WSM Project 2: Building IR systems based on the Lemur Project

111753151 ZHI RONG CHENG

December 3, 2022

1 Abstract

In this project we have implemented several different retrieval methods, i.e. algorithms that given a user's request (query) and a corpus of documents, assign a score to each document according to its relevance to the query. These retrieval methods included Vector Space Model with terms weighted by Okapi TF and Language Model with different smoothing methods (e.g. Laplace Add1 Smoothing, Jelinek-Mercer Smoothing). We also compare the effect of word stemming in each model, and it shows that in most cases stemming the corpus could make a better performance in the retrieval system.

Empirically, the language modeling approach is always very effective in retrieval performance and does better than other methods, even beating TF-IDF and BM25 weights [1]. However, in our experiment it shows that with different smoothing methods implemented the performance ratio will fluctuate accordingly, and it couldn't outperform the well-tuned Vector Space Model with Okapi term weighted.

Furthermore, we also apply a simple method which can obviously increase the retrieval performance of Laplace Add1 Smoothing in our experiment, and its outcome is nearly as good as the Okapi BM25 Vector Space Model, that is Laplace Addk Smoothing.

2 Introduction

Development and optimization of retrieval models are very important research targets in information retrieval, and the main topic of the performance evaluation is the retrieval relevance. There are two branches of methodologies using different indicators to score the retrieved documents, one is comparing the similarity between the document vectors, and the other is calculating the probability of the outcome generated [2].

The Vector Space Model is a representative model of the similarity methodology, which is a traditional way to represent the document's relevance. On the other hand, probabilistic methodology such as Regression Model, Generative Model are both recently popular research topic, and the Generative Model can even be further classified as Document Generation method (e.g. Robertson-Sparck Jones Model) and Query Generation method (e.g. Language Model).

Previous work has shown that all the effective retrieval models tend to rely on a reasonable way to combine multiple retrieval signals, such as term frequency(TF), inverse document frequency(IDF) and document length [3]. In fact, the powerful Okapi BM25 (BM is an abbreviation of best matching) formula proposed by Stephen E. Robertson, Karen Spärck Jones is a result of incorporating these heuristics into the classical probabilistic Robertson-Sparck Jones Model. In our project study, we will adjust the traditional Vector Space Model which uses the TF-IDF as its term weight, to the state-of-art Okapi BM25 Term weighted model, so as to achieve better retrieval.

What's more, although the Language Model seems to be an overpowering retrieval approach, it still has some limit that the prior parameter estimation is needed, and the smoothing method has to be considered in order to avoid the zero probability problem. In our study, we also discuss some cases with different smoothing methods, and details in how to estimate the parameters. Further information about the model functionality and formula which we have mentioned above will be explained in the following sections.

2.1 Vector Space Model with different term weight

Vector Space Model represents a document or query by a term vector as follows, a term could be defined as a single word, keyword, or longer phrase, depending on the application.

$$d_j = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})$$
$$q = (w_{1,q}, w_{2,q}, \ldots, w_{n,q})$$

In the vector each term defines one dimension, and hence with a large amount of terms it will define a high-dimensional space. Each element of the term vector corresponds to a term weight which indicates the importance of the term, and this will become the basis of calculating the similarity score. The classical Vector Space Model uses the TF-IDF (term frequency-inverse document frequency) to represent term weights, the idea of TF weighting is that a term is more important if it occurs more frequently in a document, and the idea of IDF weighting is a term will be more discriminative if it occurs only in fewer documents. There are a couple of methods which could be utilized to calculate TF score, here we list the formula below, note that $f(t, d)$ is the frequency count of term t in document d [2].

- Raw TF: $TF(t, d) = f(t, d)$
- Log TF: $TF(t, d) = log(f(t, d) + 1)$
- Maximum frequency normalization: $TF(t, d) = 0.5 + 0.5 * f(t, d)/MaxFreq(d)$
- Okapi BM25 TF: $TF(t, d) = k1 * f(t, d)/(f(t, d) + k1(1 - b + b * doclen/avgdoclen))$

The method of IDF calculation is relatively simple, we could just divide the number of documents with the number of documents with term t in the corpus, and take log to the result. The following shows the formula [2].

