**1.1 Report**

The report should provide documentation about the following:

• **Why**: Explain the purpose of the search engine and what difference you can make.

* The goal of the project is to create a search engine that revolves on the dataset that was obtained from Kaggle. The search engine will be capable of accepting a query and filters as a request from the user. Then, it will give out a list of movies which have the closest similarity to the query while at the same time filtering based on the requirements of the user. The similarity score is computed via BM25, a scoring similar to TF-IDF. Moreover, specific fields can be boosted based on the importance decided by the researchers. Thus, the output is a list of movies which show most of the important details in one screen. This makes it easier to view the list of movies compared to most search engines which just show the title, year, description, and rating. The UI is not flashy, but the results of the search engine are direct to the point which will cater to the user who just wants the list and its details, with no added clicks to navigate to different information about the movie. However, the titles are hyperlinked to the IMDB website if they do choose to get a more detailed view of the title.

• **What**: Describe the data and domain in which you build the search engine index.

Please provide details about where the data source is and how you collected the data.

* The dataset is ‘IMDb movies.csv’ which comes from Kaggle (https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset). It contains 81,273 movies with 22 attributes. However, the researchers trimmed the said dataset to retain 12 attributes. Other attributes were removed because they were either missing too much information or they were not useful for the purpose of the project.
* Using Kibana’s elasticsearch, the researchers were able to upload the dataset and at the same time leverage the built-in analyzers and similarity scoring that it already had.

• **Who**: Explain whom your search engines may serve and their basic information needs.

Ideally You should be able to describe three **use cases** (information needs).

* The search engine is for people who want a quicker or straightforward way of getting the list of movies that is related to their query. It does not boast a very excellent UI that other sites have, but that is exactly why it is better for people who just want the titles and information about it much quicker. Moreover, unlike websites like Netflix, the titles in the search engine cater to a more broad audience who may want something that is from the early 1900s, but it is fully equipped with the latest titles from 2018 as well.
* The first use case is that the user will be capable of searching the title of the movie that they are interested in
* The second use case is the user will be capable of searching similarity based on other fields other than the title.
* The third use case is that the user will be capable of filtering their query

• **How**: Describe related decisions you have made and steps you have taken to build the

Engine

Steps:

1. Obtain dataset from Kaggle
2. Preprocess the data to retain attributes of interest and remove rows with missing details
3. Setup free trial account in Kibana
4. Upload dataset in Kibana and manage the settings and mapping
5. Test queries using the dev tools of Kibana
6. Create Github repository for source control
7. Create an endpoint to utilize Kibana using Python with Flask framework
8. Translate test queries from Kibana to queries in Python
9. Design and create User Interface
10. Deploy the source code as an Azure Web Application
11. URL HERE

1. Describe the data fields related to the project and how they should be analyzed

and indexed. Select proper data types, analyzers, filters, etc. for each field, and

provide your rationale.

* Below is the index mapping which was made for Elasticsearch. The field types were assigned such that the least number of bits would be used for storing the data. For example, short was used instead of long because short was sufficient to store the data we have for the whole numbers. This is done to decrease memory footprint and try to obtain maximum processing speed when querying. Next, the analyzer decides how the contents of the field will be tokenized. For the purposes of the project, autocomplete, standard, and English were used. Autocomplete would tokenize the contents based on multiple combinations of the query, not just on the delimiters. Meanwhile, standard and english would tokenize based on the delimiter and remove stopwords as well. Next, we have the similarity which is how the scoring is decided between the query and the field. BM25 is a similar way of scoring with TF-IDF.

|  |
| --- |
| PUT /\_mapping |
|  | { |
|  | "imdb\_title\_id": { |
|  | "type": "text" |
|  | }, |
|  | "original\_title": { |
|  | "type": "text", |
|  | "analyzer": "autocomplete", |
|  | "search\_analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "year": { |
|  | "type": "short" |
|  | }, |
|  | "genre": { |
|  | "type": "text", |
|  | "analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "duration": { |
|  | "type": "short" |
|  | }, |
|  | "country": { |
|  | "type": "text", |
|  | "analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "language": { |
|  | "type": "text", |
|  | "analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "director": { |
|  | "type": "text", |
|  | "analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "writer": { |
|  | "type": "text", |
|  | "analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "production\_company": { |
|  | "type": "text", |
|  | "analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "actors": { |
|  | "type": "text", |
|  | "analyzer": "standard", |
|  | "similarity": "BM25" |
|  | }, |
|  | "description": { |
|  | "type": "text", |
|  | "analyzer": "english", |
|  | "similarity": "BM25" |
|  | }, |
|  | "avg\_vote": { |
|  | "type": "half\_float" |
|  | } |
|  | } |

