Report on Implementation of a text search engine

Liu Likun
2IE19337P
liu.likun.578@s.kyushu-u.ac.jp

***Abstract*—In this work, we've implemented a fully functional text search engine with input of keyword and output of relevant results with scores. The backend of the system was built with Python and GETA while the front-end utilized Vue framework and element UI. Two datasets in English and Japanese respectively were utilized in the system. English dataset is financial news in the United States during Jan 2018. The Japanese dataset is 250k Japanese Wikipedia data in plain text. The system can be accessed on: http://13.231.178.152:8233/ (Inside Campus) or http://dist.ecoresystems.cn/ (Globally available) and source code is available on: https://github.com/ecoresystems/distributed_system_course_project/ .**

Keywords— Search Engine, Python, Vue, Element UI, GETA

I. Introduction

In this work, we've implemented a fully functional text search engine with input of keyword and output of relevant results with scores. The backend of the system was built with Python and GETA while the front-end utilized Vue framework and element UI. Two datasets in English and Japanese respectively were utilized in the system. English dataset is financial news in the United States during Jan 2018. The Japanese dataset is 250k Japanese Wikipedia data in plain text. The original dataset was in json format and txt format. Pre-processing was utilized before the indexing procedure in conducted. In section II, we will give an overlook of our system architecture based on the data flow. Section III will give an introduction about the data pre-processing procedure as well as a brief introduction on the data itself. Section IV describes the indexing procedure, which explain the engine and method we use in detail. Section V will introduce the internal APIs and the query process. As we've built a user interface and an API for public access, section VI will describe the technology behind the user interface. In addition, a detail description of the API endpoint will also be provided in this section. In the deployment section (Section VII), we will provide a description of the deployment process along with the issues as of the time this report is written. Additionally, the thoughts about this course and acknowledgment will present at the last section. Finally, the screenshots of the system, the running log on the demo and some fix on the course's material will be provided in the Appendix.

II. System Overview

A. System Architecture

The system implements the User Interface as a web service using HTML 5 and Vue framework. All communications between user front-end and server back-end was sent though RESTful API. The server back-end utilized a web micro framework called Flask. All search requests are sent to the server asynchronously. Requests received via RESTful API are parsed and the parameters passed to corresponding search engine. The search results generated by the search engine are sent to the Flask and encoded into JSON format as response data. The full architecture of the system is shown in Fig. 1.

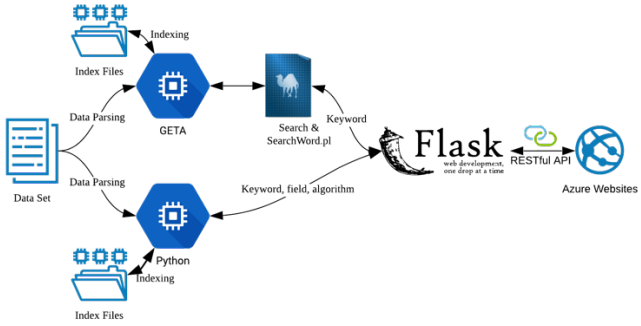


Fig. 1. System Architecture

III. Data Pre-Processing

The data used in this work are the US Financial News Articles in January 2018 and 250,000 rows of Japanese Wikipedia data. Such huge amount of date would require pre-processing before it can be indexed by a search engine. In aim of this, we've injected the data into a MySQL DBMS (Database Management System) for rapid query. Also, for the convince of reverse locating, each row of data was assigned with a UUID served as primary in the database.

A. Data Description

1) US Financial News Articles: Each news article is in a separate json file which contains various of informations including uuid, url, title, text, published date, etc. The preview of the json file is shown in Fig. 1.

Fig. 2. Preview of US Financial Articles Data

2) Japanese Wikipedia: The total number of articles is 1,132,813 and the number of unique words is 2,420,073. Each line contains each article, and sentences are separated by tabs. The preview of the Japanese data is shown in Fig. 2.