- $IDF(t) = log(n/k)$, where
  - n: number of documents.
  - k: number of documents with term t (documents frequency)

The weight of each term in the vector space will be calculated as $TF(t, d) * IDF(t)$, and after weighting the terms, we still have to decide how to rate the similarity between the query and each document. There are also a couple of options that could be selected, the main idea is if the distance between a vector pair (a query with a document, or a document with another document) is small, it represents that these two vectors are more similar with each other. Here we list some formula options below [2].

- Inner Product: $sim(\overrightarrow{Q}, \overrightarrow{D_i}) = \sum_{j=1}^{V} w_{q,j} * w_{i,j}$
- Cosine: $sim(\overrightarrow{Q}, \overrightarrow{D_i}) = \frac{\sum_{j=1}^{V} w_{q,j} * w_{ij}}{\sqrt{\sum_{j=1}^{V}(w_{q,j})^2 * \sum_{j=1}^{V}(w_{i,j})^2}}$
- Euclidean distance: $d(p, q) = \sqrt{(p - q)^2}$

The Vector Space Model is an empirically effective method in retrieval system (Top TREC performance), which is easy to implement and well-studied. However, since there are arbitrary term weighting and similarity measures that can be chosen from, it is quite a problem in picking the most suitable combinations to develop the retrieval system.

In our experiment, we select Okapi BM25 as the TF weighting, and simply use the inner product to calculate the similarity score. Okapi BM25 (BestMatch25) is a concept of probabilistic model that derived from Robertson-Sparck Jones Model (a.k.a RSJ Model). The origin RSJ model uses only term presence or absence for ranking score, and hence its performance isn't as good as well-tuned Vector Space Model. The variant of RSJ Model, which means BM25, incorporates the idea of Term Frequency(TF), Inverse Document Frequency(IDF) and document length into the formula, and hence achieve the top TREC performance. The following shows some details about Okapi BM25 TF [2].

- $TF(t,d) = k1 * f(t,d)/(f(t,d) + k1(1 - b + b * doclen/avgdoclen))$, where
  - $f(t,d)$: term frequency in document d.
  - $avgdoclen$: : length of document d (average document length in the whole collection).
  - $k1$: tuning parameter controlling the document term frequency scaling.
  - b: tuning parameter controlling the scaling by document length.

By tuning parameters $k1$ and b, we can control the different scaling of Term frequency and document length. In our testing we set $k1 = 2$ and $b = 0.75$.

2.2 Language Model with different smoothing method

Language Model (LM) is derived from the concept of finite state automaton, each model will constantly generate words with different probability, until it meets the STOP signal to form a sample(a word string, or sentence). For a given sample, we can calculate the probability that a model will randomly generate this sample by referencing its probabilistic distribution of words, and hence we can rank the relevance to this sample between different models. In the research of information retrieval, people try to incorporate the concept of Language Model so as to model the query generation process. In the LM approach, each document is treated as a language model, and hence we can rank the relevance with a given query between different documents by calculating $P(q|Md)$, where q means query and Md means document. To calculate $P(q|Md)$, we have to make some conditional independence assumption as for Naive Bayes, it means:

$$P(q|M_d) = P(\langle t_1, \ldots, t_{|q|}\rangle|M_d) = \prod_{1 \le k \le |q|} P(t_k|M_d)$$

where $|q|$ is the length of query, tk is the token occurring at position k in q. this equation is equivalent to:

$$P(q|M_d) = \prod_{distinct\ term\ t\ in\ q} P(t|M_d)^{tf_{t,q}}$$

, which can be seen as a Multinomial Model omitting its constant factor. Next, we use the Maximum Likelihood Estimates (MLE) to estimate $P(t|M_d)$, it means $\hat{P}(t|M_d) = \frac{tf_{t,d}}{|d|}$, where $|d|$ is the length of document, and $tf_{t,d}$ is the number of occurrences of a term in document. However, this equation for calculating $P(q|M_d)$ still has a problem with zeros, whenever there exist a single term with $P(t|M_d) = 0$, it will make $P(q|M_d) = \prod P(t|M_d)$ becomes zero. As a result, we need smoothing methods to avoid zeros. The key concept of smoothing is discounting the probabilities of observed words in a document, and re-allocate these extra probability to those unseen words, hence it can "smooth" the Language Model. The illustration of Figure 1 well explains the smoothing concept.

