CS 5604 (Fall 2018) – Homework 3

Deadline: 11:59pm, Oct 4nd, 2018 (EST)

Access the following resources:

- Download and uncompress the index file and other data from Canvas.
  - index_lucene_robust04_krovetz.tar.gz
  - stopwords_inquery: a list of stopwords
  - queries_robust04: 249 search queries
  - qrels_robust04: relevance judgments for the 249 queries

The index is the same as the krovetz index used in HW2. The docno and content of documents are stored in the docno and content fields, respectively.

- Check out the Lucene starter code from GitHub:
  - https://github.com/jiepujiang/CS5604_HW3

1 Unigram Language Model (20 points)

1.1 Log Probability (10 points)

Implement MySearcher.logpdc(String field, String docno). It should return the log probability of a document with the specified docno (external ID) according to the corpus language model of an index—\( P(D|\text{Corpus}) \). We assume each words are generated independently from the corpus language model.

\[
\log P(D|\text{Corpus}) = \sum_{w \in D} c(w, D) \log P(w|\text{Corpus}) \quad (1)
\]

\[
P(w|\text{Corpus}) = \frac{c(w, \text{Corpus})}{|\text{Corpus}|} \quad (2)
\]

\( D \) is a document and \( t \) denotes each term in \( D \). \( c(t, D) \) and \( c(t, \text{Corpus}) \) are the frequency of the term \( t \) in \( D \) and in the corpus, respectively. \( |\text{Corpus}| \) is the length of the whole corpus (the total number of terms). You will need to
retrieve the stored document vectors in order to compute \( \log \text{pdc} \). Please check the Lucene tutorial on GitHub if you need help with document vectors.

Report the log probability of the following documents:

| docno      | \( \log P(D|\text{Corpus}) \) |
|------------|-------------------------------|
| FBIS4-41991| -8438.50779524626             |
| FBIS4-67701|                               |
| FT921-7107 |                               |
| FR940617-0-00103 |                       |
| FR941212-0-00060 |                           |
| FBIS3-25118 |                               |

### 1.2 Comparing Log Probability (10 points)

In the query likelihood model (QL), we rank results by \( P(q|D) \) (or equivalently \( \log P(q|D) \)). If \( \log P(q_1|D_1) > \log P(q_2|D_2) \), QL believes that \( D_1 \) is more relevant to the query \( q \) than \( D_2 \) and ranks \( D_1 \) at a higher position.

Does \( \log P(q_1|D) > \log P(q_2|D) \) also suggest that \( q_1 \) is more relevant to the document \( D \) compared with \( q_2 \)?

- If yes, discuss why;
- If no, discuss how to adjust \( \log P(q|D) \) to make it appropriate for comparing the relevance/similarity of a document with different queries.

### 2 Query Likelihood and Smoothing (40 points)

#### 2.1 Implementation (10 points)

Implement two scoring functions:

- **MySearcher.QLJMSmoothing**: QL with Jelinek-Mercer smoothing:
  \[
  P(t|D) = (1 - \lambda) \cdot \frac{c(t, D)}{|D|} + \lambda \cdot P(t|\text{Corpus}) \tag{3}
  \]

- **MySearcher.QLDirichletSmoothing**: QL with Dirichlet smoothing:
  \[
  P(t|D) = \frac{c(t, D) + \mu \cdot P(t|\text{Corpus})}{|D| + \mu} \tag{4}
  \]

**MySearcher** implemented the document-at-a-time search algorithm with a replaceable scoring function (as defined in the `ScoringFunction` interface). After implementing the two scoring functions, you should be able to run the main function of **MySearcher**. It retrieves the top 1,000 results for each of the 249 queries in the Robust04 dataset using a scoring function. It reports the P@10
and average precision (AP) of each query and the mean values of P@10 and AP over the 249 queries.

