BABES-BOLYAI University of Cluj-Napoca

##### Faculty of Mathematics and Computer Science

**Proceedings of the National Symposium ZAC2014**

 (Zilele Academice Clujene, 2014)

**User-weighting query terms in text retrieval**

by

I. Badarinza and A.Sterca [[1]](#footnote-1)

**Abstract**. *In this paper we present a technique which allows the user to alter the weights of query terms in a textual retrieval system so that it returns more relevant results. This technique is not meant to increase the relevancy of results returned for general search queries, but is meant to increase the relevancy of the returned results for some specific queries in which the query terms have disproportionate IDF values.*

**1. Introduction**

In Information Retrieval (IR), the items we are trying to retrieve are called documents, and each document is described by a collection of terms. These two words, ‘document’ and ‘term’, are now traditional in the vocabulary of IR. Usually a document is seen as a piece of text, and a term as a word or phrase which helps to describe the document, and which may occur one or several times in the document. So, for example, a document might be about military activity, and could be described by corresponding terms ‘gun’, ‘action’, ‘soldier’, ‘uniform’, ‘war’, ‘captain’, ‘force’ and so on. More generally a document can be anything we want to retrieve, and a term any feature that helps describe the document.

In all IR systems, a score of each document from the collection is computed against the query and the top k documents (where k is usually 10) are returned. The score of a document is computed as a sum against each term of the query. The score of a document with respect to a query term usually has two parts:

* a component that measures the importance of the query term for this document (i.e. number of occurrences of the query term in the document; *Term Frequency*)
* a component that measures the discriminative power (i.e. specificity) of the query term inside the set of all terms from the document collection (i.e. *Inverse Document Frequency*)

More formally specified, the score of a document D against a query Q has the following form:

$$Score\left(Q,D\right)=\sum\_{i=1}^{n}tf(q\_{i}, D)×idf(q\_{i})$$

where *Q=(q1, q2, … qn)* is the query, *qi* are query terms, $tf\left(q\_{i}, D\right)$ is the term frequency of term *qi* in document *D* and $idf(q\_{i})$ is the inverse document frequency of term *qi*. The score of a document *D* against a query *Q* can be seen as a weighted sum of term frequencies, where the weight is given by the $idf\left(q\_{i}\right)$ term.

In this paper we present a technique for improving the relevancy of the documents returned by an IR system, by allowing the user to alter the default weights of the query terms in computing the document score.

**2. Ranking functions used in IR systems**

In this section we present some popular ranking functions used in IR systems, namely the Binary Independence Model used with the Probabilistic Ranking Principle, BM25 and BM25F.

Binary Independence Model and Probabilistic Ranking Principle

In the Binary Independence Model and Probabilistic Ranking Principle [4] we consider *tfi* to be a binary variable that will have only the values zero and one. We can say that this means that a term is present in a document (one) or it is not present in a document (zero). The event when the term is absent is the complement of when the term is present; probability of absence is one minus probability of presence. [5]

$$w\_{i}=log⁡(\frac{P\left(t\_{i} \right| rel) (1-P\left(t\_{i} \right| \overbar{rel}))}{(1-P\left(t\_{i} \right| rel)) P\left(t\_{i} \right| \overbar{rel})})$$

As mentioned above, the property of relevance is represented by a random variable *Rel* with two possible values: $rel$, $\overbar{rel}$ (relevant or not). Further we will use the short notation *P (rel | d, q)* to describe *P (Rel = rel | d, q)*. In the above formula: *ti* is the event that the term *ti* is present in the document; *rel* is the event that the document is relevant to the query; *P(ti | rel)* is the probability that the term *ti* is present in a document, knowing that the document is relevant; $P\left(t\_{i} \right| \overbar{rel})$ is the probability the term *ti* is present in the document, knowing that the document is not relevant. These probabilities can be estimated in the following way. Because these are conditional on the relevance property, we can assume that we have some judgements of relevance. First, we can assume that we have a random sample of the whole collection that was judged for relevance, than derive an estimator that will be used further. Let’s consider:

*N* – the size of a judged sample;

*ni* – the number of documents from the judged sample containing *ti*;

*R* – the relevant set size (i.e., number of documents judged relevant);

*ri* - the number of judged relevant documents that contain *ti*.

