Fall 2019 CS5604 Information Retrieval Final Report
Front-end and Kibana (FEK)

Edward Powell Han Liu Yanshen Sun Chao Xu
Rong Huang

January 13, 2020

Instructed by Professor Edward A. Fox
Assisted by Ziqian Song

Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

Abstract

During the last two decades, web search engines have been driven to new quality levels due to the continuous efforts made to optimize the effectiveness of information retrieval. More and more people are becoming satisfied during their information retrieval processes, and web search has gradually replaced older methods, where people obtained information from each other or from libraries. Information retrieval systems are in constant interaction with users and help users interpret and analyze data. Currently, we are building the front end of a search engine, where users can explore information related to Tobacco Settlement documents from the University of California, San Francisco, as well as the Electronic Theses and Dissertations (ETDs) of Virginia Tech (and possibly other sites). This report introduces the current work of the front-end team to build a functional user interface, which is one of the key components of a larger project to build a state-of-the-art search engine for two large datasets. We also seek to understand how users search for data, and accordingly provide the users with more insight and utilities from the two datasets with the help of the visualization tool Kibana. Already, a search website, where users can explore the two datasets, Tobacco Settlement dataset and ETDs dataset, has been created. A series of functionalities of the searching page have been realized, for instance, the login system, searching and filter functions, a Q&A page, and a visualization page.

Contents

1 Overview **5**
- 1.1 Project Introduction . . . 5
- 1.2 How Was It Managed? . . . 5
- 1.3 Problem/Challenges . . . 6
- 1.4 Solution Developed . . . 6
- 1.5 Future Work . . . 7

2 Literature Review **8**
- 2.1 The Importance of Information Retrieval Systems . . . 8
- 2.2 The Importance of Tobacco Settlement Dataset and Virginia Tech ETDs Dataset . . . 8
- 2.3 Realization of a Web-based Information Retrieval System . . . 9

3 Requirements **10**
- 3.1 Introduction . . . 10
- 3.2 Functionalities . . . 10
- 3.3 Quality/Performance . . . 10
- 3.4 User Support . . . 11
  - 3.4.1 User Manual . . . 11
  - 3.4.2 Articulatory and Semantic Feedback . . . 11

4 Design **12**
- 4.1 Kibana Utilities . . . 12
  - 4.1.1 Kibana Aggregations . . . 12
  - 4.1.2 Charts . . . 12
- 4.2 Postman Utilities . . . 13
- 4.3 Build Our Own User-friendly Website . . . 13
  - 4.3.1 Infrastructure . . . 13
  - 4.3.2 Back-end . . . 14
  - 4.3.3 Front-end . . . 16
- 4.4 Website Glance . . . 17
  - 4.4.1 Homepage . . . 17
  - 4.4.2 User Login and Register . . . 17
  - 4.4.3 Searching Page . . . 18
  - 4.4.4 Visualization . . . 20
- 4.5 Log System . . . 22
- 4.6 Recommendation . . . 23
  - 4.6.1 Ranking System in Elasticsearch . . . 23
  - 4.6.2 Recommendation Data from TML Group . . . 24

5 Implementation **26**
- 5.1 Milestones . . . 26
  - 5.1.1 Completed . . . 26
  - 5.1.2 To Be Completed . . . 27

5.2 Deliverable . . . 27
5.3 Intra-Evaluation Process . . . 27
5.4 User Log Persistence . . . 27

6 User Manual for Website **29**
6.1 A User Interested in the Tobacco Dataset . . . 29
6.1.1 Scenario 1 . . . 29
6.2 A User Interested in the ETD Dataset . . . 29
6.2.1 Scenario 2 . . . 29
6.3 User Manual for Kibana . . . 30
6.4 User Manual for FAQ Page . . . 32

7 Developer Manual **33**
7.1 Workflow . . . 33
7.2 Data Exploration . . . 33
7.3 Kibana Aggregations . . . 33
7.4 Deployment . . . 38
7.5 Requests Sample . . . 38

List of Figures

1 Postman request . . . 14
2 Infrastructure . . . 15
3 Home Page . . . 17
4 Registration Page . . . 18
5 Login Page . . . 18
6 ETD Searching Page . . . 19
7 Tobacco Searching Page . . . 19
8 ETD Searching Page . . . 20
9 Tobacco Searching Page . . . 20
10 ETD Visualization Page . . . 21
11 Tobacco Visualization Page . . . 22
12 The Process of Creating a Visualization . . . 22
13 Timeline . . . 25
14 User Workflow . . . 29
15 Database Schema for Users Table . . . 30
16 Kibana Discover . . . 30
17 Kibana Dev Tools . . . 31
18 FAQ Page . . . 32
19 FEK Team Workflow . . . 33
20 Count: return a raw count of the elements in the selected index pattern . . . 33
21 Average: return the average of a numeric field . . . 34
22 Sum: return the total sum of a numeric field . . . 34
23 Min: return the min value of a numeric field . . . 34
24 Max: return the max value of a numeric field . . . 34
25 Cardinality: return the number of unique values of a field . . . 35
26 Percentiles: divide the values in a numeric field into the specified percentile bands . . . 35
27 Percentile Rank: returns the percentile rankings for the specified values in the numeric field . . . 35
28 Range: specify ranges of values for a numeric field . . . 36
29 Histogram: specify intervals for a numeric field . . . 36
30 Terms: specify the top or bottom n elements of a given field to display . . . 37
31 Filter: specify a set of filters for the data . . . 37
32 IPv4 Range: specify ranges of IPv4 addresses . . . 38

1 Overview

Our team designed a website [1] for user interaction that allows the exploration and analysis of the Electronic Theses and Dissertations (ETD) dataset from Virginia Tech, and the tobacco dataset from the University of California, San Francisco.

The front-end and Kibana (FEK) team aimed to provide an interface between users and an accurate search system, which comprises intelligent retrieval, a complex recommendation system, and visualizations and analysis tools for both users and developers to better comprehend the data. In addition, there should be a small user study and customer discovery layer to help with later iterations of the website.

1.1 Project Introduction

The full-text search project includes a web application that provides full-text search of Virginia Tech Electronic Theses and Dissertations (ETDs), and of Tobacco Settlement documents. Aside from basic searching, the application also provides personalized recommendations and data visualizations to assist users in discovering the trends and patterns of their field of interest.

1.2 How Was It Managed?

The main task of the front end team is to connect with the data provided by the Elasticsearch team, and to produce a user-friendly interface for searching, filtering, summarizing, and visualizing the data. Aside from the web pages, the team should also produce a "bridge" for customized communications between UI and Elasticsearch. Considering that this is a small application with only a few pages in the front-end, and communication and user authority functions in the "bridge," the team decided to make the application as light-weight as they could.

As a JavaScript library, React [6] is now one of the most popular front-end development tools [10] [7]. The team made a comparison between Vue.js (a popular front-end framework) and React. Although Vue.js provides more ready-to-use function modules, the structure of React code is easier to understand. Therefore, React was eventually chosen as the front-end development tool for this project.

For "bridge" development, the team chose Flask [13] for the same reason as they chose React. Again, the major data source and processing functions of the website are provided by Elasticsearch, where the functions of the "bridge" include only data communication between front-end and Elasticsearch and a

[1] This is the present website link: http://2001.0468.0c80.6102.0001.7015.b2eb.3731.ip6.name:3000. Currently, the website can only be accessed on Virginia Tech's network. Furthermore, one can VPN into Virginia Tech's network if one needs access remotely. In the future, there will be a different link where the project is stored, and the plan is to make it accessible from anywhere.

sign in / sign up system. Flask would be a good tool for construction of a simple application that does not necessarily follow an MVC structure.

With regard to functionality, we roughly divided our functions into two parts. One is the searching and browsing of the documents, which includes basic searching and filtering of data, document detail pages, and a login system for personalized recommendations. The other is visualization of data, which aims to assist users exploring distributions and correlations of attributes of documents, and developers analyzing user behavior and monitoring performance of clusters.

The main tool employed for visualization is Kibana, a tool that is designed specifically for Elasticsearch data visualization. Both our basic searching and visualization pages are able to exchange information with the Elasticsearch container through a RESTFul API. Furthermore, basic searching in our visual pages will also allow filters. Once a filter is applied, all the charts will be updated to only show the information of remaining data.

1.3 Problem/Challenges

- How to exchange data between our web application and Elasticsearch through Python, JavaScript, and Kibana?
- How to generate queries with a user's input?
- How can the system fully support both searching activities of both amateur users and professional users?
- How to generate and persist searching logs and user clicking logs?
- How to generate a secure and fully functional user login system?
- How are admin users different from general users?
- How to produce visualizations to help users understand our datasets?
- How to acquire and visualize system monitoring data?
- How to produce a self-checking system based on unit tests?
- How to evaluate our product?

