THE UNIVERSITY *of* EDINBURGH
informatics

THE UNIVERSITY *of* EDINBURGH

Text Technologies for Data Science

Group Project Report

News Search Engine

Li Changchun

Huang Zheng

Huang Chenyu

Chen Ziheng

Li Zhezong

Jiang Ninghuan

URL: www.hanhannews.co.uk

If there is any problem with the link, please contact us through the following contact information:

TTDS Group 5: s1951405@ed.ac.uk

THE UNIVERSITY OF EDINBURGH

Part I

1 Introduction

Based on skills learnt from this course, our group finally decided to implement a news search engine from scratch. Although the idea might be somewhat outmoded, we believe that a decent news search engine reveals the outcome of the understanding of team cooperation and software development skills. Generally speaking, one piece of news appearing on a website consists of metadata including author, date, pictures, article content and a bunch of news is usually categorised depending on the most popular tags in daily life.

If taking these factors into consideration, we believe that every single task after decomposed still needs to be discussed further. This means that additional features and issue management are required. For instance, apart from basic modules like indexer, search and retrieval, we start to analyse what are the essential elements that we need to take from source, what search/retrieval algorithm best suits our target, how to implement the recommendation feature just as most news websites do, what type of database we should apply to store large data collection, how to deal with large set of queries flushing into the user interface and many coordination problems.

Once complete sorting out details, we could smoothly constitute a design draft and distribute tasks organisationally. This report aims to illustrate a comprehensive system design as well as those obstacles each member confronts during implementation.

2 System Architecture

The system architecture and the workflow are illustrated in detail in Figure 1, which includes the following 9 key elements.

1. **Data obtainment and cleaning**
   Crawler is responsible for collecting news from websites, clean and save them into local files. It runs once per day to update the latest news.
2. **Database**
   Database is responsible for storing multiple dictionaries generated by some modules, and provides the query interface for the search module.
3. **Indexing Module**
   Read and pre-process files, and create indexes of each news for later module.
4. **Retrieve Module**
   Apply TF-IDF or BM25 technique to weigh each word in each specific document.
5. **Search Module**
   Get the search query from the request, fetch necessary information from database, search the most related news and return sorted results.
6. **Recommendation Module**
   Recommend 9 most similar news of each news when the user is browsing.
7. **Classification Module**
   Classify the news into 7 main categories based on contents.

8. **Flask web frame**
   Match the predefined URL rules in the routing table according to the URL of the HTTP request and find the corresponding view function. finally, Flask operates the database and the front end for data exchange.

9. **front-end**
   Design and develop web pages with HTML, CSS and JavaScript to display the content sent by the back-end.

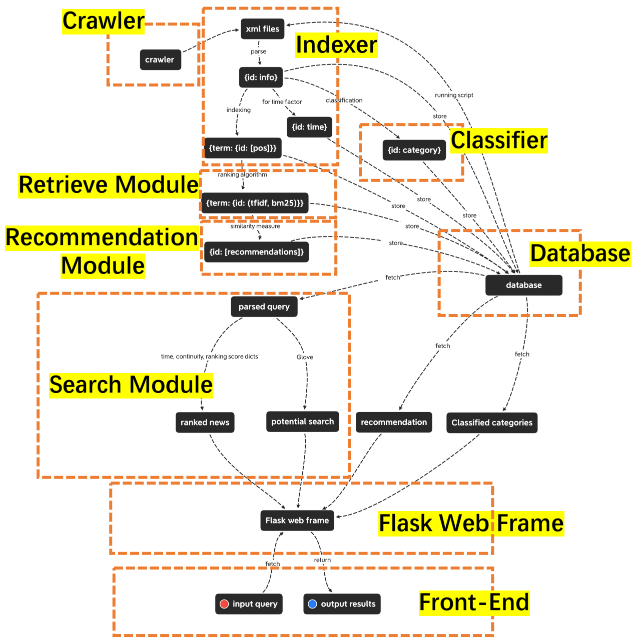


Figure 1: System Architecture and Workflow

3 Data Obtainment and Cleaning

After several times of meeting discussion about the target of our project, we eventually confirmed news to be the source. To make source control more modular, the output of the tool for getting the news information should be carefully selected. Inspired by the pipeline design pattern in computer science, the news storage format and content should serve for the input of the next programmer. Consequently, the whole group might reduce extra coordination and hence improve efficiency.

