the view count and the time of the last update, the list of unique content from different articles, a list of related news, and the comments section.

Users can click on the redirect icon below the summary text to go to the original article and read the full text from the source web site. Users can also rate the trustworthiness of the news source by clicking the green or red button to indicate that the news source is trustworthy or not. In Figure 13, users can leave comments for having discussion with other users as well and the comments can be rated by the google API to see which comment has the most positive attitude or negative for that news topic.



Fig

2.1.3 Database

The database consists of four main collections, namely **Article**, **NewsGroup**, **Comment** and **User** as well as 2 relationship tables which are **SavedArticleRelationship** and **ViewedArticleRelationship** respectively. The detailed schema for each collection is shown in Figure 14.

user	article	savedArticleRelationship	viewedArticleRelationship	newsGroup	comment
userID Integer	articleID Integer	savedArticleID Integer	viewedArticleID Integer	newsGroupID Integer	userID Integer
userName varchar(1024)	title varchar(1024)	articleID Integer	articleID Integer	articleID Integer	articleID Integer
icon blob	articleBody text	userID Integer	userID Integer	commentsID Integer	commentID Integer
followersID int	articleDate date			summary text	commentBody varchar(1284)
	url varchar(1024)			photos blob	createDate timestamp
	photos varchar			vector float	score double
	source varchar(1024)			relatedNewsGroupID Integer	magnitude double
	upvotes Integer			viewCount Integer	
	downvotes Integer			MostPositiveCommentID Integer	
	summary text			MostNegativeCommentID Integer	
	isGrouped boolean				

Figure 14. Database schema

Article stores the information of each news article. Apart from the basic information like title, text, etc, user's votes on the article is also stored in **Upvotes** and **Downvotes**. **Summary** is the per-document summary which is unique among other articles. And the **isGrouped** attributed is for determining whether an article has been grouped with other articles of the same topic.

NewsGroup stores the information of a group of similar articles, including the id of the articles in this group, the summary this group, and the number of views of this group. The **ArticleIDs** and **CommentIDs** will be used to find the articles and comments of this **NewsGroup**, which will be populated into the response data before returning from the server.

Comment stores the date of creation, text of the comment and id of the user who made the comment. In addition, the comment will have 2 new properties which are **score** and **magnitude** respectively. They are used to determine if the comment is a positive or a negative comment and the length of the description respectively. The user id will be used for finding who made the comment, so the username and profile picture can be populated to be shown in the front-end side.

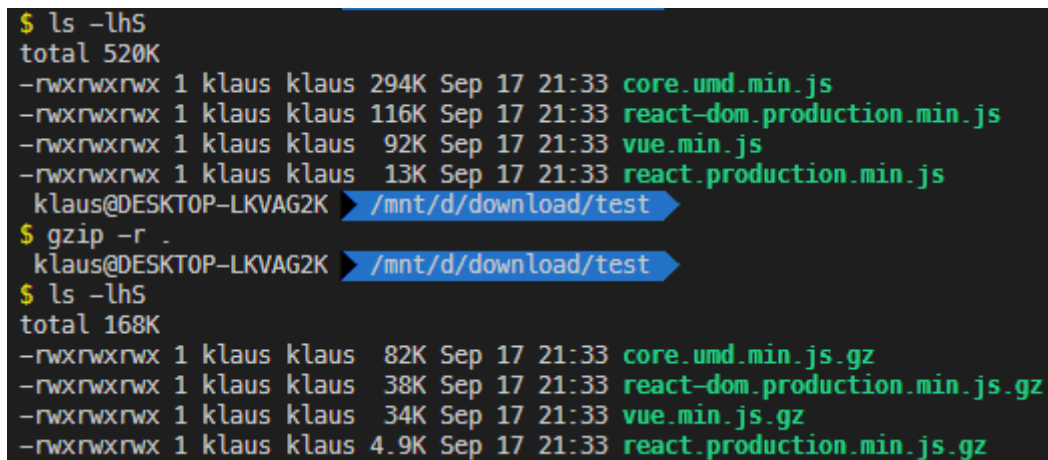
User stores the basic information of the user, such as username, profile picture, as well as id of his/her followers, saved articles, viewed articles and comments.

SavedArticleRelationship stores the information of the **articleID** and the user who saved that piece of article and this establish the relationship between user and saved article.

ViewedArticleRelationship stores the information of the **articleID** and the user who has read that piece of article and this establish the relationship between user and viewed article.

2.2 Implementation

2.2.1 User Interface

A terminal window showing the process of zipping JavaScript bundle files. The first command is 'ls -lhS' which lists files: core.umd.min.js (294K), react-dom.production.min.js (116K), vue.min.js (92K), and react.production.min.js (13K). The second command is 'gzip -r .' which zips these files. The final 'ls -lhS' command shows the zipped files: core.umd.min.js.gz (82K), react-dom.production.min.js.gz (38K), vue.min.js.gz (34K), and react.production.min.js.gz (4.9K).

```
$ ls -lhS
total 520K
-rwxrwxrwx 1 klaus klaus 294K Sep 17 21:33 core.umd.min.js
-rwxrwxrwx 1 klaus klaus 116K Sep 17 21:33 react-dom.production.min.js
-rwxrwxrwx 1 klaus klaus 92K Sep 17 21:33 vue.min.js
-rwxrwxrwx 1 klaus klaus 13K Sep 17 21:33 react.production.min.js
klaus@DESKTOP-LKVAG2K /mnt/d/download/test
$ gzip -r .
klaus@DESKTOP-LKVAG2K /mnt/d/download/test
$ ls -lhS
total 168K
-rwxrwxrwx 1 klaus klaus 82K Sep 17 21:33 core.umd.min.js.gz
-rwxrwxrwx 1 klaus klaus 38K Sep 17 21:33 react-dom.production.min.js.gz
-rwxrwxrwx 1 klaus klaus 34K Sep 17 21:33 vue.min.js.gz
-rwxrwxrwx 1 klaus klaus 4.9K Sep 17 21:33 react.production.min.js.gz
```

Figure 15. Bundle sizes of Vue.js, Angular.js, React.js + React-DOM

For design simplicity and application modularity, different UI components as described in previous front-end sections are going to be mainly implemented by Vue.js. As comparing to other popular modern frameworks out there like Angular.js and React.js, Vue.js gives a smaller package-bundle size (minified + gzipped result as shown in Figure 15) and thus a lower runtime on browser workload, in a way that can better optimize a small-to-medium-sized application like ours. Besides, it offers an accessible set of HTML-like language syntaxes and a collection of easy-to-follow application lifecycle concepts (mounted, destroyed, etc.), implying a less extreme learning curves on the developers' side and a larger flexibility. And unlike others front end libraries such as React.js which outsource some of its tools' development, developer of Vue.js also helps to establish and maintain the tools that might be needed during the development process (like those for page-routing and state-management), which makes it less vulnerable to bugs or security issues as the versions of the tools could almost always catch up with its latest official version. For these reasons Vue.js is chosen here.

With a chosen framework one could still need external supports from the popular libraries contributed by the community. In our case, Node.js is notably one of the most common library support mechanisms that we could adopt. Being a large collection of open-source libraries (commonly referred as “node-modules”), it has been refined and strengthened for over a decade, in which answers to many common-asked questions and bugs have already been given as compared with its less popular opponent Denos. Thus, Node.js is considered a plausible project dependency collection to be adopted.

Meanwhile, developers often incorporate a package-manager into Node.js-related projects, for the purpose of efficient editing and updating on the project dependencies. In our case, the manager program called Yarn is going to be adopted, for its well-received recognition and great bug-fix support from the community. Also, it is available on a number of popular operating systems, including Windows and macOS, which favours the implementation process on front-end side. Thus, yarn is selected for dependency management.

2.2.2 API Server

To achieve what we have designed in the user-interaction system flow (Figure 4), we need an application programming interface (API) server to assist frontend side on instant page updating. With Node.js's support, we develop RESTful APIs on top of the framework Express.js. To facilitate the communication between frontend and backend. In particular we establish dynamic routing based on HTTP protocol for accomplishing different tasks.

- “/register” and “/login” are end-points for registration, which handle authentication-related post-requests from client-side.
- “/news”, “/news/:news_id”, “news/deleteall” are endpoints that handle certain GET and POST and DELETE requests for the article databases
- Similarly “/comments”, “/comments/:news_id” are endpoints that take care request from client side for the comment data from users.