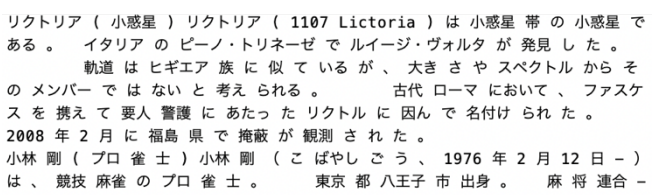


Fig. 3. Preview of Japanese Wikipedia Data

B. Data Pre-processing

1) US Financial News: The data was already well formatted but for tens of thoundans small files, it is a big challenge for I/O system to process. Therefore, we've injected the data into a DBMS. Further, to shrink the data size, we've selected UUID, URL, TITLE, CONTENT and PUBLISHED fields as our data. Fig. 3 gives a preview of the data in the database.

UUID	URL	TITLE	CONTENT	PUBLISHED
00017cde4df49b3…	https://www.reute…	Burst of snow hits the South, p…	Burst of snow hits t…	2018-01-17T22:…
0003a0ac216af0d…	https://www.wsj.c…	Dutch Skepticism About the E…	The Netherlands w…	2018-01-28T22:…
0003cb2a0212e23…	https://www.reute…	BRIEF-Aspen Aerogels Says C…	16 AM / in a few se…	2018-01-26T13:…
0004648ee4072b0…	https://www.reute…	- Chinese angels keep Wanda'…	HONG KONG Chin…	2018-01-30T15:…
0004a4fb845e3ef4…	https://uk.reuters.…	Controversial sheriff pardoned…	Controversial sheriff…	2018-01-09T20:…
00067570c4f3a75…	http://www.cnbc.…	AriseBank™ Announces First-…	DALLAS, AriseBan…	2018-01-18T17:…
0006cddbe2168df…	https://www.cnbc…	Fiat Chrysler CEO: Something…	Fiat Chrysler CEO:…	2018-01-16T16:…
0009c7f5aee3ff7c…	https://www.wsj.c…	With Workplace Suicides Risin…	As suicide rates ha…	2018-01-17T19:…
000a8229d05ba64…	https://uk.reuters.…	Tennis-Raonic slumps to early…	January 16, 2018 /…	2018-01-16T05:…
000af862b74f2524…	https://www.reute…	UK opposition party grassroots…	January 4, 2018 / 1…	2018-01-04T02:…
000b5236ba392dd…	https://www.cnbc…	UPDATE 1-U.S. crude stocks…	(Adds details, price…	2018-01-18T18:…
000c0150f279dc9f…	http://www.cnbc.…	Zapp360 Appoints New Leader…	NEW YORK, Jan. 2…	2018-01-25T16:…
000c311c22acea4…	https://uk.reuters.…	El Salvador eyes work scheme…	January 17, 2018 /…	2018-01-17T04:…
000e01843dc868d…	http://www.cnbc.…	ForgeRock Announces Key Ex…	SAN FRANCISCO,…	2018-01-23T17:…
000f0ad483fdec0b…	https://www.reute…	BRIEF-Pfizer CEO Says No Pr…	Jan 30 (Reuters) -…	2018-01-31T00:…
0011b24243020f0…	https://uk.reuters.…	Motor racing-Back with old spo…	January 18, 2018 /…	2018-01-18T21:…
0012bc725b24b3e…	https://www.cnbc…	US towns that offer financial in…	SHARES College…	2018-01-04T16:…
0012e2fc57c2911…	http://www.cnbc.…	CriticalPoint Capital Acquires t…	LOS ANGELES, Ja…	2018-01-08T16:…
0012e9f22d5029e…	http://www.cnbc.…	QuoteWizard Acquires Bantam…	Deal enhances Qu…	2018-01-11T16:…
00134ff934a4f4ec…	https://www.reute…	BRIEF-Groupe Ldlc Q3 Reven…	Jan 25 (Reuters) -…	2018-01-26T01:…

Fig. 4. US Financial News Data in a Database

2) Japanese Wikipedia: Since the data is plain text and did not contain other information, to make it identifiable in the database, we've created a random UUID served as primary key in database assigned to each row of data. Fig. 4 shows partial of the Japanese Wikipedia data in the database.