3.1 Data Crawler

Considering the best proficiency among all programming languages, we finally chose Python to implement the news crawler. The main objective is to ensure the quantity of news as well as the diversity of sources. Our ideal goal is reaching 500k pieces of news consisting of essential metadata such as title, author, description, date, image url, text and ID. Besides, diversity is guaranteed by crawling news from at least 5 websites. For this project, we particularly select famous news websites as targets. They are BBC, The Guardian, New York Times, Reuters and The Atlantic respectively.

The output file format could be significant as it is related to the input of later module. For this project, both JSON and XML are ready to use in terms of completion. However, we held different opinions. Generally speaking, JSON is more lightweight but less extensible compared with XML. This means that when the data collection reaches Gigabyte level, a lightweight structure gets better performance in terms of I/O operations. Besides, due to lack of tags, JSON files are typically smaller and easier to parse. The advantages of JSON seem to go well with our project. This is correct because the size of 500k pieces of news could at least be hundreds of Megabytes and having a good I/O saves time to train models later. Applying JSON also reduces time for updating the database.

3.2 Data cleaning

Unfortunately, despite these benefit, the overall performance of the system does not improve because the search key word only interacts with database query operation after all. Another biggest flaw of JSON is that if the source of crawler is polluted, it is difficult to remove duplication in JSON format. Worst of all, the problem will not be revealed until the last step is done. For instance, there are situations where a news website contains the exactly same news under different categories. A crawler cannot deal with duplication easily. When such problem arises, debugging might take huge amount of time because the database has to be updated repeatedly. Frankly speaking, all JSON advantages do not matter that much but fail our expectation only because the extensible ability makes it extremely hard to remove duplication. Although JavaScript could be used to support JSON and check duplication, it requires high level skills to be able to remove it in units/blocks.

This is where XML shines. To solve the problem, we need duplication removal right after crawler completion. In order to check thoroughly, we have to combine all crawled XML files into one, which is another technical hurdle for JSON files as JSON replies on arrays to extend other JSON files while XML is easy to access via tags and attributes. After combination, simply writing an XSL file to analyse the tags and structure of XML files addresses the problem. In this case, we could assign the tag title as the key to determine whether there is another tag document containing the same tag title. The removal depends on nodes, which means the extensible trait of XML helps to add/remove a child node quickly even if the XML file is bigger.

4 Database

The role of database is twofold. Firstly, database should store all the information towards news from XML file generated from online crawler; Secondly, database should efficiently and effectively give needed information back to the front end of the search engine. Thus, it is crucial to guarantee the response speed between database and search engine because a slight delay will greatly affect user's experience. Therefore, the process of implementation towards database could be divided into two parts: Database Design and Optimization of Database Performance.

Based on project's requirement, there are 3 kinds of data that need to be stored in database: information of news like title, author and so on; recommendation list for each news; word's position and its corresponding BM25 score and TFIDF score for each document. At the initial stage, the information of news is planned to be stored in one table. However, this cause a severe problem because there are too many attributes in this table, which means that loading this table into the memory buffer pool will consume more memory. As a consequence, most of the time will be wasted during the I/O operation, this is not acceptable because for search engine, the reaction time should be controlled in minimum in order to ensure the user's experience while using this search engine.

After refining the design of database through applying appropriate vertical sharding by different functionali-