1.4 Solution Developed

For the search interface part, the team decided to make use of an open-source library: React (Reactive Search). The library provides functions that can easily send requests to, and pull data from, Elasticsearch. Developers can generate Elasticsearch queries simply by filling in determined fields in a query template. It can also automatically generate decent CSS and animations for HTML elements. This enables the team to develop a user-friendly website with only a little knowledge of JavaScript and HTML programming.

Furthermore, there are users with different familiarity of the datasets as well as Elasticsearch. For example, first-time-users (amateur user) usually don't know about existing fields and searching tricks in Elasticsearch. For this kind of user, we provide auto-completion and a series of filters with unique value lists, so that amateur users can use our application just like a common search-engine interface. For those who know the field names in the datasets and are familiar with Kibana and Elasticsearh searching tricks, we also support ":" and "&&" operators in search queries, which work just the same way as in Kibana.

Elasticsearch also provides an API for Python. With the API, Python can communicate with Elasticsearch in the manner of a general HTTP request clause. To customize the format of user logs, actions of users (searching, setting filters, choosing a determined article, etc.) are first sent to Python for processing, and then saved to Elasticsearch, as well as to files in Ceph (for persistence). For the user login system, the team provides "register" and "login" interfaces for general users, and "delete user", "change username" and "change email" for admin users. For the sake of security, passwords are encoded before sending to a MySQL database. Email addresses are also requested during registration for identity validation.

After discussions with the data collection teams (CME and CMT), Pie chart, bar chart, sun burst chart, tag-cloud, heat-map, and line chart were chosen for article features. A system monitoring page is also built based on the flask monitoring dashboard, so that admin users can acquire more information about the performance of the web application.

Unit test is implemented with "unittest", a framework provided by the Python core library. More unit test functions should be produced for INT's CI/CD, so that we can guarantee that no push to GitHub brings errors to the project.

1.5 Future Work

- Complete unit tests for INT's CI/CD.
- Design evaluation module for our system.
- Build user behavior and system monitoring visualization pages.
- Add more features by working with the TML team, such as the recommendation system that they have developed with the use of the logs.
- Welcome to our Github to report issues and give feedbacks in the future.

2 Literature Review

2.1 The Importance of Information Retrieval Systems

The world is changing rapidly, not only in terms of dress, eating habits, transportation, etc., but also the way people search for information. In this so-called Information Age, hundreds of millions of people use web search engines to retrieve information every day. An information retrieval system can be considered as a system that reads the users' questions and performs a search through an internally stored collection of documents to present the most relevant information to the users [12]. Information retrieval was originally a part of librarianship. However, it has expanded into a variety of fields such as business management, office automation, genomic databases, fingerprint identification, medical image management, knowledge finding in databases, and multimedia management.

2.2 The Importance of Tobacco Settlement Dataset and Virginia Tech ETDs Dataset

Tobacco Settlement Dataset: In 1998, several million once secret documents, over 35 million pages, became available to the public as the result of legal action [8]. The documents include letters, memos, telexes, emails, and research reports; strategic, political, and organizational plans; organizational charts, lists of consultants, invoices, and copies of cheques paid; testimony in courts and before legislatures; advertising, marketing, media, and public relations strategies; and several other categories. Since 1998 the documents have been used in many ways to support the right of citizens to know how products, even legal products, affect their health, and to gain relief from past and continuing injury. The documents so used become a powerful weapon for protecting the public's health. These documents can continue to influence the public and are also used to show governments what the tobacco companies know about their products—and tried to hide—and so can lead to legislation intended to reduce the harm tobacco causes. Therefore, it is important to have more high quality platforms that people can search for these documents and understand the risks of smoking and what the tobacco companies want to hide [8].

Virginia Tech ETDs Dataset: An ETD is an electronic version of a thesis or dissertation. An ETD is formatted just like a traditional thesis or dissertation (with pagination, tables, figures, references, etc.), but is saved as a PDF file and submitted electronically to the university. An ETD makes a student's research immediately accessible to a broad audience, while reducing both printing and binding costs for the researcher. The ETDs of Virginia Tech benefit people in a variety of ways. For instance, research in VT's ETD system is widely disseminated and free to anyone, including scholars at under-resourced institutions and in developing nations, as long as there is an Internet connection. Also, these works will be preserved for decades or even centuries, unlike research posted on

a personal web page or published by a company that might go out of business. In a word, the ETD dataset is a great resource for researchers or other people to learn useful knowledge [11].

2.3 Realization of a Web-based Information Retrieval System

Elasticsearch, which is an open-source, distributed, and RESTful search engine, makes it significantly easier to build a search engine that can analyze a great amount of information effectively and deliver relevant results to users [4]. Elasticsearch is built to handle large amounts of data while maintaining user-friendly and accessible methods to retrieve data. There are a handful of APIs that allow applications from any language or framework to access an Elasticsearch index. Postman is the platform we use to guarantee the efficiency of API calls between our interface and Elasticsearch.

To ease the users' searching processes, multiple filters should be applied on the searching website for users to access the information, and present the most relevant information to the users [14]. To help users to digest the information faster, the design of the interface should be easy and appealing to use, and allow users to effectively find the answers they desire.

In addition, Kibana is a good way to search, view, and perform advanced data analysis with data stored in Elasticsearch indices using a REST API, and to create graphs and charts to help present aggregations for users to visualize in the field that they want [15]. It is built on top of Elasticsearch and provides a user-friendly interface that can easily handle a large amount of data, and helps create dashboards and query data in real-time.

3 Requirements

3.1 Introduction

Our goal is to build a user-friendly web interface, which provides several basic functionalities, clear and well-formatted layouts, and a feedback window for user support.

3.2 Functionalities

Below is the list of functionalities our team will provide:

- ETD searching
- Tobacco searching
- Recommendations for searching
- Content filtering by author, date, location, etc.
- List of content sorted by relevance score
- Detailed content found by clicking on any content anchor in the list
- Login page and sign up page
- Log system to record user activities
- Project introduction page with project overview, all six teams' profiles, and descriptions
- Pie charts

3.3 Quality/Performance

The quality of a front-end interface can be measured by considering appearance and functionalities. For appearance, the interface should be both concise and informative. "Concise" here means that the level and partitions of sections should be identifiable, so that users can easily find the information they are looking for. "Informative" indicates that the interface should provide enough messages for (1) operational guidance, (2) connections among records, and (3) connections between a record and the group containing it. For functionalities, the aim is to reduce the time when searching for a useful item. For example, a recommendation should be provided so that they can find related documents without thinking of a new query.

The performance of front-end pages can be assessed from three aspects:

1. basic elements loading time.
    - text loading time.

- image loading time.
- front-end functions running time.

2. front-end response time.
3. back-end response time.

The priorities of these three elements should be: texts>functions>images. Front-end response time includes the time consumed by a front-end function processing a user action. Back-end response time is the time interval between sending a request and getting a response from the back-end. Besides, the total time consumption and the changing rate of response time with the increment of concurrency should also be examined. All of the above measurements will be checked with online tools such as Siege (front-end time consumption) and Locust.io (server response time). The system will be modified until the total delay is less than 300ms.

3.4 User Support

3.4.1 User Manual

To make users have a positive navigating experience on our website, we will provide a user manual. The user manual provides detailed instructions about how to utilize functionalities on the website.

3.4.2 Articulatory and Semantic Feedback

Articulatory and semantic feedback will be given once users encounter any possible problems or errors such as those listed below:

- Invalid password or username on login page.
  Solution: printing out an error message "Invalid password or username."
- An email address has been registered by previous users.
  Solution: printing out an error message "This email address has been used."
- Connection error.
  Solution: navigate to the 404 page.

4 Design

4.1 Kibana Utilities

Kibana is a tool that we can use to search, view, and perform advanced data analysis with data stored in Elasticsearch indices, and visualize the data in a variety of tables, charts, histograms, tables, and maps [5]. It is built on top of Elasticsearch and leverages the functionalities of Elasticsearch. It is a part of the ELK stack, which consists of Elasticsearch, Logstash, and Kibana. It connects to Elasticsearch using the REST API. It provides a user-friendly interface that can easily handle a large amount of data and helps create dashboards that are easy to build. It aids advanced users who query data in real-time. In our project, we will utilize Kibana to create a series of charts and graphs, including pie charts, bar graphs, line graphs, etc. A collection of these visualizations related to the ETD dataset and tobacco dataset will be provided on our website for users to explore. Users can search across all documents with any filters or text. Kibana also supplies many other features. Inspection of the network traffic on Kibana also will be conducted to see how it works with Elasticsearch.

