Search Engine Design

Introduction

The Search Engine is an application that develops the algorithm to search the best match between the user's full-text query and the document collection by similarity occurrence of terms and term weight. Search Engine will search, interpret, and organise the content to provide the best relevance to answer the user's query. Examples that use Search Engine are keywords search by a search website, Search Engine for Google, Long Tail Keywords Search etc.

Aim

To develop and implement a search engine with various Ensemble models helping in indexing of documents and ranking them based on the user Query. The documents are indexed by storing in a structured data for search optimisation by various models. Then based on the Query we rank the documents to topmost relevant documents.

Objectives

1. Data collection

To gather data from a source that is relevant to the term from the document collection of the unstructured data. Search engines allow users to filter and analyse its full-text query from the document collection.

2. Data Gathering and Analysis

Download the dataset example LoTTe from the open-source git. Read and analyse the data for each category, to understand how many Queries, and documents. Helps in manually validating the dataset, and understanding if we need to filter data points. A sanity check.

3. Data Engineering (pre-processing)

An example of three sentences are searched by pre-processing the collection and created into data table by its terms with its frequencies across the collection (Dictionary) and which terms are appeared in which documents (Postings)

Document 1: The Aces check their sleeves
Document 2: Trickery by crown of aces
Document 3: Aces have more convenient lives

Dictionary		Documents
Terms	**Frequency**	**Postings**
aces	3	1, 2, 3
by	1	2
check	1	1
convenient	1	3
crown	1	2
have	1	3
lives	1	3
more	1	3
of	1	2
sleeves	1	1
the	1	1
their	1	1
trickery	1	2

Table 1: Term Frequencies

4. Retrieval Model Design (Modelling)
Design a model to train a dataset for testing to evaluate the accuracy of the retrieval ranking of the documents for a given full-text query by relevance score. The models will be implemented using BM25, TF-IDF, BERT and if time is permitted ColBERT.

5. Search Engine Simple User Interface
The simple User Interface (UI) of the search engine will be connected to a dataset through document collection, where all the data will be searched, collected, filtered, and extracted from the dataset.

6. Evaluation Module
Evaluate the indexing of the document collection and optimise the search engine performance by implementing and experimenting the design of the search engine with a range of full-text queries and the Ground Truth correlation.

7. Hyperparameter Tuning
A planned three phase experiment would be done, to have results of each phase record and record the progress with the pipeline and newer models.

Dataset
The base dataset for our experimentation would be the LoTTE dataset, which has collection of questions and answers from GooAQ and stack Exchange:

LoTTE Dataset
- The Long-Tail Topic Stratified Evaluation (LoTTE) concentrates on long-tail user queries that might not be addressed by an entity-centric information store like Wikipedia.
- LoTTE includes 6 domain-specific datasets. The domains include writing, recreation, science, technology, pooled, and lifestyle, each with 500–2000 questions and 100k–2M passages. Since it consists of two types of queries such as search and forum queries, the whole dataset sums up to 12 datasets. These datasets are derived from StackExchange and GooAQ.
- It consists of two sets of queries: search-based queries and forum queries. The search queries are brief, knowledge-based questions which have a direct answer, and these queries were taken from GooAQ dataset. The forum queries are more open- ended questions with kind of a mixed and indirect answer, and these queries are taken from StackExchange.
- The test sets are clearly separated into topical groups, and each test set is followed by a validation set of connected but unrelated questions and passages.

