COL764
Assignment 2

Document Reranking Task

Debraj Dey
2021ME10973

Prof. Srikanta Bedathur

Indian Institute of Technology Delhi

1 Language Modeling based Retrieval Model

The simplest document language model is the maximum likelihood model $\mathcal{M}_d^{\mathrm{ml}}(t)$ based on the counts of the terms appearing in the document:

$$\mathcal{M}_d^{\mathrm{ml}}(t) = \frac{f_{t,d}}{l_d}.$$

$f_{t,d}$ is the number of times t appears in d and l_d is the length of the document. Thus, for terms not appearing in the document $\mathcal{M}_d^{\mathrm{ml}}(t) = 0$.

Because $\mathcal{M}_d^{\mathrm{ml}}(t)$ is nothing more than term frequency scaled by document length, we might suspect that by itself it may not be sufficient to compute estimates of $p(q|d)$ that will provide a satisfactory document ranking. In particular, given that d is just a single example of a relevant document and that d may consist of only a few hundred words, the estimates it provides have the potential to be wildly inaccurate. Moreover, the model assigns a probability of 0 to all terms not appearing in the document, implying that it is impossible for these terms to appear in the query.

To address these problems when using language models based on documents or examples of similar size, it is common in information retrieval, as well as in other areas such as speech recognition, to *smooth* these models using a *background language model* in the hope of improving accuracy. In information retrieval the collection as a whole provides a convenient basis for this background model.

We define $\mathcal{M}_C(t)$ as the maximum likelihood language model based on term frequencies in the collection as a whole:

$$\mathcal{M}_C(t) = \frac{l_t}{l_C},$$

where l_t is the number of times t occurs in the collection C and l_C is the total number of tokens in the collection.

The maximum likelihood language model based on *Dirichlet smoothing* is

$$\mathcal{M}_d^{\mu}(t) = \frac{f_{t,d} + \mu \cdot \mathcal{M}_C(t)}{l_d + \mu},$$

where $f_{t,d}$ is the number of times t appears in the original document and l_d is the original length.

If a term t does not appear in a document d, then $\mathcal{M}_d^{\mu}(t) = (\mu / l_d + \mu) \cdot \mathcal{M}_C(t)$.

Uni-gram language model probabilities are computed for each document in the top-100 documents retrieved for each query, and also uni-gram probabilities for background model using all documents retrieved collectively.

To handle out of vocabulary terms, some probability mass is reserved for the <UNK> token. Any token in the query or otherwise, not in the documents retrieved are treated as the <UNK> token.

2 Ranking using Relevance Model

The Kullback-Leibler divergence measures the difference between two probability distributions. Given the *true* probability distribution P and another distribution Q that is an approximation to P, the KL divergence is defined as:

$$KL(P \parallel Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)}$$

Since KL-divergence is always positive and is larger for distributions that are further apart, we use the negative KL-divergence as the basis for the ranking function. In addition, KL-divergence is not symmetric, and it matters which distribution we pick as the true distribution. If we assume the true distribution to be the relevance model for the query (R) and the approximation to be the document language model (D), then the negative KL-divergence can be expressed as

$$\sum_{w \in V} P(w|R) \log P(w|D) - \sum_{w \in V} P(w|R) \log P(w|R)$$

where the summation is over all words w in the vocabulary V . The second term on the right-hand side of this equation does not depend on the document, and can be ignored for ranking.

More formally, we can relate the probability of w to the conditional probability of observing w given that we just observed the query words $q_1 \dots q_n$ by the approximation:

$$P(w|R) \approx P(w|q_1 \dots q_n)$$

By definition, we can express the conditional probability in terms of the joint probability of observing w with the query words:

$$P(w|R) \approx \frac{P(w, q_1 \dots q_n)}{P(q_1 \dots q_n)}$$

$P(q_1 \dots q_n)$ is a normalizing constant and is calculated as:

$$P(q_1 \dots q_n) = \sum_{w \in V} P(w, q_1 \dots q_n)$$

Now the question is how to estimate the joint probability $P(w, q_1 \dots q_n)$. Given a set of documents C represented by language models, we can calculate the joint probability as follows:

$$P(w, q_1 \dots q_n) = \sum_{D \in C} p(D) P(w, q_1 \dots q_n|D)$$

We can also make the assumption that:

$$P(w, q_1 \ldots q_n|D) = P(w|D) \prod_{i=1}^{n} P(q_i|D)$$

When we substitute this expression for $P(w, q_1 \ldots q_n|D)$ into the previous equation, we get the following estimate for the joint probability:

$$P(w, q_1 \ldots q_n) = \sum_{D \in C} P(D)P(w|D) \prod_{i=1}^{n} P(q_i|D)$$

The prior probability $P(D)$ is usually assumed to be uniform and can be ignored. The expression $\prod_{i=1}^{n} P(q_i|D)$ is, in fact, the query likelihood score for the document D. This means that the estimate for $P(w, q_1 \cdots q_n)$ is simply a weighted average of the language model probabilities for w in a set of documents, where the weights are the query likelihood scores for those documents.