Researchers have proposed several smoothing methods, such as Jelinek-Mercer smoothing and Dirichlet smoothing method, to solve the zero probability problem. We will further explain this in the following subsections.

2.2.1 Jelinek-Mercer smoothing

Jelinek-Mercer smoothing, which is also known as linear interpolation method, is a form of mixture model that mixes the probability from the document with the general collection frequency of the word. Below we show the formula:

$$P(q|d, C) = \lambda * P(q|M_d) + (1 - \lambda) * P(q|M_c)$$

With the high value of λ, the search system tends to retrieve documents containing all query words, on the contrary with low value of λ the search system will be more disjunctive and suitable for long queries. Note that the $1 - \lambda$ is a constant value shrinking uniformly toward $P(q|M_c)$, and it brings some variants of Jelinek-Mercer smoothing by replacing λ in the formula, such as Dirichlet smoothing. In our experiment we set λ as 0.8.

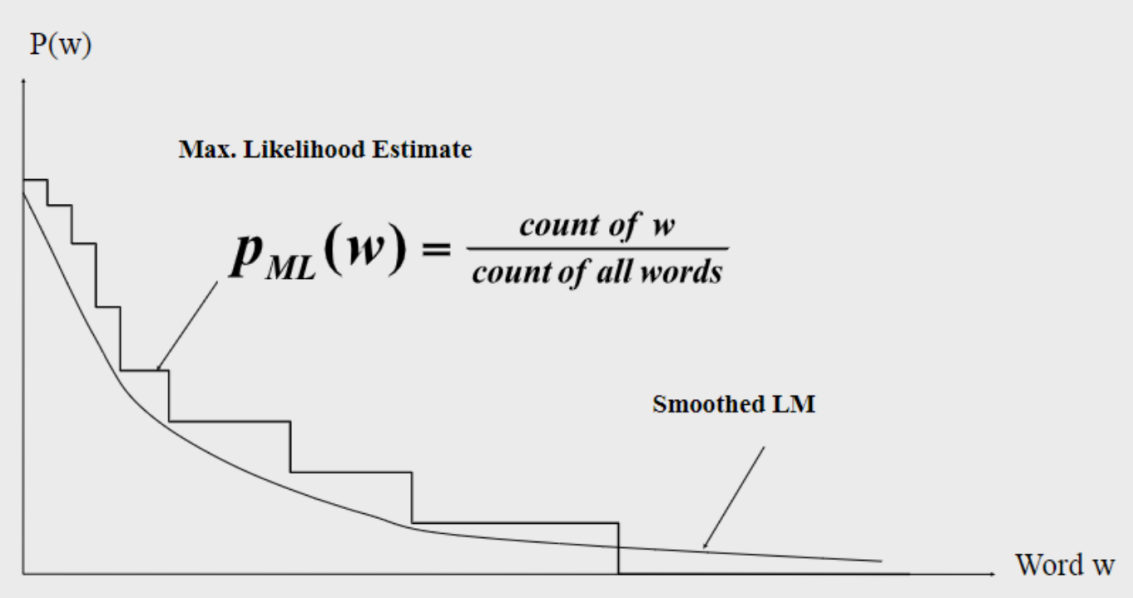

Figure 1: The concept of smoothing

2.2.2 Dirichlet smoothing

Dirichlet smoothing is a special case of Jelinek-Mercer smoothing assuming pseudo counts by $\mu * p(w|C)$. Here we shows the formula:

$$P(w|d) = \frac{c(w,d) + \mu p(w|C)}{|d| + \mu} = \frac{|d|}{|d| + \mu} P_{ml}(w|d) + \frac{\mu}{|d| + \mu} P(w|C)$$

It shows that the origin $1 - \lambda$ parameter in Jelinek-Mercer smoothing has been replaced by $\lambda = 1 - \frac{|d|}{|d|+\mu}$ and the value m setted will affect the result of smoothing. In this model, as the document size $|d|$ getting larger, it will take $P_{ml}(w|d)$ more into account while scoring this document. By the way, we can also adjust some parameters of the Dirichlet smoothing method to get the variant formula, in our case it is the Laplace smoothing.