UUID	CONTENT
00007b3041b0…	ブジェジンカ ブジェジンカ （ Brzezinka ） は 、 ポーランド 南部 、 クラクフ から 約 60 キロメートル に ある 広大…
000133f641ab1…	田中 亘 田中 亘 （ た なか わたる ） は 、 日本 の 法学 者 。 東京大学 社会 科学 研究所 教授 。 専門 は 商法 、 法…
00014dcc41af1…	上杉 美 浩 上杉 美 浩 （ うえ すぎ み ひろ 、 1982 年 10 月 6 日 - ） は 日本 の 女優 。岐阜 県 出身 。
0001c4c641aa…	スタンリー・クラーク・バンド フィーチャリング 上原 ひろみ 『スタンリー・クラーク・バンド フィーチャリング…
0001d0c641ad…	マンダレイ・ベイ・トラム マンダレイ・ベイ・トラム （ 英 : ） は 、 ネバダ 州 パラダイス に ある ラスベガス ・ス…
0001ea5c41a8…	ジャック・ドンズロ ジャック・ドンズロ （ Jacques Donzelot , 1943 年 - ） は 、 フランス の 歴史 社会 学者 、 都市…
0001fe3e41ae1…	酒井 直次 (陸軍 軍人) 酒井 直次 （ さかい な おじ 、 1891 年 （ 明治 24 年 ） 3 月 26 日 - 1942 年 （ 昭和 17 年 ）…
000208904 1b2…	フェ リット フェリット (Felitto) は 、 人口 1 , 390 人 の イタリア 共和 国 カンパニア 州 サレルノ 県 の コムーネ の…
00021dc841a9…	渡辺 寿 (名望 家) ひさし ） は 、 幕末 から 明治 時代 の 、 山梨 県 の 名望 家 、 政治 家 、 厳書 家 。字 を 俊…
00022aa641b3…	シュトルーヴィーナ (小惑星) シュトルーヴィーナ (768 Struveana) は 小惑星 帯 の 小惑星 で ある 。クリミア半島…
0002658041b0…	村山 三男 村山 三男 （ むら やま みつお 、 1920 年 4 月 1 日 - 1979 年 7 月 29 日 ） は 、 日本 の 映画 監督 。新潟…
000272ee41b1…	李 施愛 の 乱 李 施愛 の 乱 （ り しあい (イ・シエ) の らん ） は 、 1467 年 （ 世 祖 12 年 ） に 成鏡 道 で 起こっ…
0002a88241ac…	チェルヴェーレ チェルヴェーレ （ ） は 、 イタリア 共和 国 ピエモンテ 州 クーネオ 県 に ある 、 人口 約 2 , 300 人…
00040c8641b3…	カミーユ・ビダン カミーユ・ビダン (Kamille Bidan) は 、 アニメ 『 機動 戦士 Z ガンダム 』 に 登場 する 架空 の…
00043f4041ab1…	PQ 2 船団 PQ 2 船団 は 、 第 二 次 世界 大戦 中 に イギリス から ソ連 へ 支援 物資 を 送る ため に 運航 され た 3…
0004591841af1…	モンフォルテ・ダルパ モンフォルテ・ダルパ （ ） は 、 イタリア 共和 国 ピエモンテ 州 クーネオ 県 に ある 、 人口…
0004eae241b2…	サプライ サイド 経済 学 サプライ サイド 経済 学 （ サプライ サイド けいざい がく 、 ） は 、 マクロ 経済 学 の ……
0004f4d441aa1…	三星 鎮 (金堂 県) 三星 鎮 （ さん せい ちん ） は 中華人民共和国 四川 省 成 都市 金堂 県 の 鎮 。三星 鎮 は 以下 の…
000502e441ad…	社会民主党 (東ティモール) 社会民主党 （ しゃかいみんしゅとう 、 略称 PSD ） は 、 東ティモール の 穏健 派 中…
000514de41a8…	エンドレスギャルズ エンドレスギャルズ は 、 1984 年 （ 昭和 59 年 ） 7 月 から 1990 年 （ 平成 2 年 ） 6 月 まで…
0005305c41ae…	ラロンデ・ゴードン ラロンデ・ゴードン （ Lalonde Keida Gordon 、 1988 年 11 月 25 日 - ） は 、 トリニダード・…
00054fcc41a91…	認証 制度 共同 事務 局 認証 制度 共同 事務 局 （ にん しょうせいどきょうどうじむきょく 、 英語 表記 : Secretari…

