Exercise 1.1 - TF-IDF, BM25 & BM25L, and evaluation (60/100 points)

188.980 Advanced Information Retrieval 2018S

March 13, 2018

Abstract

At the end of this exercise, you should have an understanding of the practical issues of creating an inverted index for information retrieval, being able to implement basic scoring, as well as being able to comprehend and apply advance scoring functions from research papers. The exercise should be done in pairs, but each of the members has to be able to answer questions about the implementation provided. **Make sure you go through the lecture slides and relevant book chapters at least once before starting on this exercise.**

Task

In Exercise 1.1, your task is to index and search **without** the use of any library (like: Lucene, Terrier, etc.[1]), to create a scorer that implements both TF-IDF and BM25 scoring functions, and to implement an improved version of BM25 based on the article published by Yuanhua Lv and ChengXiang Zhai in 2011 at SIGIR and available online at `http://sifaka.cs.uiuc.edu/~ylv2/pub/sigir11-bm25l.pdf` as well as at the end of this Exercise sheet. You also have to evaluate your results by using the `trec_eval` program. Finally, you need to write a short report, upload everything to TUWEL, and present your results in person.

Warning

About a week before the hand-in of this assignment we offer the possibility to do a pre-hand-in (i.e. an optional checkpoint) in order to solve any sort of technical and non-technical problems you may encounter during the execution of the assignment, so please take advantage of it. **Excuses about technical problems at the presentation session will not be tolerated**.

Functionality (70%)

Index

Build an inverted index from the document collection CD4&5 of the TIPSTER collection that is available in TUWEL: TREC8Adhoc.tar.bz2 in TREC8all.zip.

Each file in the provided dataset contains several documents (indicated by the `<DOC>...</DOC>` tags). Single documents contain an ID (`<DOCNO>...</DOCNO>`) and text (`<TEXT>...</TEXT>`). Documents that do not have a `<TEXT>` tag can be ignored.

If you encounter encoding problems reading the dataset, make sure that you use the correct file encoding `ISO 8859-1`.

[1] However, you may use a library for stemming or lemmatizing

Before indexing, apply any combination of the techniques described in Chapter 2 of the *Introduction to Information Retrieval* book (case folding, removing stop-words, stemming, lemmatization). These options should be exposed as parameters in the index creation phase. For the stemming and lemmatization components, you can use a library at your choice.

You should provide an executable file `indexer.sh` (*.bat for windows) that given the necessary parameters generates the required index.

You may use any format for storing the index.

Search

Implement a basic search functionality and provide it as a command line interface (CLI). No GUI is required. The CLI allows the user to pass search parameters and a file that contains the search topics (i.e. `topicsTREC8Adhoc.txt`). The system then returns a list of documents ranked by a similarity function of your choice (in Exercise 1.1 either: TF-IDF, BM25 or BM25L). The scoring functions and their individual parameters need to be exposed as parameters to the CLI.

Details regarding the BM25 modification: Your task is to modify your implementation of the BM25 scoring function and create a new similarity class called BM25L as in the paper, which introduces a parameter δ. For your experiments, you may fix the parameter to a standard value 0.5.

You are provided with a set of 50 topics to search for. The topic file is contained in the dataset and has a simple SGML structure. Each topic is delimited by `<TOP>` tags and you are free to use all or part of their content in your search.

As a recommendation, try to implement topic processing and the actual scoring/ranking as two separate tasks. By topic processing, we understand here simply getting the terms that will be searched in the index. Consider the use or not use of the different elements in the topic document.

Your search should be (moderately) efficient. It is, for example, not OK to perform your collection indexing only at search time, and also loading the inverted index should be reasonably in time, or at least not happen before each search.

The output of the CLI given the previously described parameters should be a ranked list of up to 1000 documents, where each line is in the following format:

$$\{\textit{topic-id}\}\ \textit{Q0}\ \{\textit{document-id}\}\ \{\textit{rank}\}\ \{\textit{score}\}\ \{\textit{run-name}\}$$

where

topic-id is an integer number between 401 and 450;

document-id is an identifier for the document (e.g. LA122690-0033);

Q0 this field is unnecessary to our purposes but must be present;

rank is an integer indicating the rank of the object in the sorted list (normally, this should be ascending from the first line to the last line)

score the similarity score calculated by (it has to not exceed the float precision and normally, this should be descending from the first line to the last line)

run-name a name you give to your experiment (free for you to choose)

Here is a short example of a potential output for all the topics:

```
401 Q0 FBIS3-10899 1 2.813525 grp5-exp1
401 Q0 FBIS3-13322 2 1.0114759 grp5-exp1
...
450 Q0 LA043089-0083 1000 0.08848727 grp5-exp1
```

You should provide an executable `searcher.sh` (*.bat for windows) file that given the necessary parameters (i.e. `[scoring function] [scoring function's paramters] [topic file]`) generates a ranked list as above.

