COMP 5790/ 6790/ 6796

Special Topics: Information Retrieval

Instructor: Shubhra ("Santu") Karmaker

Assignment #4: Language Models [100 points]

⚠ **Notice:** This assignment is due **Friday, March 4, 2020 at 11:59pm**.

Please submit your solutions via Canvas (https://auburn.instructure.com/). You should submit your assignment as a **typeset PDF**. Please do not include scanned or photographed equations as they are difficult for us to grade.

1. Language Models with Smoothing [50 points]

Recall that, under the query-likelihood derivation, we model the probability of relevance with

$$P(R = 1 \mid D, Q) \approx P(Q \mid R = 1, D)$$

If we assume a multinomial distribution for the query generation, we arrive at the general formula

$$P(Q \mid D) = \prod_{i=1}^{|Q|} p(q_i \mid \theta_D)$$

The main difference between retrieval functions of this form comes from their different choice of smoothing method that is applied to the unigram language model θ_D. In general, when we smooth with a collection background language model, we can write this probability as

$$p(w \mid \theta_D) = \begin{cases} p_s(w \mid \theta_D) & \text{if } w \in D \\ \alpha_D p(w \mid C) & \text{otherwise} \end{cases}$$

where $p_s(w \mid \theta_D)$ is the discounted maximum likelihood estimate of observing word w in document D and α_D is a document- specific coefficient that controls the amount of probability mass assigned to unseen words to ensure that all of the probabilities sum to one. Noting that log is a monotonic transform (thus leading to equivalent results under ranking), and using the above smoothing formulation, we can show the following:

$$\begin{aligned}
\log P(Q \mid D) &= \sum_{i=1}^{|Q|} \log p(q_i \mid \theta_D) \\
&= \sum_{w \in V} c(w, Q) \log p(w \mid \theta_D) \\
&= \sum_{w \in D} c(w, Q) \log p_s(w \mid \theta_D) + \sum_{w \notin D} c(w, Q) \log \alpha_D p(w \mid C) \\
&= \sum_{w \in D} c(w, Q) \log p_s(w \mid \theta_D) + \sum_{w \in V} c(w, Q) \log \alpha_D p(w \mid C) - \sum_{w \in D} c(w, Q) \log \alpha_D p(w \mid C) \\
&= \sum_{w \in D} c(w, Q) \log \frac{p_s(w \mid \theta_D)}{\alpha_D p(w \mid C)} + |Q| \log \alpha_D + \sum_{w \in V} c(w, Q) \log p(w \mid C) \\
&\overset{\text{rank}}{=} \sum_{w \in D} c(w, Q) \log \frac{p_s(w \mid \theta_D)}{\alpha_D p(w \mid C)} + |Q| \log \alpha_D
\end{aligned}$$

a. **[8 pts]** Show that if we use the query-likelihood scoring method (i.e., $p(Q|D)$) and the Jelinek-Mercer smoothing method (i.e., fixed co-efficient interpolation with smoothing parameter λ) for retrieval, we can rank documents based on the following scoring function:

$$score(Q,D) = \sum_{w \in Q \cap D} c(w,Q) \log\left(1 + \frac{(1-\lambda) \times c(w,D)}{\lambda \times p(w \mid REF) \times |D|}\right)$$

where the sum is taken over all the matched query terms in D, $|D|$ is the document length, $c(w,D)$ is the count of word w in document D (i.e., how many times w occurs in D), $c(w,Q)$ is the count of word w in Q, λ is the smoothing parameter, and $p(w|REF)$ is the probability of word w given by the reference language model estimated using the whole collection.

b. **[7 pts]** This scoring function above can also be interpreted as a vector space model. If we make this interpretation, what would be the query vector? What would be the document vector? What would be the similarity function? Does the term weight in the document vector capture TF-IDF weighting and document length normalization heuristics? Why?