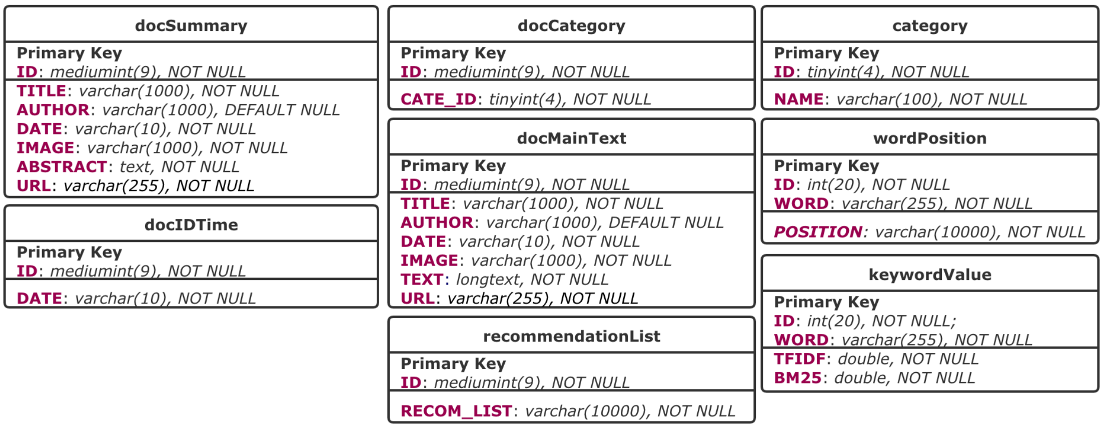

docSummary
Primary Key
ID: mediumint(9), NOT NULL
TITLE: varchar(1000), NOT NULL
AUTHOR: varchar(1000), DEFAULT NULL
DATE: varchar(10), NOT NULL
IMAGE: varchar(1000), NOT NULL
ABSTRACT: text, NOT NULL
URL: varchar(255), NOT NULL
docIDTime
Primary Key
ID: mediumint(9), NOT NULL
DATE: varchar(10), NOT NULL
docCategory
Primary Key
ID: mediumint(9), NOT NULL
CATE_ID: tinyint(4), NOT NULL
docMainText
Primary Key
ID: mediumint(9), NOT NULL
TITLE: varchar(1000), NOT NULL
AUTHOR: varchar(1000), DEFAULT NULL
DATE: varchar(10), NOT NULL
IMAGE: varchar(1000), NOT NULL
TEXT: longtext, NOT NULL
URL: varchar(255), NOT NULL
recommendationList
Primary Key
ID: mediumint(9), NOT NULL
RECOM_LIST: varchar(10000), NOT NULL
category
Primary Key
ID: tinyint(4), NOT NULL
NAME: varchar(100), NOT NULL
wordPosition
Primary Key
ID: int(20), NOT NULL
WORD: varchar(255), NOT NULL
POSITION: varchar(10000), NOT NULL
keywordValue
Primary Key
ID: int(20), NOT NULL;
WORD: varchar(255), NOT NULL
TFIDF: double, NOT NULL
BM25: double, NOT NULL

Figure 2: Final Design of Database

ties in search engine, the final structure of database is shown in Figure 2. In this structure, instead of storing all the information of news in one table, now the information is stored in three tables:‘docSummary’, which is used for displaying the search result generated from user’s query;‘docMainText’, which store the main contents of news;‘docIDTime’, which is used for news sorting by date. Also, there is a table call‘docCatetory’, which is used to store the result of classification for each news. In addition to these four tables, there are four remaining tables like ‘keywordValue’used for storing the BM25 score and TFIDF score of each word in different document, and ‘wordPosition’used for storing the position of each word in different document, and ‘recommendationList’which stores the ID of news that is similar for each news, and ‘category’used for storing the information of category after classification.

5 Indexing Module

Different from the coursework 1, there are nearly 500k pieces of news which need to be pre-processed this time. They are divided into several XML files since a single XML file including all the news is easy to cause memory errors. Therefore, first of all, we use *os.walk* method to get all the path of the files and write a XML parser function to load the XML file iteratively and extract key information of the news. During the process of extraction, we also do tokenization, stopping and stemming the same as coursework 1, considering all the operation above can help us remove invalid information.

After the parse of the files, we get 3 dictionaries, dictionary $\{id : \{title :< str >, author :< str >, url :< str >, ...\}\}$ which contains all the information of a piece of news for future display in our search engine website, dictionary $\{id : time\}$ for computing time factor in search module, dictionary $\{id : text\}$ which will be processed in this module to get indexing dictionary $\{term : \{id : position\}\}$. All these dictionaries are saved into database for later use.

6 Retrieve Module

The retrieve module applies TF-IDF and BM25 techniques. There are multiple versions of them. In our project, we use the formulas shown as follows.

$$W_{TFIDF}(t,d) = (1 + log tf(t,d)) log(\frac{N}{df(t)}) \quad (1)$$

$$W_{BM25}(t,d) = \frac{tf(t,d)(k_1+1)}{tf(t,d) + k_1(1 - b + b(l(d)/\bar{l}))) log(1 + (N - df(t)) + 0.5/(df(t) + 0.5))} \quad (2)$$

where t and d denote term and document respectively, tf is term frequency, N is the number of documents in total, l is the length of document, $\bar{l}$ is the average length, k_1 controls non-linear term frequency normalisation (saturation), and b controls to what degree document length normalises tf values. In Apache Lucene, the default values are 1.2 and 0.75 respectively.

