CS 6550: Introduction to Information Retrieval

Assignment 2

UID: u1255635 *Student Name: Hasan Pourmahmoodaghababa*

Question 1.

Language modeling approaches are text dependent but classical probabilistic models are not. Language modeling approaches give a probability distribution over word sequences and so a different approach for scoring documents according to a query. It helps us to measure the uncertainties associated with the use of natural language. In fact, using a language model one can sample word sequences according to that model to get a piece of text. Therefore, language models are appropriate for query generation. However, classical probabilistic models are utilized for document generation. Both models are useful for quantifying how likely a document is similar/relevant to a query but from different perspectives. All in all, they both are trying to score documents according to a query but from different approaches (language modeling are data-dependent but classical probabilistic models are not).

Question 2.

(a) The unigram language model probability of a term in a document is $L(d) = \Pi_{t \in d}\, p(t|\theta_d)$, where d shows the document, t shows a term, θ_d represents the distribution of d and $L(d)$ is the likelihood of the document d. Notice terms are assumed to be independent of each other given a document.

Now we are going to maximize the likelihood of the document using maximum likelihood estimation in order to obtain $p(t|\theta_d)$. To accomplish this we first take log from $L(d)$, which we denote by $LL(d)$, and try to maximize it (remember that log is an increasing function and so maximizing $LL(d)$ will result in maximizing $L(d)$). So we have $LL(d) = \sum_{t \in d} \log(p(t|\theta_d))$ as the objective function and $\sum_{t \in V} p(t|\theta_d) = 1$ as the only constraint, where V denotes the unique terms in d. Applying Lagrangian multiplier method we can consider the function

$$f(d, \lambda) = \sum_{t \in d} \log(p(t|\theta_d)) + \lambda(1 - \sum_{t \in V} p(t|\theta_d)),$$

as the objective function. Taking the derivative of $f(d, \lambda)$ with respect to a term $p(t_1|\theta_d)$ for some term t_1 we get

$$\frac{\partial f(d, \lambda)}{p(t_1|\theta_d)} = \sum_{t \in d,\, t = t_1} \frac{1}{p(t|\theta_d)} - \lambda = \frac{count(t_1, d)}{p(t_1|\theta_d)} - \lambda.$$

Setting $\dfrac{\partial f(d, \lambda)}{\partial p(t_1|\theta_d)} = 0$ implies that $p(t_1|\theta_d) = \frac{count(t_1,d)}{\lambda}$.

Since this is true for any $t_1 \in d$, we use t instead of t_1 thereafter. Now employing the condition $\sum_{t \in V} p(t|\theta_d) = 1$ we get $\sum_{t \in V} \frac{count(t,d)}{\lambda} = 1$, which implies $\lambda = |d|$. Therefore, $p(t|\theta_d) = \frac{count(t,d)}{|d|}$ and the model is $MLE(d) = \Pi_{t \in d} \frac{count(t,d)}{|d|}$.

(b) Probabilities are as follows:

- $p(Information) = \dfrac{count(Information, d)}{|d|} = \dfrac{4}{23} = 0.1739,$
- $p(resources) = \dfrac{count(resources, d)}{|d|} = \dfrac{2}{23} = 0.0870,$
- $p(system) = \dfrac{count(system, d)}{|d|} = \dfrac{1}{23} = 0.0435.$

Question 3.

Solving the zero probability problem is one of the main reasons for smoothing. In fact, if a term in a query does not occur in a document, that term would receive 0 probability and so the whole product in the model would be 0. This happens because using MLE we allow the model to overfit the data. To solve this problem we need to have $p(t|\theta_d) > 0$ for all terms t in the vocabulary/corpus (not document). This will make sure that all documents have a chance, although a tiny chance, to be considered for a query and participate in scoring.

The next reason I would say is that without smoothing, general terms in the query, like stop words, would get quite a high probability and so the score for a document will be affected by them. Smoothing overcomes this issue and helps to reduce the impact of general terms in scoring documents for a query.

Lastly, smoothing methods are simple enough to use.

Question 4.

We know that log-likelihood without JM smoothing uses $p_{MLE}(t|\theta_d) = \frac{count(t,d)}{|d|}$ and so it is

$$\log p_{MLE}(q|\theta_d) = \sum_{t \in q} \log p_{MLE}(t|\theta_d) = \sum_{t \in q} \log \Big(\frac{count(t,d)}{|d|} \Big) = \log(\frac{4}{23}) + \log(\frac{1}{23}) + \log(\frac{0}{23}) = -\infty,$$

which, I guess, it means that this document is not considered relevant for that query as score is $-\infty$. Also, the likelihood is $e^{-\infty} = 0$.

However, the log-likelihood with JM smoothing utilizes

$$p_{JM}(t|\theta_d) = \lambda\, p_{MLE}(t|d) + (1-\lambda)\, p_{MLE}(t|C) = \lambda\, \frac{count(t,d)}{|d|} + (1-\lambda)\, \frac{count(t,C)}{|C|},$$

(here $\lambda = 0.5$) and thus log-likelihood with JM smoothing is

$$\begin{aligned} \log p_{JM}(q|\theta_d) &= \textstyle\sum_{t \in q} \log p_{JM}(t|\theta_d) = \sum_{t \in q} \log(\frac{1}{2}) + \log\big(\frac{count(t,d)}{|d|} + \frac{count(t,C)}{|C|}\big) \\ &= 3\log(\tfrac{1}{2}) + \log(\tfrac{4}{23} + \tfrac{100}{200*200}) + \log(\tfrac{1}{23} + \tfrac{40}{200*200}) + \log(\tfrac{0}{23} + \tfrac{60}{200*200}) \\ &= -13.4294. \end{aligned}$$

So the likelihood is $e^{-13.4294} = 0.00000147$.