2.2.3 Laplace smoothing

There are two ways to explain this method. Firstable, it could be seen as the variation of Dirichlet smoothing by setting the μ value as k (number of unique terms in corpus) and replacing $P(w|C)$ with $\frac{1}{k}$. The second perspective is that we look back to the fundamental definition of Maximum Likelihood method (MLE), we estimate $P(t|M_d)$ by MLE with the formula below:

$$MLE: \quad P(w_i) = \frac{C(wi)}{\sum_j C(w_j)} = \frac{C(w_i)}{N}$$

, where $\sum_j C(w_j)$ equals the document length. Now, we are interested in eliminating the effect of zero probability. Actually we can just add one to each word count while estimating, and hence we get the formula of Laplace Add1 Smoothing method.

$$Add\ One: \quad P(wi) = \frac{C(w_i) + 1}{\sum_j C(w_j) + 1} = \frac{C(w_i) + 1}{N + V}$$

The advantage of Laplace Add1 smoothing is that it is quite simple to implement, however, the disadvantage of this approach is it takes away too much probability mass from seen words, and assigns too much total probability mass to unseen words. As a result, we can just lower the assigned value from 1 to k, where $0 < k < 1$, so as to alleviate the side effect, and this is known as Laplace Addk smoothing. In our experiment, we set $k = 0.25$ and find that it really does improve this IR model. In the next section we will showcase our experiment results.

3 Experiments

3.1 Testing Collections and Evaluation

We utilize the Lemur toolkit and the Indri search engine (http://www.lemurproject.org/) to carry out our experiment, and also take advantage of the TREC Web Corpus, WT2G to test our algorithm. We construct two indexes (a) With stemming (b) Without stemming for WT2G corpus, both indexes contain stopwords. For stemming, we use the Porter stemmer in our experiment, Figure 2 shows the information of our datasets.

	documents	unique terms	total terms
stemming	247491	1342611	261742791
Without stemming	247491	1525589	261742791

Figure 2: Datasets information

For the query input, we use a set of 50 TREC queries for the WT2G corpus, and only use the topic title field of the standard TREC format. We run this set of queries against the collection, return a ranked list of documents (the top 1000 for each query) in a particular format, and use a program of $trec_eval.pl$ to evaluate the retrieval quality. The models we have tested are the Vector Space Model with Okapi BM25, Jelinek-Mercer smoothing, Laplace Add1 smoothing and its improvement, Laplace Addk smoothing. Here we list the parameters setted in each model:

- Vector Space Model with Okapi BM25: $k1 = 2, b = 0.75$
- Laplace Addk smoothing: $k = 0.25$
- Jelinek-Mercer smoothing: $\lambda = 0.8$

3.2 Testing results

We basically analyze these four models with their relevant document retrieved, Un-interpolated mean average precision and Precision at rank 10 between stemming and without-stemming index, while we also further showcase their average interpolated Recall-Precision curve and recall precision for the more detailed comparison. The code name we use to express each model are listed below:

- Vector Space Model with Okapi BM25: **tfidf**
- Jelinek-Mercer Smoothing: **jelinek**
- Laplace Add1 smoothing: **Laplace**
- Laplace Addk smoothing: **addK**

Later we'll use these code names to introduce our analysis, without using the verbose model names. First we have noticed that with the index stemmed, the relevant document retrieved will be obviously increased (See Figure 3, Figure 4).

The most likely reason is that all the models we use are based on term matching, which is suffering from vocabulary mismatch, hence pre-stemming the corpus will have a positive effect on retrieval performance. Furthermore, the table shows that the tfidf method performs the best in terms of the relevant document retrieved, and the ranking order of these four models is **tfidf > jelinek > Laplace > addK**. Besides, we can find that our improving method, Laplace Addk smoothing, doesn't beat the original Add1 method in this performance ratio.

However, as we continue to look at the performance of un-interpolated mean average precision and precision at 10 in these models, the result shows that the tfidf method still outperforms the other approaches, but our implemented addK method obviously perform better than the origin add1 method, and it is almost comparable with the tfidf method (See Figure 5 to Figure 8). This outcome shows

	Retrieved	Relevant	Relevent_retrieved
addK	48185	2279	1198
jelinek	48185	2279	1440
Laplace	48185	2279	1328
tfidf	48185	2279	1491

Figure 3: Document without stemming

	Retrieved	Relevant	Relevent_retrieved
addK	48887	2279	1421
jelinek	48887	2279	1581
Laplace	48887	2279	1495
tfidf	48887	2279	1699

Figure 4: Document with stemming

that the improvement based on reducing total probability mass assigned to unseen words does work in some performance ratio, but in some cases if we need to get the relevant document retrieved as much as possible, then the addK method might not be the best approach we should consider to use.