Fig. 5 Japanese Wikipedia Data in a Database

IV. Data Parsing and Indexing

In this section, we've parsed and indexed the data in two different systems. The first system was implemented by Python called whoosh (whooshjp for Japanese package). The second system was GETA (Generic Engine for Transposable Association). As the data format required by different system as well as the nature of the data itself differs in 2 languages, we've parsed the data accordingly.

A. Python

The index process in python was straight forward. For English data (US Financial News Articles), we've set the UUID as ID and indexed other fields including title, content, URL and published date. For Japanese data (Japanese Wikipedia), since there wasn't enough field provide by the data, we've only indexed the content.

B. GETA

The GETA runs on Linux OS and only accept a plain text file with a special format to build the index. The sample file provides tools that written in Perl to create frequency file for both English and Japanese data. As we mainly built the system on Python, we've modified the tools to accept arguments in aim of make it callable from the system.

1) US Financial News Articles: To provide as much agilability of searhing as possible, we've provide two search field for the data: the title and the content. However, as the nature of the GETA, the two fields has to be treated as two different datasets using UUID to link togather. We have added two flags to the original tool "mkfreq.pl": the "i" flags which stands for initial ID, and the "u" flag which will accept the UUID of the current content. The parsing process will produce a large frequency file. The sample of the frequency file is shown in Fig. 5. Note that this sample is also the same format as the text field as well as the format of the Japanese Wikipedia data (data format required by GETA). Then we used GETA to generate the corresponding wams data for indexing.

```
@75def43c-4f40-4e93-8366-fa111348cdd3 (UUID)
1 z
1 i:1
1 …
2 …
```

Fig. 6. Sample of the frequency file

2) Japanese Wikipedia: Similar to the process done to the US Financial News Articles, the Japanese Wikipedia data firstly us Mecab to divide sentence into words and then generated a frequency file. From there, the GETA engine was utilized to generate the wams file for the Japanese Wikipeida data.

V. Interface and Query

Subsequentially, the next process is to provide a query interface for the applications to access. While the query interface provided by the system is a generic web API (RESTful API), there were also two sub-APIs for python to call.

A. Interfaces

1) The Python Interface: The python interface accepts serval parameters including: "fields" which distinguish the corresponding search target, "query_str" which indicates the corresponding search keyword string , "item_count" was the maximum number of the returned reslt , "data_source" was the selection of the data source, a selector to identify US Financial News Data and Japanese Data. "weighting_alg": we've provided 3 algorithms for the search process: Frequency, TF-IDF and BM25F.

2) The GETA Interface: The GETA interface was written in Perl, the Perl code used in the system has some modifications to the output statement to make the output easy for python to process. First, we've canceled the result serial number. Second, we've disabled the formatting strings in print statement e.g. %2d was reduced to %d. The out put of search.pl and searchWord.pl is shown in Fig. 7.