With sufficient server's support, we can now cover the mechanism on the instant page-updating feature. In fact, such feature can be achieved by libraries like SocketIO [6]. According to the documentation, SocketIO allows real-time and bidirectional communication between the browser and the server to be done gracefully, meaning application users on several local browsers can interact with each other through a pre-designed server, which well suits our purpose of including users' communication into the application. Moreover, SocketIO is famous for its event-driven aspect, which makes it compatible with the traditional JavaScript event flow and hence sufficient to be a strong candidate for our user-based feature. Thus, SocketIO is selected here.

2.2.3 News Fetching And Processing

This section presents the implantation of news fetching and processing pipeline. News are first fetched from various sources, which will then be grouped according to event. Next, multi-document summarization will be performed and unique sentences from each article will be extracted. Related news will then be computed. The whole set of procedure is set to be repeated every 45 minutes, for ensuring the newness of the news information. Figure 16 shows the flowchart for the pipeline.

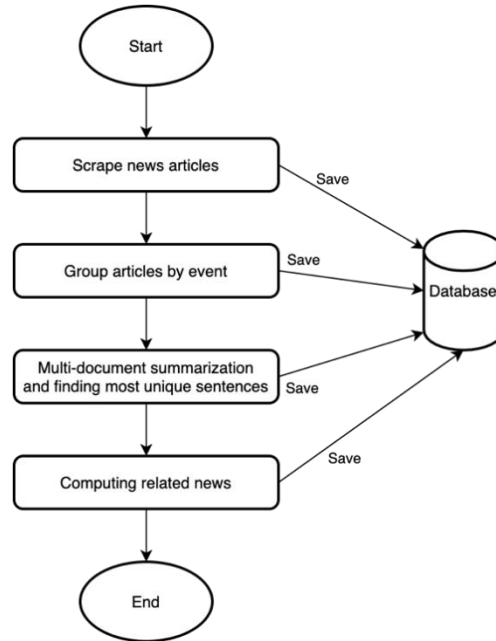


Figure 16. Flowchart of news fetching and processing pipeline

2.2.3.1 Web Scraper

For the purpose of efficiently collecting news data from various sources, and facilitating noise pre-filtering for subsequent computation, a sophisticated web scraping routine is implemented to retrieve news data for subsequent processes. Such routine contains three main steps, which would be explained one-by-one.

The first step is to retrieve existing article urls, so that we could have a base of comparison and could make sure the routine would not process the same article upon iterations over time. Currently since all the processed news are stored as a large json in a fixed api-endpoint, so a simple GET request is the quickest solution at this point. From such request, we get back our computed article data, and with python's efficient list processing, their urls can then be easily collected.

The second step is to scrape a new set of articles' urls based on predefined news sources. It can be further decomposed several sub-steps, namely the selection mechanism on various news sources, and the method on generating news articles urls, and the approach on filtering.

- I. We currently put four local (SCMP, RTHK news, The Standard, Hong Kong Free Press), and six foreign brands (BBC, Reuters, Fox News, Guardian News, USA TODAY, NBC) into our concerns, so as to make sure the news that would be presented in the frontend news sites are from trustworthy media. One thing to note is, the routine here is in no way confined to these pre-selected brands. As shown in later texts, it could be easily extended to more than 10 modern news sites, given sufficient computation time and resources.
- II. For obtaining news articles urls, multiple approaches are applied with an aim to derive the most completed list of url data.
 - a. For each news source (represented by the url of its main landing page), we first try with the third-party module in python called newspaper3k, in which it

contains a sophisticated logic handling on HTML and HTTP network specialized on news webpage for generating our desired url data.

- b. However, by manual testing, we find the library not reliable on some local news sites (e.g. Hong Kong Free Press). Then we in turn fall back to the conventional python requests module for performing simple GET requests to obtain html. By assuming necessary consistent structure of these html script upon each scraping, we can use the python bs4 module to understand and analysis the script structure and obtain the article urls we want.
 - c. In some “extreme” (yet not uncommon) cases, the above approaches may still fail, since the news page can be not as “static” as expected, in a sense that one single GET request would not be enough for our purpose. To handle such dynamic webpages, we would turn to a popular python module called selenium, which works by simulating a modern browser behaviour on dynamic webpage loading.
- III. It is worth to mention that not every url obtained in the previous step are appropriate to be processes in later steps; The content contained in these urls could be non-textual based (like in video format), or the content can sometimes be considered as “impure” news material (say, some can be structured as opinion pieces), or the url can simply be repeated to the existing one. For these reasons, we perform further filtering, with the result in previous step and a regular expression checking strategy, to see if the article url is concerning itself with special patterns that make it not a valid candidate to be analysed.

The third step is to obtain and process the news data contained in each valid url. For each url, we scrape some useful content (such as article’s title, date, texts, etc) by the python newspaper3k module, but again some information obtained could contain “noise” might not be accurate. Sources of noises may include,

- Hyperlink of advertisements or other articles (in article’s text attribute)
- Description texts from various charts or images (in article’s text attribute)
- Urls of company logos or other related assets (in article’s photo attribute)

Moreover, sometimes the date attribute may not be correctly scraped (probably due to various date representation format that the external newspaper3k module could not recognise). To handle for all these marginal cases, we perform manual inspection on the article’s raw html, and introduce a hopefully general and consistent fix to the data.

- For reducing the article’s text noisiness, with the use of the python bs4 module, we collect texts in paragraph tags in those html (possibly within a special div) and exclude texts in tags with specific class name(s) that we do not want, instead of completely relying on results from the newspaper3k module.
- For handling photos information, a similar approach is adopted (by assuming article’s photos are usually placed with the article’s main texts).

After these pre-processing, a less noisy news data would be passed to the next steps with further handling.

2.2.3.2 Grouping related news articles

The scraped articles shall now be grouped to form NewsGroups. Each NewsGroup is composed of at least two articles reporting the same event.

To determine whether two articles should belong to the same group, a binary classification model is trained to determine if two articles talk about the same topic (class 1), or they are not about the same topic (class 0). This method is chosen over unsupervised clustering algorithms; As time goes by, there are more and more articles adding to the database. Hence if an unsupervised approach is adopted, existing groups of articles could potentially be scattered and clustered into different groups in later runs, but then it would certainly break the consistency of news story and summary on the readers' point of views. On the other hand, our proposed model does not incur such issue, and it also favours the prediction of output of the model and evaluation of the target achievement.

To train the model, 93 and 51 NewsGroups are scraped to be the training set and test set respectively. Each NewsGroup consists of 2 to 6 articles from sources such as The Guardian, Fox News, and Reuters.

Each article is then pre-processed by removing stop words in the title and the paragraphs. In addition, it is found that out of the 229 articles in the training set, 65 of them (or 28.4%) contains one of the three abbreviations: "COVID", "COVID19" or "COVID-19". So, it is worth adding a special pre-processing rule during this pandemic to replace them with the word "coronavirus". Once this rule is applied, empirically the F1-score of the model increases by around 4%. And since the word "coronavirus" has already existed long time ago, it is more likely than the other three abbreviations, that the pretrained word2vec models contain its vector. Hence, "coronavirus" is chosen to be the replacement.

The next step is to turn the text data into features. For the titles, they will be passed to the spaCy [7] library to generate two word2vec vectors, and the cosine similarity between the vectors will be calculated, which will form the first feature — `title_cosine_similarity`. For the paragraphs, they are passed to scikit-learn's `CountVectorizer` [8], which maps the number of occurrences of the words into vectors. These vectors will be reduced into a scalar using cosine similarity, becoming the second feature — `text_count_cosine_similarity`. Figure 17 shows the visual representation of the process.