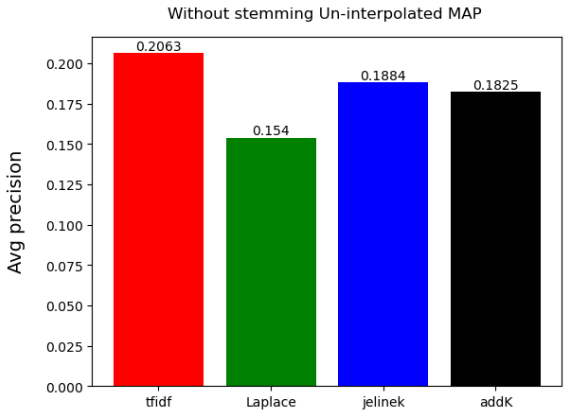

Figure 5:

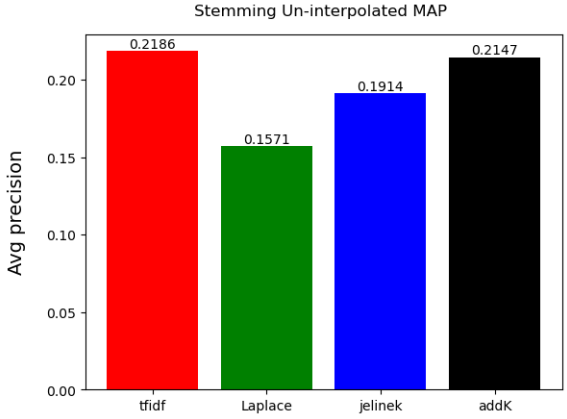

Figure 6:

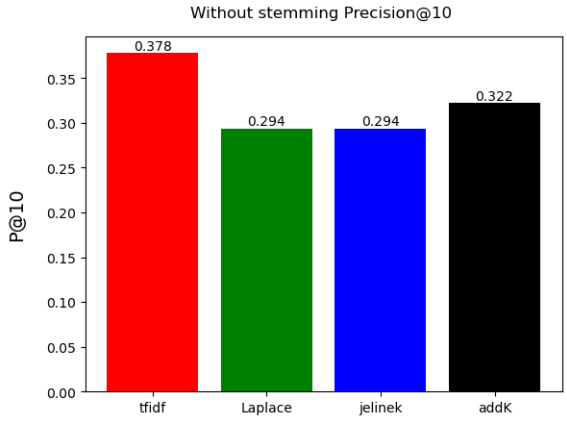

Figure 7:

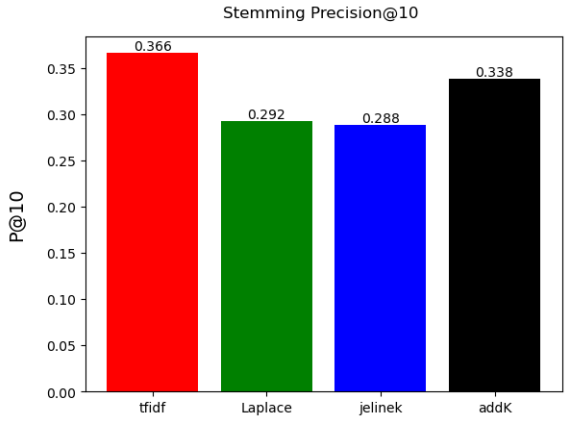

Figure 8:

Here we also showcase the analysis results of average interpolated Recall-Precision curve and R-Precision, for further conclusion. (See Figure 9 to Figure 12) In terms of the overall performance based on each ratio, we could say that basically the retrieval ranking order for our testing is **tfidf > addK > jelinek > add1**.

4 Conclusions

In this project, we have researched three popular retrieval models plus one improvement model based on Laplace Add1 smoothing Language Model in IR field, and actually do some experiments on these models by utilizing the TREC WT2G datasets. Our conclusion is that the Vector Space Model with well tuned Okapi BM25 term weights can perform the best in our datasets, and the improvement of Add1 smoothing, Addk smoothing, does work in most performance ratios. For more details about our experiment, we find that our set of 50 TREC queries for the WT2G corpus fail to retrieve documents if we haven't done some pre-process of each query, such as replacing the punctuation marks. Besides, since we only use the topic title field of the standard TREC format, which means short query for testing, it doesn't mean that our experiment result will always act the same in other cases. Further and more precise experiments should be designed, and that will be our next target in the relevant research.