4.1.1 Kibana Aggregations

Aggregations represent a collection of documents from a specific search query, and can lead to desired visualizations in Kibana. Some of the important aggregations include metric aggregation and bucket aggregation. Metric aggregations compute metrics based on values extracted in a specified way from the fields of documents. Bucket aggregations build buckets of documents depending on certain criteria. All types of aggregations are listed below. In the visualization builder, we can choose metric aggregation for the visualization's Y-axis, and bucket aggregation for the X-axis.

4.1.2 Charts

To create a Kibana visualization, a variety of choices are available to choose from, such as charts, data, metrics, maps, and time series. Each of them has its own focus on how to present extracted data.

Basic Charts:

- Pie charts are used to display the parts of some whole. In Kibana, they are determined by metrics aggregation.
- Vertical bar charts are used for clarifying trends with a large dataset. Kibana determines it by metrics aggregation.
- A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors. The color for each matrix position is determined by the metric aggregation.

Data:

- A data table is used to display the raw data of a composed aggregation.
- Goal and gauge visualization enables displaying the progress toward an identified goal.

Maps:

- A coordinate map associates the results of aggregation with geographic locations.
- Thematic maps use shapes' color intensities to represent a metric's value.

Time Series:

- Timelion is a visualization tool for time series in Kibana. It has the ability to add multiple time series to one chart.
- Time Series Visual Builder allows you to combine an infinite number of aggregations and pipeline aggregations to display complex data in a meaningful way.

4.2 Postman Utilities

Postman is a collaborative platform for API development. Users can quickly and easily send REST requests directly within Postman. In our case, we use Postman to quickly check Elasticsearch APIs and try to display the response data in a helpful manner. Figure 1 shows the request demo for accessing Elasticsearch via Postman. Postman helps with understanding the requests to Kibana and Elasticsearch, allowing us to structure our back-end system more quickly and efficiently. Users also can use Postman directly to search for data on Elasticsearch [2].

4.3 Build Our Own User-friendly Website

Our main goal is to build a user-friendly website for users to retrieve data from Elasticsearch. The website architecture should include back-end and front-end systems. Our back-end system is mainly for the user system and log system in our website infrastructure. Elasticsearch is also the back-end for us to get data and collaborate with other teams.

4.3.1 Infrastructure

As shown in Figure 2, we depict the whole infrastructure of our website. It consists of 5 main modules; in each module there are several parts. We would

[2]This link provides examples for how to create and get from an Elasticsearch index using Postman [9].

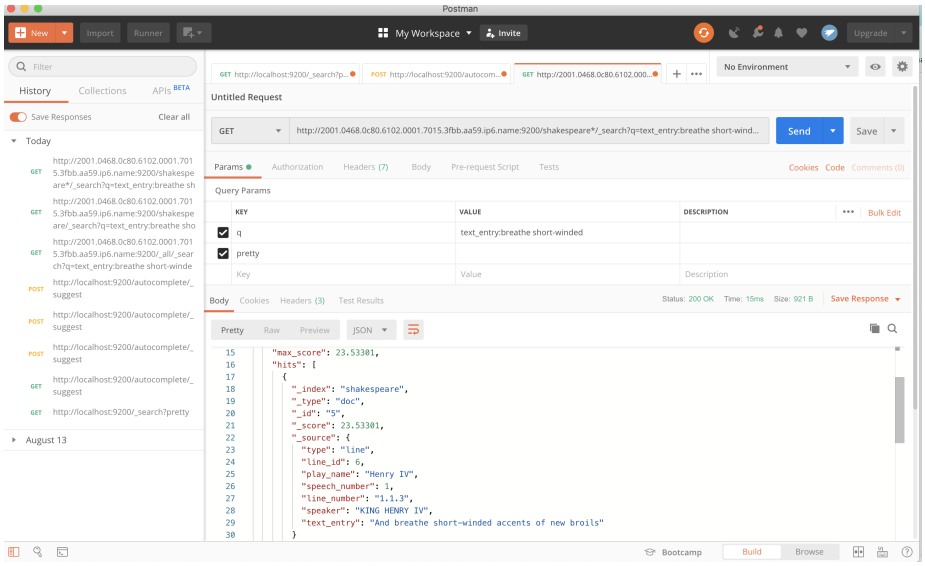

Figure 1: Postman request

also have connections to MySQL, Elasticsearch, and Kibana for storing and retrieving data.

4.3.2 Back-end

For the back-end of this system, Python and MySQL will be primarily used. Data and documents are retrieved from the datasets by using the Elasticsearch API.

Connection to Kibana and Elasticsearch: The connection to Kibana and Elasticsearch is mainly executed by the Python request package and REST APIs on Kibana and Elasticsearch. This is direct and simple, but it could not provide us a convenient way to build a whole searching page with fields and filters. Accordingly, we use ReactiveSearch, which is an Elasticsearch UI components library for React and React Native. This allows us to use CORS (Cross-Origin Resource Sharing) HTTP requests to connect to Elasticsearch. Basically, it still works on the original REST APIs of Elasticsearch. For example, we send our searching queries to Elasticsearch via API call: " POST /index/_msearch? ". The details are discussed in Section 7.

Connection to Front-end: We use Python Flask to connect our back-end system to the front-end system. We use Flask Render to build our website on the targeted URL path. Plus, some data processing would be done in the back-end system and then transferred to the front-end to be displayed. Then,

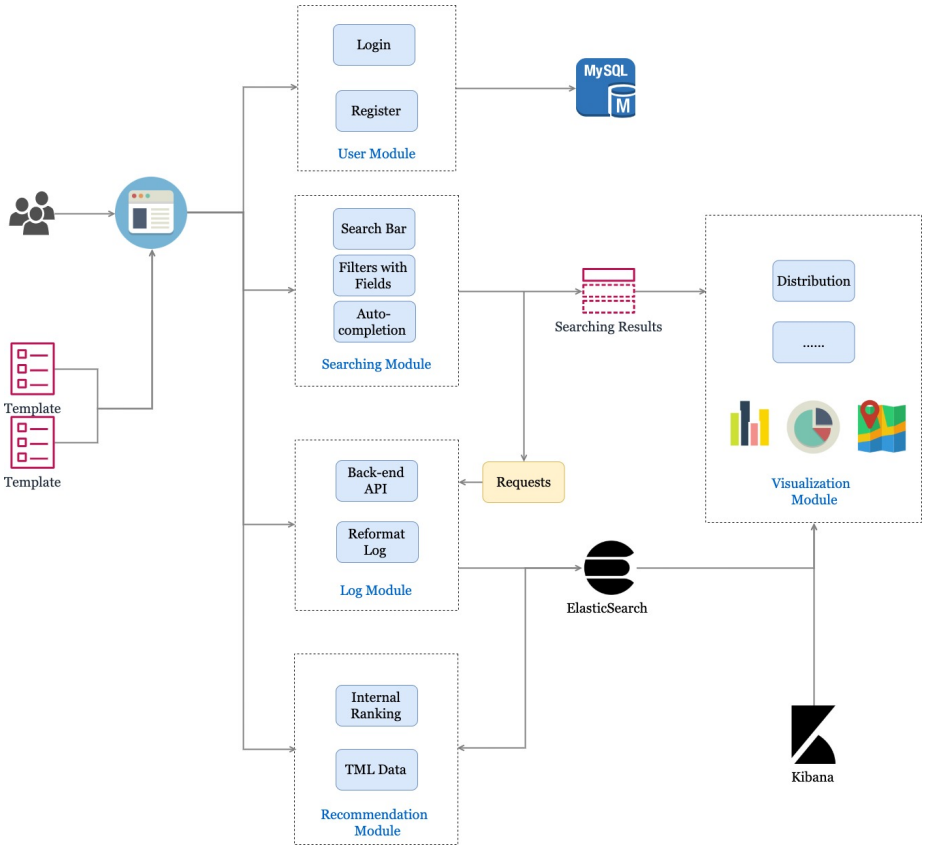

Figure 2: Infrastructure

its JavaScript code would further process the received data and display the expected format. Additionally, we write the POST API to receive log data posted from the front-end system.

User login system: Users may register on the website, and the user information will be stored in MySQL. We require standard input for usernames such as a unique name in the database that has a minimum of 4 characters and a maximum of 20. We run a MySQL container on the cloud server and use HTTP to connect to MySQL. Considering it would lose user data once the machines shut down, we are going to persist user data with Ceph. The user validation should also be processed in this system. We save and retrieve data in MySQL to maintain the user system.

4.3.3 Front-end

Utilizing the Flask framework, our website runs on Python, JavaScript, HTML, and CSS. Our homepage contains navigation bars and a login/register button for users. Also, considering we have two different datasets, we offer two searching pages for users to choose from. There are two links for two searching pages on the homepage. In each searching page, there are three parts presented to the user. There is a search bar, filters with fields, and a results block.