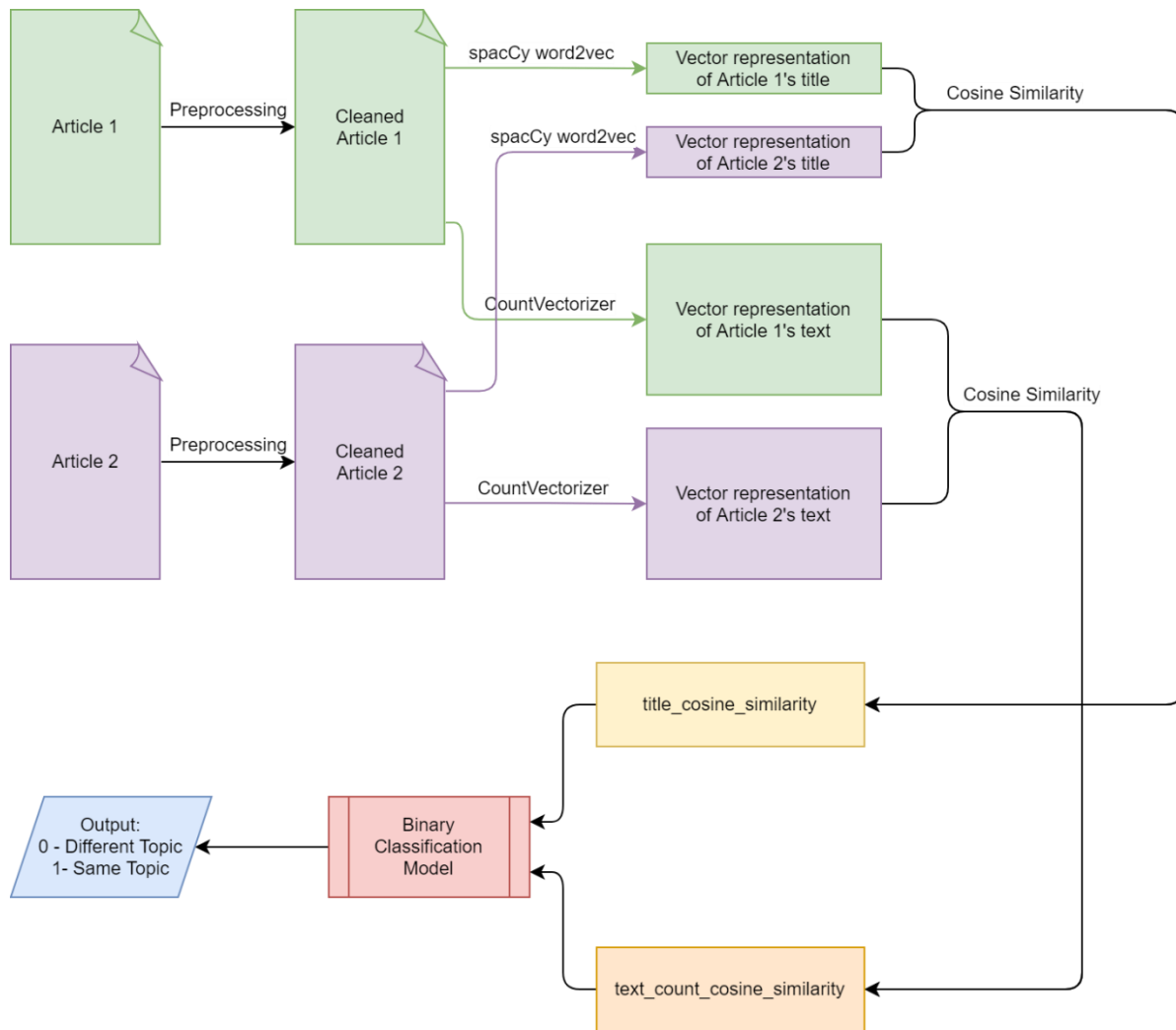


Figure 17. Processing articles into features

To make things simpler, a manual rule was added — if the post time of the two input articles differs by a specific timeframe, which is set to 24 hours for now, they will be treated as if they are in different topics and so will not go through the above pipeline. This is because the newer article is supposed to contain more complete or comprehensive information, and hence having another `NewsGroup` summarizing the newer articles is more appropriate.

Before determining which binary classification model to use, it is worth analyzing how the features and the corresponding class look like in the visualization in Figure 18. From the graph, article pairs with high `title_cosine_similarity` and `text_count_cosine_similarity` are likely to be in class 1 (same topic).

SVMs from `scikit-learn` [9] are tested first since a geometric model might be suitable at first glance. Using either linear kernel or RBF kernel, each with different `C` values, the accuracy is listed in Table 1. It is as expected that the F1-score becomes higher as `C` increases, since there are noises in the region of class 1. Also, using RBF kernel increases the recall by around 10% while the precision is decreased by 2% or so.

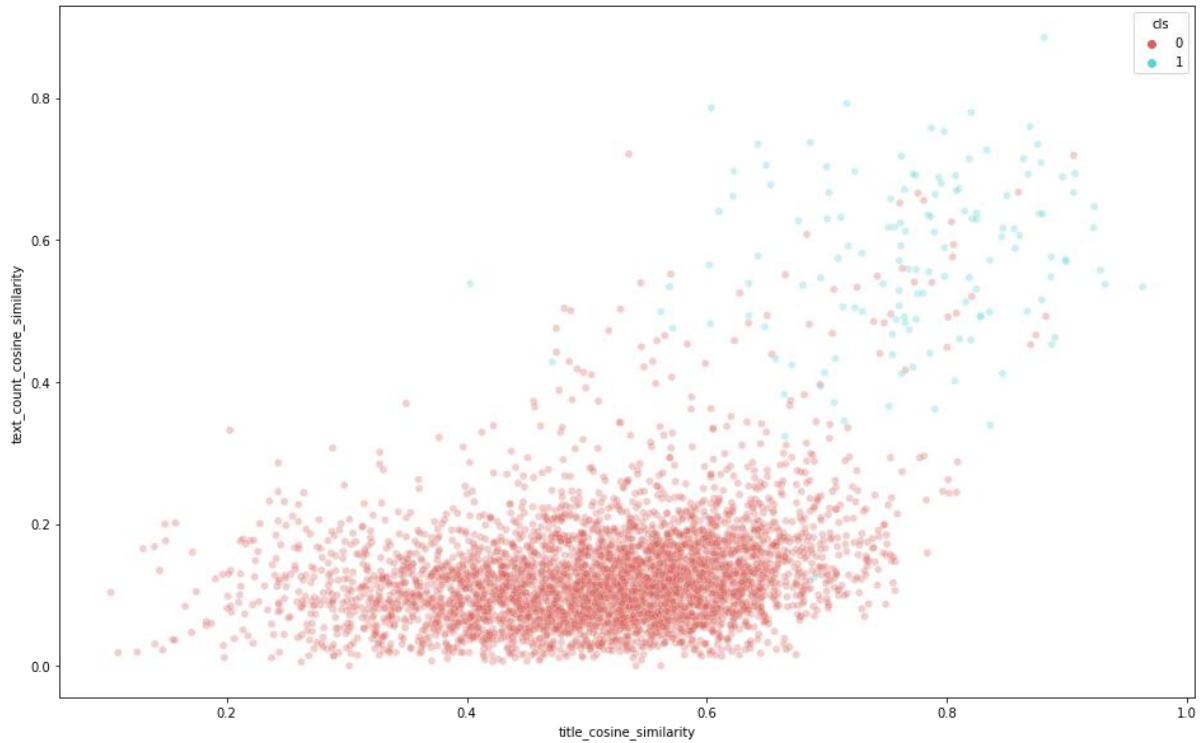


Figure 18. Visualization of the data points

Model	Kernel	C	Recall	Precision	F1-Score
SVM	Linear	0.5	0.818	0.783	0.8
SVM	Linear	1	0.825	0.785	0.804
SVM	Linear	40	0.847	0.779	0.811
SVM	RBF	0.5	0.927	0.770	0.841
SVM	RBF	1	0.927	0.770	0.841
SVM	RBF	40	0.934	0.766	0.842

Table 1. Accuracy of SVM

Next, neural networks were built with Tensorflow Keras [10] to see if they can learn a better decision boundary than SVMs. Multiple neural networks, each with different hyperparameters (eg. number of hidden layers, number of neurons per layer, learning rate, etc), and it was found that by using the neural network structure shown in the Figure 19, the recall, precision, and F1-score of the model became 0.949, 0.765 and 0.846 respectively.

```

model = Sequential()
model.add(Dense(8, input_dim=X_train.shape[1], activation='sigmoid'))
model.add(Dense(8, activation='sigmoid'))
model.add(Dense(8, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer="adam", metrics=[Recall_, Precision_])
model.fit(X_train, y_train, epochs=1000, verbose=1)

```

Figure 19. Structure of the neural network

Logistic regression with L2 penalty was also tried, the recall, precision and F1-score are 0.620, 0.841 and 0.714 respectively.

For now, the neural network model is used, because it has the highest F1-score, and when looking at the false positive article pairs as shown in Figure 20, we found that some of them are actually acceptable, which means that the precision might actually be higher than the score suggested above. So, later we will inspect the performance of the classifier in production and see if there are too many false positives affecting the readers' experience, and may switch to the logistic regression model which has higher precision if necessary.