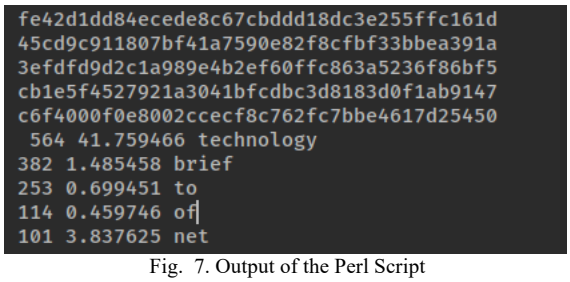


Fig. 7. Output of the Perl Script

B. *Query Process*

1) The Python Process: When receiving a python query with parameters, the system will firstly determine the data source and open the corresponding index folder. Then the field and the key word string will be passed into the engine for querying. For the US Financial News Articles data, a result with title, score, url, content and uuid will be returned for each hit. Also, the runtime of the search and the number of hits is also returned. For the Japanese Wikipedia data, a result with title, score, content and uuid with runtime and total hist were returned.

2) The GETA Query: We've modified the ci.conf file to provide flexibility for the GETA to query. When receiving a request, the system will call "search.pl" and "searchWord.pl" to get the result and the corresponding score. Next, the system will query the MySQL database with the uuids of the searched result to get the content. We've tried to make the response as consistant as possible. As a result, a similar data response with score, url, title, content and uuid along with runtime and total hits was returned. A preview of the response data is shown in Fig. 8.

Fig. 8 Response Preview

VI. The User Interface

To provide relatively optimum user experience, we've built the user interface as a web application. The Front-end of the systems is written in HTML 5 and JavaScript based on Vue Framework and Element UI. Interactions between the front-end and the back end was done asynchronizely via AJAX.

We've provided a total of 6 options:

Key word: The search word.

Data Source: switch between US Financial News Articles

Fields: switch between title and text

Engine: switch between Python and GETA

Algorithm: algorithm used for searching (Python engine only)

Limits: Maximum hits

The result is listed in a table with 4 fields: title, score, url and uuid. We've also added some additional features to the system. As the content might be too long to display, we will only display the title in the table. However, the content is available via a single click on the title. For the US Financial News Articles data, we've also preserved the original URL to the article itself for easy access. In addition, a popup window will be displayed upon each successful search indicating the total hits and the runtime of the search.

VII. Deployment

Since the search engine is a functional system, we've deployed the system on two different servers: Kyushu University's QUEENS (HTTP Endpoint: http://http://13.231.178.152:8233/ OS: Amazon Linux, Inside Kyushu University Only) and Microsoft Azure Cloud (HTTP Endpoint: https://dist.ecoresystems.cn/ OS: Ubuntu Server 18.04 LTS, Globally Available). We've installed apache2 and MySQL DBMS on both system and installed the wsgi module to allow flask to serve the pages and provide the relative API (API is available on endpoint "/doc_search"). The API will accept XXX parameters. A complete parameters description is shown in Table 1.

Parameter	***Type***	***Value***	***Description***
data_source	String	US Financial News, Japanese Wiki	Specify which data to search.
search_engine	String	Python, GETA	Specify the search engine
key_words	String	User specific	Specify the search word
limit	Integer	User specific	Specify the max number of results
search_algoritm	String	TF-IDF, Frequency, BM25F	Specify the algorithm for the search
fields	String	Title, Text	Specify the fields for the search

Table 1. Required Request Parameters for the API

However, as of the time this report is written, there were still some issues during the deployment:

1) Due to the limited storage on the QUEENS (8.4GB total) and the huge data size from two datasets as well as index file for the both engine, it is confirmed that memory error is likely to occur during the query of Japanese Wikipedia data. (Insufficient swap memory)

2) As the flask is running on the root directory, the calls for Perl from Python is likely to fail and therefore affects the return results from GETA.

Nevertheless, the system is fully functional in localhost (local environment) and we will provide screenshots of the operational system in the appendix section. In addition, we will continue to resolve these problems and hopefully make both sites operational when this report is graded.

Thoughts Abot The Lecture and Acknowledgement

This lecture provides an introduction about the World Wide Web and the search engine mechanism. However, as there exist some bugs in the program as well as some minor issues in the course materials (sample program) as we will address in the appendix, the development process was severely delayed due to these issues. Also, as a conventional habit, this report uses we or our to indicate first person, but it is actually I since this work is one-man's work.

