Aniket Panda

CSC 583 Watson Report

How to Run

To run the project code, simply cd into top level of the repository and type

```
mvn test
mvn clean
```

into the terminal.

The project will first output the name of each of the eighty text files with prefix “enwiki-” along with a count of how many documents was created from each one. This is to track how far along the index construction, the most time-consuming task, is. So long as lemmatization is not being used, the whole process should only take a few minutes.

Next, the project will print out one hundred lines, one per question in the questions.txt file. Each line shows the correct answer to the question, Watson’s top answer, and the score, separated by commas for ease of entry into a csv file. An example is below.

```
The Washington Post,Media of the United States, 39.38503646850586
```

Code Description

The project is structured almost identically to the repo for homework 3. There are two code files that contain the bulk of the important code: QueryEngine.java and TestQ1_1.java.

QueryEngine.java, located in the src/main/java/edu/arizona/cs subdirectory, contains all code for building the index and executing queries. Its constructor takes a boolean argument which indicates whether to use cosine similarity or BM25 probabilistic scoring, as well as an integer argument which indicates whether to use stemming, lemmatization, biword search or none of the above. The method runQuery() takes a string argument representing an index query and returns a single ResultClass object representing the top scoring document.

TestQ1_1.java, located in the src/test/java/edu/arizona/cs subdirectory, acts as a sort of main() function. It instantiates QueryEngine, loops through questions.txt, and queries the index using each category-clue pair. It then prints out each question’s correct answer, Watson’s answer, and the score to the terminal. TestQ1_1.java is also where the scoring strategy and token normalization can be changed. The boolean instance variable scoringStrategy is set to true or false for cosine similarity or BM25 probabilistic scoring, respectively. The integer instance variable tokenNorm can take on values of 0 for none of the following, 1 for stemming, 2 for lemmatization, and 3 for biword search. It is not recommended to use lemmatization first as it can take several hours. The other options only take a few minutes.

The two scoringStrategy options and four tokenNorm options give a total of eight possible configurations. Outputs for each of these, with correctness manually added, are in the results directory in the form of PDFs with charts. I recommend these as they are much easier to read

than the terminal output. The src/main/resources subdirectory contains questions.txt and all eighty “enwiki-” text files.

Indexing and Retrieval Questions

I implemented four different options for indexing terms. The first, and default, option is to simply use a StandardAnalyzer, which filters stop words and lowercases all tokens. The second option improved upon the first option by adding stemming. It uses the EnglishAnalyzer which adds Porter stemming and the removal of possessives to StandardAnalyzer. The third option eschews stemming, but adds lemmatization on top of StandardAnalyzer. It uses a StanfordCoreNLP pipeline to do so.

The fourth option was part of the graduate student “improving retrieval” requirement. I wanted to add positional information to the search rather than just relying on a bag of words approach. Therefore, I implemented a biword search. This was done by taking each consecutive pair of terms in a query and converting it to a proximity search phrase with a range of one. This allows a pair of words to be transposed, but I found the tolerance beneficial.

All queries start out as concatenations of the category and the clue before being subjected to the normalization described in the previous paragraphs. That is, all words from the clue and category are used except for stop words.

The only issue I encountered specific to wikipedia’s articles is the fact that various non-alphanumeric symbols are often directly concatenated with normal words. This can render stopword filter less effective because the common words concatenated with strange symbols are not in the stop word list, so they are not removed, which adds noise to scoring process. It can also reduce the term frequencies of important terms by changing them to a form not found in the query. I tried to solve this by removing punctuation from the documents, but it did not help much.

Results

The results are summarized in the table below.

Precision at 1

Scoring	Standard	Stemming	Lemmatization	Biword
Probabilistic	0.20	0.21	0.22	0.11
Cosine Similarity	0.02	0.01	0.02	0.02

Performance was measured using precision at 1. Jeopardy, unlike web search, only accepts one answer. Therefore, only to top scoring result is truly relevant. This makes precision at 1 the more appropriate chose over mean reciprocal rank (MRR), which relies on there being multiple results. Additionally, the relevance, or correctness, of the answer is binary, which eliminates normalized discounted cumulative gain (NDCG) as an option.

Once again, the full output for each configuration, with correctness labels manually added, is in the results folder.

Analysis

The scoring function proved to be the most impactful parameter to tune by a wide margin. For all tokenization/normalization choices, using probabilistic scoring rather than tf-idf over cosine similarity yielded an order of magnitude improvement in the performance metric. By comparison, switching between stemming, lemmatization, and neither changed the performance very little.

When using BM25, the standard method (no stemming or lemmatization), stemming, and lemmatization had performance values of 0.2, 0.21, and 0.22 respectively. Technically, stemming is better than the standard method and lemmatization is better still. However, the differences are so small among the three that it is more appropriate to say that stemming and lemmatization are not significant factors in driving performance.

The use of biword search roughly halved the performance value when compared to the three token normalization methods. It failed its intended purpose of improving upon the bag of words approach. However, I do not think refutes the premise that adding positional could improve this version of Watson. It just suggests that a biword approach is too crude. A more nuanced approach might be to allow a pair of words be anywhere in the document, but have the contribution of that pairing to the scoring decrease as they get further apart.

The performance differences for biword search and the three token normalization techniques also had little effect when using tf-idf scoring. However, in that case the performance values are so small that there is little room to gauge meaningful differences anyhow.

BM25 scoring with lemmatization was ultimately the best configuration for the system. The precision at one was .22 with one hundred questions in the test set, so it answered 22 out of 100 questions correctly.

To understand why certain questions were answerable by such a simple system while others were error-prone, it is useful to analyze them in terms of the technique used to score them. That is, in terms of BM25. When evaluating the weight of each term on the score, BM25 penalizes document frequency. This means the rarer words in the question will be given the most weight. Questions are more likely to be answered correctly if their rarer words are specific

the answer's corresponding article and less likely if their rarer words are specific to another article.

For example, the following question was answered correctly.

> THE RESIDENTS
> Title residence of Otter, Flounder, Pinto & Bluto in a 1978 comedy
> Animal House

This question was not.

> OLD YEAR'S RESOLUTIONS
> The practice of pre-authorizing presidential use of force dates to a 1955 resolution re: this island near mainland China
> Taiwan

The first question is almost entirely composed of terms that are highly specific, such as names and dates. Watson answered the second question with "United Nations General Assembly Resolution 505" because the bulk of the question's rarer terms are specific to that resolution.

Another factor that impacts whether a question is answered correctly is whether natural subtleties of natural language factor into it. For example, the following question was incorrectly answered with "Dragon (Middle-earth).

> CONSERVATION
> Indonesia's largest lizard, it's protected from poachers, though we wish it could breathe fire to do the job itself
> Komodo dragon

This error is clearly because the bag of words approach cannot understand humor well enough to discount the joke about breathing fire as irrelevant.

A final source of error arises from the fact that although lemmatization can reduce words to their base form, it cannot account for semantic similarity by matching synonyms or near-synonyms.

These three categories or errors, lack of specific terms, lack of tonal comprehension, and lack of semantic comprehension, contribute to the low performance of Watson.