Evaluation

You must use `trec_eval` (available at: `https://github.com/usnistgov/trec_eval`) to calculate the Mean Average Precision (MAP) of your result lists over the 50 topics provided. `trec_eval` is a small C application, which means that you have to compile it for your own machine. In principle, you can use any C compiler (most Linux distributions already come with one), and just run the `make` command in the folder[2].

```
$trec_eval -q -m map -c qrels_file your_results_file
```

where the parameters are[3]:

-m measure: Add 'measure' to the lists of measures to calculate and print. If 'measure' contains a '.', then the name of the measure is everything preceding the period, and everything to the right of the period is assumed to be a list of parameters for the measure, separated by ','. There can be multiple occurrences of the -m flag. 'measure' can also be a nickname for a set of measures. Current nicknames include

'official' : the main measures often used by TREC

'all_trec' : all measures calculated with the standard TREC results and rel_info format files.

'set' : subset of all_trec that calculates unranked values.

'prefs' : Measures not in all_trec that calculate preference measures.

-c: Average over the complete set of queries in the relevance judgements instead of the queries in the intersection of relevance judgements and results. Missing queries will contribute a value of 0 to all evaluation measures (which may or may not be reasonable for a particular evaluation measure, but is reasonable for standard TREC measures.) Default is off.

-q: In addition to summary evaluation, give evaluation for each query/topic

and where

`qrels_file` is the file available on the resources on TUWEL.

`your_results_file` is your results file using the format indicated above, where all topics are present.

In the end, your results should be in a table like:

	TF-IDF	BM25	BM25L
401	0.0301	0.0252	0.0251
...	...	...	...
450	0.0230	0.0350	0.0241
AVG	0.0210	0.0220	0.0151

[2]Note that last year some students could not compile the latest trec_eval on Windows, but managed to compile version 8.1 (available here `http://trec.nist.gov/trec_eval/`). If that happens to you, you may use 8.1 as well.

[3]taken from the help menu of trec_eval. Type `trec_eval -h` to see the full list of options

Point allocation

The 70 percentual points are allocated as follows:

Document processing	
basic tokenizer	12
case folding	2
special strings	2
stemming	3
lemmatization	3
Index creation	
Simple posting list, Hash or B-Tree dictionary	13
Single Pass In Memory Indexing[4]	6
Map-Reduce	6
Search	
TF-IDF	3
BM25	3
Paper scoring function (BM25L)	7
Evaluation	
simple	12
calculate statistical significance for all pairings (=3 pairs) of scoring functions	2
Total	**74**

Report (15%)

Describe your prototype in a report. Describe briefly your index, in particular which additional information is required to be saved, if any. Show how the score is calculated for two documents adjacent in the ranked list. For this, you may use a query of your choosing (the simpler the query, the simpler the explanation, but you must have at least two terms in the query). Finally, show your MAP results calculated using `trec_eval`. **Try to find a justification of why some of your retrieval models performed better than the others.** The report must explain how to run the prototype. Maximum size: 4 pages or 2000 words.

Hand-in Presentation (15%)

As part of the presentation, general knowledge about the system you have used will be tested (e.g. what a tokenizer is, which stemmer have you used, how to change it, how to change the scoring function, what are the language-specific features).

- All the source code must be uploaded on a public Git repository, and the coordinators invited (markus.zlabinger@tuwien.ac.at and lipani@ifs.tuwien.ac.at);
- Prototypes are presented to the coordinator in one of the labs;
- Final deadline is **April 18th - 23:59** . For that, you must upload a zipped file to TUWEL (report, source code, and corresponding executable files incl. all dependencies);
- Your submission has to be self-contained;
- You must book a time on TUWEL for the presentation;
- You present on your own notebook to the coordinator in the lab;
- Lab locations are on TISS.

Note: You must have 35 points for this exercise in order to pass the course.