8156	0.780511	0.655621	1	0	0.798453	Donald Trump has 'mild symptoms' but 'in good spirits' after contracting coronavirus	Trump coronavirus: White House has not been 'open at all' about president's condition, says former UK ambassador to US
8157	0.775703	0.665638	1	0	0.799544	Donald Trump has 'mild symptoms' but 'in good spirits' after contracting coronavirus	Trump says he is doing well, but next couple of days the 'real test'
8162	0.747488	0.483074	1	0	0.725615	Trump, first lady experiencing 'mild symptoms' after testing positive for coronavirus	Trump coronavirus: White House has not been 'open at all' about president's condition, says former UK ambassador to US
8163	0.760729	0.652036	1	0	0.795879	Trump, first lady experiencing 'mild symptoms' after testing positive for coronavirus	Trump says he is doing well, but next couple of days the 'real test'
8164	0.569917	0.551832	1	0	0.688082	Covid: Donald Trump and Melania test positive	Trump coronavirus: White House has not been 'open at all' about president's condition, says former UK ambassador to US
8165	0.535060	0.720756	1	0	0.780454	Covid: Donald Trump and Melania test positive	Trump says he is doing well, but next couple of days the 'real test'

Figure 20. Examples of acceptable false positive misclassifications

The program fetches ungrouped articles from the API and uses the chosen model to group articles to form NewsGroups. Each article is given a boolean attribute called `isGrouped`. If two articles that are determined to belong to the same topic have falsy `isGrouped` attribute, they will form a new NewsGroup object and in turn be saved in the database, with their `isGrouped` flag be switched to true state. Otherwise, if any of each has truthy `isGrouped` attribute, the new ungrouped article is appended to the existing NewsGroup.

2.2.3.3 Multi-document extractive summarization

In order to give readers an overview about the event mentioned in the NewsGroup, a 3-sentence summary would be produced using the algorithm proposed in *An unsupervised method for extractive multi-document summarization based on centroid approach and sentence embeddings* [11] by Salima Lamsiyaha, Abdelkader El Mahdaouy, Bernard Espinasse and Saïd El Alaoui Ouatik, which is presented below.

Given a NewsGroup N containing m Articles ($N = [A_1, A_2, \dots, A_m]$), the first step is to pre-process the articles by splitting the text of all the articles into sentences using NLTK sentences tokenizer function [12], so the NewsGroup N now contains P sentences, denoted as $N = [S_1, S_2, \dots, S_P]$.

Each sentence S_i in N will then be converted into a vector representation with the aid of a pre-trained sentence embeddings model. The authors of the paper experimented with various sentence embeddings such as uSIF, Universal Sentence Encoder and ELMo, and compared their ROUGE score on DUC'2002–2004 datasets, and Universal Sentence Encoder (transformer architecture) was one of the sentence embeddings models that perform the best, and can be used easily thanks to TensorFlow hub [13]. So, Universal Sentence Encoder is chosen to be the sentence embedding model to map the sentences to vectors of 512 dimension, and now we have $\vec{N} = [\vec{S}_1, \vec{S}_2, \dots, \vec{S}_P]$.

To get the centroid embedding $\vec{\bar{N}}$ for the NewsGroup, simply average all sentence vectors with the formula $\vec{\bar{N}} = \frac{1}{P} \sum_{i=1}^P \vec{S}_i$. It is worth-mentioning that the centroid embedding $\vec{\bar{N}}$ will be stored in the database for later use in computing related news.

We can now rank the sentences by how relevant the sentence is to the main idea of the NewsGroup (sentence content relevance score [11]), how original the sentence is (sentence novelty score [11]), and whether the sentence is mentioned in the beginning of the article (sentence position score [11]).

The sentence content relevance score of sentence S_i in NewsGroup N is defined by:

$$score^{contentRelevance}(S_i, N) = cosineSimilarity(\vec{S}_i, \vec{\bar{N}}) = \frac{\vec{S}_i \cdot \vec{\bar{N}}}{\|\vec{S}_i\| \|\vec{\bar{N}}\|}$$

The higher the sentence content relevance score, the closer to the main idea the sentence.

The sentence novelty score of sentence S_i in NewsGroup N is defined by:

$$score^{novelty}(S_i, N) = \begin{cases} 1, & \text{if } \max(sim(S_i, S_k)) < \tau, 1 \leq k \leq P, i \neq k \\ 1, & \text{if } \max(sim(S_i, S_k)) > \tau \text{ and } score^{contentRelevance}(S_i, N) > score^{contentRelevance}(S_l, N), \\ & l = \arg \max(sim(S_i, S_k)), 1 \leq k \leq N, i \neq k \\ 1 - \max(sim(S_i, S_k)), & \text{otherwise} \end{cases}$$

where $sim(S_i, S_k) = cosineSimilarity(\vec{S}_i, \vec{S}_k) = \frac{\vec{S}_i \cdot \vec{S}_k}{\|\vec{S}_i\| \|\vec{S}_k\|}$, and value of τ is suggested by the authors to be 0.95 for the best performance [11]. The first case will set the sentence novelty score to 1 if the cosine similarities between S_i and all other sentences are less than τ , which implies the semantic of the sentence is rather unique in the NewsGroup; The second case will set the sentence novelty score to 1 if the maximum cosine similarities between S_i and all other sentences are greater than τ , and the sentence S_i is closer to the mean vector $\vec{\bar{N}}$ than all other sentences. This is important because there are many sentences that are related to the main points of the NewsGroup which convey similar meanings, yet we only want to select one of those sentences that is closer to the main idea to avoid having repeated messages in the summary. The third case handles the scenarios that do not belong to the former two, and the sentence novelty score will be one minus the maximum pairwise cosine similarity between the sentence and all the other sentences.

The sentence novelty score of sentence S_i in Article A_j with Q sentences is defined by:

$$score^{position}(S_i, A_j) = \max(0.5, \exp\left(\frac{-p(S_i)}{\sqrt[3]{Q}}\right))$$

where $-p(S_i)$ is the index of the sentence S_i in Article A , $1 \leq -p(S_i) \leq Q$. The sentences placed in the front part of an article will be given a higher score, since the most important ideas in news articles are often placed in the front.

A linear combination of the above three scores will be computed to be the final score $score^{final}$ for each sentence, and three sentences with highest scores will be extracted to form

the summary, so that ideally the summary can convey the main idea as much as possible and the sentences included in the summary would not have much overlapping content.

$$score^{final}(S_i, N, A_j) = \alpha \cdot score^{contentRelevance}(S_i, N) + \beta \cdot score^{novelty}(S_i, N) + \lambda \cdot score^{position}(S_i, A_j)$$

where α, β, λ are hyperparameters and experimented with different values by the authors. They are eventually chosen to be 0.6, 0.2, 0.2 respectively [11].

According to the authors, the method above achieves ROUGE-1 and ROUGE-2 score of 39.84 and 9.53 respectively, which is better than traditional algorithms like LexRank and many other neural-network based approaches such as PG-MMR [11]. And through our manual examination, the algorithm is good since it can summarize the main ideas in the NewsGroup without too much duplicated content. An example of the summary generated about Micron's stock after releasing 2020 Q4 fiscal results is shown in Figure 21.

```
"summary": {
  "top": [
    "Micron Technology (NASDAQ:MU) shares were taking a hit late Tuesday afternoon, following the company's release of its fourth quarter of fiscal 2020 results after market hours.",
    "Micron Technology Inc. shares slid in the extended session Tuesday after the chip maker noted challenges going forward after nearly doubling its profit and adding more than $1 billion in sales from last year thanks to strong cloud and consumer sales during the COVID-19 pandemic.",
    "Micron MU, +1.99% said it expects adjusted fiscal first-quarter earnings of 40 cents to 54 cents a share on revenue of $5 billion to $5.4 billion, while analysts had forecast earnings of 66 cents a share on revenue of $5.27 billion."
  ],
}
```

Figure 21. Example summary about Micron's stock after releasing 2020 Q4 fiscal results

2.2.3.4 Finding the most unique sentences

Given that one of the objectives of this final year project is to give users a more complete view on the event, two semantically unique sentences from each article shall also be extracted, in addition to the summary that constitutes the main idea of the NewsGroup.

