Josh Reitz

Professor Carsten Eickhoff

BIOL 1595

4 May 2020

Assignment 2 Report

For Assignment 2 of BIOL 1595, my group and I created a search engine that responds to free-text queries with a ranked list of the 100 most relevant articles of the CORD-19 Collection. This program first downloads the CORD-19 dataset of over 30,000 articles relating to various coronaviruses, then extracts, parses, and organizes the text within and calculates term frequency-inverse document frequency (TF-IDF) scores to implement a search engine over the database. This report will formally detail our program and explain its successes and shortcomings.

This project had three main components: preprocessing, organization, and retrieval. The preprocessing of the text of the articles had several intended benefits. First, we tokenized each word in the text and removed any punctuation. We thought punctuation wouldn’t be relevant to our queries and tokenization allowed us to process the text on a word-by-word basis and obtain a corpus for the dataset. Next, the removal of stopwords such as “the” and “a” cut down on the amount of data we would have to store and process. This is also necessary for accurate search results since these extremely common words could clutter our TF-IDF rankings. The last step of our text processing is lemmatization. Lemmatization is essential to a search engine because it removes all inflectional endings of a word and returns the base form, or lemma, of each token of our articles. This allows for the various forms of a word to all be referenced as the same lemma for TF-IDF calculations and query matching.

The organization of our code varied as we progressed through the project. We began using an inverted index. This index mapped each word in our corpus to its TF-IDF score in each document. This could be used to assess the relevance of a query term to each document. We discarded this approach in favor of a vector space model. Our vector space model created a vector of each word in our corpus associated with its corresponding TF-IDF score for all documents in the corpus. For each query, we then created a vector of the same length that mapped the query terms’ TF-IDF scores to each document in the corpus. After this, the cosine\_similarity function from Sklearn would measure the similarity of the query to each document. However, this method only produced about 25% of the same search results as the CORD-19 Engine, which encouraged us to explore other methods.

The BM25 package ranks the relevance of a query to all documents in our dataset. It also uses elements of the TF-IDF formula when calculating query relevance, such as emphasizing rare and repeated words, but also incorporates elements such as document length normalization and the relative current document length. We found this method to produce about 40% of the CORD-19 Engine’s results. Thus, we used it as our final submission.

We faced several important challenges during our implementation. To begin, we had to filter out faulty articles from the dataset. Many of the articles were in another language or lacked a necessary field, such as Abstract, and had to be excluded from our engine as a result. Additionally, we faced memory issues when storing our TF-IDF vectors. To address this, we stored all of our data structures in Pickle files to minimize memory usage. We also extracted the JSON files to a .csv file. Using these additional files also shrank our preprocessing runtime to a reasonable 45-60 minutes. We also needed to weigh the relative contributions of the Title and Abstract fields. We tuned this parameter by meticulously checking the accuracy of our results in comparison to the CORD-19 Engine. We settled on a 3:1 ratio that emphasized the Title field.

In the future, we thought that accessing more features of the article could allow us to better refine our search by increasing the number of relevant terms associated with each article. Our current project only processed the Title and Abstract of each article, but adding fields such as Body or Authors would likely improve our engine’s usability but at a significant processing cost. Lastly, improving the interface of our engine would make it more user-friendly and practical. Possible additional features include returning hyperlinks instead of solely article ID’s to the user and implementing the program into a Read-Eval-Print Loop that could continually process user queries.

We submitted our project to Kaggle’s COVID-19 Open Research Dataset Challenge. Here is a link to our submission. https://www.kaggle.com/joshreitz/biol1595-assign2