[4]Depending on your hardware, you will probably have enough memory to index everything without the creation of additional blocks. Despite of having enough memory, make sure that you create at least 2 blocks when using SPIMI.

When Documents Are Very Long, BM25 Fails!

Yuanhua Lv
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
ylv2@uiuc.edu

ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
czhai@cs.uiuc.edu

ABSTRACT

We reveal that the Okapi BM25 retrieval function tends to *overly penalize very long documents*. To address this problem, we present a simple yet effective extension of BM25, namely **BM25L**, which "shifts" the term frequency normalization formula to boost scores of very long documents. Our experiments show that BM25L, with the same computation cost, is more effective and robust than the standard BM25.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms

Keywords

BM25, BM25L, term frequency, very long documents

1. MOTIVATION

The Okapi BM25 retrieval function [2, 3] has been the state-of-the-art for nearly two decades. BM25 scores a document D with respect to query Q as follows:

$$\sum_{q \in Q \cap D} \frac{(k_3+1)c(q,Q)}{k_3+c(q,Q)} \cdot f(q,D) \cdot \log \frac{N+1}{df(q)+0.5} \quad (1)$$

where $c(q,Q)$ is the count of q in Q, N is the total number of documents, $df(q)$ is the document frequency of q, and k_3 is a parameter. Following [1], we use a modified IDF formula in BM25 to avoid its problem of possibly negative IDF values.

A key component of BM25 contributing to its success is its sub-linear term frequency (TF) normalization formula:

$$f(q,D) = \frac{(k_1+1)c(q,D)}{k_1\left(1-b+b\frac{|D|}{avdl}\right)+c(q,D)} = \frac{(k_1+1)c'(q,D)}{k_1+c'(q,D)} \quad (2)$$

where $|D|$ represents document length, $avdl$ stands for average document length, $c(q,D)$ is the raw TF of q in D, and b and k_1 are two parameters. $c'(q,D)$ is the normalized TF by document length using pivoted length normalization [4].

$$c'(q,D) = \frac{c(q,D)}{1-b+b\frac{|D|}{avdl}} \quad (3)$$

SIGIR'11, July 24–28, 2011, Beijing, China.
ACM 978-1-4503-0757-4/11/07.

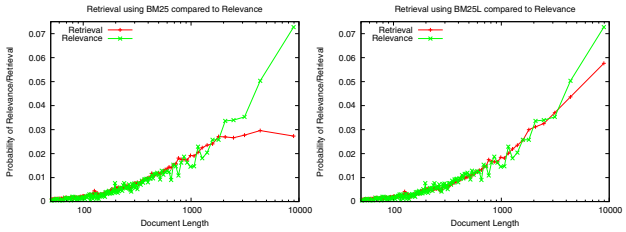

Figure 1: Comparison of retrieval and relevance probabilities against all document lengths.

When a document is very long, we can see that $c'(q,D)$ could be very small and approach 0. Consequently, $f(q,D)$ will also approach 0 as if q did not occur in D. That is, the presence of q in a very long document D fails to differentiate D clearly from other documents where q is absent. This suggests that the occurrences of a query term in very long documents may not be rewarded properly by BM25, and thus those very long documents can be overly penalized. (See Figure 1 for empirical evidence of this problem.)

2. BOOSTING VERY LONG DOCUMENTS

In order to avoid overly-penalizing very long documents, we need to add a constraint in TF normalization to make sure that the "score gap" of $f(q,D)$ between $c'(q,D)=0$ and $c'(q,D)>0$ be sufficiently large. However, we do not want that the addition of this new constraint violates those existing properties of BM25 [2], which have been shown to work quite well. Thus what we want is an improved sub-linear TF normalization formula $f'(q,D)$ that has all the following characteristics: (I) It is zero for $c'(q,D)=0$; (II) it increases monotonically as $c'(q,D)$ increases but to an asymptotic maximum; (III) it decreases monotonically as $c'(q,D)>0$ decreases but to *an asymptotic minimum that is sufficiently large.* Here (I) and (II) are characteristics of the TF normalization formula of the original BM25 [2].