You can verify your Dirichlet smoothing by comparing your results with those by running LuceneQLSearcher’s main function. LuceneQLSearcher includes an implementation of QL with Dirichlet smoothing using a slightly faster approach optimized for term-at-a-time search.

We will use QL with Dirichlet smoothing in follow-up questions. You can use either your own implementation or LuceneQLSearcher in follow-up questions. They take similar input parameters, but the latter is probably faster.

2.2 Smoothing Parameters (20 points)

Divide the 249 queries into a training set (No. 301–450) and a testing set (No. 601–700). Separately plot the mean P@10 and AP of queries in the training and the testing set for:

- QL with Jelinek-Mercer smoothing where $\lambda$ ranges from 0.1 to 0.9 with step 0.1 (set x-axis to $\lambda$ and y-axis to P@10 or AP).
- QL with Dirichlet smoothing where $\mu$ ranges from 500 to 5000 with step 500 (set x-axis to $\mu$ and y-axis to P@10 or AP).

To examine the stability of the two smoothing approaches across different query sets, we apply the best $\lambda$ or $\mu$ in the training set (the value that gives the best mean P@10 or AP) to the testing set. Usually the results would be worse than those by using the best $\lambda$ or $\mu$ in the testing set. Report the differences in mean P@10 and AP comparing to those by using the best $\lambda$ or $\mu$ in the testing set. Discuss which smoothing approach seems more effective and stable on the Robust04 dataset based on your results.

2.3 Smoothing, TF, and Document Length (10 points)

We discussed in class that the log query likelihood score can be considered as a summation over each query term’s scores in the document as in Equation 6, where a term’s score in a document is: $\text{score}(t, D) = \log P(t|D)$.\n
\[
\text{score}(q, D) = \sum_{t \in q} \text{score}(t, D) \tag{5}
\]

\[
\text{score}(t, D) = \log P(t|D) \tag{6}
\]

For convenience, we can consider $\text{score}(t, D)$ as a continuous and derivable function of $c_{t,D}$ (the frequency of $t$ in the document $D$) and other variables. This makes it easier to examine the influence of $c_{t,D}$ on $\text{score}(t, D)$ and consequently on $\text{score}(q, D)$.

\[
\frac{\partial \text{score}(t, D)}{\partial c_{t,D}} = \frac{1}{P(t|D)} \frac{\partial P(t|D)}{\partial c_{t,D}} \tag{7}
\]
For Jelinek-Mercer smoothing:
\[
\frac{\partial \text{score}(t, D)}{\partial c_{t,D}} = \frac{1}{c_{t,D} + \lambda \cdot |D| \cdot P(t|\text{Corpus})} \tag{8}
\]

For Dirichlet smoothing:
\[
\frac{\partial \text{score}(t, D)}{\partial c_{t,D}} = \frac{1}{c_{t,D} + \mu \cdot P(t|\text{Corpus})} \tag{9}
\]

As we discussed in class, according to Equation 7, 8, and 9:

- When \(c_{t,D}\) increases, \(\frac{\partial \text{score}(t, D)}{\partial c_{t,D}}\) decreases: thus QL penalizes repeated occurrences of the same term (regardless of which smoothing approach is employed).

- \(\text{score}(t, D)\) increases by a slower rate as \(c_{t,D}\) increases if \(t\) is a common term in the corpus—\(t\) has a higher \(P(t|\text{Corpus})\)—compared with a less common term. Thus, QL with Jelinek-Mercer or Dirichlet smoothing gives less credit for matching a common term in the corpus (similar to IDF).

Now discuss the following questions:

- **2.3.1** – According to Equation 8 and 9, how do the two approaches differ in terms of discounting repeated term occurrences? How do the length of the document and the smoothing parameters influence the discount? (5 points)

- **2.3.2** – Analyze to which extent the two smoothing approaches penalize longer documents. Discuss while other conditions (\(c_{t,D}\) and \(P(t|\text{Corpus})\)) being equal, when will one smoothing approach penalize longer documents by a greater extent than another? (5 points)

Note that 2.3.1 asks the influence of \(|D|\) and smoothing parameters on \(\frac{\partial \text{score}(t, D)}{\partial c_{t,D}}\), while 2.3.2 asks the influence of \(|D|\) and smoothing parameters on \(\text{score}(t, D)\).