From this example we can learn that without smoothing if a query has at least one term that is not in a document, the log-likelihood will assign $-\infty$ score to that document and so it will not be retrieved as a relevant document to that query. Nevertheless, with smoothing, the model partially considers that term (assuming it is in the corpus) and so assigns a positive probability for that term. So it never will pass 0 to the log function and as a result no document will get a $-\infty$ as a score. Therefore, all the documents will get a chance to be retrieved even though they do not consist some terms from a query.

Question 5.

Dirichlet prior smoothing is expected to perform best and additive smoothing is expected to perform the worst. Of course, JM smoothing also will perform well assuming the hyperparameter λ is chosen appropriately. The reason that we expect additive smoothing would not perform well is that it is not incorporating the corpus information (like term count in the corpus) and the impact of document length is not appropriate. However, Dirichlet prior and JM smoothing consider the corpus effect but Dirichlet prior smoothing also take the length of the document appropriately into account (the λ value is fixed for JM smoothing but it differs document by document for Dirichlet smoothing). So, we expect Dirichlet smoothing performs better than JM smoothing.

Question 6.

First part: The order for α-iDCG that I have chosen is $d_8, d_3, d_5, d_1, d_6, d_2, d_4, d_7$.

$\alpha\text{-iDCG} = 4 + \frac{2.5}{\log_2 3} + \frac{1}{\log_2 4} + \frac{0.5}{\log_2 5} + \frac{0.25}{\log_2 6} = 6.3894$

- List 1: $\alpha\text{-DCG} = 1 + \frac{2}{\log_2 3} + \frac{3}{\log_2 4} + \frac{0.5}{\log_2 7} + \frac{1.75}{\log_2 9} = 4.492 \Longrightarrow$
  $\alpha\text{-nDCG} = \frac{\alpha\text{-DCG}}{\alpha\text{-iDCG}} = \frac{4.492}{6.3894} = \mathbf{0.703}$
- List 2: $\alpha\text{-DCG} = 1 + \frac{2}{\log_2 3} + \frac{2.5}{\log_2 4} + \frac{0.25}{\log_2 7} + \frac{2.5}{\log_2 9} = 4.3896 \Longrightarrow$
  $\alpha\text{-nDCG} = \frac{\alpha\text{-DCG}}{\alpha\text{-iDCG}} = \frac{4.3896}{6.3894} = \mathbf{0.687}$

Second part:

$\text{iDCG} = 4 + \frac{3}{\log_2 3} + \frac{2}{\log_2 4} + \frac{1}{\log_2 5} + \frac{1}{\log_2 6} = 7.7103$

- List 1: $\text{DCG} = 1 + \frac{2}{\log_2 3} + \frac{3}{\log_2 4} + \frac{1}{\log_2 7} + \frac{4}{\log_2 9} = 5.3799 \Longrightarrow$
  $\text{nDCG} = \frac{\text{DCG}}{\text{iDCG}} = \frac{5.3799}{7.7103} = \mathbf{0.6978}$
- List 2: $\text{DCG} = 1 + \frac{2}{\log_2 3} + \frac{4}{\log_2 4} + \frac{1}{\log_2 7} + \frac{3}{\log_2 9} = 5.5645 \Longrightarrow$
  $\text{nDCG} = \frac{\text{DCG}}{\text{iDCG}} = \frac{5.5645}{7.7103} = \mathbf{0.7217}$

Third part:

We can see that α-nDCG from List 1 to List 2 is decreased but nDCG is increased. The main reason is that when we use α-nDCG, we penalize similar items but in nDCG we don't. In

fact, the relevance value for each document (i.e. r_i) does not depend on its position in the list in nDCG-calculation but it (i.e. $rel(i)$) differs for α-nDCG-calculation according to the novelty in covering subtopics.

We know that the place of documents d_3 and d_8 is changed in Lists 1 and 2 and that d_8 is related to more subtopics than d_3 (i.e. d_8 has a higher relevance value than d_3 in nDCG-calculation). Therefore, nDCG will be higher if d_8 appears before d_3. However, for α-nDCG, document d_8 that contains more similar items than d_3 with respect to d_1 and d_5 (that have already been retrieved), will gain lower weight in α-nDCG-calculation if d_8 appears before d_3.

Question 7.

One reason is that when we force diversity, we are biasing the retrieval results towards covering more diverse topics related to the query. It means that the model tries to prevent retrieving similar documents. So, without considering diversification, if a query is ambiguous, then it is more likely that the model retrieves completely non-relevant documents. On the other hand, by diversification, it is more probable that some of the retrieved results be related to the intent of the user. Thus, the relevance will be improved for such queries. For non-ambiguous queries, of course, the search result will focus on the right topic/keywords and retrieve the relevant documents, as the query is not ambiguous. Therefore, the over all relevance will be improved.