One heuristic way to achieve this goal is to define $f'(q,D)$ as follows:

$$f'(q,D) = \begin{cases} \frac{(k_1+1)\cdot[c'(q,D)+\delta]}{k_1+[c'(q,D)+\delta]} & \text{if } c'(q,D)>0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

which is essentially a shifted version of $f(q,D)$ by addition of a shift parameter $\delta > 0$. It is easy to verify that $f'(q,D)$ still satisfies both properties (I) and (II). Moreover, $f'(q,D)$ has a *positive* lower bound $(k_1+1)\delta/(k_1+\delta)$ for $c'(q,D)>0$. In this sub-linear function $f'(q,D)$, the score increase from the addition of δ is decreasing as $c'(q,D)$ increases. Therefore,

Method	Terabyte		WT10G		WT2G		Robust04		TREC8		AP8889	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	0.2942	0.5703	0.2099	0.3031	0.3198	0.4620	0.2543	0.4345	0.2557	0.4580	0.2631	0.4071
BM25L	**0.2999**	0.5703	**0.2154**	0.3072	**0.3260**	0.4780	0.2553	0.4390	0.2571	0.4560	0.2650	0.4152

Table 1: Comparison of optimal performance. δ is fixed to 0.5 in BM25L. Bold font indicates that the corresponding MAP improvement is statistically significant using the Wilcoxon test ($p < 0.05$).

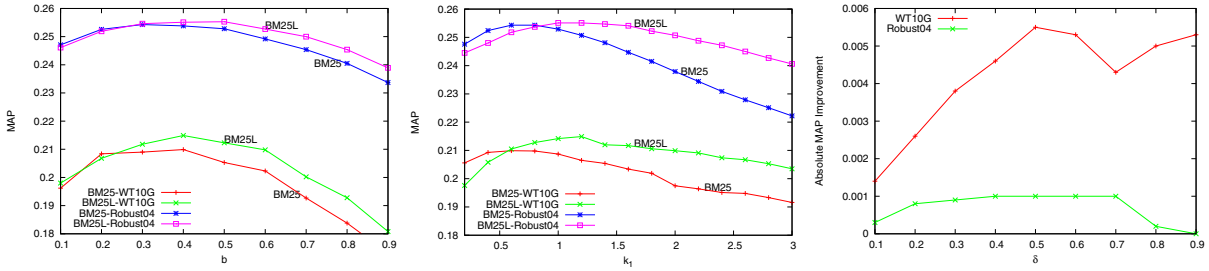

Figure 2: Performance Sensitivity to b (left), k_1 (middle), and δ (right).

$f'(q, D)$ tends to favor small $c'(q, D)$ values more, which would intuitively boost very long documents.

Finally, substituting $f'(q, D)$ into Equation 1 to replace $f(q, D)$, we obtain a new retrieval function, namely **BM25L**.

3. EXPERIMENTS

We compared BM25L with BM25 using six TREC collections: Terabyte, WT10G, and WT2G are web datasets with queries 701-850, 451-550, and 401-450 respectively; Robust04, TREC8, and AP8889 represent news datasets with queries 301-450&601-700, 401-450, and 51-150 respectively. The preprocessing included Porter stemming and removal of standard InQuery stopwords. For both BM25 and BM25L, we set $k_3 = 1000$ and tuned $b \in [0.1, 0.9]$ and $k_1 \in [0.2, 3.0]$ to optimize MAP. The parameter δ in BM25L was empirically set to **0.5** unless otherwise specified.

3.1 Retrieval Pattern VS. Relevance Pattern

Inspired by Singhal et al.'s finding that a good retrieval function should retrieve documents of all lengths with similar chances as their likelihood of relevance [4], we also compare the retrieval pattern of BM25 with the relevance pattern. We follow the binning analysis strategy proposed in [4] and plot the two patterns against all document lengths on WT10G in Figure 1 (left). It confirms our previous analysis that *BM25 retrieves very long documents with chances much lower than their likelihood of relevance.*

We also plot the retrieval and relevance patterns for BM25L in Figure 1 (right). As expected, BM25L indeed alleviates the problem of over-penalizing very long documents clearly.

3.2 Experimental Results

We first compare the optimal performance of BM25 and BM25L in Table 1. We observe that BM25L outperforms the well-tuned BM25 consistently in both MAP and P@10. Besides, BM25L works better on web collections than on news collections. This is likely because there are generally more very long documents in web collections, where the problem of BM25, i.e., overly-penalizing very long documents, would presumably be more severe.