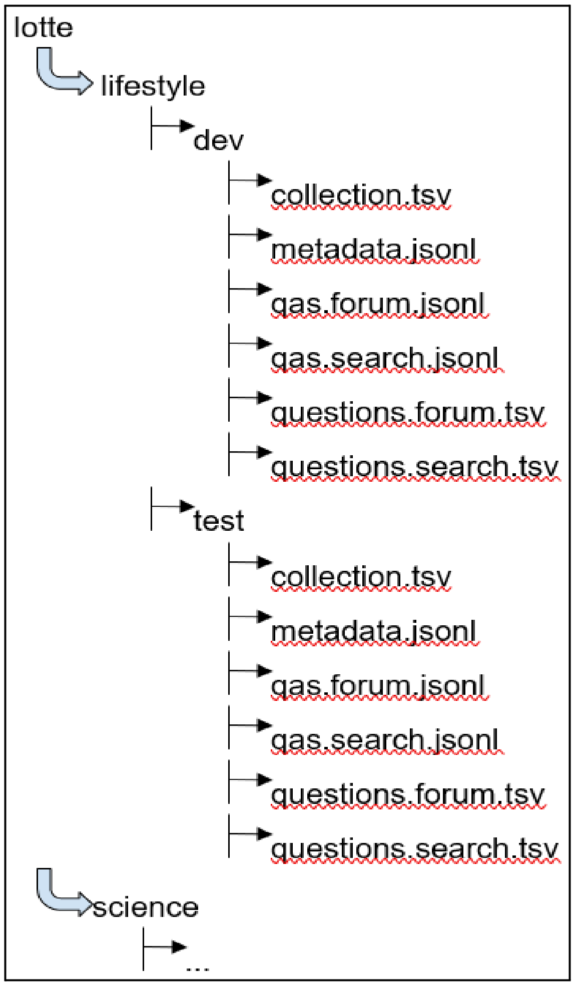
Count of the Documents and Queries

	Search		**Forum**	
Category	**Document**	**Query**	**Document**	**Query**
Lifestyle	268893 | 119461	417 | 661	268893 | 119461	2076 | 2002
Pooled	2428854 | 2819103	2931 | 3869	2428854 | 2819103	10097 | 10025
Recreation	263025 | 166975	563 | 924	263025 | 166975	2002 | 2002
Science	343642 | 1694164	538 | 617	343642 | 1694164	2013 | 2017
Technology	1276222 | 638509	916 | 596	1276222 | 638509	2003 | 2004
Writing	277072 | 199994	497 | 1071	277072 | 199994	2003 | 2000

Table 2: Dataset counts **Legend**: DevCount **|** TestCount

Layout Of The Data In The Dataset

The dataset is organised in the following manner[3]:



Queries and some samples

Search Queries:

- These queries are collected from GooAQ dataset, a dataset of Google search-autocomplete queries and their answers.
- By using a range of indications for relevance, such as expert annotations, user clicks, and hyperlinks as well as specialized QA components for different question kinds with access to the post title and question body, Google Search probably translates these natural questions to their responses.
- Search queries are concise, knowledge-based questions with direct responses.

Forum Queries:

- In order to gather the forum queries, the post titles from the StackExchange community are extracted and utilised as queries. Then, the posts that correlate to the answers are gathered as targets.
- The most popular questions are prioritised, and the remaining questions are sampled based on their relative contributions within each topic.
- Unlike search queries which exhibit more natural and consistent patterns, these queries tend to have more wider diversity.
- Forum queries have wider and open-ended questions with more implicit responses.

Architecture Diagram

Given below is an Architecture proposal for the TF4ces Search Engine.