By using the above formulas, the retrieve module can form dictionary $\{term : \{id : (tfidf, bm25)\}$. The TF-IDF score is mainly used in recommendation module while BM25 is used in search module as a base ranking score. For such a huge dataset with 500k pieces of news, the calculation time is long. However, it is necessary for the search engine system to pre-process and store data, hence improve the search speed.

7 Search Module

A comprehensive search module should take multiple factors into consideration, such as the selection of ranking techniques, time, continuity, etc. Further, these factors should be adjusted to match the ultimate goal, which is to find related news then rank them according to a specific ranking scheme.

In our project, the search module mainly supports three types of search modes with different priority levels. The highest priority is given to exact-match search, the second is synonyms search, and the last one, single-word search is set as a compromise on hard-to-search queries. A complete search procedure is illustrated in Figure 3.

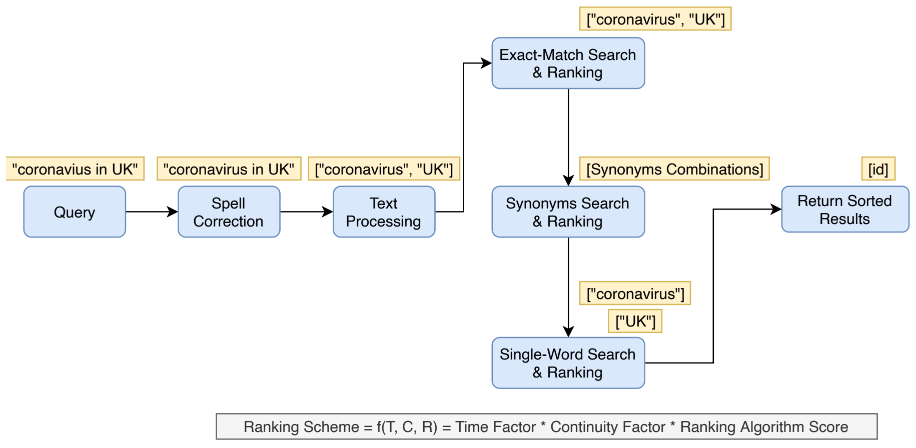


Figure 3: Search Module Flowchart

As shown in Figure 3, the first step is to ensure the spell correctness of the search query. After that, to be consistent with the contents in database, the corrected query should be case-folded, tokenized, stopwords-removed, and stemmed. Then we start to find the related news by AND logic operation among the words, then sort the results according to our ranking scheme. In this case, we search "coronavirus" AND "UK" in exact-match mode, potential synonyms combinations in synonyms search mode (achieved by calling a low-latency API which returns synonyms),

synonyms ombinations = [["coronavirus", "britain"], ["coronavirus", "great britain"], ["coronavirus", "united kingdom"], ["coronavirus", "united kingdom of great britain and northern ireland"]]

and if the number of search results is still not large enough, the search module applies single-word mode, which is to search word by word. In each mode, the search results are ranked based on the function $f(T, C, S)$ which combines the time factor T, continuity factor C, and ranking base score S calculated by TFIDF or BM25 (both are implemented).

$$f(T, C, R) = T * C * S \tag{3}$$

$$T = \frac{1}{\delta * R + 1} \tag{4}$$

where δ is the sensitivity, R is the rank starting from 0 and sorted by time from new to old. Hence, the latest related news can have the time factor T to be 1, and any others are penalised by the worse freshness.

$$C = 1 + b * (L - 1) \tag{5}$$

where b is the bonus parameter, L is the length of longest continuous search words. For example, the final score of the news is awarded if there is a continuous phrase "UK coronavirus".

Finally, all the sorted results are appended to a list and returned. Unlike coursework 1, we only need to fetch a few existing data of related news from the database instead of loading all the data in the memory, then calculating the ranking scores with latency. Furthermore, the algorithm is designed to match the characteristics of news search task, especially the time factor. Hence, memory cost, computational speed, and search quality are significantly optimised.

Additionally, the search module can provide some potential search options based on the current query. This function is realised by Glove. For example, if we search "Edinburgh food", our website can show "Edinburgh meat", "London food", etc. aside the page.