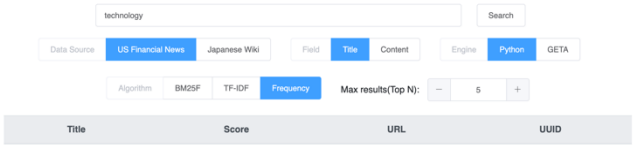
In addition, we would like to thank Microsoft for providing virtual machine and corresponding resources for free.

APPENDIX

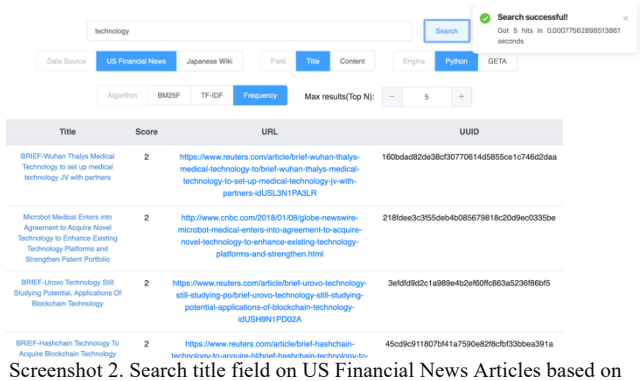
A. Known Issues In Sample:

The GETA 2 program is known to have missed an INT_MAX value, originally I've resolve this with adding #define at each .c file, but there's a better solution by adding the #define INT_MAX <Max value of INT here> to the "limits.h" file located in include floder.
The mkfreq.pl file will generate empty word every now and then, this will cause GETA to malfunction in some cases. Also, the end of line should be <FF> instead of "/".

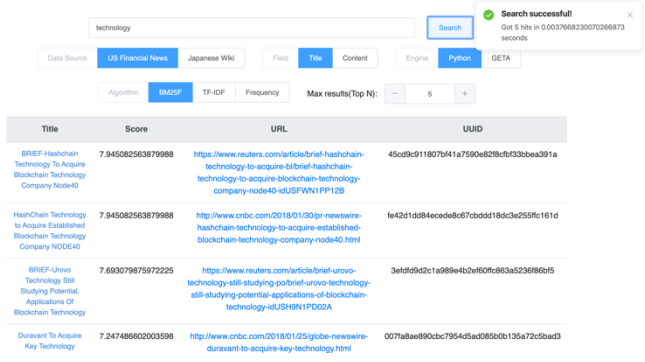
B. Screenshots of the system in Development environment:



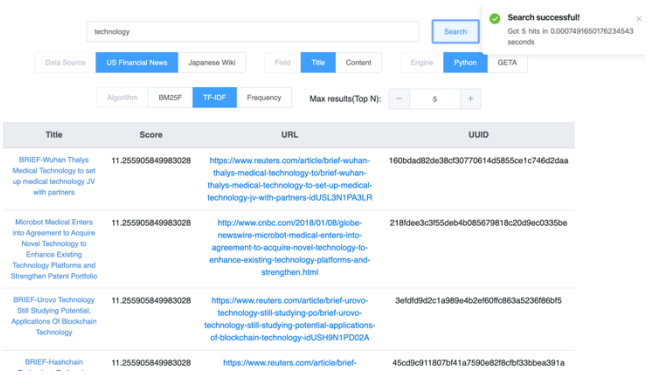
Screenshot 1. Initial Interface



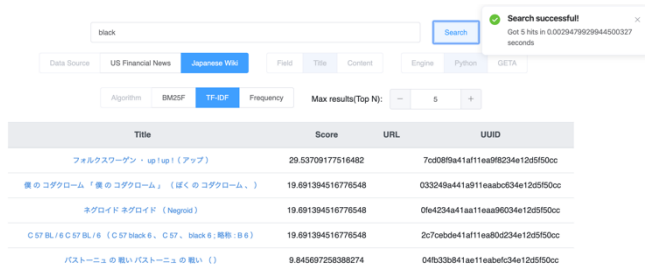
Screenshot 2. Search title field on US Financial News Articles based on Frequency