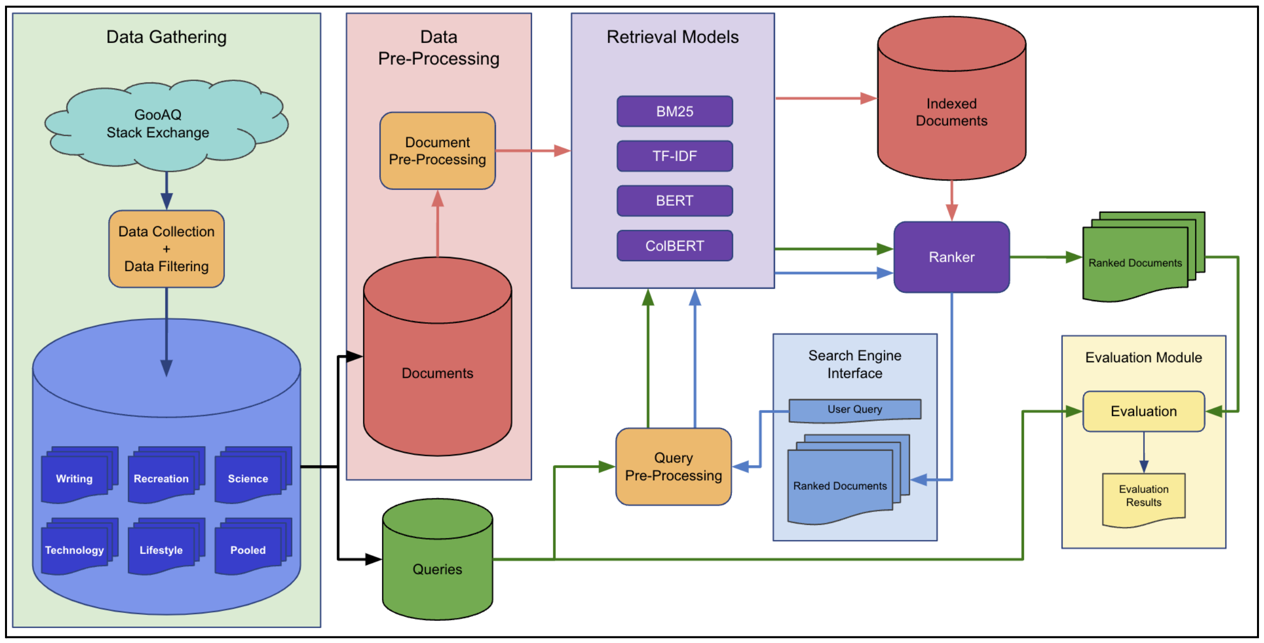


Fig 1: Architecture of TF4ces Search Engine

Retrieval Models

We aim to incorporate the following retrieval models in our search engine design:

BM25[4]:

- BM25 is a statistical algorithm that calculates the probability of relevance between the query and the document, based on the frequency of terms from the query appearing in the document.
- It has a term frequency normalisation factor to address the issue of term saturation.
- It has been proven as an effective ranking model in many IR applications.

TF-IDF[5]:

- TFIDF measures the importance of term in the collection of documents.
- It assigns weights to each term which is proportional to its frequency in the document and inversely proportional to the frequency in the entire collection of documents.
- It gives high weights to the terms that are important to a specific document but is rare in the collection of documents. It gives low weight to the terms that are common in the collection of documents but is of less importance to the specific document.

BERT[7]:

- BERT is a transformer based neural network that can learn contextual relations between the words and the sentences.
- BERT can be fine-tuned on wide range of NLP tasks like Information retrieval, text classification, named entity recognition, etc.
- There are two ways in which BERT can be used for IR task:
- Fine tuning the model on query-document pairs, so that it can rank the documents for a given query, based on the relevance score.
- BERT can be used to encode long or complex queries and documents, which can then be used in traditional IR pipeline such as BM25, to generate the score.

ColBERT[8][9]:

- ColBERT introduces a late interaction step which encodes the query and document independently using BERT, and then uses the interaction step to model their similarity. The late interaction mechanism introduced here tackles the problem of high computational cost.
- Late interactions, indexing document representations offline, its interaction via MaxSim (Maximum Cosine Similarity) operator and crucial design choice in the BERT based encoder are all essential to ColBERT's effectiveness.
- colBERT can re-rank the outputs of other retrieval models.

Apart from those models, ElasticSearch[6] was also looked into, but it won't be going to be used in our experiments for the limited low level debugging capabilities offered by the API.

Evaluation Metrics used by above Models:

- **Mean Average Precision (MAP)**
- **Normalised Discounted Cumulative Gain (NDCG)**
- **Mean Reciprocal Rank (MRR)**
- **Recall @ K**

Tools

Software/Tools	Description
PyCharm	IDE used for python programming.
GitHub	Software development and version control.
Trello	Project Management Tool

Table 3: Software tools.