8 Classification Module

Since news crawled from 5 main news websites have totally different categories and naming systems, we expect to classify these news into 7 main categories(bussiness, entertainment, health, politics, science & technology, sport, world) for two applications. The first one is to demonstrate the news according to individual class in the homepage of our website. In addition, we hope this work is able to reduce the workload of computation and increase the accuracy in the part of recommendation.

The first step is to build a classifier based on the dataset we collected specifically for training(7000 pieces of news in total and 1000 pieces of news each class, 80% for training, 20% for testing). Perform the data cleaning and unify the format of news for further processing. We, firstly find the corpus(all the non-repeating words in 7000 pieces of news) of news and label each term with unique id. The total number of id is around 28 k. The next step is to convert each news into the predefined format as following 'label feature_id:value(tfidf)' by leveraging the dictionaires: {term:{news_id:tfidf}} and {term:term_id} we construct earlier. After that, apply the functions imported from the sklearn package: *sklearn.multiclass.OneVsRestClassifier* and *sklearn.svm.SVC* to train the SVM classifier. Here, our strategy is to train 7 individual classifier for each class which is called ‘one Vs all’. The advantage of this method is that we can gain knowledge about the class by inspecting its corresponding classifier. In addition, the performance could be more convincing and stable. The kernel of SVM is set to ‘sigmoid’ according to the performance result whose accuracy is greater than 94%.

After obtaining the classifier, we subsequently do the conversion for the real unlabelled news (500k) and make prediction, then save the results in the database.

9 Recommendation Module

The purpose of this part is to find 9 pieces of most similar news of each news(500k) and subsequently recommend 9 pieces of news to the user when reading the target news. Basically, this recommendation is based on the content of the news which consist of two parts: title and main text. The assumption is that the news is similar if they have similar words regardless of position in the news and semantics.

The first procedure is to convert news into vectors so that the dimensionality or features need to be identified in advance. Two methods have been implemented to find the features. The first one is to choose the top 25 terms of each news according to the value of TFIDF. Then merge and filter them eventually. Another one is to adopt the function *sklearn.feature_extraction.text.TfidfVectorizer* to extract the top 10000 terms across the 20k pieces of news according to term frequencies. The results illustrate that the second method is superior to the first one in terms of speed, but the first method could guarantee better accuracy.

After comprehensive consideration, the second method was adopted since the speed is much more important in the real-time news search engine and its accuracy is also satisfactory when extracting more than 20000 features. After that, each news is converted into vectors with 25000 dimensionalities. Values are their TFIDF which is got from a dictionary({term:{news_id:tfidf}}) we built before.

The next procedure is to compute the similarity between each vector in this big matrix (20W * 2.5W). It turns out that the PC cannot handle such big data directly. We might need the assistance of parallel computing or more powerful machine, however, the solution we proposed here is to break this big matrix into 7 pieces according to their categories (we have classified news into 7 classes in total). This operation could substantially reduce the computation each time and, to some extent, increase the accuracy of recommendation since news from the same category is more likely to be similar. Obtain 9 pieces of most similar news for each news and save them in the database.

10 Flask web frame

Flask is a micro web development framework for Python. When we visit a website through a URL, we send a request to the Web server and the server will send the request to the corresponding Web program for processing according to the URL. Flask retrieves the data required by the search engine through the database pair operation and renders it to the front-end pages respectively.

Pagination processing is one of the best ways to speed up the front-end response. First, we calculate the number of pages by searching the id number returned by the model, and the page number is returned to the front-end display. Then the database is segmented and extracted based on the predefined number of news per page, and the pages returned by the front-end are forwarded to the corresponding urls via flask routing. For example:

$$/search/page/ < int : page = 3 > / \tag{6}$$

the obtained page value is multiplied by each The number of page news is passed into the database as the starting value to fetch the corresponding news.

11 Front-End

Front-end development mainly uses HTML, CSS, JavaScript development languages and Bootstrap framework. The implemented pages mainly include search homepage, search results page and news content page(appendix for details). The homepage uses HTML5 <method = ‘get’> to get the user ’s query, sends it to flask through back-end processing, flask then forwards the search results to the search results page through routing. In order to take into account the mobile display, we mainly use the Grid system of the bootstrap framework. Also, we use this framework to implement a variety of functions such as top navigation bar, push-button drop-down menu and so on. Jquery aims to provide form processing, dynamic tab switching, keyword highlighting and other functions for search engine.

