Deep Finders

CS 410 Course Project | Theme: Intelligent Browsing

Team Members

Diego Carreno (lead)	diegoac3	diegoac3@illinois.edu
Jason Ho	jasonho4	jasonho4@illinois.edu
Chris Kabat	cjkabat2	cjkabat2@illinois.edu
Robbie Li	robbiel2	robbiel2@illinois.edu

Problem Statement and Relevance

Search engines excel at surfacing results to users' queries. However, after the user has clicked into a returned result, they lose context on what parts of the document match their original query. Generally, users use the native browser find (i.e. CTRL + F or Cmd + F) and cycle through matched results to find relevant passages in the document. However, there are 2 downsides of this approach that can lead to user frustration. First, longer documents, such as technical documents, can provide too many matches than are useful to the user (ex. 100+), requiring the user to make multiple queries. Second, providing results based on exact matches could potentially miss on key pieces of information in the document that the user isn't aware of.

This topic relates to the intelligent browsing theme because it seeks to provide additional intelligence to the user in their search experience after they have selected a document from the initial search results. It relates to topics covered in class because it is an information retrieval problem and presents an opportunity to leverage some of the techniques we learned in class to provide a better search experience.

Project Goals

Provide a mechanism for users to search over a document using BM25 by providing a ranked list of passages in the document that are most relevant to the user's query. The user will be able to click on each search result and be taken directly to that passage in the document.

Technical Approach

We plan to develop a Google Chrome extension that allows a user to enter in a query. The extension will do some preliminary parsing of the page (i.e., removing noise and HTML tags) and send the data to a backend service that will index the page, separate the page into separate passages, and use the BM25 algorithm to provide a list of ranked results. The Chrome extension will provide a UI for the user to select and navigate to the passages on the original document. A stretch goal for this project is to set up a feedback mechanism to inform us of the efficacy of our model and help to tune our model's parameters (ex. k and b values).

We plan to use metapy or a similar library (ex. nltk, spaCy, etc.) to provide basic functionality such as stop tokenization, word removal, NGram analysis, etc. We will use Python for the backend and JavaScript (along with HTML and CSS) for the browser extension. We plan on hosting the backend on Microsoft Azure using Azure functions.

Testing and Validation

We will initially tune the model based on a selection of documents that we think are good use cases for this extension, and the 4 members will provide relevance feedback on these documents as training data. We will test and validate this extension by collecting feedback based on how a user interacts with the results, and asking fellow classmates to also use the extension to provide us with more data points.

Workload Estimation Breakdown

80 hours (4 members, 20 hours each) broken down as:

- Research - 10 hours
    - 10 hours to identify text retrieval approaches to use and for overall system design
- Backend - 30 hours
    - 8 hours to set up database in cloud platform
    - 8 hours to implement ranking algorithm API
    - 8 hours to implement web scraper
    - 6 hours to implement feedback collection system
- Frontend - 30 hours
    - 30 hours to implement chrome extension
- QA and testing - 10 hours
    - 8 hours for overall QA and system evaluation
    - 2 hours tuning model assumptions (i.e., *k* and *b* values for BM25)