Library	Description
Apache Lucene	It's an open-source search engine library that provides advanced search and indexing capabilities.
Scikit-Learn	Provides TF-IDF vectorizer.
Hugging Face Transformers	For pretrained transformer models like BERT and ColBERT
PyTorch and TensorFlow	Libraries used to implement ML models in the IR pipeline.

Table 4: Python libraries

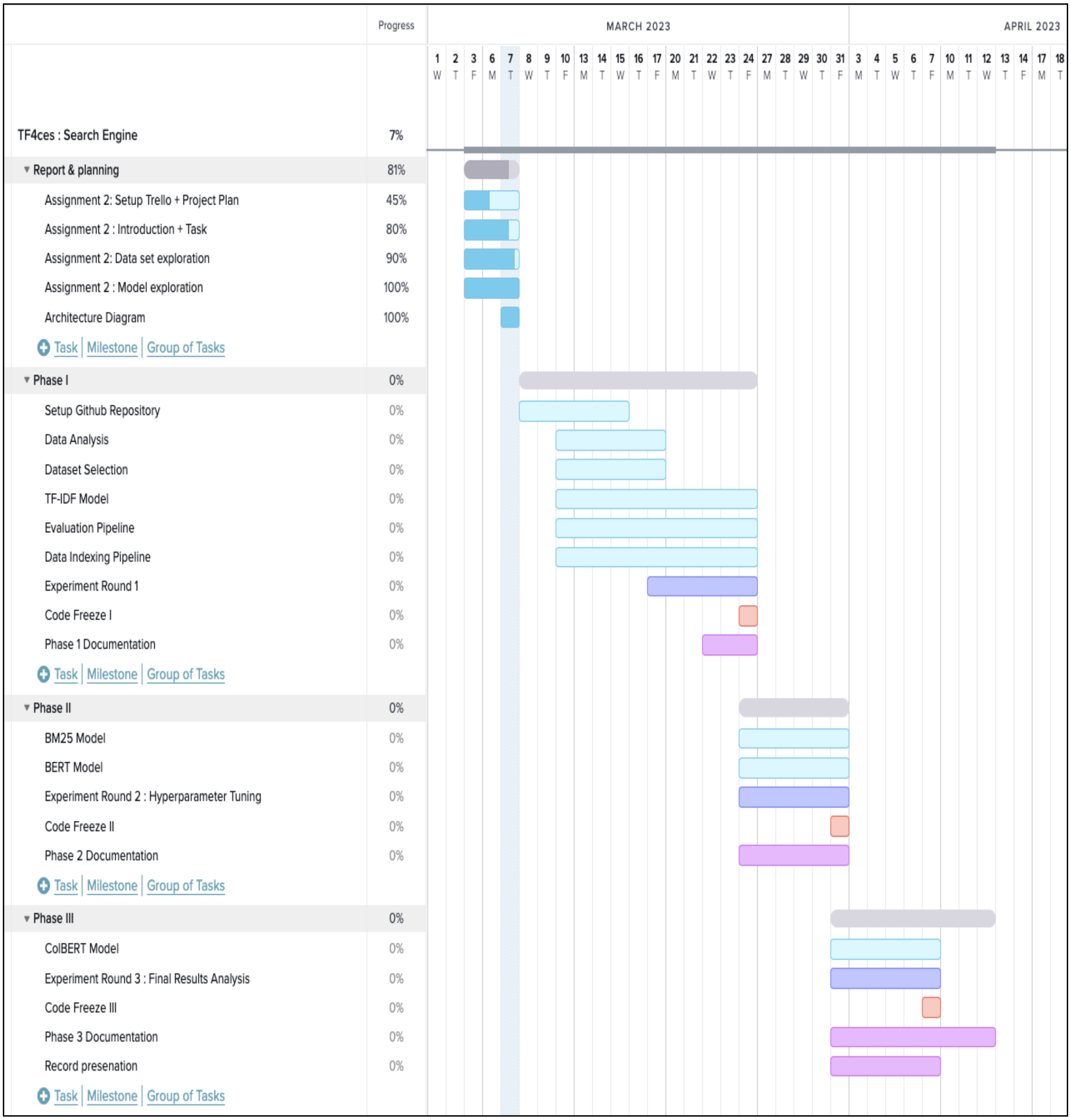
Task Delegation

Team	Roles
Mohammed Ataaur Rahaman	Team Lead, Code Reviewer 1, Modeling, Evaluation, Documenter
Edmund Lepre	ML Operations, Data Engineering, Modeling, Evaluation, Documenter
Arvind Jadhav	ML Engineering, Code Reviewer 2, Modeling, Experimentation, Documenter
Ayushi Choudhury	Data Analytics, Data Pipelining, Modeling, ML Engineering, Documenter

Table 5: The Four Aces Team, and primary responsibility

Project Plan

The project is planned to be completed in three phases as shown below from TeamGantt and Trello.



	Progress
TF4ces : Search Engine	7%
▾ Report & planning	81%
Assignment 2: Setup Trello + Project Plan	45%
Assignment 2 : Introduction + Task	80%
Assignment 2: Data set exploration	90%
Assignment 2 : Model exploration	100%
Architecture Diagram	100%
Task | Milestone | Group of Tasks	
▾ Phase I	0%
Setup Github Repository	0%
Data Analysis	0%
Dataset Selection	0%
TF-IDF Model	0%
Evaluation Pipeline	0%
Data Indexing Pipeline	0%
Experiment Round 1	0%
Code Freeze I	0%
Phase 1 Documentation	0%
Task | Milestone | Group of Tasks	
▾ Phase II	0%
BM25 Model	0%
BERT Model	0%
Experiment Round 2 : Hyperparameter Tuning	0%
Code Freeze II	0%
Phase 2 Documentation	0%
Task | Milestone | Group of Tasks	
▾ Phase III	0%
ColBERT Model	0%
Experiment Round 3 : Final Results Analysis	0%
Code Freeze III	0%
Phase 3 Documentation	0%
Record presenation	0%
Task | Milestone | Group of Tasks	