The following is a summary of the steps involved in ranking using relevance models:

1. Rank documents using the query likelihood score for query Q.
2. Select some number of the top-ranked documents to be the set C.
3. Calculate the relevance model probabilities $P(w|R)$ using the estimate for $P(w, q_1 \ldots q_n)$.
4. Rank documents again using the KL-divergence score:

$$\sum_{w} P(w|R) \log \frac{P(w|R)}{P(w|D)}$$

The document language model probabilities ($P(w|D)$) should be estimated using Dirichlet smoothing.

3 Document Pre-Processing Techniques

The following text pre-processing techniques were used:

- **Casing** : Convert the text into lowercase or not.
- **Stopwords** : Remove stopwords given in ntlk.corpus.stopwords
- **Contractions** : Expand contractions, like you're to you are, from a contractions dictionary
- **Punctuations** : Remove punctuations or keep them
- **Digits** : Remove digits or keep them
- **Stemming** : Option to do stemming using the Snowball Stemmer from NLTK

4 Embeddings for Query Expansion

For local embeddings, first relevance corpus is created from the top-K (top-100) documents retrieved for each query. This corpus is used to train the *Word2Vec* embeddings for each query. This gives an embeddings matrix $\mathbf{U}$ of size $V \times d$, where V is the size of the vocabulary and d is the dimensions of the embeddings trained.

The embeddings matrix $\mathbf{U}$ is given in the case of generic embeddings.

We create the query vector $\mathbf{q}$ of dimension $V \times 1$, where each element denotes how many times a vocabulary term is present in the query. If a query term is not in the vocabulary, then it is treated as the <UNK> token.

We then compute $\mathbf{U}\mathbf{U}^T\mathbf{q}$, which is a $V \times 1$ vector denoting the proximity of each term in the vocabulary to the query terms. We pick the *Top-K* terms for this, and these are the **Query Expansion** terms.

We use the weights of the previous multiplication and normalize them to get the Language Model corresponding to the expansion terms, where terms closer to the query terms are given more weight. This model is called p_{q^+}.

We interpolate the original query language model (p_q), with this new model, to get the final query language model, according to the following equation.

$$p_{q'}(w) = (1 - \lambda)p_q(w) + \lambda p_{q^+}(w)$$

To re-rank the documents, we then use the *KL-Divergence* between the Document Language models and $p_{q'}$:

$$\mathcal{D}(q'||d) = \sum_{w \in \mathcal{V}} p_{q'}(w) \log \frac{p_q(d)}{p_d(d)}$$

$\mathcal{D}$ is the scoring function we use for ranking.

5 Experiments and Results

Experiments were conducted for all of the below choices for local embeddings:

- **Word2Vec Model** : Skip-Gram or CBOW (1 or 0)
- **Model window** : Maximum distance between the current and predicted word within a sentence (3, 5, 7, 9)
- **Dirichlet Mu** : (250, 500, 750, 1000, 1250)
- **Top-N** : Top N most similar words (5, 10, 15, 20)

- **Lambda** : Weight given to expanded query terms (0.15, 0.3, 0.45, 0.6, 0.75, 0.9)

