Old Dominion University
Department of Computer Science
CS834: Introduction to Information Retrieval
Fall 2017
Assignment 3
Professor: Dr. Michael Nelson

Hussam Hallak
CS Master's Student

November 8, 2017

Contents

List of Figures

List of Tables

Question 1:

Exercise 6.2:

Create a simple spelling corrector based on the noisy channel model. Use a single-word language model, and an error model where all errors with the same edit distance have the same probability. Only consider edit distances of 1 or 2. Implement your own edit distance calculator (example code can easily be found on the Web).

Answer:

I have used the WikiSmall collection provided on the book website as a sample:
http://www.search-engines-book.com/collections/

Google found about 377,000,000 results for the term “Memory”.

Figure 1: Query: Memory, Search Engine: Google

Question 2:

Exercise 6.4:

Assuming you had a gazetteer of place names available, sketch out an algorithm for detecting place names or locations in queries. Show examples of the types of queries where your algorithm would succeed and where it would fail.

Answer:

The easiest way to detect place names or locations in a query is to tokenize the query and run all tokens against the place names dictionary, gazetteer. This method would be computationally expensive if the gazetteer has a large number of place names. It will sometimes fail, generate some false positives. For example:

Query: Virginia Woolf

The term “Virginia” will be falsely identified as a place name, but the user meant Virginia Woolf, the writer.

A better, less expensive, algorithm can be implemented by defining patterns that will help detecting place names. These patterns can be defined using prepositions or other words that are used with places in English language. Examples of these prepositions and words include: From, in, near, to, of, ...etc.

Pattern: $* + ('in'||'from'||'near'||'to'||'of')+ < placename >$

Example queries:

Celebrities born in Virginia

When did MalcomX go to Egypt

Universities in Hampton

Homes for sale near Suffolk

Is Michael Nelson from Norfolk

Does Syria lie north of Jordan?

The approach would begin by scanning the query to find these words or prepositions from, in, to, ...etc. If one of the words is detected, only the following word is run through the gazetteer. This method will sometimes fail, generate some false positives as well as false negatives.

False positive example:

Query: A recipe for a pie made out of avocado

The term “avocado” will be falsely detected as a place name, Avocado, California. However, the user meant avocado, the fruit.

False negative example:

Query: Where is Avocado located in California?

Unless the defined patterns are advanced enough to detect place names in such queries, “Avocado”, the place in California will not be detected.

Assuming that the defined patterns are advanced enough to identify all possible place names, practically impossible, a large amount of False positives will be generated.

Pattern: $< placename > +location$

Query 1: Washington location on the map

Query 2: George Washington location of birth

In query 1, the term “Washington” will be detected as a place name, which is true. However, the same term “Washington” in query 2 will also be identified as a place name, which is false.

Also using patterns will have to be accompanied with a rule to handle single term queries. This will also generate a large number of false positives. For example:

Query: Lincoln

The term “Lincoln” will be identified as the place name Lincoln in Alabama, while the user’s intention was to look up Lincoln, the president or Lincoln, the car or Lincoln, the welding machine.

One more rule must be added to the patterns to handle place names that are not a single terms. Examples include place names like “Virginia Beach”, “Newport News”, “Washington DC”, ...etc.

Example:

Pattern: $* + ('in'||'from'||'near'||'to'||'of') + < placename >$

Query: Hospitals in San Francisco

A rule to handle queries similar to the query above must be added. The first word of each place name that is n-gram where $n > 1$ must be added to the gazetteer and then mapped to the next term in the place name.

Here is my simple algorithm/implementation to detect place names in python-like code:

Success example:

Query: Schools in Norfolk

Failure example:

Query: Casinos in Las Vegas

Question 3:

Exercise 6.9:
Give five examples of web page translation that you think is poor. Why do you think the translation failed?

Answer:

Question 4:

Exercise 7.2:

Can you think of another measure of similarity that could be used in the vector space model? Compare your measure with the cosine correlation using some example documents and queries with made-up weights. Browse the IR literature on the Web and see whether your measure has been studied (start with van RijsbergenâĂŹs book).

Answer:

Question 5:

Exercise 7.5:

Implement a BM25 module for Galago. Show that it works and document it.

Answer:

I wrote the code for BM25 in Java since that is the language in which Galago is written. The file is named “BM25.java”. Some variables should be passed as arguments, but for the purpose of demonstration, I manually set them within the class BM25. Here is the code I wrote:

I downloaded the source code for Galago. BM25 is already implemented in the file “BM25Scorer.java”. I noticed that IDF is calculated using a different formula. Here is the formula used in their implementation:

```
idf = Math.log(documentCount / (df + 0.5));
```

Question 6:

Exercise 7.7:

What is the “bucket” analogy for a bigram language model? Give examples.

Answer:

Question 7:

MLN1: Using the small wikipedia example, choose 10 words and create stem classes as per the algorithm on pp. 191-192.

Answer:

Question 8:

MLN2: Using the small wikipedia example, choose 10 words and compute MIM, EMIM, chi square, dice association measures for full document & 5 word windows (cf. pp. 203-205)

Answer:

References

[1] Stackoverflow. https://stackoverflow.com/questions/tagged/python.

[2] https://stackoverflow.com/questions/10369393/need-a-python-module-for-stemming-of-text-documents

[3] https://www.rosettacode.org/wiki/Inverted_index#Simple_inverted_index

[4] https://en.wikipedia.org/wiki/Pigeonhole_principle