Introduction to Web Search & Mining

# Group Project

Final Report, Code Due: June 5th

Demo Due: June 12th

## Introduction

This is a group-based project. Each group should contain maximum 3 students. In this project, there are four options. Each one of you should choose a preferred option and find partners to form a group, and then register your name and student ID to the corresponding slot on our shared Tencent Doc.
【腾讯文档】WSM Project Groups
https://docs.qq.com/sheet/DWEZqUHFuVWpHU0Fk
Each Option can only be chosen by 12 groups at most, so if a project idea is chosen by too many groups, it will be allocated on a first-come, first-serve basis.
The deadline of project selection is April 15th.

## Option A

In this project, you are required to crawl classical Chinese poems (诗、词、曲…) online and building a simple search engine. Detailed requirements are as follows:

1. **Data Crawling and Preprocessing (20%)**. You need to crawl classical Chinese poems from websites like:
   古诗文网-古诗文经典传承_(gushiwen.cn)
   古诗文网_(gswen.cn)
   文库 · 中华诗词库_(xcz.im)
   After that, you have to do some necessary data preprocessing steps to transform the raw data into structure records, with each having the following attributes: title, author, dynasty(朝代), content(正文), and optionally other information if available: translation, annotation(注释), appreciation(赏析), background(创作背景), etc.
   You should crawl at least 1,000 poems with different authors and dynasties.
2. **Search Engine(60%).** Based on the poems, you need to implement a simple search engine supporting the following functionalities:
   a) ***Boolean search (20%)***: Users provide search keys and operations between keys. The system needs to return all the relevant poems (Either title or content meets the requirement of the query). The query language must include operations such as AND, OR, NOT.
   b) ***Zone-specific search (20%)***: Allow users to specify the attribute to filter/search.

For enumerable attributes like author and dynasty, the results should only contain those exactly match the query.
For other attributes like title and content, a Boolean or ranked search can be performed.
You can use either designs of UI or query (e.g. author(李白), title(酒)) to support such specification.

c) ***Pinyin-based Tolerant (Fuzzy) Search (10%):*** Just like SOUNDEX introduced in class, we can use pinyin for Chinese phonetic tolerant search. This can be especially useful when the user enters a wrong word with the same sound, or the poem itself contains 通假字.
Therefore, your search engine should support a special query language SOUND(x) which can retrieve poems with any word having the same pinyin as x. For example, we should be able to get "最爱湖东行不足，绿杨阴里白沙堤。" with SOUND(荫).

d) ***Ranked Search (10%):*** Return the search results with certain order that may be favorable by the users. The factors to consider may include: semantic relevance, fame of the author and the poem itself, etc. You can use any ranking method and any available information only to achieve this. we will prepare several test cases and score the system according to our search experience.

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

## Option B

In this project, you are going to build a search system for COVID-19 open research papers based on the CORD-19 dataset, which contains the parsed JSON data for tens of thousands of COVID-19 related open paper.
You can download the data from: https://ai2-semanticscholar-cord-19.s3-us-west-2.amazonaws.com/historical_releases.html (To save space, we recommend you to download the 1.4GB cord-19_2020-04-03.tar.gz)
We provide sample code to read the data without decompressing all the large gz files in https://github.com/blmoistawinde/cord_reader_example
Your system should have the following functionalities:

1. **Boolean search**(20%). Users provide search keys and operations between keys. The system needs to return all the relevant papers (Either title or abstract meets the requirement of the query. Search within body_text is not mandatory.). The query language must include operations such as AND, OR, NOT.
2. **Tolerant (Fuzzy) Search** (20%). The search engine should implement search strategies that tolerates the changes of Pluralization/singularization (infection-infections), or the changes in the suffix of words between its different morphological variations (infect-infection-infective). You should use at least 2 methods to implement this, like lemmatization.
3. **Support for multiple Ranking methods** (40%)**.** Both textual content as well as the citations(links) between papers are present in the dataset. Therefore, you should implement several strategies to rank the search results, from these aspects:

*a)* **Semantic Relevance:** at least 2 methods, like TF.