|  |
| --- |
| PUT /\_settings |
|  | { |
|  | "analysis": { |
|  | "filter": { |
|  | "autocomplete\_filter": { |
|  | "type": "edge\_ngram", |
|  | "min\_gram": 1, |
|  | "max\_gram": 20 |
|  | } |
|  | }, |
|  | "analyzer": { |
|  | "autocomplete": { |
|  | "type": "custom", |
|  | "tokenizer": "standard", |
|  | "filter": [ |
|  | "lowercase", |
|  | "autocomplete\_filter" |
|  | ] |
|  | } |
|  | } |
|  | } |
|  | } |

2. Describe your search queries in terms the use cases (needs):

**–** What **keywords** and **query structure** should be used?

**–** What **fields** should be **searched for** potential **matches**?

**–** What **fields** should be included in the **scoring** (ranking)? In what manner?

|  |
| --- |
| GET /\_search |
|  | { |
|  | "query": { |
|  | "bool": { |
|  | "should": [ |
|  | { "match": { "original\_title": {"query": "Avengers Chris Zak Penn", "boost": 5 }}}, |
|  | { "match": { "actors": {"query": "Avengers Chris Zak Penn", "boost": 1 }}}, |
|  | { "match": { "genre": {"query": "Avengers Chris Zak Penn", "boost": 1 }}}, |
|  | { "match": { "country": {"query": "Avengers Chris Zak Penn", "boost": 1 }}}, |
|  | { "match": { "language": {"query": "Avengers Chris Zak Penn", "boost": 1 }}}, |
|  | { "match": { "director": {"query": "Avengers Chris Zak Penn", "boost": 1 }}}, |
|  | { "match": { "writer": {"query": "Avengers Chris Zak Penn", "boost": 1 }}}, |
|  | { "match": { "production\_company": {"query": "Avengers Chris Zak Penn", "boost": 1 }}}, |
|  | { "match": { "description": {"query": "Avengers Chris Zak Penn", "boost": 1 }}} |
|  | ], |
|  | "filter": [ |
|  | {"bool": {"must": [ |
|  | {"match": {"genre": {"query": "Biography, History"}}}, |
|  | {"match": {"language": {"query": "German"}}}, |
|  | {"match": {"country": {"query": "UK"}}} |
|  | ] |
|  | }}, |
|  | {"range": { "year": { "gte": 1966, "lte": 2020}}}, |
|  | {"range": { "duration": { "gte": 0, "lte": 300}}}, |
|  | {"range": { "avg\_vote": { "gte": 0 }}} |
|  | ] |
|  | } |
|  | } |
|  | } |

3. Select related similarity, scoring, and boosting methods accordingly. Describe

them and provide your reasons.

* Specific fields have their scores boosted because the researchers decided that certain fields are more important than others. The main focus were the title, actors, director, and genre, which were boosted by 4, 3, 3, and 2, respectively. However, scores are also calculated based on the similarity to the country, language, writer, production\_company, and description fields.

4. Create your index, mappings, and load data.

• **How good**: Test and evaluate your search engine index

1. Test your query for each use case (information need); you may limit the number

of hits for your search.

2. Examine the results and judge each document’s relevance in terms of the information

need.

3. Provide a formal evaluation in terms of metrics such as precision, recall, F1,

and/or nDCG.

4. Discuss the results. You are encouraged to try different settings and compare

their results.

• **Where**: Where is your search engine (index)? Provide the names of your indices. • **Experiences**: Discuss your team experiences working on the search engine. What

works? What doesn’t? What could have been done better? What have you learned as

a team? And your thoughts on future works?

- The search engine index is located in a free trial account in Kibana Elastic Search. The indices are named faf42\_movies1, faf42\_movies2, faf42\_movies3, faf42\_movies4, faf42\_movies5, faf42\_movies6, and faf42\_movies7. Indices seemed to have a limit of 10,000 lines and the researchers had to create 7 indices to accommodate the entirety of the dataset.

**1.2 Code and Interface (if applicable)**

If you have built a front end (or anything related) to your search engine index, please

provide the **link** to your (web) interface and/or **source code** with **instructions**.