Suppose in the NewsGroup N , there are m Articles ($N = [A_1, A_2, \dots, A_m]$), and we would like to find two unique sentences from article A_x which are not mentioned in all other articles $A_j, j \neq x$. First, the sentences in A_x are compared with all other sentences from other articles using cosine similarity, and each sentence in A_x will be given a sentence dissimilar score, higher represents the sentence is more unique among the NewsGroup. More formally,

$$score^{dissimilar}(A_{x,p}) = 1 - \max(\cosineSimilarity(\overline{A_{x,p}}, \overline{A_{j,q}})), 1 \leq j \leq m, j \neq x$$

where $A_{c,d}$ denoted the d^{th} sentence in article A_c and $1 \leq d \leq \text{number of sentences in } A_c$

After that, an originality score, which is a linear combination of the sentence dissimilar score and sentence novelty score from the previous section will be calculated for each sentence.

$$score^{originality}(A_{x,p}, N) = 0.9 \cdot score^{dissimilar}(A_{x,p}) + 0.1 \cdot score^{novelty}(A_{x,p}, N)$$

In addition, convolution is performed to the above sentence originality score using the kernel $[0.05, 0.9, 0.05]$ to generate the final sentence originality score, because from human language point of view, it is likely that the sentence i is related to sentence $i - 1$ and $i + 1$, and if the

author uses several sentences to express an idea, that idea is probably more important. For example, the sentence originality score for S_{i-1}, S_i, S_{i+1} are 0.6, 0.7, 0.5 respectively, and that for S_{j-1}, S_j, S_{j+1} are 0.2, 0.7, 0.4 respectively, S_j will be penalized because its neighbouring sentences are not very original, which implies that S_j might not be as important as S_i , or might be rare noise that cannot be filtered, so S_i will be selected as the unique sentence in this example.

$$score^{originalityFinal}(A_x, N) = [0.05, 0.9, 0.05] * score^{originality}(A_x, N)$$

A simple evaluation is done to have a preliminary understanding of how well the method performs. Sentences from seven NewsGroups, each with two to four articles, were labelled as either common, unique, none or noise. Common means the meaning of a particular sentence appears in all articles in the NewsGroup; Unique means the meaning of the sentence appears in one article only. It is found that 23 out of 24, or 67.6% of sentences selected by the algorithm matches the unique label. On the other hand, around 18% of the selected sentences are actually noise that failed to be removed by the initial version of the web scrapper, which have been improved to reduce noise in the past months. The full results are presented in Table 2 below.

Label	Percentage
Common	8.82%
Unique	67.6%
None	5.88%
Noise	17.6%

Table 2. Labels of sentences selected by the method

2.2.3.5 Related news

Recall that in section 2.2.3.3, the centroid embedding $\bar{\bar{N}}$ for each NewsGroup is stored in the database. Since the centroid embeddings represent the semantics of the NewsGroup, it can be used to find other NewsGroups which convey similar ideas or belong to the same category.

To compute the list of related NewsGroups, all NewsGroup ids along with the centroid vectors are fetched from the API, then the cosine similarities of the current centroid vector against all other centroid vectors are calculated and sorted in descending order. Two NewsGroups are said to be related if their cosine similarity is at least 0.4. Finally, the ids of related NewsGroups are returned, and will be saved to the database through the API.

An example of the results of this approach for finding related news is shown in Figure 22. The list of related news generated for the news about H&M planning to close physical stores (the master news) includes economic depression in the United Kingdom amid the pandemic (related news ②) and companies cutting jobs (related news ⑤ and ⑥). It is not obvious how the other three news are related to the master news solely by looking at the summarized title. Since the related news are generated by the centroid embedding on all sentences, there are some clues about how they are related when examining the body texts of the articles. For example, consumers' demand on online shopping platform due to COVID-19 were mentioned in both

the master news and in related news ③, so they are determined to be related. Related news ① is negative event in the financial market, which share the same sentiment and category as the master news, so as a result, they are determined to be related. From the above example, it seems like the proposed approach can capture not only news that are closely related, but also news that are interlinked at a wider level. This not only can cater the needs for readers who would like to read news that are closely related, but also useful for readers who wish to read news of the same category without getting bored by consuming content that are too alike. Hence, both types of curious readers can be satisfied.

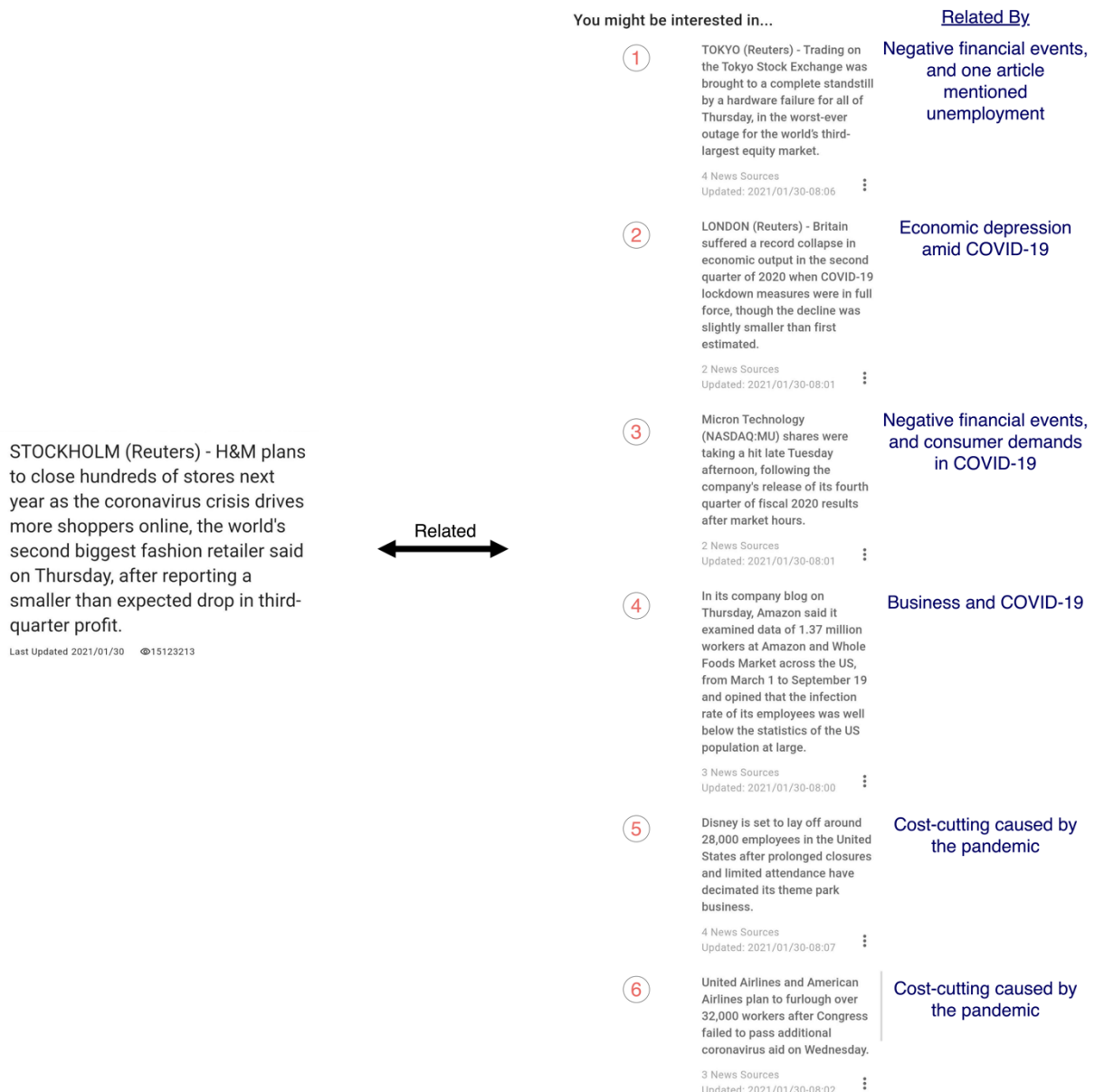


Figure 22. Example output for related news

2.2.4 Database

We have switched back to MongoDB to be our database. Although as mentioned in previous reports, MongoDB is limited by its size of declaration and that causes inconvenient for the users, but we find that for our small-scale application, building relationship for various database

may not be very crucial. Plus, in mongoose (the Node.js module for supporting MongoDB), initiating Schemas is way easier with the chosen Express.js framework. Hence by considering the development cost as compared to the possible affects on performance, we pick MongoDB as the database developemt tool.

2.3 Features Implemented and Objectives

Thus far, the majority of the proposed features have been implemented and tested. Yet, there are still some features being implemented and rooms to improve current user experience. It is anticipated that this final year project can be completed on time with all features programmed by April.