Parameter b controls the influence of document length in TF normalization. We draw the sensitivity curves of BM25 and BM25L to b on WT10G and Robust04 in Figure 2 (left), where k_1 is optimized for each method. It shows that BM25L is more robust than BM25; if we increase b, the MAP of BM25 drops more quickly than BM25L. It intuitively makes sense in that, increasing b in BM25 would overly penalize very long documents even more. Overall, BM25L works effectively with $b \in [0.3, 0.6]$.

We also draw in Figure 2 (middle) the sensitivity curves of BM25 and BM25L through varying k_1, where b is optimized for each method. We can see that BM25 drops dramatically when increasing k_1, while BM25L appears more stable. This observation is expected: according to Formula 2, for a small $c'(q, D)$ in very long documents, the larger k_1 is, the smaller $f(q, D)$ will be; as a result, increasing k_1 would exacerbate the problem of BM25. However, the $f'(q, D)$ of BM25L is guaranteed to have a document-independent lower bound $(k_1 + 1)\delta/(k_1 + \delta)$ to avoid overly penalizing very long documents. Additionally, the optimal k_1 of BM25L is also larger than that of BM25 due to the effect of "shifting"; setting $k_1 \in [1.0, 2.0]$ usually works well for BM25L.

In previous experiments, we empirically set the shift parameter $\delta = 0.5$ in BM25L. To examine how δ affects the performance of BM25L, we plot the absolute MAP improvements against different values of δ in Figure 2 (right). It shows that the simple shifting strategy can successfully improve performance steadily and setting $\delta = 0.5$ works well for both web and news datasets. Besides, it confirms our finding that BM25L works more effectively on web data.

4. CONCLUSIONS

We proposed BM25L, a simple yet effective extension of BM25, to overcome the problem of BM25 which tends to over-penalize very long documents. Our experiments show that BM25L, with the same computation cost, is more effective and robust than BM25. BM25L can potentially replace the standard BM25 in all retrieval applications.

5. REFERENCES

[1] H. Fang, T. Tao, and C. Zhai. A formal study of information retrieval heuristics. In *SIGIR '04*, pages 49–56, 2004.

[2] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94*, pages 232–241, 1994.

[3] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC '94*, pages 109–126, 1994.

[4] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *SIGIR '96*, pages 21–29, 1996.

Exercise 1.2 - TF-IDF, BM25 & BM25Fs, and evaluation (85/100 points)

188.980 Advanced Information Retrieval 2018S

March 13, 2018

Abstract

At the end of this exercise, you should have an understanding of the practical issues of creating an inverted index for information retrieval, being able to implement basic scoring, as well as being able to comprehend and apply advance scoring functions from research papers. The exercise should be done in pairs, but each of the members has to be able to answer questions about the implementation provided. **Make sure you go through the lecture slides and relevant book chapters at least once before starting on this exercise.**

Task

In Exercise 1.2, your task is to index and search **without** the use of any library (like: Lucene, Terrier, etc.[1]), to create a scorer that implements both TF-IDF and BM25 scoring functions, and to implement an improved version of BM25 based on the article published by Arjien P. de Vries et al. 2005 at SIGIR and available online at `http://homepages.cwi.nl/~arjen/pub/f330-devries.pdf` as well as at the end of this Exercise sheet. You also have to evaluate your results by using the `trec_eval` program. Finally, you need to write a short report, upload everything to TUWEL, and present your results in person.

Warning

About a week before the hand-in of this assignment we offer the possibility to do a pre-hand-in (i.e. an optional checkpoint) in order to solve any sort of technical and non-technical problems you may encounter during the execution of the assignment, so please take advantage of it. **Excuses about technical problems at the presentation session will not be tolerated**.

Functionality (70%)

Index

Build an inverted index from the document collection CD4&5 of the TIPSTER collection that is available in TUWEL: TREC8Adhoc.tar.bz2 in TREC8all.zip.

Each file in the provided dataset contains several documents (indicated by the `<DOC>...</DOC>` tags). Single documents contain an ID (`<DOCNO>...</DOCNO>`) and text (`<TEXT>...</TEXT>`). Documents that do not have a `<TEXT>` tag can be ignored.

If you encounter encoding problems reading the dataset, make sure that you use the correct file encoding `ISO 8859-1`.

[1] However, you may use a library for stemming or lemmatizing

Before indexing, apply any combination of the techniques described in Chapter 2 of the *Introduction to Information Retrieval* book (case folding, removing stop-words, stemming, lemmatization). These options should be exposed as parameters in the index creation phase. For the stemming and lemmatization components, you can use a library at your choice.