*b)* **Link Property:** at least 2 methods, like number of citations.

*c)* **Combination of (a) and (b):** at least 1 method.

We will prepare several test cases to check the quality of your best ranking method, so you may use any information to improve your search results (e.g. using full body_text).

4. **Nice Web Interface** for demo and **Project Report** (20%).
   Basic information of each search result should be properly displayed, including: **cord_uid, title, author and abstract**(can be partially displayed for brevity)

## Option C

In this project, you are asked to build a simple news website. Your news should be crawled from multiple data source including but not limited to:

- Fox News: https://www.foxnews.com/
- AP News: https://apnews.com/
- China Daily: https://www.chinadaily.com.cn/
- USA today: https://www.usatoday.com/
- Global Times: https://www.globaltimes.cn/

The followings are the detailed requirements:

1. **Data Crawling and Preprocessing** (20%). You are supposed to crawl at least 10,000 news starting from year 2019 (including 2019) up to now. For storage, the news data should be processed into json format. For each news, the corresponding json format should be like: doc= {“title”: “xxx”, “time”: “xxx”, “content”: “xxx”, “source”: “xxx” , “url”: “xxx”}.
2. **Search System** (40%). Your search system should meet the following requirements:
   a) **Boolean Search** (20%): Users provide search keys and operations between keys. The system needs to return all the relevant news (Either title or content meets the requirement of the query). The query language must include operations such as AND, OR, NOT.
   b) **Specific search** (10%): Allow users to specify their search query by time or source. For example, users can limit their search only to news within 30 days from CNN.
   c) **Ranked Search** (10%): Given a search query, the search system is supposed to return a ranked list of search results. You need to consider factors like semantic relevance and freshness. To implement this, you may use any ranking method.
3. Category Classification (20%). Each news has a category. For example, Some News website has categories named "Health", "Business", "Sports" and so on. You need to classify the crawled news into different categories, which can be either manually designed or automatically generated (like clustering). You should

also allow users to search by categories. They can specify their search query within several (or one) categories.

4. Implement **GUI Interface** for demo and **Project Report** (20%)**.**

## Option D

The category structure in English Wikipedia resembles a tree. For example, both "Nature history" and "Nature sciences" belong to the same parent category named "Nature". In this project, you need to construct the category tree structure in English Wikipedia, and then implement a simple search system for users to search categories. You can directly download the data from the Wiki Dump dataset. If the dataset is too space-consuming to process, an alternative is to crawl it from the website. The followings are the detailed requirements for you project:

1. **Data preprocessing** (20%): You are required to construct tree structure from the category data in Wikipedia. Note that there exist cycles among Wikipedia categories, which need to be resolved. You need to classify the sub-categories into main categories like "Business", "Culture", "People" and so on. For detailed information, you can refer to Main Topic Classifications. Given the large quantity of Wikipedia categories, you may process only a small portion of the given data. The minimum requirement for quantity is at least 10 main categories and 1,000 sub-categories for each main category.
2. **Tolerant (Fuzzy) Search** (20%): Users may wrongly type some words or wish to do wild-card queries. For example, the query "peo*" and "pwople" might match "people". Your search system should thus support fuzzy search.
3. **Category-specific Search** (20%): Allow users to specify their search query to a certain query. The search results should only include sub-categories to the given category.
4. **Ranked Search** (20%): You need to implement at least five ranking methods. The factors to consider may include: semantic relevance and other category attributes (like generality).
5. Provide a simple **GUI Interface** for demo and well-written **Project Report** (20%)**.**

## Deliverables

The final deliverables to submit.

| **Item** | **Description** |
|---|---|
| Report | A well-written report to describe your ideas, design, implementation, example queries, results, conclusion, etc. |
| Web demo | A web demo. (You need to display your demo to TAs before the due date.) |
| Source code | Source code of your whole implementation. |