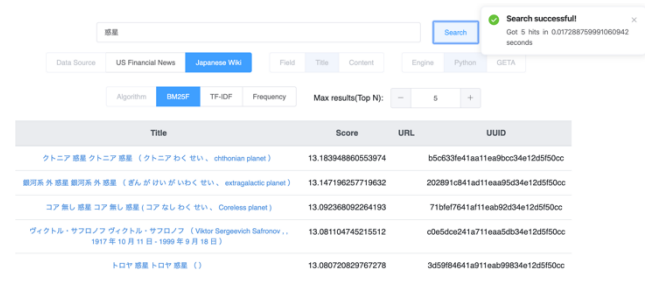
Screenshot 3. Search title field on US Financial News Articles based on BM25F



Screenshot 4. Search title field on US Financial News Articles based on TF-IDF

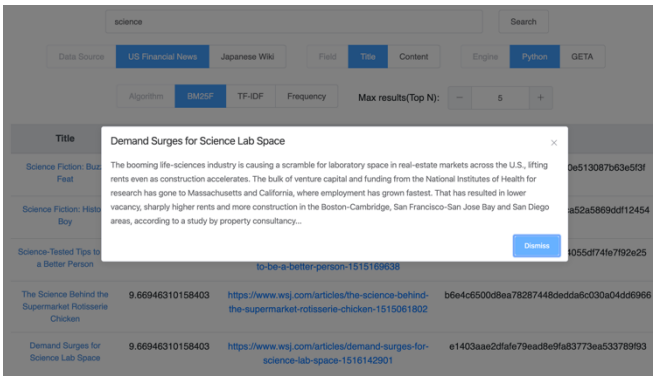


Screenshot 5. Search on Japanese Wikipedia based on TF-IDF

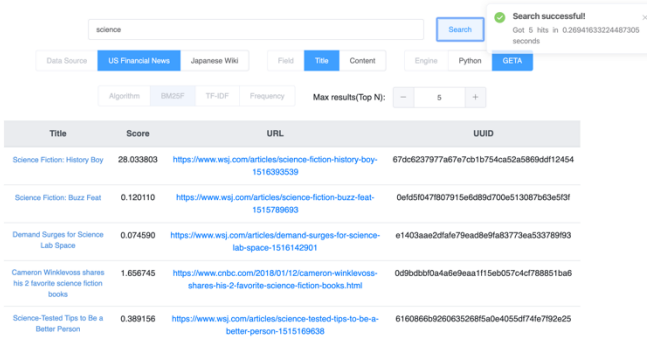


Screenshot 6. Search on Japanese Wikipedia based on BM25F

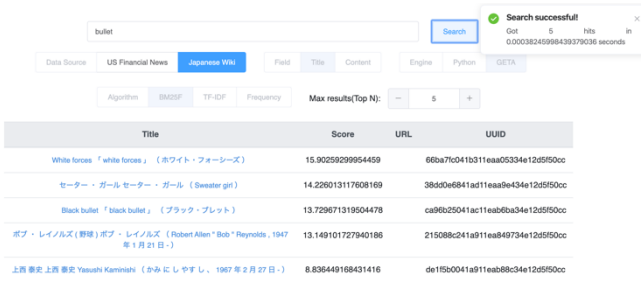
Screenshot 7. Article view of Japanese Wikipedia



Screenshot 8. Article view of US Financial News Articles



Screenshot 9. Search on US Financial News Articles based on GETA



Screenshot 10. Search on Japanese Wikipedia data

Screenshot 11. Content view of Japanese Wikipedia data

C. Running Log of the demo system

The running log was uploaded to GitHub named debug.log and can be accessed via: https://github.com/ecoresystems/distributed_system_course_project/