Introduction to Web Search & Mining

Group Project

Final Report, Code, Demo Due: June 19th

Introduction

This is a group-based project. Each group should contain maximum 3 students. In this project, there are four options. Each group must email TA (wsm2021spring@163.com) your choice. There should be **3 choices** in the email (First choice, Second choice and Third choice ranked by your preference). The topic of the email should be named as ***[Project Choice]+group leader's name+group leader's student id***. If a project idea is chosen by too many groups (each option can be selected about 1/4 groups), it will be allocated on a first-come, first-serve basis. The deadline of project selection is April 26th.

Option A

In this project, you are required to crawl Ubuntu IRC dialogues online and building a simple search engine. Detailed requirements are as follows:

1. **Data Crawling and Preprocessing (30%)**. You need to crawl the raw Ubuntu IRC data(2004~2021) from https://irclogs.ubuntu.com/. After that, you have to do some necessary data preprocessing steps to transform the raw logs into multi-party dialogues. It should be noted that time、utterance speaker、utterance receiver(if available)、utterance text and etc. are all key values for each dialogue session. Spell-correction is not required during preprocessing.
2. **Search and Ranking (50%).** Based on the dialogues, a simple search engine is required to be capable of boolean search and spell-correction.
   - **a)** ***Boolean search***: Users provide search keys and operations between keys. The system needs to return all the relevant dialogue sessions. Types of search keys include but are not limited to a user name, the role(speaker/receiver) of the user, a linux/ubuntu command, etc. The query language must include operations such as AND, OR, NOT, and proximity (e.g., /3 /p /s as in the Westlaw system), as well as association by parentheses (check Page 47 and L1).
   - ***b)*** ***Spell-correction with Tolerant (Fuzzy) Search:*** Sometimes user might make some spelling mistakes while querying (such as: rooot account), your system should also return suggested alternative queries with the correct spelling like : *Do you mean by "root account"?* It should be noted that there also exist spelling mistakes in dialogues. So, the search engine should return dialogues containing "root account", "root account", "roooot accounts", etc. These

fuzzy results should be ranked according to the relevance to users' search keys.

The system should provide options for users to sort the returned dialogues by length, time, etc.

3. **Simple Web Interface** for demo and **Project Report (20%)**.

Option B

In this project, you are building a simple book search system. You are first asked to crawl all the free books and metadata from the webpage(https://www.smashwords.com/). Then you are asked to build a search engine that supports searching for **author, title, publish year, description and content**. You are required to build index or other data structures to support the following operations, and create a nice **interface** for demo.

1. **Boolean search**(about 20% scores). Users provide search keys and operations between keys. The system needs to return all the relevant documents or data. Types of search keys include but are not limited to an author name, a title, publish year etc. Operations include but are not limited to AND, OR, NOT, proximity, etc.
2. **Wild-card queries**(about 20% scores). While searching data with attributes including author name, title, publish year etc., users may process wild-card queries. For example, Alex* will match Alexander.
3. **Query mis-spellings**(about 20% scores)**.** Sometimes user might make some spelling mistakes while querying, your system should return several suggested alternative queries with the correct spelling like : *Do you mean…?*

The final results will also depend on the following aspects:

1. **Speed**(about 20% scores): Given a query, you need to quickly return the ranked list. You should show the effectiveness and search time of each ranking algorithm you implement.
2. **Functionality**(about 20% scores)**:** You should build a full-featured search engine, including support for title, author, content search and an easy-to-use interface. The result provided by your search engine should be sorted by relevance.

Option C

In this project, you are asked to build a search engine for Wikipedia with existing dumps. Data source is provided here : https://dumps.wikimedia.org/enwiki/latest/

You are required to implement **searching and ranking algorithms** to support the following requirements.

1. (about 30% scores) **Support tolerant(fuzzy) search**. Users may wrongly type some information or process wild-card queries and the system is required to do fuzzy

search.

2. (about 30% scores) **Support several ranking methods**. You need to implement at least five kinds of ranking algorithms. Note that you are not allowed to use the existing frameworks such as Lucene.
3. (about 20% scores) **Speed**. Given a query, you need to quickly return the ranked list. You should show the effectiveness and search time of each ranking algorithm you implement.
4. (about 20% scores) **Provide friendly search system GUI.**

Option D

In this project, you are asked to build a search engine with existing dataset: https://zenodo.org/record/1043504/files/corpus-webis-tldr-17.zip?download=1
This dataset contains 3 Million pairs of content and self-written summaries mined from Reddit. The format is a json file where each line is a JSON object representing a post. The schema of each post is shown below:

- author: string (nullable = true)
- body: string (nullable = true)
- normalizedBody: string (nullable = true)
- content: string (nullable = true)
- content_len: long (nullable = true)
- summary: string (nullable = true)
- summary_len: long (nullable = true)
- id: string (nullable = true)
- subreddit: string (nullable = true)
- subreddit_id: string (nullable = true)
- title: string (nullable = true)

You are required to implement **searching and ranking algorithms** to support the following requirements.
1. **Boolean search(20%)**. Users provide search keys, operations between keys and domain restrictions for keys (if exists). The system needs to return all the relevant samples. Operations include but are not limited to AND, OR, NOT, proximity, etc. For example: “title: show AND content: dramatically AND brilliant”. The system should return data with word “show” in title and word “dramatically” in content and “brilliant” in anywhere.
2. **Wild-card Queries(20%)**. Users may process wild-card queries. For example, in the query “There's a diff* between”, “diff*” may match “difference”, “different”, etc.
3. **Soundex and Spell-correction(20%).** Users may not sure about spelling of some words, the search engine should support queries like: “SOUNDEX(conthideriball) SPELL(anti-intellectualizm)”, return the most possible suggested alternative query with the correct spelling like: *Do you mean by “considerable anti-intellectualism”?*, and also return the data related to it.

4. **Several Ranking Methods(20%)**. You need to implement at least five kinds of ranking algorithms. Note that you are not allowed to use the existing frameworks such as Lucene.
5. **Friendly GUI and Project Report (20%).** For example, the matched words should be highlighted in the returned samples.

Deliverables

The final deliverables are different in specific options, please pay attention to what you need to submit for your option.

Item	**Description**	**Option**
Report	A well-written report to describe your ideas, design, implementation, example queries, results, conclusion, etc.	A, B, C, D
Crawled data	Zipped archive of the entire crawled data.	A, B
Web demo	A web demo. (You need to display your demo to TAs before the due date.)	A, B, C, D
Source code	Source code of your whole implementation.	A, B, C ,D

In addition, each member of the group needs to submit a peer review form to the TA separately to review group members performance. This form is also downloadable from the course website.