User login: The entry offers users the ability to register and login to our system. From their profile page, users can see their searching history on the website. There will also be a logout button. Users who do not want to create an account will still be able to take advantage of many of the features the website will offer.
There are two kinds of users in our system: normal users and admin users. Normal users would have the ability to search, and to look at visualization pages. Admin users would have access to the admin page where they can be CRUD users in MySQL. Also, admin users can upload new files to two datasets and they can employ the monitoring dashboard where they can monitor all APIs in this project.

Searching bar: Users send queries by putting some text in the search bar. Possible suggestions are displayed as the user enters characters. The auto-complete part is also based on the response from Elasticsearch. The auto-completion would start from the third character that a user types in. Additionally, we support searching in specific fields via "fields" + "text", like "title: march". We also set different weights for various searching fields, e.g., we recognize that title is more important than other fields when users search for something.

Filters: Users will be able to self-define filters to request specific results. For example, users could limit their searching fields by selecting some target fields. This part would also show details of each field. For example, it would show the number of related results in such fields.

Results display: The searching results will be displayed here. It would automatically be updated as long as searching requests change. When the user clicks one of the filters, then the results block would refresh immediately. At the top of the results block, it shows the total number of results and the time cost. This will give users an intuitive feeling of searching speed and quantity. For users' convenience, we add a hyperlink on each title of results. Users could click on the title and access the original PDF file. Once a user clicks on the title and redirects to the original PDF file, we will record those activities in the log.

4.4 Website Glance

4.4.1 Homepage

The homepage is shown as Figure 3.

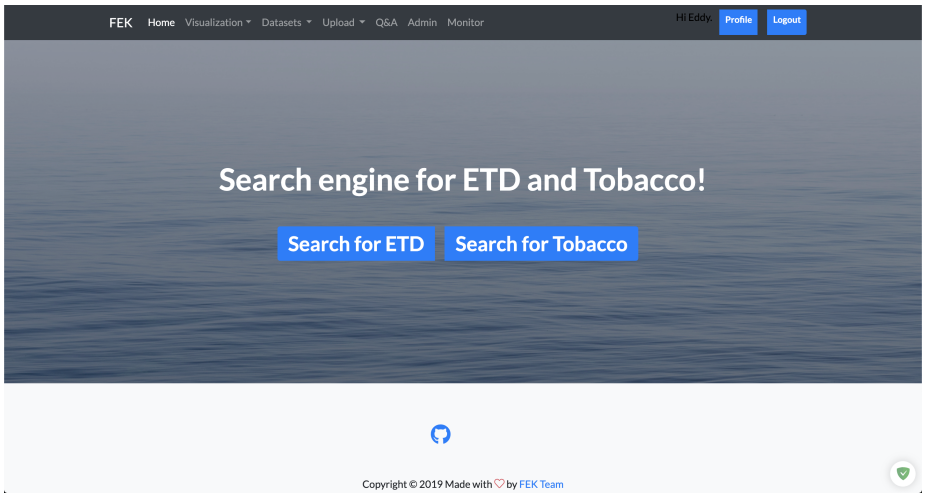

Figure 3: Home Page

4.4.2 User Login and Register

The login page is shown as Figure 5, and the registration page is shown as Figure 4.

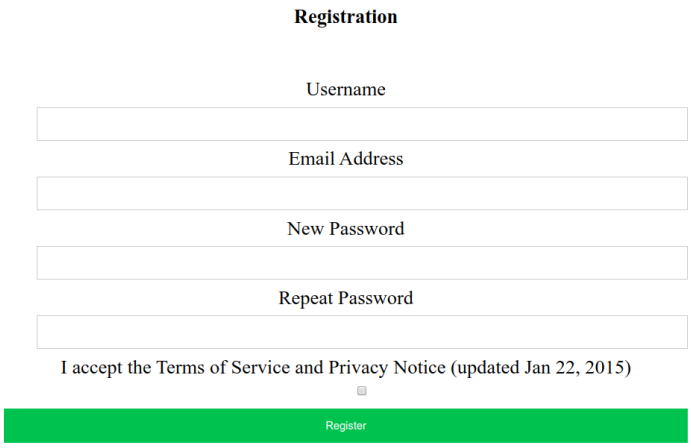

Figure 4: Registration Page

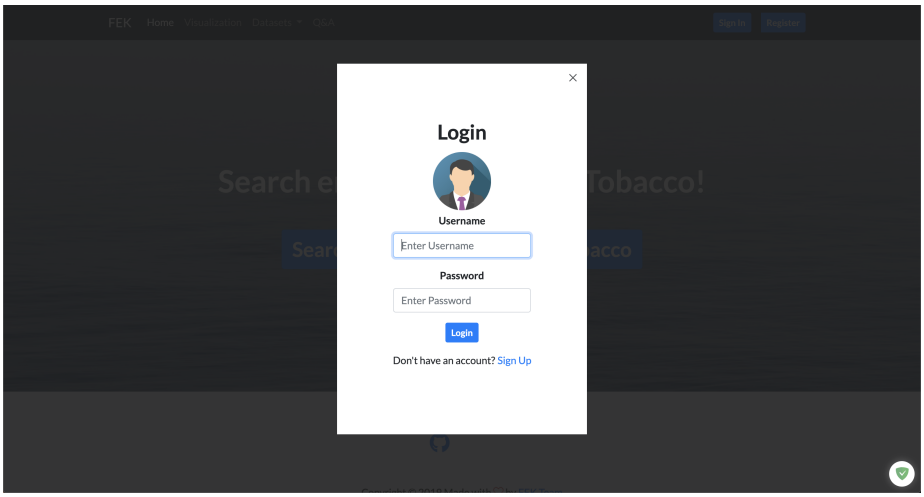

Figure 5: Login Page

4.4.3 Searching Page

There are two searching pages. One is for searching the ETD dataset; the other is for searching the tobacco dataset. The ETD searching page is shown as Figure 6. The tobacco searching page is shown as Figure 7.

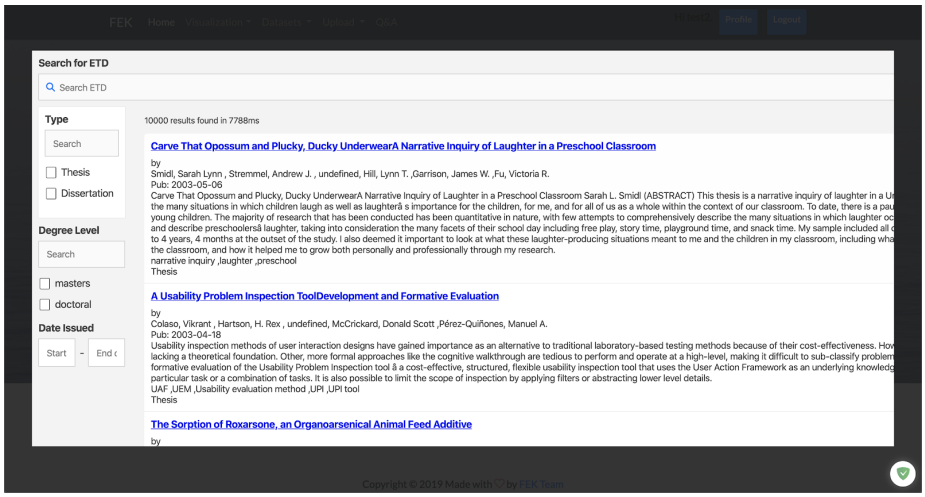

Figure 6: ETD Searching Page

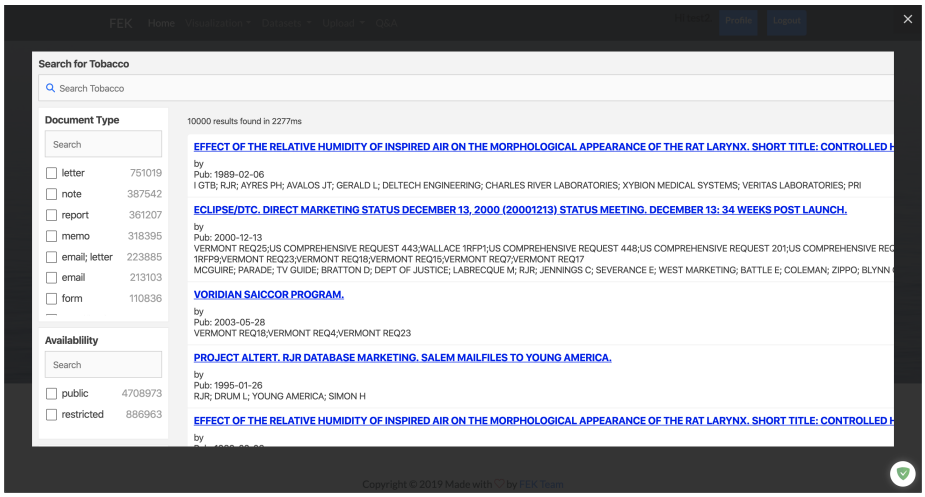

Figure 7: Tobacco Searching Page

Figures 8 and 9 show some examples for searching in the two datasets.