Objective	Completed
1.To aid the user on analysing news material from multiple web sources.	<ul style="list-style-type: none">• Model to group articles about the same event• Algorithm to perform multi-document summarization• Algorithm to find two unique sentences from each article in the NewsGroup
2.To efficiently scrape and store news from multiple trustworthy news sites	<ul style="list-style-type: none">• Web scrapper to scrape news articles from ten different sources• Schemas have been completed and are integrated with various API endpoints
3.To deliver a good user interaction experience on reading summarized news result	<ul style="list-style-type: none">• A simple and comprehensive user interface in smartphone devices that is easy to follow• User commenting section is mostly completed
4.To employ basic security measure for securing application users' privacy	<ul style="list-style-type: none">• A simple user-registration and login form is established

2.4 Testing

Below is a rough layout on the test strategies that are going to be adopted in this project.

2.4.1 Testing the User Interface

The user interface will be tested with black box testing to verify that the layout is shown as in the prototype. Each interactive component, such as buttons and drop-down lists, will be inspected to see if the designated actions will be triggered or not when those components are clicked.

2.4.2 Testing the API server

Unit test will be used for testing the API server. A few test cases will be written, including querying user data, NewsGroup comments, etc. In each test case compare the expected output of the API with the actual output, and this process could be automated. Stress test can also be conducted to ensure that the server can process a large number of requests at the same time.

2.4.3 Testing of News Fetching and Processing

To make sure the news fetching, and processing pipeline works as mentioned, every step that have been described above was tested carefully. For the news scrapper, manual inspection will be made to ensure the web scrapper can correctly identify and save the title, body text, date, etc, given a link. Next, the step of grouping news articles was tested by calculating the accuracy of the grouping result, since we have labelled testing data set containing articles belong to the same group. Finally, the summarization algorithm was tested by black box testing to see if the summary includes both the most important sentences in the NewsGroup and unique sentence from each article, and that the summary should be saved to the database as expected. An internal sentence labelling web pages was developed for this purpose to label every sentence into one of the following categories, namely common, unique, none and noise.

2.5 Evaluation

In the user acceptance tests, have created a Goole Form Survey to collect user feedback as well as providing external tester invitation links on the google form. At the end, we have managed to collect an effective 21 reponses.

App Target Users

In the current phase, it only provides the English news with English description, so, the target user group is focused on the educated class or people who are native in English.

Survery Overview

As mentioned previously that our application aims to aid users to analyzing news material from multiple online news sources. And with 21 effective reponses collected, evaluation of the questionnaire will be done to see if we have met our project objectives or not. The details of the survey can be found in the Appendix B.

Survey Summary

Since we have managed to collect opinions from different gender and hopefully that the views on the aestic of the user interface design is more balanced.

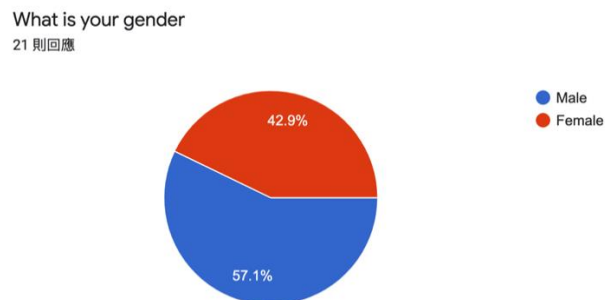


Figure 23. Gender distribution of respondent

By analyzing the responses on the questions regardless of the user interface(question 3 to 5) that our users agree that our user interface is quite neatly designed as we have referenced other user interface of the current news application as our inspiration such as the Stand News which is mentioned in previous litertaure review. So, we think that we have achieved the 3rd objective of delivering a good user interaction experience on reading summarized news result.

In general, do you agree that our UI is easy to use

21 則回應

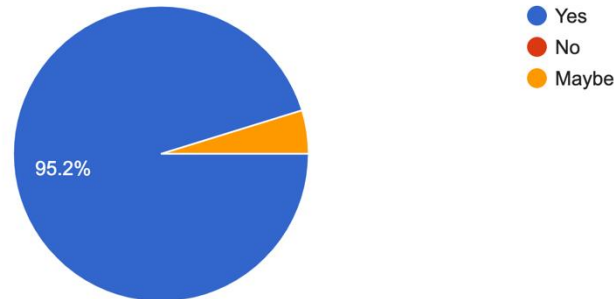


Figure 24. Views on the user interface to see if it is easy to use or not

For the evaluation of the performance of the AI, question 6,7 plays a vital role in understanding the precision as well as the conciseness for our readers to understand the news or not. And by the responses gathered, over 89% of them agrees that the summarization is both precise and concise which helps them to understand the news topic more quickly. And we have helped our users to analyse online news from multiple news sources

How would you rate the preciseness and conciseness of the summarization?

21 則回應

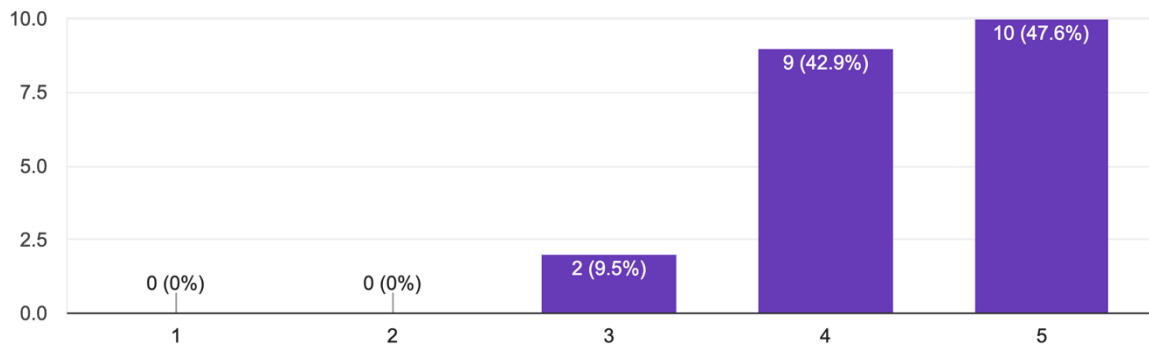


Figure 25. Preciseness and conciseness of summarization

Do you think that the generated summary from the AI can help you understand the news more quickly?

21 則回應

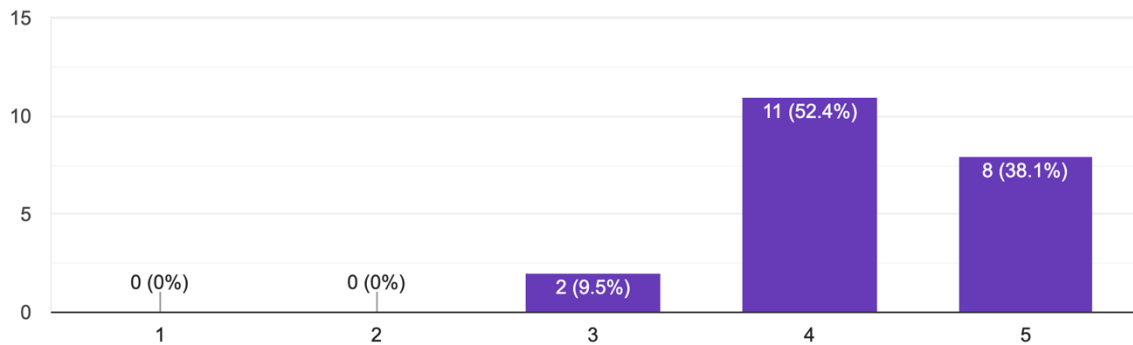


Figure 26. Rating on whether the AI can help them to understand news more quickly or not

In this question, over 81% of them agrees that we have enough news sources, which means that we have achieved the objectives of providing multiple news sources to our users. And they are indeed trustworthy sources as “BBC news”, “The Guardian”, “South China Morning Post” both have reputations in the industry as well. So we have achieved the objective of providing multiple as well as trustworthy news source.

Do you think that we have enough news sources?

21 則回應

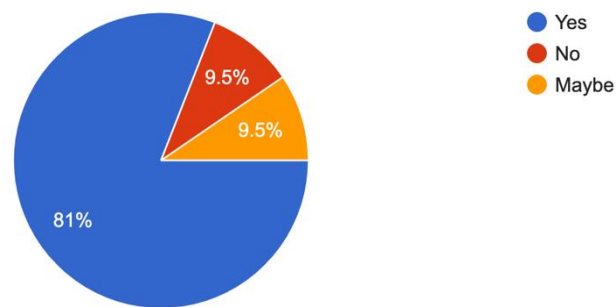


Figure 27. Response on whether the app has enough new sources or not

3 Discussion and Future Work

Future Work

Due to the limited time, we mainly focus on the development of features which are related to the objectives of the project. And by doing that, some functions of the app can be further refined.

