CS 598cxz (/course/598f16) Fall 2016

Assignment #3: Probabilistic Retrieval Models

> ⚠ **Notice:** This assignment is due **Saturday, September 24th at 11:59pm**.
>
> Please submit your solutions via Compass (https://compass2g.illinois.edu). You should submit your assignment as a **PDF**.

This assignment tests your understanding and analysis of the mathematical background underlying probabilistic retrieval models. In particular, you will be walking through a series of hands-on problems designed to examine and reinforce your ability to recognize and appreciate how the retrieval formulas are constructed, what each component of the formulas is used for, and why the retrieval models could achieve satisfactory retrieval results. Remember that you are expected to complete the assignment on your own.

1. Classic Probabilistic Retrieval Model [50 pts]

a. **[20 pts]** In the derivation of the Robertson-Sparck-Jones (RSJ) model (notes/rsj-derivation.pdf), a multi-variate Bernoulli model was used to model term presence/absence in a relevant document and a non-relevant document. Suppose, we change the model to a multinomial model (see the slide that covers both models for computing query likelihood). Using a similar independence assumption as we used in deriving RSJ, show that ranking based on probability that a document is relevant to a query Q, i.e., $p(R = 1 \mid D, Q)$, is equivalent to ranking based on the following formula:

$$score(Q, D) = \sum_{w \in V} c(w, D) \log \frac{p(w \mid Q, R = 1)}{p(w \mid Q, R = 0)}$$

where the sum is taken over all the words in our vocabulary (denoted by V), and $c(w, D)$ is the count of word w in document D (i.e., how many times w occurs in D). How many parameters are there in such a retrieval model that we have to estimate?

(Hint: ranking based on probability of relevance is equivalent to ranking based on the odds ratio of relevance, just like we did in RSJ.)

b. **[5 pts]** The retrieval function above won't work unless we can estimate all the parameters. Suppose we use the entire collection $C = D_1, \ldots, D_n$ as an approximation of the examples of non-relevant documents. Give the formula for the Maximum Likelihood estimate of $p(w \mid Q, R = 0)$.

c. **[5 pts]** Suppose we use the query as the only example of a relevant document. Give the formula for the Maximum Likelihood estimate of $p(w \mid Q, R = 1)$ based on this single example.

d. **[5 pts]** One problem with the maximum likelihood estimate of $p(w \mid Q, R = 1)$ is that many words would have zero probability, which limits its accuracy of modeling words in relevant documents. Give the formula for smoothing this maximum likelihood estimate using fixed coefficient linear interpolation (i.e., Jelinek-Mercer) with a collection language model.

e. **[15 pts]** With the two estimates you proposed, i.e., the estimate of $p(w \mid Q, R = 0)$ based on the collection and the estimate of $p(w \mid Q, R = 1)$ based on the query with smoothing, your should now have a retrieval function that can be used to compute a score for any document D and any query Q. Write down your retrieval function by plugging in the two estimates. Can your retrieval function capture the three major retrieval heuristics (i.e., TF, IDF, and document length normalization)? How?

2. Language Models [50 pts]

a. **[20 pts]** Show that if we use the query-likelihood scoring method (i.e., $p(Q \mid D)$) 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 $p(w \mid REF)$ is the probability of word w given by the reference language model estimated using the whole collection.

b. **[10 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. **[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.