Given this information, we can estimate the probabilities from as follows:

$$P\left(t\_{i} \right| rel)=\frac{n\_{i}}{R}$$

$$P\left(t\_{i} \right| \overbar{rel})=\frac{n\_{i}-r\_{i}}{N-R}$$

After replacing the probabilities and trying to obtain a more robust estimator by introducing a pseudo-count of frequency 0.5, as demonstrated in [6] we get the well-known Robertson/Sparck Jones weight [5]:

$$w\_{i}=log\left(\frac{\left(r\_{i}+0.5\right) (N-R-n\_{i}+r\_{i}+0.5)}{\left(n\_{i}-r\_{i}+0.5\right) (R-r\_{i}+0.5)}\right)$$

BM25

BM25 if the fundamental result of the Probabilistic Relevance Framework for document retrieval created in 1970-1980’s. BM25 is one of the most successful algorithms used in information retrieval systems. In an information retrieval system we cannot know the values of the relevance property for each document so we might say that the information given by the system is probabilistic. Based on probabilistic 2-Poisson model, BM25 algorithm will return the documents that are potentially relevant with our information need. The score formula of the BM25 algorithm is [1]:

$$Score\left(Q,D\right)=\sum\_{i=1}^{n}log\left(\frac{N-n\_{i}+0.5}{n\_{i}+0.5}\right)\frac{f\left(q\_{i},D\right)\*(k\_{1}+1)}{f\left(q\_{i},D\right)+k\_{1}\*(1-b+b\*\frac{|D|}{avgdl})}$$

where $f\left(q\_{i},D\right)$ is $q\_{i}$’s term frequency in the document *D*, *|D|*is the length of the document *D* in words, and *avgdl* is the average document length in the text collection from which documents are drawn. $k\_{1} $and *b* are free parameters, usually chosen, in absence of an advanced optimization, as $k\_{1}$ ∈ [1.2, 2] and *b* = 0.75 [1]. In this formula the term weight is:

$$log\left(\frac{N-n\_{i}+0.5}{n\_{i}+0.5}\right)$$

BM25F

BM25F is an extended model of BM25 that also incorporates the structure of the documents into the scoring process. One final equation form of BM25F can be sees as [2]:

$$Score\left(Q,D\right)=\sum\_{t in Q}^{}idf\left(t\right)\* \frac{weight(t, D)}{k\_{1}+weight(t, D)}$$

where *idf(t)* is the inverse document frequency and $weight(t, D)$ is defined as [2]:

$$weight\left(t, D\right)= \sum\_{c in D}^{}\frac{occurs\_{t,c}^{D}\*boost\_{c}}{(\left(1-b\_{c}\right)+b\_{c}\*\frac{l\_{c}}{avl\_{c}})}$$

where $l\_{c}$ is the field length, $avl\_{c} $is the average length for field *c*, $b\_{c}$ is a constant related to the field length which is similar to *b* in BM25 and $boost\_{c}$ is the boost factor applied to field *c*.