12 Conclusion

Whilst we have successfully built the prototype of a search engine with fundamental features, the project leaves room for improvement. Considering that it is group work, maintaining a good synchronicity from each step is not that easy, especially for a big project. Each member also gains valuable experience in his section. During the process of finding/solving problems, we realise that communication is crucial as the ability to explain problems intelligibly requires the condition where one should not illustrate the problem too technically. Similarly, co-ordination between team members is sometimes subtle and once fully exploited, everyone benefits from it. Encouraged by these positive effects, our group results in a good work efficiency.

Part II

13 Task Distribution

This section indicates each member's workload respectively.

13.1 Huang Chenyu

- Specify the content of the entire project and completed the allocation of work in the beginning with Changchun and Huang Zheng.
- In charge of implementations of news classification module, news recommendation module and also their corresponding descriptions in the report.
- Responsible for running main.py to get all files of this project that will be stored in database and will keep running to have engine reached real time. (more than 2h/test, more than 12 tests)
- Implement title cleaning and author extraction of news and also contribute to part of job in indexer module.

13.2 Huang Zheng

- Code and debug indexing module, retrieve module (BM25), and search module (design and implement three-level-priority ranking algorithms combining multiple factors, and similar search suggestions).
- Outline the key elements and workflow of the project, and participate in task distributions.
- Write report on System Architecture, BM25 part of Retrieve Module, and Search Module sections.
- Help my dear mates to debug classifier and other kinds of issues.

13.3 Li Changchun

- All flask web development work including database and search model related docking and integration
- All front-end development work plus web and interaction design
- Team management including project idea, project planning, allocation of task, organise meetings etc.
- Project Testing, operation and maintenance including Debug the entire project feedback to each team member including me
- Cloud server deployment and operation

13.4 Li Zhezong

- Code to pre-process the dataset from Crawler
- Help optimise and debug indexer module
- Write report on Indexer Module and Retrieve Module sections
- Proofread the report

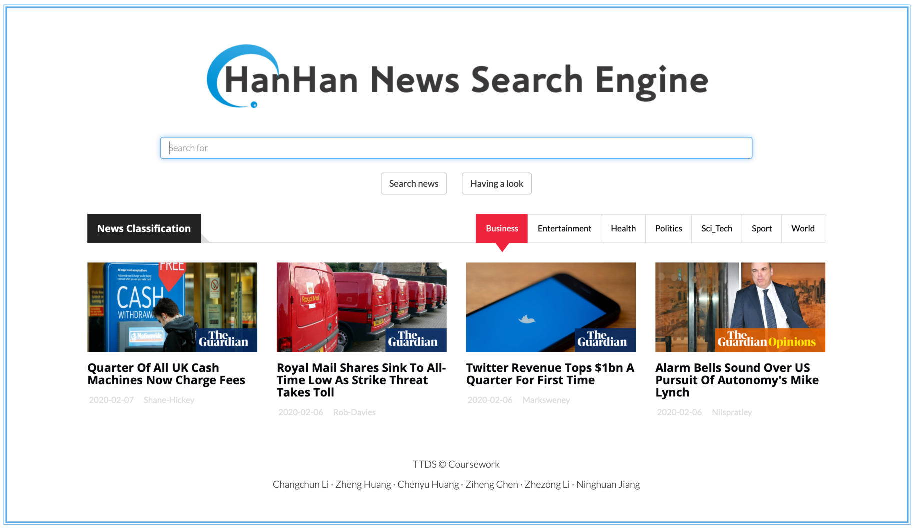
13.5 Chen Ziheng

- The Structure Design and Refinement of Database
- The Deployment of Cloud Database
- SQL search sentence Optimisation
- Coding for API for getting information from database

13.6 Jiang Ninghuan

- Crawl news up to 500k.
- Remove duplicated news.
- Write the report for Section Data Obtainment, Introduction and Conclusion.

Appendices



Quarter Of All UK Cash Machines Now Charge Fees
2020-02-07 Shane-Hickey

Royal Mail Shares Sink To All-Time Low As Strike Threat Takes Toll
2020-02-06 Rob-Davies

Twitter Revenue Tops $1bn A Quarter For First Time
2020-02-06 Marksweney

Alarm Bells Sound Over US Pursuit Of Autonomy's Mike Lynch
2020-02-06 Nilspratley

TTDS © Coursework

Changchun Li · Zheng Huang · Chenyu Huang · Ziheng Chen · Zhezong Li · Ninghuan Jiang

Figure 4: Homepage of HanHan News Search Engine