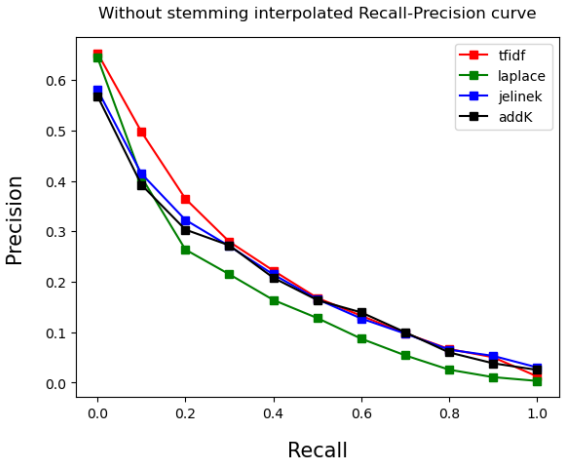

Figure 9:

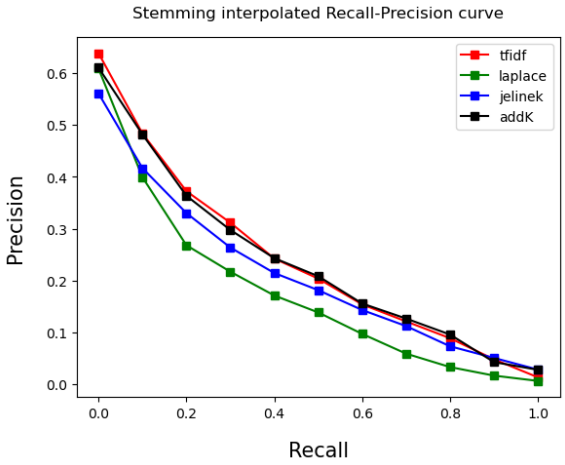

Figure 10:

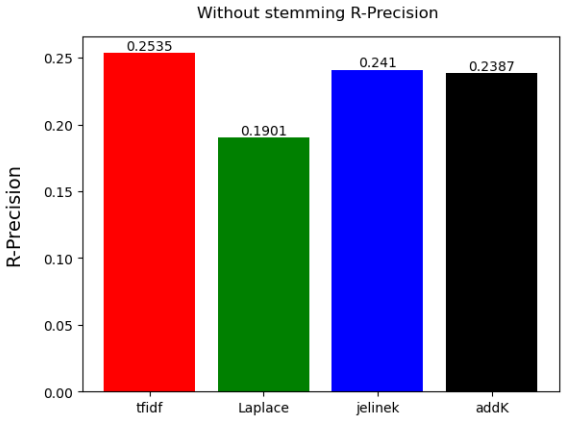

Figure 11:

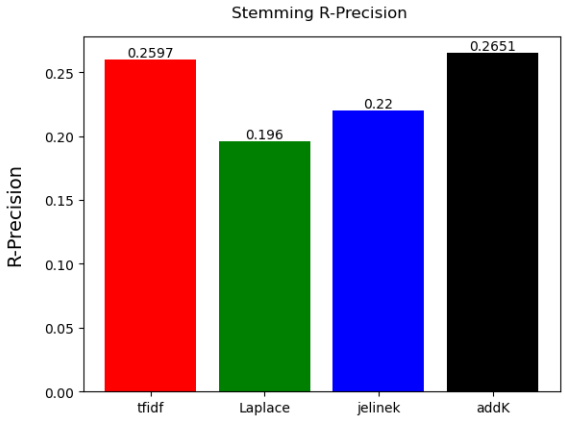

Figure 12:

5 References

1 Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

2 Ming-Feng Tsai (Victor Tsai) Dept. of Computer Science National Chengchi University, Web Search and Mining.

3 Lv, Yuanhua, and ChengXiang Zhai. "Lower-bounding term frequency normalization." Proceedings of the 20th ACM international conference on Information and knowledge management. 2011.