 **3. Partial user weighting of query terms**

We consider the case when a query has multiple terms, but a query term which has a lower IDF ranking is more relevant for the information need (expressed by the query) than some other query term which has a higher IDF weight (i.e. it appears less frequent in the document collection than the first query term). In probabilistic ranking IR, the score of a document relative to a given query is equal to the sum of the scores of the given document with respect to each query term. The score of the document *D* with respect to query term *ti* can be viewed as $w\left(t\_{i}\right)\*s\left(t\_{i}, D\right)$ where $w\left(t\_{i}\right)$ is a variant of IDF which measures the term's $t\_{i}$ specificity in the whole document collection (i.e. it is the weight of term's $t\_{i}$ importance for the returned results) and $s\left(t\_{i}, D\right)$ is a function of the term frequency of $t\_{i}$ which ranks the current document against other documents containing $t\_{i}$ (i.e. it is the actual score of the document relative to term $t\_{i}$). For example, for the BM25 algorithm, $w\left(t\_{i}\right)=log\frac{N-n\_{i}+0.5}{n\_{i}+0.5}$ and $s\left(t\_{i}, D\right)=\frac{f\left(q\_{i},D\right)\*(k\_{1}+1)}{f\left(q\_{i},D\right)+k\_{1}\*(1-b+b\*\frac{|D|}{avgdl})}$. In this perspective, the total score of a document relative to a given query can be considered a weighted sum of scores, the scores of this document relative to each of the query terms. So, the weight of each score of the document relative to a query term depends only on the specificity (depends on the IDF) of that query term in the document collection. This constitutes a problem for queries where not all query terms are equally important. For example, in a two terms query where one term is a polysemantic, high IDF term (i.e. it's weight/IDF is high) and the other term is a low IDF term with the role of discriminating between the meanings of the high IDF term, the results returned by the BM25 algorithm can be overwhelmed by documents containing non-relevant meanings (relative to the query) of the high IDF term. For example if the query is "jaguar lifespan" and we are searching for the lifespan of the animal jaguar on Google, we get as results many documents about Jaguar, the car, which are not relevant to our query (the documents returned don't even contain the word "lifespan"). This is because the IDF of "Jaguar" is a lot higher than the IDF of "lifespan". Another example is when we search for the height of the TV presenter Bear Grylls using the query "bear grylls height", we get many documents containing data about Bear Grylls but no height information, because the term "Bear Grylls" has a much higher IDF (i.e. weight) than the term "height" which is far more common. Another example is if we look up "Transilvania bani" on Google.ro in order to obtain information about the Transilvania Bank which is a Romanian financial institution. But among the results we also get documents about the Transilvania highway. The term “bani” should discriminate between the two meanings, but its IDF is much lower than the IDF of Transilvania.

 The queries presented above are just specific examples where allowing the user to alter the IDF weights of query terms would help in increasing the relevancy of the returned results. But in a document collection there are numerous pairs of query terms, one with a low IDF and the other with a high IDF, which would benefit from our technique. For example, in the Reuters-RCV1 collection [3], from 21578 indexed documents and 1217726 indexed terms the distribution of IDFs is depicted in Fig.1 and Fig. 2. In Fig. 1 we see for each integer IDF value, the number of indexed terms which have an IDF close to that specific IDF. Because most IDF values in the index are real values (not integer), the IDF of each term is rounded to the closest integer and then, for each integer IDF value the number of terms which have a rounded IDF equal to this one is counted. We can see in Fig. 1 that most indexed terms have an IDF between 3 and 6. But there are numerous terms which have an IDF smaller than 3 or larger than 6, as we can see in this figure. If the query given by the user is formed by a term with an IDF < 3 and another term with an IDF > 6, our technique should improve the relevancy of the returned results.



Fig. 1. Number of indexed terms for specific IDF value in the Reuters collection



Fig. 2. IDF value distribution among indexed terms in the Reuters collection

The same type of IDF values distribution can be seen in Fig. 2 in which the number of indexed terms with a specific IDF value is plotted as a proportion of the total number of indexed terms.

 Our technique of allowing the user to alter the IDF weights of query terms is described in this paragraph. We consider a query made from two terms: *t1* and *t2*. Let’s assume that IDF(*t1*) is large and IDF(*t2*) is small. We want our IR system to give a weight to *t2* larger or just as high as the one of *t1* because *t2* is more relevant than *t1* in the opinion of the user issuing the query. If there is no document in the collection that contains both *t1* and *t2*, a classical IR system will probably return in the first k=10 results only documents containing *t1* (and not *t2*). Or even if there are documents in the collection that contain both *t1* and *t2*, it may still happen that only documents containing *t1* are returned in the top k=10 results, because the IDF(*t1*) is much higher than the IDF(*t2*). On the contrary, we want the IR system to return among top k=10 results also documents that contain term *t2*, because *t2* is more important for the user than *t1*. Assuming that the ranking function of our IR system is:

$$Score\left(Q,D\right)=w\left(t\_{1}\right)\*s\left(t\_{1}, D\right)+w\left(t\_{2}\right)\*s\left(t\_{2}, D\right)$$

where *w(ti)* is the inverse document frequency of *ti* and *s(ti, D)* is a function of the term frequency of $t\_{i}$ in document *D*, we introduce two mechanisms through which the user can alter the default weight of each query term when he specifies the query:

**1) t1 t2:++n** => by specifying a query like this one, the weight of *t2* will become $w\left(t\_{2}\right)+n$, where *n* = 0.5, 1, 1.5, …

**2) t1 t2:+n** => by specifying a query like this one, the weight of *t2* will become $w\left(t\_{2}\right)+n\*0.1\*(w\left(t\_{1}\right)-w\left(t\_{2}\right))$, where *n* = 1,2,3, … 10. When *n=1*, the weight of *t2* will increase with 10% of the initial weight difference between *t1* and *t2*.