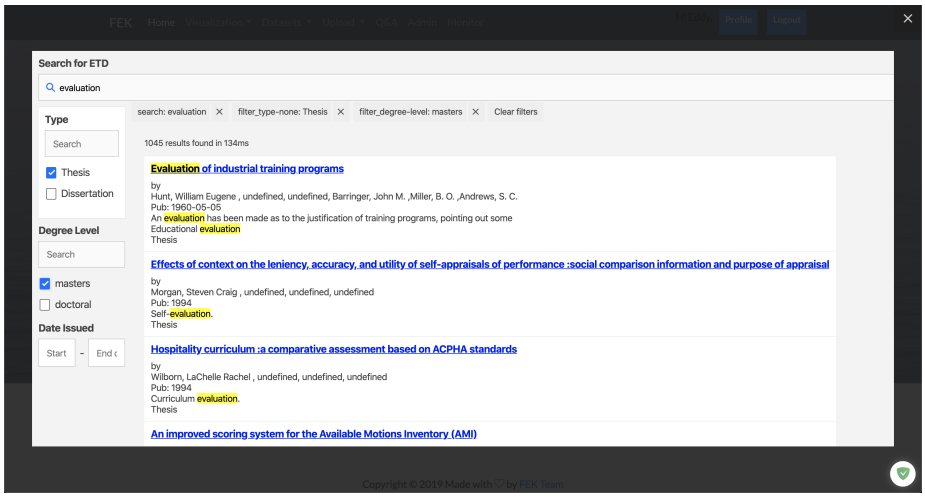

Figure 8: ETD Searching Page

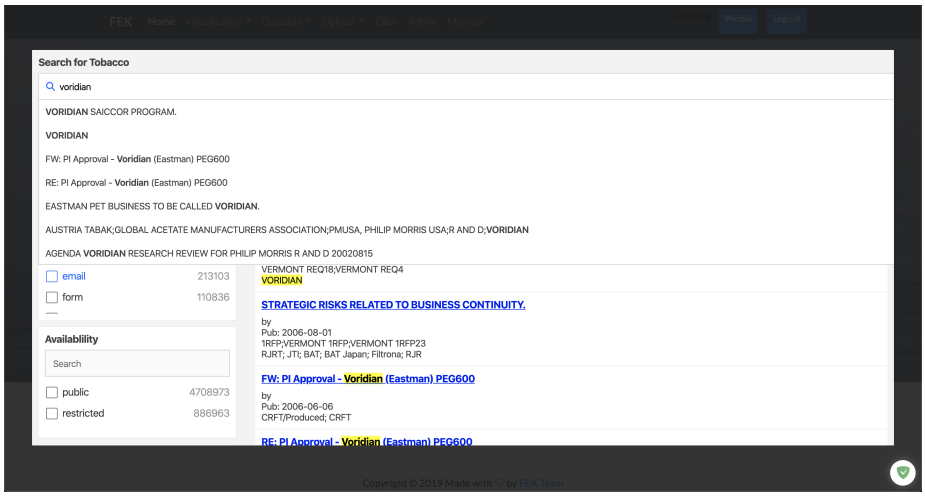

Figure 9: Tobacco Searching Page

4.4.4 Visualization

We have built two visualization pages. One is for ETD settlement documents and the other is for tobacco settlement documents. Different types of visualization have been applied, such as charts, data tables, and maps.

The visualization of ETD data is an interactive, comprehensive, and graphic page for users to have a better understanding of the data distribution by time, departments, degree level, and degree name. Visualizations are shown in Figure

10.

The visualization of tobacco data contains several types of graphs such as Data Tables, Tag Graph, Pie Chart, Area Graph, and Gauge. These are shown in Figure 11. The keywords utilized mainly include brands, cases, languages, and topics.

Several steps are included in the process of creating visualizations. They are shown below, as well as in Figure 12.

- Select the proper type of graph for visualizing the data.
- Pick up a data set.
- Add buckets or aggregations for the graph.
- Add the created visualization into the dashboard.

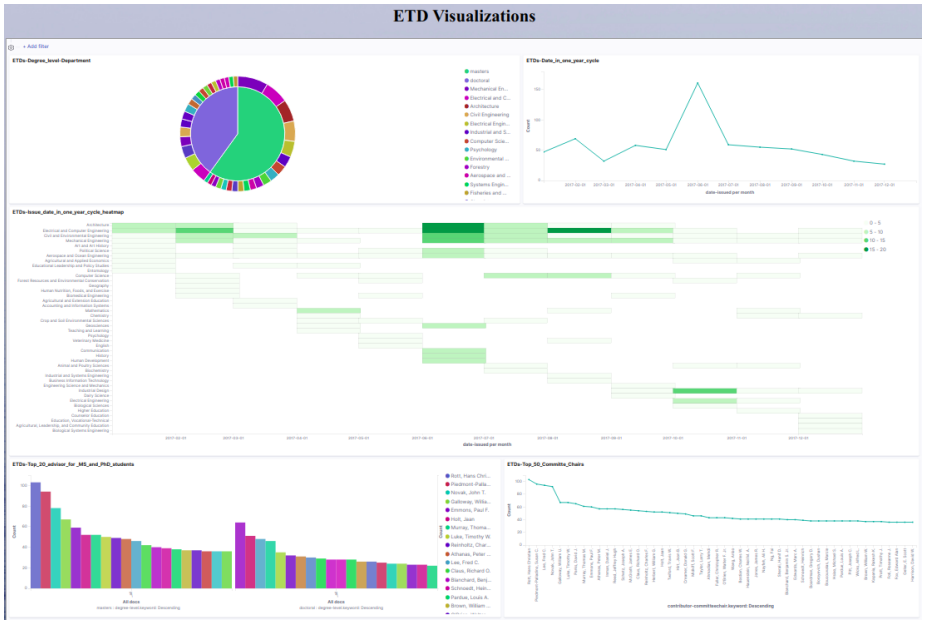

Figure 10: ETD Visualization Page

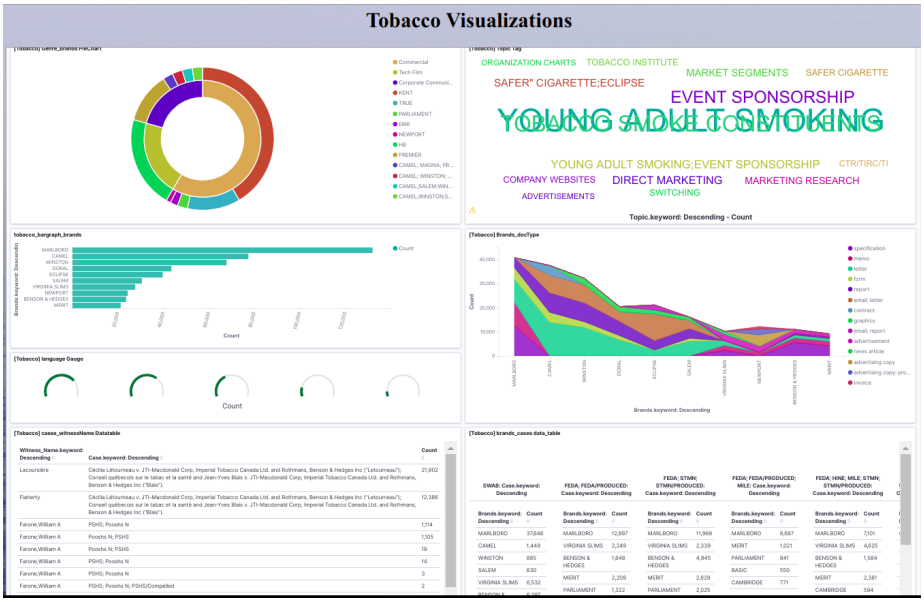

Figure 11: Tobacco Visualization Page

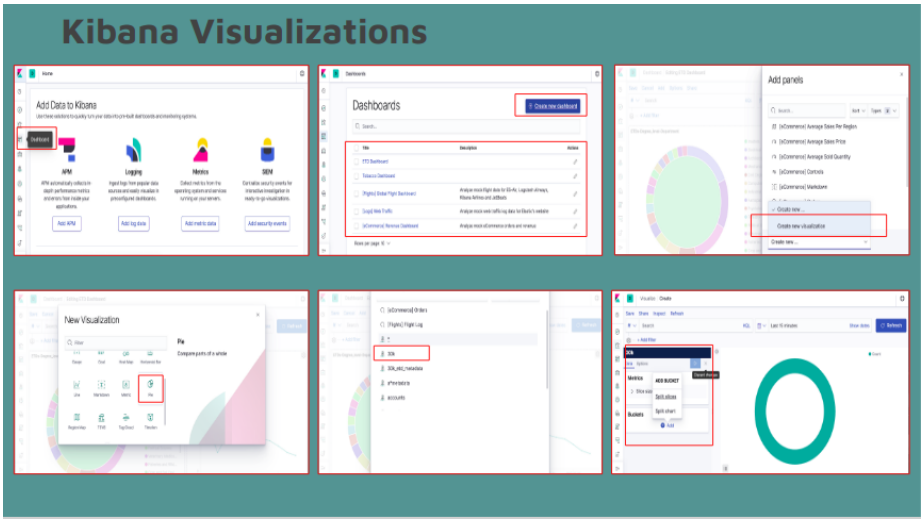

Figure 12: The Process of Creating a Visualization

4.5 Log System

Each time the user searches something in the searching bar, it will send a request to Elasticsearch. We jsonify the requests data and send it to the

back-end API we created for processing requests data. As the request data are poorly summarized and do not completely overlap with what should be in the log record, we re-format the JSON data by combining the JSON and user-session information, and merging query items. Finally, the re-formatted JSON data is sent to Elasticsearch for persistence and further analysis.

