Lab 2: Information Retrieval

Due date: Thursday, October 8.

Overview

In this assignment you will perform a number of Information Retrieval tasks over a small collection of documents. The collection is a set of utterances spoken by California State Legislators, and public speakers during the discussion of SB 277, the 2015 bill to mandate vaccinations of children attending schools.

Assignment Preparation

This is a pair programming assignment. Each student teams up with a partner. Each team submits only one copy of the assignment deliverables.

Data

You will be working with a dataset called VACCINATION-DISCUSSION, a collection of utterances spoken during the 2015 California State Legislative Session in legislatice Committee hearings while discussing SB 277, a bill designed to require mandatory vaccinations for all children attending public and private schools in the State of California[1] The dataset is courtesy of the Digital Democracy project (http://www.digitaldemocracy.org), which provides searchable transcripts of California Legislative Committee hearings.

Information about SB 277, as well is links to committee hearings about it can be found at the following URL on the Digital Democracy portal:

http://www.digitaldemocracy.org/bill/201520160SB277

[1] This specific bill was chosen because of all the pieces of legislation during the most recent session, this one attracted the most attention of the public, and lead to most heated discussions.

(A link to this page will be made available on the course web site).

The Dataset

The data is provided to you as a single file `SB277Utter.json`. The dataset is a collection of *comma-separated* JSON objects, with each object representing a single utterance.

The utterances, i.e., the individual portions of a hearing, spoken by a single person at a specific time, are provided in the `SB277Utter.json` file in no specific chronological, or other order. While for a more complicated task, preserving the order of utterances within a single hearing is a good idea, for the purposes of this lab, we treat each utterance as an independent document, not connected with others.

Below, we show a sample JSON object representing a single utterance (the JSON object is formatted for easier reading here, in the actual file, no formatting exists):

```
{"pid":43,
 "first":"Travis",
 "last":"Allen",
 "PersonType":"Legislator",
 "date":"28-04-15",
 "house":"Senate",
 "Committee":"Judiciary",
 "text":"We are committed to working to make sure that the
         medical exemption ensures, you know, the freedom of
         professional practice for doctors. We're very concerned
         about making sure that we are respecting people's rights
         but as we have discussed and as are..."
}
```

A JSON object, as seen from the example above is a nested collection of key-value pairs, where the key represents the name of a specific attribute of an object, and the value contains its value - simple or compound. In the example above, the JSON object consists of eight simple attributes. The first seven attributes, `pid, first, last, PersonType, date, house, Committee` represent some of the meta-data the Digital Democracy project collects about the utterance. The eighth attribute, `text` contains the text of utterance. A brief explanation of the metadata attributes is below:

`pid`	The unique Id of the person who spoke the utterance
`first`, `last`	First and last name of the person who spoke the utterance
`PersonType`	Category of person's involvement with the legislative process (see below)
`date`	Date the hearing took place
`house`	The legislative body whose hearing it was: either Senate or Assembly
`Committee`	The Committee where the hearing it taking place

Note: There are seven categories of speakers in the Legislative Committee hearings in California: lawmakers, lobbyists, general public representatives, legislative staff, legislative analyst office staff, constitutional office representative, and government office representative.

In your work with the dataset, you need to pay attention, first and foremost, to the text of each utterance – this is the information you will be modeling. However, some of the meta-data: the speaker, the type of the speaker, the name of the committee, may be useful in answering some of the questions for this lab (or may have to be included in your answers).

Lab Assignment

In this assignment, you have to complete the following tasks.

1. **Model the dataset.** You will parse all utterances from the dataset, and will create the vector space model representations of them.
2. **Query answering.** You will implement matching for vector space model representations of documents, and using your implementation, you will find the documents (utterances) relevant to a number of queries provided by the instructor. For each query, you will measure the precision of your answer and will report it.
3. **Information need modeling.** You are provided with a number of user information needs. Your goal is to find the utterances that best match these needs. You will develop appropriate queries, and will run your matching procedure to determine the documents relevant to your queries. You will report the documents (utterances) you recovered, and the accuracy (precision) of your retrieval effort.
4. **Report.** The main deliverable for this assignment is a report documenting both your IR system (what decisions you made, what you have experimented with, and so on), and the results you obtained for the query answering and information need modeling tasks.

The detailed description of each component of the assignment is provided below.

Modeling the Dataset

Your first task is to model the dataset provided to you. In doing so, you shall select a version of the Vector Space Model discussed in class, and implement parsing and processing the input data — the utterances from the JSON objects — and turning them into vectors of keyword weights. You will also build and maintain any other data structures necessary for modeling the data and useful for Information Retrieval and query answering.

Specifically, you need to implement the following.

Parsing and tokenization. You shall implement a document parser, that takes as input a single JSON object, extracts from it all metadata you want to store in the system, extracts the text of the utterance, and tokenizes the text.

Stopword Removal. You shall implement a simple stopword removal mechanism. The stopword removal procedure will accept as input a single token produced by your parser/tokenizer and will determine whether the token contains a stopword (in which case it shall be removed from the token stream) or a keyword (in which case it shall be forwarded onto the next processing step).

It is up to each team to come up with a good list of stopwords. Some suggestions will be made available to you (I will put up links to some stopword lists on the web page), but I urge each team to spend some time studying the utterances that you are processing (either by reading them, or by listening to the hearings) and determining the appropriate list of stopwords.