c. **[8 pts]** Show that if we use the query-likelihood scoring method (i.e., $p(Q|D)$) and the Dirichlet prior smoothing method for retrieval, we can rank documents based on the following scoring function:

$$score(Q,D) = \sum_{w \in Q \cap D} c(w,Q) \log\left\{1 + \frac{c(w,D)}{\mu P(w|C)}\right\} - |Q| \log(|D| + \mu)$$

d. **[7 pts]** This scoring function above can again be interpreted as a vector space model. If we make this interpretation, what would be the query vector? What would be the document vector? What would be the similarity function? Does the term weight in the document vector capture TF-IDF weighting and document length normalization heuristics? Why?

e. **[20 pts]** One way to check whether a retrieval function would over-penalize a long document is to do the following: Imagine that we duplicate a document D k times to generate a new document D' so that each term would have k times more occurrences (naturally, the new document D' is also k times longer than the original D). Intuitively, the score for D' shouldn't be less than the score of D for any query. Check if this is true for the query likelihood retrieval function with both Jelinek-Mercer smoothing and Dirichlet prior smoothing, respectively.

2. Language models Implementation [50 pts]

It is now time get your hands dirty with some real experiments. We will implement a simple retrieval interface and play with a toy dataset named "Cranfield Dataset". Cranfield is a small curated dataset that is very extensively used in the information retrieval experiments. In the dataset, there are 226 queries (search terms), 1400 documents, and 1837 (evaluations). The dataset is supposed to be complete in the sense that the documents that should be returned for each query are known. This makes the evaluation easier. For more details and history on Cranfield Experiments, refer to the following Wikipedia page: https://en.wikipedia.org/wiki/Cranfield_experiments.

Cranfield dataset consists of 3 different files all of which have been provided to you as part of this assignment:

i. cran.qry.json – Contains the queries with ID and query text
ii. cranfield_data.json – Contains documents with ID, title, author and body
iii. cranqrel.json – Contains the relevance judgements

The relevance judgements have been coded as follows:

1. References which are a complete answer to the question.
2. References of a high degree of relevance, the lack of which either would have made the research impracticable or would have resulted in a considerable amount of extra work.
3. References which were useful, either as general background to the work or as suggesting methods of tackling certain aspects of the work.
4. References of minimum interest, for example, those that have been included from an historical viewpoint.
5. References of no interest.

Obviously no 5's are included in the cranqrel.json. If you cannot find a query-document pair in the cranqrel.json, assume the relevance score to be 5.

a. [15 pts] Retrieval Functions: You will now implement two retrieval functions as follows:

i. Write a function using any programming language which takes as input a query-document pair and outputs the Jelinek-Mercer smoothed query-likelihood score corresponding to the equation you derived in question 1(a). Let us call this output as JM (Jelinek-Mercer) score.
ii. Repeat the exercise above for Dirichlet prior smoothed query-likelihood score corresponding to the equation you derived in question 1(c). Let us call this output as DP (Dirichlet prior) score.

b. [35 pts] Evaluation:

i. **[10 pts]** Set $\lambda = 0$. For each query-document pair (Q,D), compute JM(Q,D) using the function from 2(a)-i. Then, for each query, rank the top 5 documents according to their JM(Q,D) scores and compute the average of their relevance positions provided in the cranqrel.json file. This way you will be able to generate a performance score for each query, i.e., 226 performance scores in total. Now, plot a histogram of this performance score across all queries (consider each query-level-score as a data point and use 0 - 5 scale for x axis with bins [0-1-2-3-4-5]). Further, compute the average over all individual query scores to derive a single performance score for the whole dataset. Congratulations!!! you have finally generated a performance score and a histogram plot corresponding to $\lambda = 0$ for the whole dataset.
ii. **[5 pts]** Now, repeat 2(b)-i for $\lambda \in \{0.05,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}$ and compute a performance number as well as a histogram plot for each λ. In total, you will generate 10 numbers and 10 plots in this step.
iii. **[10 pts]** Now, generate a line-plot for λ vs dataset-level-performance. What pattern do you see? Explain your observations.
iv. **[10 pts]** Now, repeat 2(b) i-iii for Dirichlet prior smoothed query-likelihood and generate plots and performance metrics with:

$$\mu \in \{0,100,500,1000,2000,4000,8000,10000\}$$

In total, you will generate 8 numbers and 8 histogram plots in this step.
Also, generate a line-plot for μ vs dataset-level-performance. What pattern do you see? Explain your observations.