MARCH 2023
APRIL 2023
1 W 2 T 3 F 6 M 7 T 8 W 9 T 10 F 13 M 14 T 15 W 16 T 17 F 20 M 21 T 22 W 23 T 24 F 27 M 28 T 29 W 30 T 31 F
3 M 4 T 5 W 6 T 7 F 10 M 11 T 12 W 13 T 14 F 17 M 18 T

Figure 2: project plan in TeamGantt + Trello

References

[1] Stanford Future Data Lab. (n.d.). LoTTE.md. GitHub.
https://github.com/stanford-futuredata/ColBERT/blob/main/LoTTE.md

[2] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, Matei Zaharia. (2022). ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. ArXiv.
https://arxiv.org/pdf/2112.01488.pdf

[3] Hugging Face Datasets. (n.d.). "lotte". Hugging Face.
https://huggingface.co/datasets/colbertv2/lotte/tree/main

[4] Shane Connelly. (2018). Practical BM25 - Part 2: The BM25 Algorithm and its Variables. Elastic.
https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables

[5] Gerard Salton, Christopher Buckley. (1988). Term-weighting approaches in automatic text retrieval. Elsevier.
https://doi.org/10.1016/0306-4573(88)90021-0

[6] Pratik Jogdand. (2021). Information Retrieval in ElasticSearch. University of Maryland, Baltimore County.
https://redirect.cs.umbc.edu/courses/graduate/CMSC676/SP2021/termpapers/CMSC476676-TermPaperJogdandPratik.pdf

[7] Trabelsi, Mohamed & Chen, Zhiyu & Davison, Brian & Heflin, Jeff. (2021). Neural ranking models for document retrieval. Information Retrieval Journal. 24. 10.1007/s10791-021-09398-0.
https://www.researchgate.net/publication/355418275_Neural_ranking_models_for_document_retrieval

[8] Omar Khattab, Matei Zaharia. (2020). ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. ArXiv.
https://arxiv.org/abs/2004.12832

[9] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, Matei Zaharia. (2022). ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. ArXiv.
https://arxiv.org/abs/2112.01488