Such refinement can be done in the search function of searching news article. Our current implementation is a keyword-match on the title of the news. However, this has some drawback as since the returned result might not be as close as what the users want. In addition, in some complex search function such as the google search, it is capable of handling auto-completion and spelling-correction of common English words. Such refinement could be a possible extension in the current application in order to improve the user experience.

Another possible extension is to incorporate an open-standard authorization protocol like OAuth for security purpose. With this, users can register an account more quickly by using their existing google account, which is just within several clicks. It helps to save the time for manual registration and authentication on users' side, and in turn reduce the system load on the server's side. Moreover, it allows users to enjoy a highly secured web service with the help in authentication offered by big, trustworthy enterprise, which could in turn attract potential users to use our application. But due to time limits, this could not be accomplished and serves as a possible extension in the future.

Refinement on web accessibility is also worth studying. Different age group of users may want to achieve different level of understanding or visualization on the application content due to their physical condition. Then it would be nice if the application further allows for customized configuration on the user-interface, such as adjustment on font-size, font-colour, or news card layout, etc. A more desirable solution is to make the application responsive to various mobile devices in order to suit for users on various platforms. Thus, such user interface refinement serves as another possible extension.

More multi-document summarization algorithms, including both extractive summarization and abstractive summarization, can be explored. Although the current algorithm performs quite well, there are on occasion some duplicated information presented. Summarization algorithms that make use of the latest natural language processing techniques can be examined in the future, with the aim of providing a summary that is more accurate and concise than the current one.

4 Conclusion

In this project, we have managed to successfully develop the SumNews as those objectives are met. In this project we aim to provide and with our survey collected that over 84% of them will continue to use such an application to help them to read news which means that it is actually an attractive app and we will continue to develop and scale up the application in order to cope with different kind of users' need as well. In addition, this application can certainly help the users to cope with such an era of excess information and help them to understand news matter faster and a better way. It is believed that With more users onboard in this app, the accuracy of the AI will further improve as well and this will further improve the overall user experience as well.

Will you keep using this application in order to read the latest news?

21 則回應

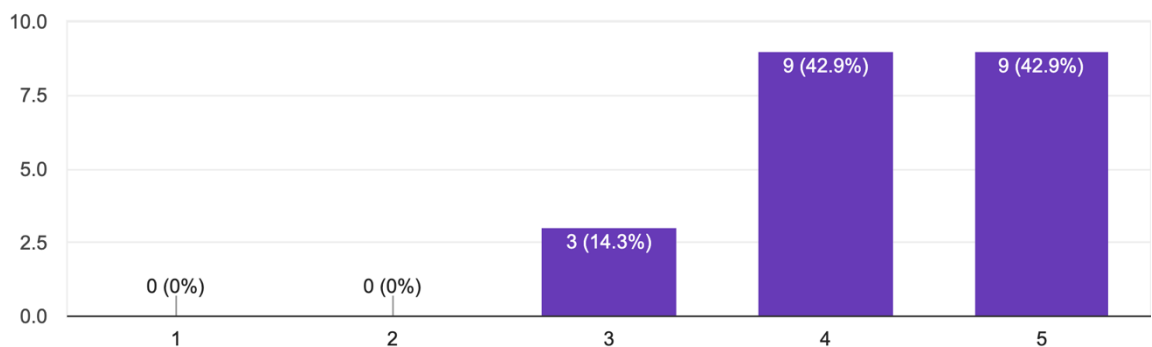


Figure 28. Respondent on continue to use this app in the future

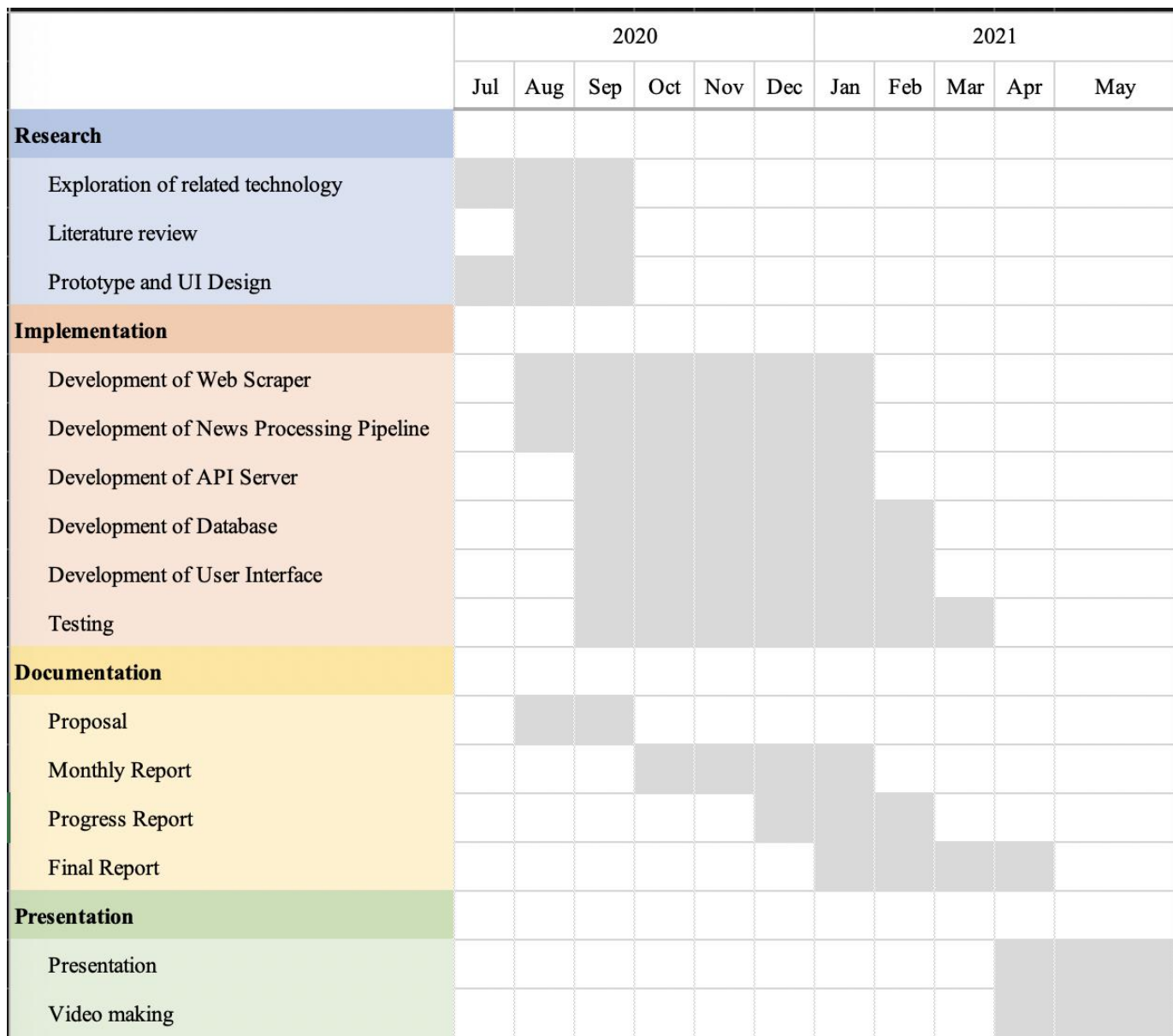
5 Project Evaluation

5.1 Division of Work

	CHAU Shun Wai	LEUNG Hang Kam	LEUNG Hin Fung	TO Ming San
Research				
Exploration of related technology	A	A	L	A
Literature review	A	A	L	A
Prototype and UI Design	L	A	A	L
Implementation				
Development of Web Scraper	A	L	A	A
Development of News Processing Pipeline	A	A	L	A
Development of API Server	L	A	A	A
Development of Database	L	A	A	A
Development of User Interface	L	A	A	L
Testing	L	A	A	A
Documentation				
Proposal	A	A	L	L
Monthly Report	A	A	A	L
Progress Report	A	A	A	L
Final Report	A	A	A	L
Presentation				
Presentation	A	A	A	L
Video making	A	A	A	L