You should provide an executable file `indexer.sh` (*.bat for windows) that given the necessary parameters generates the required index.

You may use any format for storing the index.

Search

Implement a basic search functionality and provide it as a command line interface (CLI). No GUI is required. The CLI allows the user to pass search parameters and a file that contains the search topics (i.e. `topicsTREC8Adhoc.txt`). The system then returns a list of documents ranked by a similarity function of your choice (in Exercise 1.2 either: TF-IDF, BM25 or BM25Fs). The scoring functions and their individual parameters need to be exposed as parameters to the CLI.

Details regarding the BM25 modification: Your task is to modify your implementation of the BM25 scoring function and create a new similarity class called BM25Fs, which introduces the four different ways to calculate the IDF of a term (F1, F2, F3, and F4). For your experiments, you may fix the method to F1.

You are provided with a set of 50 topics to search for. The topic file is contained in the dataset and has a simple SGML structure. Each topic is delimited by `<TOP>` tags and you are free to use all or part of their content in your search.

As a recommendation, try to implement topic processing and the actual scoring/ranking as two separate tasks. By topic processing, we understand here simply getting the terms that will be searched in the index. Consider the use or not use of the different elements in the topic document.

Your search should be (moderately) efficient. It is, for example, not OK to perform your collection indexing only at search time, and also loading the inverted index should be reasonably in time, or at least not happen before each search.

The output of the CLI given the previously described parameters should be a ranked list of up to 1000 documents, where each line is in the following format:

$$\{\textit{topic-id}\}\ \textit{Q0}\ \{\textit{document-id}\}\ \{\textit{rank}\}\ \{\textit{score}\}\ \{\textit{run-name}\}$$

where

topic-id is an integer number between 401 and 450;

document-id is an identifier for the document (e.g. LA122690-0033);

Q0 this field is unnecessary to our purposes but must be present;

rank is an integer indicating the rank of the object in the sorted list (normally, this should be ascending from the first line to the last line)

score the similarity score calculated by (it has to not exceed the float precision and normally, this should be descending from the first line to the last line)

run-name a name you give to your experiment (free for you to choose)

Here is a short example of a potential output for all the topics:

```
401 Q0 FBIS3-10899 1 2.813525 grp5-exp1
401 Q0 FBIS3-13322 2 1.0114759 grp5-exp1
...
450 Q0 LA043089-0083 1000 0.08848727 grp5-exp1
```

You should provide an executable `searcher.sh` (*.bat for windows) file that given the necessary parameters (i.e. `[scoring function] [scoring function's paramters] [topic file]`) generates a ranked list as above.

Evaluation

You must use `trec_eval` (available at: `https://github.com/usnistgov/trec_eval`) to calculate the Mean Average Precision (MAP) of your result lists over the 50 topics provided. `trec_eval` is a small C application, which means that you have to compile it for your own machine. In principle, you can use any C compiler (most Linux distributions already come with one), and just run the `make` command in the folder[2].

```
$trec_eval -q -m map -c qrels_file your_results_file
```

where the parameters are[3]:

-m measure: Add 'measure' to the lists of measures to calculate and print. If 'measure' contains a '.', then the name of the measure is everything preceding the period, and everything to the right of the period is assumed to be a list of parameters for the measure, separated by ','. There can be multiple occurrences of the -m flag. 'measure' can also be a nickname for a set of measures. Current nicknames include

'official' : the main measures often used by TREC

'all_trec' : all measures calculated with the standard TREC results and rel_info format files.

'set' : subset of all_trec that calculates unranked values.

'prefs' : Measures not in all_trec that calculate preference measures.

-c: Average over the complete set of queries in the relevance judgements instead of the queries in the intersection of relevance judgements and results. Missing queries will contribute a value of 0 to all evaluation measures (which may or may not be reasonable for a particular evaluation measure, but is reasonable for standard TREC measures.) Default is off.

-q: In addition to summary evaluation, give evaluation for each query/topic

and where

`qrels_file` is the file available on the resources on TUWEL.

`your_results_file` is your results file using the format indicated above, where all topics are present.