Stemming. You can use a ready-made implementation of the Porter's stemming algorithm - no need to reinvent the wheel here.

Building Vectors of Keyword Weights. You shall implement the procedure of constructing a vector of keyword weights from a given stream of tokens representing a single document (utterance) in the collection. You can choose any term weighting scheme discussed in class (or any other scheme you find in literature), as long as your implementation properly computes both *term frequency* and *document frequency/inverse document frequency* of the keywords. We will have discussed in class the means of computing both during the dataset processing stage.

Query processing. You shall implement a procedure for parsing a query to produce a vector of keyword weights (note: the procedure may differ somewhat from the procedure for modeling utterances, but is should be compatible with it), and a matching algorithm for finding the relevance of each document (utterance) in the dataset to a given query. The output of your matching procedure shall generate a list of utterances presumed relevant to the query, ordered by level of relevance in decreasing order. As an option, your matching algorithm can take a parameter specifying the desired number of matches and return no more than that number of utterances.

Note: You are allowed, and in fact, encouraged to experiment with different ways to process/model/match the data. As such, while implementing some components in your IR system (stopword removal, stemming, comuting *tf* and *idf* is required), your final IR process may choose, *after careful and documented experimentation*, to omit any of these items. For example, it is quite possible, that you get better accuracy when answering queries without stemming your keywords. If you discover that - you may omit stemming from your IR pipeline, but you must document your reasons for it in your report.

Query Answering

To test your IR system, we are providing a set of queries. The queries are available as plain text files named `queryXX.txt` from the course web page. Each file contains the verbatim query you are being asked to run. For this assignment, you cannot modify the text of the query.

You are asked to run the query through your IR system and retrieve the top 10 (or fewer, if the system cannot retrieve 10) utterances deemed relevant. The documents shall be presented in descending order of perceived relevance, and a relevance score needs to be associated and reported for each document.

For each query, you must examine the retreived documents (utterances) and, to the best of your ability, determine, whether the retrieved utterance is relevant to the query. Based on this determination, you shall compute the *precision* of each query. You can also compute other precision-related measures, for example, *average expected precision* for each query.

Because this dataset is "fresh", obtaining recall information from it may be difficult. If your team **wants to earn extra credit**, you may choose to determine *manually* the list of all relevant utterances for each query (or for some of the queries) and include this information in your report. However, this may be a time-consuming process, so this is not a requirement.

Finally, you shall compute the overall and/or average precision your IR system achieved on all queries and report it.

Note: You may choose to run a series of experiments comparing the accuracy of alternative implementations of your IR system (e.g., with or without stemming, or with different lists of stopwords). If you do, please make sure you document your entire set of experiements, and explicitly state what configuration of your IR system you consider to be the best. Particularly well-designed experiments comparing multiple IR system implementations in an apples-to-apples way will **earn extra credit**.

Infomration Need Matching

In addition to rigidly defined queries, you are also provided with a list of information needs. Each information need is provided to you in a file named `InfoNeedXX.txt`. Unlike the queries, information needs are informal descriptions of the information a specific user wants to find from the IR system.

Your task is as follows.

1. For each information need, figure out a good query to submit to your IR system. You can experiment with multiple queries, but at the end,

only the results of one query can be presented.

2. Report the query, and the results of running the query through the matching process of your IR system. Just as with the query answering, you shall return up to 10 responses in descending order of the perceived relevance.

3. Evaluate the relevance of each retreived utterance to the *original information need*.

Report

The main deliverable – the one that will actually be graded – for this lab is your lab report. The report shall be a professionally written and typeset document, submitted in **PDF format**. (You can use LaTeX or MS Word, or Googledocs, or any other word processing/typesetting system). Please use a *serif* font, like *Times New Roman* as the main font for your document[2]. Also, please justify your paragraphs both and left and right margins.

Your report needs not adopt a journal or a conference proceedings style, but it will be treated as a piece of academic writing. The report must have a title, a list of authors, a short abstract, and the following sections:

1. **Introduction.** A short outline of everything you've done for this lab.

2. **System Design and Implementation.** The description of your IR System, and its implementation. Describe the components of the system you implemented, the methods you used to obtain term weights for the keywords in the document and in the query, the similarity measure you use in the matching algorithm, and any additional data structures to implemented to support the querying process. If you have developed and tested alternative implementations, outline the alternatives you have experimented with.

3. **Query Answering.** Show the results of running your IR system on the provided queries, provide the precision and any other accuracy assessments.

4. **Information Need Matching.** Outline the queries you tried for the provided information needs, show the results of matching together with the accuracy assessments.

5. **Analysis and Conclusions.** Discuss what you have observed. If you experimented with multiple systems, describe which version behaved better and why. Assess the overall accuracy of your system. If your system is not too accurate, try to analyze why (based on the results observed) and suggest the further means of improvement.

[2]It improves readability and makes your report look more professional

Deliverables and submission instructions

This lab has only electronic deliverables. Submit the following:

- Source code for the IR system.
- A `README` file describing how to run the system.
- Your report.

Submit your electronic deliverables using handin:

```
$ handin dekhtyar lab02-466 <files>
```

You can zip all your source files into a single archive, but I ask you to submit your report as a single, unzipped file.