To determine what exact documents users are searching, we also record click activities when users select a hyperlink associated with a title. The JSON log data will contain user information and document information. Those kinds of logs should be used for better user recommendation.

These two types of logs are stored in separate indices in Elasticsearch. The logs are also saved in JSON files in Ceph as backups.

4.6 Recommendation

Search quality evaluation starts with looking at the users of our search application and taking into account the things for which they are searching. The challenge is to tweak this translation process from user entries to a concrete query, leading to search results that contain the most relevant information with respect to a user's information need. There are two alternatives.

4.6.1 Ranking System in Elasticsearch

Basically, we use the default ranking of response data from Elasticsearch. The ranking evaluation API [5] allows users to evaluate the quality of ranked search results over a set of typical search queries. It supports various evaluation metrics for ranking. Currently, the following metrics are supported.

Precision at K (P@k): This measures the number of relevant results in the top k search results. It is a form of the well-known precision metric that only looks at the top k documents. It is the fraction of relevant documents in those first k results.

Mean Reciprocal Rank: For every query in the test suite, this metric indicates the reciprocal of the rank of the first relevant document. For example, finding the first relevant result in position 3 means the reciprocal rank is 1/3. The reciprocal rank for each query is averaged across all queries in the test suite to give the mean reciprocal rank.

Discounted Cumulative Gain (DCG): In contrast to the two metrics above, discounted cumulative gain takes into account both the rank and the rating of the search results. The assumption is that highly relevant documents are more useful for the user when appearing at the top of the result list. Therefore, the DCG formula reduces the contribution that high ratings for documents on lower search ranks have on the overall DCG metric.

Expected Reciprocal Rank (ERR): Expected Reciprocal Rank (ERR) is an extension of the classical reciprocal rank for the graded relevance case [2]. It is based on the assumption of a cascade model of search, in which a user scans through ranked search results in order, and stops at the first document that satisfies the information need.

4.6.2 Recommendation Data from TML Group

The TML group is still working on this part. We have not been notified about data content so we have not collaborated yet. Basically, it could play a role similar to the internal ranking system in Elasticsearch but with improvements and greater accuracy. Additionally, we could list some specific recommendations for specific users.

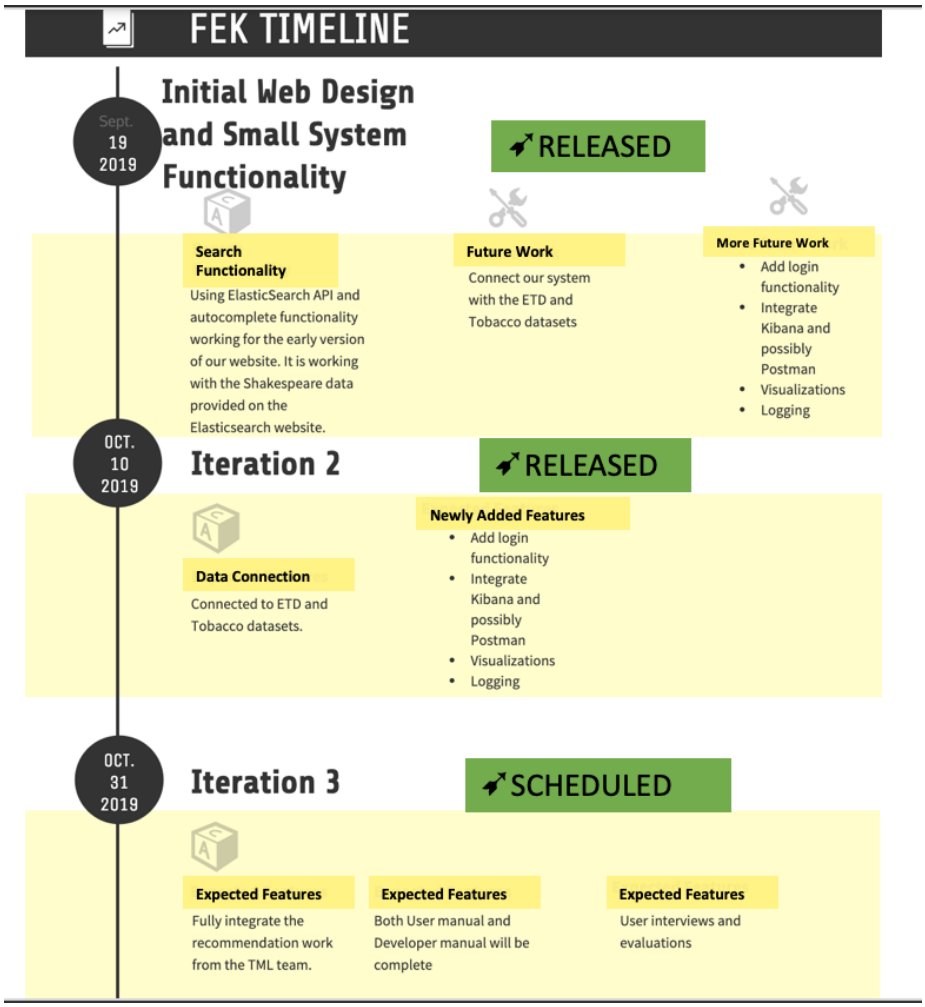

Figure 13: Timeline

5 Implementation

The tools that we utilized for our implementation were primarily: Kibana, Python, JavaScript, HTML, CSS, and Elasticsearch.

5.1 Milestones

5.1.1 Completed

- System running with Shakespeare data
- System running with a subset of both datasets
- System running with the full ingestion of both datasets
- Shakespeare data running with Kibana and testing its various modules and visualizations
- Shakespeare data tested with Postman
- Login and Register system
- Administrator has standard CRUD operations
- Filters that are specific to the ETD and Tobacco datasets are shown as facets in our interface
- Put our system into the CS testing container cluster
- Connected Kibana to Tobacco and ETD datasets
- Searching pages connected to two datasets: Tobacco and ETD
- Hyperlinking to original documents implemented
- Logs generated for TML team
- ETD dataset has an editable dashboard with a set of visualizations connected with Kibana.
- Tobacco dataset has an editable dashboard with a set of visualizations connected with Kibana.
- Built a framework for a FAQ page

5.1.2 To Be Completed

- Implement an improved recommendation system based on the TML team's work
- Administrator upload for a new file for both datasets is ready but a back-end endpoint is needed from the INT group.
- A profile page that users can utilize and navigate should be developed. This page should also allow users to recover and change their password.
- A small user study to test the feasibility and usability of the website's functionality and UX

5.2 Deliverable

Our main deliverable is the front end website itself. This deliverable is paramount. It enables access to the entire class project and will help display all of the teams' work.

5.3 Intra-Evaluation Process

For Elasticsearch request/response simulation, we have tested out Postman [1] and dejavu [17]. Postman was eventually chosen as it had a concise interface and JSON could help us understand the structure of data. Elasticsearch-GUI [3] and ElasticHQ [16] were also considered for the template of our searching page. However, the team finally decided to develop the pages by themselves, as it would be more flexible and the underlying functionalities could be easier to understand. Initially, our group was going to use and had an early version of our website, that utilized PHP, but we decided to use Python because most of the group members are familiar with it, potential visualizations are easily obtainable, and there is an abundance of documentation for the language.

5.4 User Log Persistence

There are mainly three functions to be considered regarding user log persistence. The first one is passing the request information and user information from the front-end to the back-end (the "bridge"). Then queries generated by the front-end instead of user activity need to be discarded, while the remaining requests should be reformatted. Finally, the reformatted JSON data is sent to Elasticsearch, as well as to Ceph (for persistence).

In the first function, searching queries and article choosing requests are sent separately. The former are sent during rendering, while the latter are sent by an "onClick" function through AJAX.