L=Leader, **A**=Assistant

5.2 Gantt Chart



6 Hardware and Software Requirements

6.1 Hardware Requirements

Hardware	Requirement	Usage
iPhone 11	OS: iOS 14	Testing the app on iOS

6.2 Software Requirements

Software	Version	Usage
BlueStacks	4.60	App testing on different versions of Android OS
Visual Studio Code with git bash		Code editing and versioning
Yarn	1.19.1	Node-modules management
Node version manager (nvm)		Help installing and applying node-modules with specified version

6.3 Cloud

Cloud Service	Usage
MongoDB	Store our articles as well as pictures.
Heroku	Host the API Server

7 References

- [1] J. Grabmeier, “Study: Americans choose media messages that agree with their views,” 28 May 2009. [Online]. Available: <https://phys.org/news/2009-05-americans-media-messages-views.html>. [Accessed 18 September 2020].
- [2] “Google News,” Google, [Online]. Available: <https://news.google.com/>. [Accessed 10 September 2020].
- [3] “Welcome to Feedly,” Feedly, [Online]. Available: <https://feedly.com/i/welcome>. [Accessed 10 September 2020].
- [4] “nwsty – Read Less – Know All,” Instance A, [Online]. Available: <https://nwsty.com/>. [Accessed 10 September 2020].
- [5] “Figma: the collaborative interface design tool,” [Online]. Available: <https://www.figma.com/>. [Accessed 10 February 2021].
- [6] “Socket.IO,” [Online]. Available: <https://socket.io/>. [Accessed 17 September 2020].
- [7] “English · spaCy Models Documentation,” Explosion AI, [Online]. Available: <https://spacy.io/models/en>. [Accessed 6 September 2020].
- [8] scikit-learn developers, “sklearn.feature_extraction.text.CountVectorizer — scikit-learn 0.24.1 documentation,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Accessed 1 December 2020].
- [9] scikit-learn developers, “sklearn.svm.SVC — scikit-learn 0.24.1 documentation,” [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>. [Accessed 1 December 2020].
- [10] Google, “Module: tf.keras | TensorFlow Core v2.4.1,” [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras. [Accessed 1 February 2021].
- [11] SalimaLamsiyah, A. E. Mahdaouy, B. Espinasse and S. E. A. Ouatic, “An unsupervised method for extractive multi-document summarization based on centroid approach and sentence embeddings,” *Expert Systems with Applications*.
- [12] NLTK Project, “nltk.tokenize package — NLTK 3.5 documentation,” [Online]. Available: <https://www.nltk.org/api/nltk.tokenize.html>. [Accessed 2 December 2020].
- [13] Google, “universal-sentence-encoder-large,” Google, [Online]. Available: <https://tfhub.dev/google/universal-sentence-encoder-large/5>. [Accessed 2 December 2020].
- [14] “The most popular database for modern apps | MongoDB,” MongoDB, Inc., [Online]. Available: <https://www.mongodb.com/>. [Accessed 2 September 2020].
- [15] “Word Vectors and Semantic Similarity,” Explosion AI, [Online]. Available: <https://spacy.io/usage/vectors-similarity>. [Accessed 6 September 2020].
- [16] “English word vectors,” Facebook Inc., [Online]. Available: <https://fasttext.cc/docs/en/english-vectors.html>. [Accessed 6 September 2020].
- [17] J. Pennington, R. Socher and C. D. Manning, “GloVe: Global Vectors for Word Representation,” [Online]. Available: <https://nlp.stanford.edu/projects/glove/>. [Accessed 6 September 2020].

- [18 “scikit-learn: machine learning in Python,” [Online]. Available: <https://scikit-learn.org/stable/>. [Accessed 6 September 2020].
- [19 “TensorFlow,” Google, [Online]. Available: <https://www.tensorflow.org/>. [Accessed 6 September 2020].
- [20 “ Welcome to Python.org,” Python Software Foundation, [Online]. Available: <https://www.python.org/>. [Accessed 6 September 2020].
- [21 “Sharding — MongoDB Manual,” [Online]. Available: <https://docs.mongodb.com/>. [Accessed 17 September 2020].
- [22 “Data Modeling Introduction — MongoDB Manual,” [Online]. Available: <https://docs.mongodb.com/manual/core/data-modeling-introduction/>. [Accessed 17 September 2020].
- [23 Restuta, “framework-sizes.md,” 13 October 2019. [Online]. Available: <https://gist.github.com/Restuta/cda69e50a853aa64912d>.

8 Appendix A: Meeting Minutes

8.1 Minutes of the 6th project meeting

Date: 11/7/2020

Time: 20:00

Place: Microsoft Teams meeting

Present: Dr. Cecia Ki CHAN, CHAU Shun Wai, LEUNG Hin Fung, LEUNG Hang Kam, TO Ming San

Absent: N/A

Recorder: TO Ming San

Items discussed

- App design
may have to be determined by the end of this month
- Recent technology exploration
- Brief report layout
- Could take reference to provided template from past year
- Need specify the **project objective**, which shall be determined, before the proposal deadline
- Literature review for similar work
- to make the app stand out in 2nd reader's point of view
- Brief plan of workload distribution
- Better at least have 2 people in each task
- Web-service purchase
What we could purchase: database, cloud-service etc.
May need a more detail plan because of limited budget

Goals for the coming weeks

- Phase planning
- Come up with some features
- Find suitable tools (database / technologies)
- Literature review

Meeting adjournment and the next meeting

The meeting was adjourned at 21:03 and the next meeting will be held on 25/7/2020 at 20:00 on Microsoft Teams meeting.

8.2 Minutes of the 23rd project meeting

Date: 10/4/2021

Time: 21:00

Place: Microsoft Teams meeting

Present: Dr. Cecia Ki CHAN, CHAU Shun Wai, LEUNG Hin Fung, LEUNG Hang Kam, TO Ming San

Absent: N/A

Recorder: TO Ming San

Items discussed

- Review of the questionnaire for the evaluation
e.g. more concise wordings for the question 6
- Reminder of submission of final report and self-evaluation form

Goals for the coming weeks

- Send out questionnaires in order to complete the evaluation and the discussion part of the report as well
- Submit the draft of Final Report to Dr Chan on 12/4 noon

Meeting adjournment and the next meeting

The meeting was adjourned at 22:00 and the next meeting will be held on Teams meeting at 1830.

8.3 Minutes of the 24th project meeting

Date: 12/4/2021

Time: 18:30

Place: Microsoft Teams meeting

Present: Dr. Cecilia Ki CHAN, CHAU Shun Wai, LEUNG Hin Fung, LEUNG Hang Kam, TO Ming San

Absent: N/A

Recorder: TO Ming San

Items discussed

- Collected result for the questionnaires
- Review of the final report
e.g. Missing picture in literature review
Review on the future works' part
- Reminder of submission of self-evaluation form

Goals for the coming weeks

- Fix the mentioned problem on the draft of final report
- Submit self-evaluation form as well

Meeting adjournment and the next meeting

The meeting was adjourned at 19:30

9 Appendix B: Questionnaire for Evaluation

Question 1: What is your gender

Options:

- Male
- Female

Question 2 : How often do you read news via web? (Background Evaluation of user)

Options:

- I read online news everyday
- I sometimes read online news (i.e. at least once in 3 days)
- I seldom read online news(i.e. once a week)
- I never read news via web

Question 3: Do you think the design of the home page is neatly designed?

Answer: Scaling scale from 5 to 1 with 5 being “Very neatly design” and 1 being “Not neatly design at all”

Question 4: How satisfied are you with the flow of the pages of the app

Answer: Scaling scale from 5 to 1 with 5 being “Very neatly design” and 1 being “Not neatly design at all”

Question 5: In general, do you agree that our UI is easy to use

Options:

- Yes
- No
- Maybe

Question 6: How would you rate the preciseness and conciseness of the summarization?

Answer: Scaling scale from 5 to 1 with 5 being “Very precise and concise” and 1 being “Not precise and concise”

Question 7: Do you think that the generated summary from the AI can help you understand the news more quickly?

Answer: Scaling scale from 5 to 1 with 5 being “It helps a lot” and 1 being “It helps a little”

Question 8 : Do you think that we have enough news sources?

Options:

- Yes
- No
- Maybe

Question 9: Will you keep using this application in order to read the latest news?

Answer: Scaling scale from 5 to 1 with 5 being “Definitely will” and 1 being “Definitely will not”

Question 10: Anything you want to say to us about this app? (i.e.: Any new features/functionalities to add to this existing app)

Question 11: Are there any features that needs to be improved with our existing application?