final_answer_w2v_local

Filename	Avg_nDCG@5	Avg_nDCG@10	Avg_nDCG@50	Avg_nDCG
output-file_Skip_win_5@mu_750@Top_15@LAMBDA_0.15	0.4457650774936070	0.43817196578542600	0.4523426925212010	0.445426578600078
output-file_Skip_win_5@mu_1250@Top_20@LAMBDA_0.15	0.4447736293462410	0.43576574751759000	0.45272859965344000	0.44442265883909
output-file_Skip_win_9@mu_1000@Top_10@LAMBDA_0.15	0.4418502429376640	0.44050757453904800	0.44958301083548600	0.443980276104066
output-file_Skip_win_5@mu_750@Top_10@LAMBDA_0.15	0.44820754778141400	0.43645098095452200	0.44692702197086000	0.44386185023559900
output-file_Skip_win_7@mu_1250@Top_20@LAMBDA_0.15	0.4452487555155880	0.4390917239782650	0.44635097378768100	0.443563817760511
output-file_Skip_win_7@mu_1000@Top_5@LAMBDA_0.15	0.4432492530868930	0.4343769782986100	0.4530378989096170	0.4435547100983730
output-file_Skip_win_7@mu_750@Top_10@LAMBDA_0.15	0.44388111919439100	0.44014824154978100	0.44604510730737900	0.44335815601718400
output-file_Skip_win_7@mu_750@Top_5@LAMBDA_0.45	0.44384099249537700	0.4293728887233480	0.4567651425498150	0.44332634125618
output-file_Skip_win_5@mu_1000@Top_15@LAMBDA_0.15	0.4448193188801520	0.4357774083282830	0.4474344269258830	0.4426770513781060
output-file_Skip_win_9@mu_750@Top_5@LAMBDA_0.15	0.44301127315992900	0.43402856318845300	0.45090781829319000	0.44264921821385700
output-file_Skip_win_5@mu_1000@Top_10@LAMBDA_0.15	0.4441284930658720	0.4382144129978860	0.44542546587250300	0.442589457312087
output-file_Skip_win_7@mu_1000@Top_15@LAMBDA_0.3	0.4400808372150420	0.4382336284187120	0.4490739667671470	0.4424628108003000
output-file_Skip_win_7@mu_500@Top_5@LAMBDA_0.3	0.4375010630327110	0.4293400182838820	0.45995139471438200	0.442264158676992
output-file_CBOW_win_5@mu_1000@Top_10@LAMBDA_0.15	0.4430234639350680	0.4358643905440250	0.44752599489520400	0.442137949791432

Figure 1: Word2Vec Local Embeddings Parameter Results

Out of 960 permutations of parameters, the best set of parameters are as follows:

Parameter	**Choice**
W2V Model	Skip-Gram
Model Window	5
Smoothing Constant (Dirichlet μ)	750
Top-N	15
Expansion Interpolation Constant (λ)	0.15

Table 1: Parameter values for Text Pre-Processing

Similarly for generic embeddings, best results are obtained for

Using Word2Vec : Top-N $= 5$ and $\lambda = 0.15$
Using Glove : Top-N $= 20$ and $\lambda = 0.15$

For choices of parameters of text pre-processing, the results are as follows:

Permutation	Avg_nDCG@5	Avg_nDCG@10	Avg_nDCG@50	Avg_nDCG
lowercase_True	0.430845284448932	0.423292575342471	0.439446315500343	0.431194725097249
lowercase_False	0.407419406210777	0.420107742153921	0.431756854742594	0.419761334369097
Remove_stopwords_True	0.430845284448932	0.423292575342471	0.439446315500343	0.431194725097249
Remove_stopwords_False	0.375257107523368	0.374285537934846	0.413674215442402	0.387738953633539
Expand_Contractions_True	0.430845284448932	0.423292575342471	0.439446315500343	0.431194725097249
Expand_Contractions_False	0.427338198037238	0.423834421066937	0.438357413729283	0.429843344277819
Remove_Punctuations_True	0.430845284448932	0.423292575342471	0.439446315500343	0.431194725097249
Remove_Punctuations_False	0.425582948516144	0.422402630663633	0.437822243200021	0.428602607459933
Remove_Digits_False	0.430845284448932	0.423292575342471	0.439446315500343	0.431194725097249
Remove_Digits_True	0.426849697917287	0.430080924731191	0.439848397668624	0.432259673439034
Stemming_False	0.426849697917287	0.430080924731191	0.439848397668624	0.432259673439034
Stemming_True	0.398125548588306	0.414421727631391	0.437579020170998	0.416708765463565

Figure 2: Text Pre-Processing Parameter Results

Hence the best set of parameters for text pre-processing are as follows:

Parameter	**Choice**
Lowercase Text	True
Remove Stopword	True
Expand Contractions	True
Remove Punctuation	True
Remove Digits	True
Stemming	False

Table 2: Parameter values for Text Pre-Processing

Finally comparing the scores for the three models:

MODEL	Avg_nDCG@5	Avg_nDCG@10	Avg_nDCG@50	Avg_nDCG
W2V Local	0.445765077493607	0.438171965785426	0.452342692521201	0.445426578600078
W2V Generic	0.425610255343933	0.425293098842938	0.434910043245275	0.428604465810715
Glove Generic	0.413562787642334	0.413645478024595	0.434345139966010	0.420517801877646

Figure 3: Model Comparison Results

Hence, the best performance is when Word2Vec is trained in local embeddings.