Although only searching and article choosing actions need to be recorded in the logs, there are many different types of requests sent to the back-end when a user interacts with the searching page. For example, to load the initial list of articles, the front end needs to send a "match all" request to Elasticsearch.

To achieve “auto-completion” while a user is typing in the searching bar, the front-end is asked to keep sending ”prefix” queries to Elasticsearch. Therefore, a filter is supposed to be set so that the log could only contain a completed search or article choosing action.

After acquiring the request bodies, the useful information is extracted and reformatted as log objects in JSON format. The log is then stored in Elasticsearch through its API and in a file in Ceph through an OS operation.

6 User Manual for Website

The user manual is intended to help users navigate the website and understand the interface. The figure below depicts the basic workflow for an end user on the website.

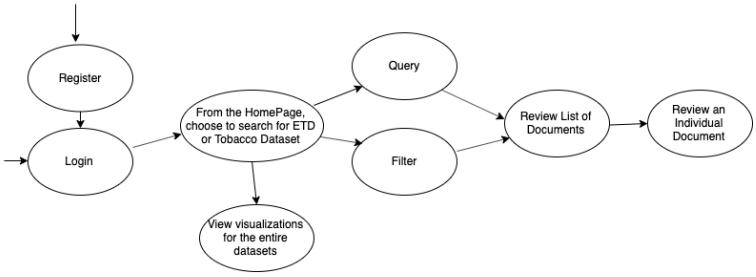

Figure 14: User Workflow

6.1 A User Interested in the Tobacco Dataset

6.1.1 Scenario 1

1. A user will register for an account.
2. A user will login to their account.
3. From the homepage, a user will click on "Search for Tobacco".
4. A user can begin to type in a keyword such as "Reynolds" to help narrow their search.
5. Note: A user could click on a document from the auto-complete search bar.
6. A user enters their search query and reviews the list of documents.
7. A user clicks on a document to review the original PDF version.

6.2 A User Interested in the ETD Dataset

6.2.1 Scenario 2

1. A user will register for an account.
2. A user will login to their account.
3. From the homepage, a user will click on "Search for ETD".
4. A user can begin to type in a keyword such as "Doctoral" to help narrow their search.

5. Note: A user could click on a document from the auto-complete search bar.
6. A user enters their search query and reviews the list of documents.
7. A user clicks on a document to review the original PDF version.

Figure 15 shows the basic database schema for our users table at the moment. There are plans to add more fields that will be useful for the TML team as well as help log the system status.

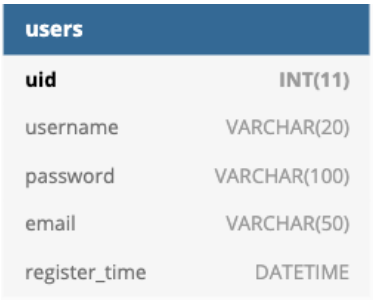

Figure 15: Database Schema for Users Table

6.3 User Manual for Kibana

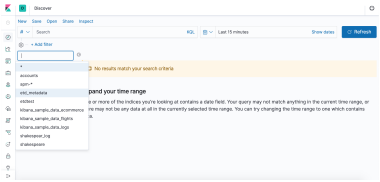

Figure 16: Kibana Discover

To access Kibana that is connected to our Elasticsearch indices, a user must be on the VT network to access our website [3] The Discover tab is the easiest way for a novice to interact with the datasets by taking advantage of the graphical user interface of Kibana with preset filters. Once on the Discover tab, a user needs to select a dataset as is shown in Figure 16. From here, users can filter their results and use the drop-down bar to see all the fields in the index. Additionally, multiple fields can be filtered at one time.

[3]This is the present website link: http://2001.0468.0c80.6102.0001.7015.b2eb.3731.ip6.name:3000. Currently, the website can only be accessed on Virginia Tech's network. Furthermore, one can VPN into Virginia Tech's network if one needs access remotely. In the future, there will be a different link where the project is stored, and the plan is to make it accessible from anywhere.

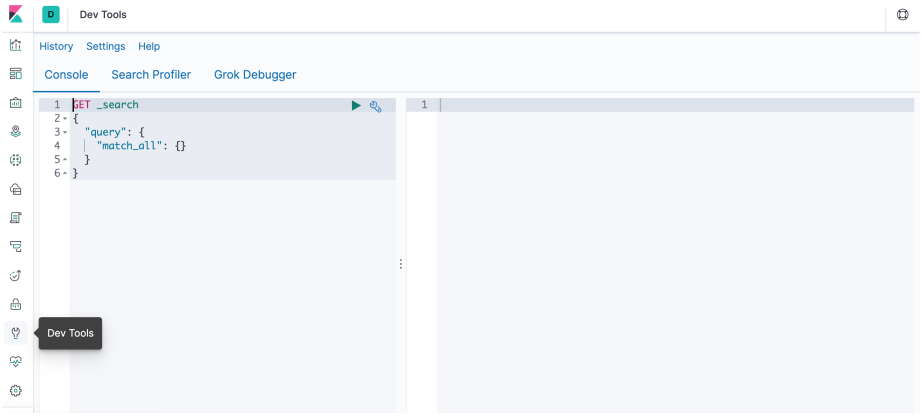

Figure 17: Kibana Dev Tools

For a more robust and fine-tuned way to control the data, a user can click on the tool gear and manipulate the indices with more complex queries. The page is displayed in Figure 17. These queries have a lot more potential, power, and flexibility than the defaults provided in the Discover page. Weights of various fields can also be "boosted" here.

Logs that show the usage of Elasticsearch can be found through Kibana. These are automatically generated. This could also be useful for the TML team. Kibana has some visualizations that help to aggregate the data. More details of this were explained in the Visualization Section 4.4.4.

6.4 User Manual for FAQ Page

A frequently asked questions (FAQ) page has been developed on the website to help users understand the significance of this website and how to use it.

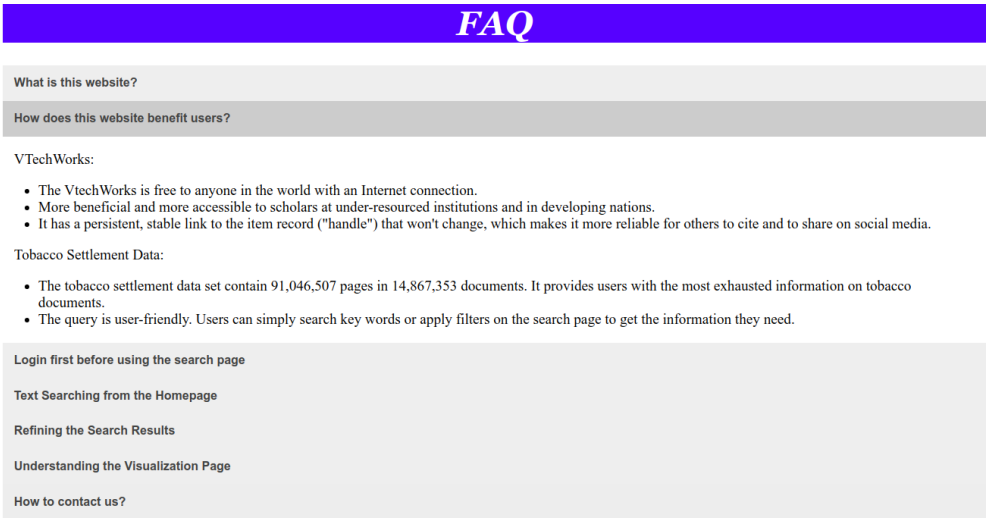

Figure 18: FAQ Page

7 Developer Manual

7.1 Workflow

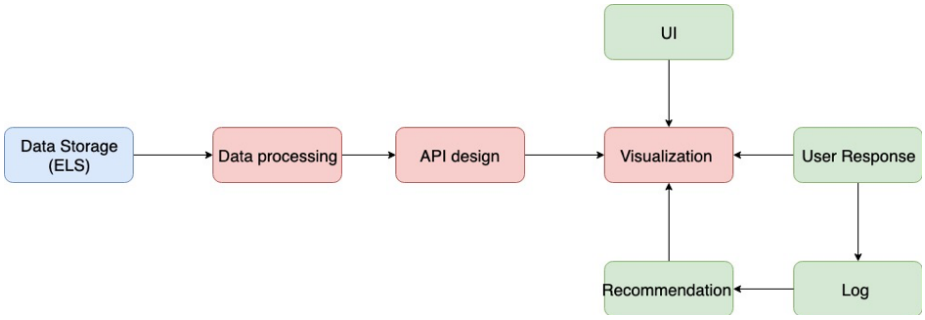

Figure 19: FEK Team Workflow

The workflow of our team is shown in Figure 19.

7.2 Data Exploration

Data exploration for an index can be discovered using Elasticsearch. The documentation can be found here.