### 3 Naïve Bayes and Relevance Feedback (40 points)

We have discussed in class Naïve Bayes classification using multinomial language models. Let \(X\) and \(Y\) be two classes and \(P(w|X)\) and \(P(w|Y)\) be the language models of the two classes. Assuming that words in a document are independently generated from \(X\) and \(Y\), we can classify a document \(D\) by:

\[
\frac{P(X|D)}{P(Y|D)} = \frac{P(X)}{P(Y)} \cdot \frac{P(D|X)}{P(D|Y)} = \frac{P(X)}{P(Y)} \cdot \prod_{w \in D} \frac{P(w|X)^{c(w,D)}}{P(w|Y)^{c(w,D)}}
\]

Consider the following scenario:
A user submitted a query to an IR system;

The system returned the first page of 10 results to the user;

The user examined all the 10 results on the first page and judged which ones are relevant and non-relevant;

Now the user turns to the second page of search results ...

We hope to design a way to re-rank the rest of the search results according to the user’s relevance feedback on the first page. We can consider this problem as a binary classification problem, where our task is to classify documents into relevant ($R$) or non-relevant ($NR$). The user’s relevance judgments on the first page can be used as training data for estimating $R$ and $NR$.

Please implement and evaluate this re-ranking method using the Robust04 dataset:

- To start with, for each query in Robust04, you need to use QL to retrieve the top $k = 1000$ search results (please use QL with Dirichlet smoothing and set $\mu = 1000$ in your experiment).

- Although we do not have real users here, we can use relevance judgments in the qrels to simulate users—simply assume a user’s judgments for the top 10 results will be the same as those in the qrels.

- Estimate language models $P(w|R)$ and $P(w|NR)$ based on the relevance judgments in the top 10 results. Let $D_R$ and $D_{NR}$ be the set of relevant and non-relevant documents in the top 10 results of QL. We have:

$$P(w|R) = \frac{1}{|D_R|} \cdot \sum_{D \in D_R} P(w|D), P(w|NR) = \frac{1}{|D_{NR}|} \cdot \sum_{D \in D_{NR}} P(w|D)$$

Where: $|D_R|$ and $|D_{NR}|$ are the number of relevant and non-relevant documents in the top 10 results; $P(w|D)$ is the document language model for $D$, estimated using Dirichlet smoothing ($\mu = 1000$). (10 points)

- Re-rank the rest of the search results (starting from the second page) by:

$$\frac{P(R|D)}{P(NR|D)} \propto \prod_{w \in D} \frac{P(w|R)^{c(w,D)}}{P(w|NR)^{c(w,D)}}$$

$P(w|R)$ and $P(w|NR)$ are the probability of $w$ from $R$ and $NR$; $c(w,D)$ is the frequency of $w$ in $D$.

(10 points)
• To make sure we have some relevant and non-relevant documents to estimate $R$ and $NR$, our experiments only select the subset of queries in Robust04 where the top 10 QL results included both relevant and non-relevant ones.

• Remember we are considering the case that the user turns to the second page of search results. So we can evaluate by the precision of the second page—still, we show 10 results on the second page. Compare the effectiveness of this re-ranking method and QL by the precision of the second page (report the mean value on the selected subset of queries in Robust04). Discuss which one is better and why. (20 points)

A A Checklist for Your Submission

• A report including the experiment results and answers to all questions (in pdf).

• All your source code. Write a brief readme file if necessary.

Pack all the stuff as a .zip or .tar.gz file and upload to Canvas’s HW3 submission link. If you decide to use the 5-day extension (you can only use it once during the whole semester), send an email and let the instructor and the TA know.