DD2477 Search Engines and Information Retrieval Systems

Project topics, ir23

Project 1: News recommendation

Recommendation services give suggestions to users what to read, watch or listen to next, based on feedback users have provided previously. In particular, a news recommendation system will suggest newly published articles that a user is likely to appreciate. In this project, the task is to write a program that

- crawls the web and finds news articles
- presents a selection of links of those news articles to the user, perhaps based on a query from the user
- gives the users the possibility of providing feedback on what articles they liked or didn't like
- gradually adjusts its selection of articles based on the feedback

This task is inspired by the NewsREEL CLEF challenge:: http://www.clef-newsreel.org/
The NewsREEL Replay data set may be useful as an experimental base. As a basis for the implementation, preferably use *Elasticsearch* (https://github.com/elastic/elasticsearch).

Project 2: Personalized search

Modern commercial search engines like Google compute search results to a given query not just based on measures like *tf-idf* and pagerank, but also on the user's past behavior. In particular, what links users have selected in preceding searches will heavily influence the ranking of the results of new searches.

The assignment: Write a search engine application that continuously builds and maintains a profile of a user, based on search logs (containing past searches, and the search results the user has clicked on following those searches), and possibly other material about the user, such as social media profiles or text content the user has chosen to expose to the system. The user profile should then influence the ranking of search results somehow. As a basis for the implementation, use *Elasticsearch* (https://github.com/elastic/elasticsearch). If you have time: Create a dashboard for visualizing the statistics you have gathered in a visualization tool, e.g Kibana (https://www.elastic.co/kibana).

This task is inspired by the PIR-CLEF challenge:
http://clef2017.clef-initiative.eu/index.php?page=Pages/labs_info.php

Project 3: GitHub search

Have you ever written methods/functions where you realize that it must have been written before? On GitHub, millions of people are publishing their code for anyone to use, but there is no obvious way of finding the code snippet that you are looking for. The goal of this project is to solve just that! Thus:

- Crawl (a part of) the publicly available GitHub code.
- Filter out one programming language that you feel comfortable with.
- Process the files and separate class names, method names, modifiers (for example public, private, static, final etc.), variable names – things that you may want to search and filter!
- Index it into *Elasticsearch* (https://github.com/elastic/elasticsearch), or another search engine of your choice.
- Create an interface where you can search and filter methods or classes based on the metadata you have created.
- A sample query could be `methodName:quicksort AND returnType:List<Number>` i.e. search for quicksort, and filter by methods with returnType List. What would you want to search for?

Project 4: Searching the scientific literature

Contact: *Markus Ekvall, Terran AI, marekv@kth.se*

When searching for scientific literature on a specific topic, it is possible to consult a database like Diva (https://kth.diva-portal.org/smash/builder.jsf?searchType=FEED&dswid=-8952) . However, to find the most relevant articles, it is helpful for the system to know something about the user. If the user has written some scientific articles herself, this information could be used to create a user profile, perhaps with additional information about previous searches the user has made. The result list for a specific query should then be influenced by both the query and the user profile. The task would be:

- Download a subset of articles and/or reports published by KTH researchers and/or students, using the Diva interface.
- Index the dataset into *Elasticsearch* (https://github.com/elastic/elasticsearch), or another search engine of your choice.
- Come up with a way to construct a user profile for particular authors in the database, and a way to rank search results that depend both on the query and the user profile.
- Create an interface where you can search articles and reports, and visualize the results.

Terran AI is a recently started company working on the problem of searching the scientific literature and visualizing the search results in interesting ways. Groups that select this topic are welcome (but not obliged) to contact Markus Ekvall at Terran AI to discuss ideas for a more alternative or more specific topic within the realm of scientific search.

Project 5: Podcast search

Wouldn't it be great to find podcasts that discuss exactly what you are interested in at the moment? Or even better, to find the exact part of the podcast that discusses your topic of interest? To this end, Spotify has provided us with a nice data set of podcasts, consisting of audio files + transcriptions, and time markers (https://podcastsdataset.byspotify.com). The task is thus to:

- Use *Elasticsearch* (https://github.com/elastic/elasticsearch) to index the transcriptions of the podcasts (You will obtain this data from Johan).
- Create an interface where a user can enter a search query, e.g. "*Higgs Boson*", "*terrorism*" or "*what Jesus means to me*", to find and rank *n*-minute clips from podcasts that treat this subject (e.g. 2-minute-clips, or *n* can be selectable).
- The relevant portions of the transcriptions of the clips can be shown as search results.
- (Optionally, if you have time, you might want to create an interface where the relevant audio clips are played. However, the data is very large (2TB), so that might be unfeasible.)

This task is inspired by the TREC 2020 podcasts track, task 1 (https://www.aclweb.org/portal/content/trec-2020-podcasts-track-guidelines)

Project 6: Book recommendation engine

Compiling a reading list can be quite challenging given the number of new books that are published every year. Recommending new books automatically is an interesting, but non-trivial task requiring matching the genre a person likes, comparing the abstract of the book to the books previously read, checking what other users with similar preferences have read, etc. In this project the task is to:

- Scrape the GoodReads website, and store information about some books and their reviews in Elasticsearch, https://github.com/elastic/elasticsearch

- Get information from the user about her previously read books.
- When given a query, e.g., “romance”, “adventure”, or “dragons and trolls”, generate recommendations for new books, using the query + the information about the user, the books, and their reviews.
- Provide recommendations with short descriptions of the books.