Kibana has even better tools to explore an Elasticsearch index. Documentation for the data exploration can be found here and documentation for visualizations can be found here.

7.3 Kibana Aggregations

We focus on metric aggregations and bucket aggregations currently.

Metric Aggregation: Metric aggregations compute metrics based on values extracted in a specified way from the fields of documents. Metrics enable a variety of functionalities. The count, average, sum, min, max, cardinality, percentiles, and percentile rank aggregations are shown in Figures 20 to 27, respectively.

```
{
    "aggs" : {
        "grades_count" : { "value_count" : { "field" : "grade" } }
    }
}
```

Figure 20: Count: return a raw count of the elements in the selected index pattern

```
{
    "aggs" : {
        "avg_grade" : { "avg" : { "field" : "grade" } }
    }
}
```

Figure 21: Average: return the average of a numeric field

```
"query" : {
    "constant_score" : {
        "filter" : {
            "range" : { "timestamp" : { "from" : "now/1d+9.5h", "to" : "now
        }
    }
},
"aggs" : {
    "intraday_return" : { "sum" : { "field" : "change" } }
}
```

Figure 22: Sum: return the total sum of a numeric field

```
{
    "aggs" : {
        "min_price" : { "min" : { "field" : "price" } }
    }
}
```

Figure 23: Min: return the min value of a numeric field

```
{
    "aggs" : {
        "max_price" : { "max" : { "field" : "price" } }
    }
}
```

Figure 24: Max: return the max value of a numeric field

```
{
    "aggs" : {
        "author_count" : {
            "cardinality" : {
                "field" : "author"
            }
        }
    }
}
```

Figure 25: Cardinality: return the number of unique values of a field

```
{
    "aggs" : {
        "load_time_outlier" : {
            "percentiles" : {
                "field" : "load_time"
            }
        }
    }
}
```

Figure 26: Percentiles: divide the values in a numeric field into the specified percentile bands

```
{
    "aggs" : {
        "load_time_outlier" : {
            "percentile_ranks" : {
                "field" : "load_time",
                "values" : [15, 30]
            }
        }
    }
}
```

Figure 27: Percentile Rank: returns the percentile rankings for the specified values in the numeric field

Bucket Aggregation: Bucket aggregations build buckets of documents depending on certain criteria. All types of these aggregations are listed below.

```
GET /_search
{
    "aggs" : {
        "price_ranges" : {
            "range" : {
                "field" : "price",
                "keyed" : true,
                "ranges" : [
                    { "key" : "cheap", "to" : 100 },
                    { "key" : "average", "from" : 100, "to" : 200 },
                    { "key" : "expensive", "from" : 200 }
                ]
            }
        }
    }
}
```

Figure 28: Range: specify ranges of values for a numeric field

```
POST /sales/_search?size=0
{
    "query" : {
        "constant_score" : { "filter": { "range" : { "price" : { "to" : "500
    },
    "aggs" : {
        "prices" : {
            "histogram" : {
                "field" : "price",
                "interval" : 50,
                "extended_bounds" : {
                    "min" : 0,
                    "max" : 500
                }
            }
        }
    }
}
```

Figure 29: Histogram: specify intervals for a numeric field

```
GET /_search
{
    "aggs" : {
        "products" : {
            "terms" : {
                "field" : "product",
                "size" : 5
            }
        }
    }
}
```

Figure 30: Terms: specify the top or bottom n elements of a given field to display

```
GET logs/_search
{
  "size": 0,
  "aggs" : {
    "messages" : {
      "filters" : {
        "filters" : {
          "errors" :   { "match" : { "body" : "error"   }},
          "warnings" : { "match" : { "body" : "warning" }}
        }
      }
    }
  }
}
```

Figure 31: Filter: specify a set of filters for the data

```
{
    "aggs" : {
        "ip_ranges" : {
            "ip_range" : {
                "field" : "ip",
                "ranges" : [
                    { "to" : "10.0.0.5" },
                    { "from" : "10.0.0.5" }
                ]
            }
        }
    }
}
```

Figure 32: IPv4 Range: specify ranges of IPv4 addresses

7.4 Deployment

We have made careful and detailed documentation in our repository for the project that explains how to deploy our website both locally and also remotely using the Kubernetes & Docker platforms.

7.5 Requests Sample

The Code 1 is the request demo for searching. The request is sent to Elasticsearch (for retrieving searching results) and the back-end (for persistence of searching logs) separately. The back-end will receive the jsonfy data through "/emitlogs" and then send a re-formatted JSON (Code 2) with the es.index() function.

```
POST /index/_msearch?
{
   "preference":"List"
}{
   "query":{
      "bool":{
         "must":[
            {
               "bool":{
                  "must":[
                     {
                        "bool":{
                           "should":[
                              {
                                 "multi_match":{
                                    "query":"asd",
                                    "fields":["text_entry"],
                                    "type":"best_fields",
                                    "operator":"or",
                                    "fuzziness":0
                                 }
                              },
                              {
                                 "multi_match":{
                                    "query":"asd",
                                    "fields":["text_entry"],
                                    "type":"phrase_prefix",
                                    "operator":"or"
                                 }
                              }
                           ],
                           "minimum_should_match":"1"
                        }
                     }
                  ]
               }
            }
         ]
      }
   },
   "highlight":{
      "pre_tags":["<mark>"],
      "post_tags":["</mark>"],
      "fields":{
         "text_entry":{}
      },
      "require_field_match":false
   },
   "size":20
}
```

Code 1: Searching Request

```
/emitlogs
{
   "status":200,
   "message": "Success",
    {
       "data":{
          "user":{
             "username": "username",
             "email": "email"
             },
            "activity":{
                "url": "http://url_Elasticsearch:9200/index/_msearch?",
                "search_text": "asd",
                "filters": {"text_entry": "asd"},
            }
        },
        "dataset": "dataset",
        "time": "time",
        "ip": "user_ip"
    }
}
```

Code 2: Searching Log

References

[1] Abhinav Asthana, Ankit Sobti, A. K. Postman. https://www.getpostman.com/, 2018. Accessed: 2019-12-10.

[2] Chapelle, O., Metlzer, D., Zhang, Y., and Grinspan, P. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management* (2009), ACM, pp. 621–630.

[3] Coenradie, J. Elasticsearch-GUI. http://www.gridshore.nl/esgui, 2015. Accessed: 2019-12-10.

[4] Divya, M. S., and Goyal, S. K. Elasticsearch: An advanced and quick search technique to handle voluminous data. *Compusoft 2*, 6 (2013), 171.

[5] Elastic. ElasticSearch Ranking Evaluation API. https://www.elastic.co/guide/en/elasticsearch/reference/current/search-rank-eval.html, 2019. Accessed: 2019-12-10.

[6] Fackbook. ReactJS. https://reactjs.org, 2019. Accessed: 2019-12-10.

[7] Greif, S. Front-end Frameworks-Overview (2018). https://2018.stateofjs.com/front-end-frameworks/overview/, 2018. Accessed: 2019-12-10.

[8] Hirschhorn, N., Initiative, W. T. F., et al. The tobacco industry documents: what they are, what they tell us, and how to search them: a practical manual.

[9] Hoy, J. Getting Started With Elasticsearch. https://blog.red-badger.com/2013/11/08/getting-started-with-elasticsearch, 2016. Accessed: 2019-12-10.

[10] JetBrain. The State of Developer Ecosystem Survey in 2018. https://www.jetbrains.com/research/devecosystem-2018/javascript/, 2018. Accessed: 2019-12-10.

[11] Libraries, V. T. U. ETDs: Virginia Tech Electronic Theses and Dissertations. https://vtechworks.lib.vt.edu/handle/10919/5534, 2019. Blacksburg, VA 24061. Accessed: 2019-12-10.

[12] Manning, C., Raghavan, P., and Schütze, H. Introduction to information retrieval. *Natural Language Engineering 16*, 1 (2010), 100–103.

[13] Mönnich, A. Flask. http://flask.palletsprojects.com/en/1.1.x/, 2018. Accessed: 2019-12-10.

[14] Niu, X., and Hemminger, B. Analyzing the interaction patterns in a faceted search interface. *Journal of the Association for Information Science and Technology 66*, 5 (2015), 1030–1047.

[15] Robles-Gómez, A., Ros, S., Martínez-Gámez, A., Hernández, R., Tobarra, L., Pastor, R., Caminero, A. C., and Cano, J. Using Kibana and Elasticsearch for the recommendation of job offers to students. In *LASI-SPAIN* (2017), pp. 93–99.

[16] Russo, R. ElasticHQ. http://www.elastichq.org/, 2018. Accessed: 2019-12-10.

[17] Samani, L. dejavu. https://opensource.appbase.io/dejavu/, 2018. Accessed: 2019-12-10.