 **4. Evaluations**

In order to evaluate the two mechanisms of specifying weights for the query terms, we built an IR system and indexed the Reuters document collection [3]. We considered a query formed by two terms, *coke* and *prices*, where the IDF of *prices* is 2.52 and the IDF of *coke* is 7.07. For evaluating the first mechanism, we ran 11 searches using the query *“coke prices:++n”* where *n* is 0 in the first search, 0.5 in the second search, 1 in the third one and so on until *n*=5 in the 11th search. In each search we took the top 15 documents returned. Considering all 11 searches, there were 27 distinct documents returned. Some of those 27 documents appeared in the top 15 results of all 11 searches performed and some other appeared only in a subset of top 15 results of the 11 searches. In Fig. 3 we can see the rank (i.e. position from 1 to 15; rank 1 represents the most relevant document/position) of each of the 27 documents in each search performed. Out of the 27 documents depicted, documents with IDs 1-7,9-12 contain only the term *“coke”*, documents with IDs 13-27 contain only the term *“prices”* and the document *doc8* contains both terms. For a better view, the ranks achieved by each document are connected by a line. Also for a better view, we showed in Fig. 4 only the documents containing only the term *“coke”* and in Fig. 5 we showed only the documents containing only the term *“prices”* and the document *doc8* which contains both terms. We can see in these 3 figures, that as parameter *n* increases across searches, the documents containing the low IDF term, *“prices”*, start to get a higher rank (i.e. a “high rank” actually means a value closer to 1 in these figures).

****

Fig. 3 The rank of the returned documents in each of the 11 searches (a rank closer to 1 means a more relevant document)

****

Fig. 4 The rank of the returned documents which contain only the term “coke” (a rank closer to 1 means a more relevant document)

****

Fig. 5 The rank of the returned documents containing the term *“prices”* (a rank closer to 1 means a more relevant document)

We performed a similar evaluation for the second mechanism where the weight of the low IDF term is increased with a percent from the weight difference between the two terms. This time we used the query *“coke prices:+n”* for our 11 searches and in the first search n=0, then n=1 for the second search and so on until n=10 for the 11th search. We also considered the top 15 results of each search and we obtain 26 distinct documents from all 11 searches. We can see in figures 6,7 and 8 the same type of results as we have seen in the evaluation of the first mechanism. As *n* increases and so the weight of *“prices”* increase the documents containing this term get a higher rank (i.e. a rank closer to 1).

****

Fig. 6 The rank of the returned documents in each of the 11 searches (a rank closer to 1 means a more relevant document)

****

Fig. 7 The rank of the returned documents which contain only the term “coke” (a rank closer to 1 means a more relevant document)

****

Fig. 8 The rank of the returned documents which contain only the term “prices” (a rank closer to 1 means a more relevant document)

**5. Conclusions**

In this paper we presented two mechanisms which allow the user to artificially increase the weight of a query term in order to improve the relevancy of the returned results in an IR system. We also evaluated these two mechanisms and showed that they can improve the results retrieved by an IR system for the case of specific user queries.

**References**

1. Manning C.D., Raghavan P., Schutze H., An introduction to Information Retrieval, Cambridge University Press, 2009.

2. Perez-Iglesias J., Perez-Aguera J., Fresno V., Feinstein Y. Z., Integrating Probabilistic

Model BM25/BM25F into Lucene, 2009.

3. Reuters Corpus Volume 1, http://trec.nist.gov/data/reuters/reuters.html

4. Rijsbergen V., Joost C., Information Retrieval, 2nd edition, Butterworths 1979.

5. Robertson S., Zaragoza H., The Probabilistic Relevance Framework: BM25 and Beyond.

6. S. E. Robertson and K. Sparck Jones, Relevance weighting of search terms, Journal of

the American Society for Information Science, 1977.

1. Babes-Bolyai University Cluj-Napoca, *ionutb/forest@cs.ubbcluj.ro*  [↑](#footnote-ref-1)