In the end, your results should be in a table like:

	TF-IDF	BM25	BM25Fs
401	0.0301	0.0252	0.0251
...	...	...	...
450	0.0230	0.0350	0.0241
AVG	0.0210	0.0220	0.0151

[2]Note that last year some students could not compile the latest trec_eval on Windows, but managed to compile version 8.1 (available here `http://trec.nist.gov/trec_eval/`). If that happens to you, you may use 8.1 as well.

[3]taken from the help menu of trec_eval. Type `trec_eval -h` to see the full list of options

Point allocation

The 70 percentual points are allocated as follows:

Document processing	
basic tokenizer	12
case folding	2
special strings	2
stemming	3
lemmatization	3
Index creation	
Simple posting list, Hash or B-Tree dictionary	13
Single Pass In Memory Indexing[4]	6
Map-Reduce	6
Search	
TF-IDF	3
BM25	3
Paper scoring function (BM25Fs)	7
Evaluation	
simple	12
calculate statistical significance for all pairings (=3 pairs) of scoring functions	2
Total	**74**

Report (15%)

Describe your prototype in a report. Describe briefly your index, in particular which additional information is required to be saved, if any. Show how the score is calculated for two documents adjacent in the ranked list. For this, you may use a query of your choosing (the simpler the query, the simpler the explanation, but you must have at least two terms in the query). Finally, show your MAP results calculated using `trec_eval`. **Try to find a justification of why some of your retrieval models performed better than the others.** The report must explain how to run the prototype. Maximum size: 4 pages or 2000 words.

Hand-in Presentation (15%)

As part of the presentation, general knowledge about the system you have used will be tested (e.g. what a tokenizer is, which stemmer have you used, how to change it, how to change the scoring function, what are the language-specific features).

- All the source code must be uploaded on a public Git repository, and the coordinators invited (markus.zlabinger@tuwien.ac.at and lipani@ifs.tuwien.ac.at);
- Prototypes are presented to the coordinator in one of the labs;
- Final deadline is **April 18th - 23:59** . For that, you must upload a zipped file to TUWEL (report, source code, and corresponding executable files incl. all dependencies);
- Your submission has to be self-contained;
- You must book a time on TUWEL for the presentation;
- You present on your own notebook to the coordinator in the lab;
- Lab locations are on TISS.

Note: You must have 35 points for this exercise in order to pass the course.

[4]Depending on your hardware, you will probably have enough memory to index everything without the creation of additional blocks. Despite of having enough memory, make sure that you create at least 2 blocks when using SPIMI.

Relevance Information: A Loss of Entropy but a Gain for IDF?

Arjen P. de Vries
CWI
PO Box 94079
1090 GB Amsterdam
The Netherlands
arjen@acm.org

Thomas Roelleke
Queen Mary University of London
Mile End Road
London, E1 4NS
United Kingdom
thor@dcs.qmul.ac.uk

ABSTRACT

When investigating alternative estimates for term discriminativeness, we discovered that relevance information and *idf* are much closer related than formulated in classical literature. Therefore, we revisited the justification of *idf* as it follows from the binary independent retrieval (BIR) model. The main result is a formal framework uncovering the close relationship of a generalised *idf* and the BIR model. The framework makes explicit how to incorporate relevance information into any retrieval function that involves an *idf*-component.

In addition to the *idf*-based formulation of the BIR model, we propose Poisson-based estimates as an alternative to the classical estimates, this being motivated by the superiority of Poisson-based estimates for the within-document term frequencies. The main experimental finding is that a Poisson-based *idf* is superior to the classical *idf*, where the superiority is particularly evident for long queries.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

General Terms: Theory, Experimentation

Keywords: Information retrieval, relevance feedback, probability of relevance, binary independent retrieval model, inverse document frequency

1. INTRODUCTION

Virtually all information retrieval systems estimate the relevance of documents to a query by comparing term statistics for the document at hand to those of the collection and, if available, a set of known relevant and/or non-relevant documents. Combining within-document term frequency (referred to as *tf*) with the inverse document frequency (referred to as *idf*) has consistently been proven successful on a variety of retrieval benchmarks.

SIGIR'05, August 15–19, 2005, Salvador, Brazil.

The intuition underlying this combination of *tf* and *idf* is that frequent occurrence of query terms in a document makes the document more likely to be relevant to the query, but only if these terms are also discriminative. A term is discriminative if it does not occur in many different documents, or, in other words, when its inverse document frequency is high. Many publications point out how a $tf \cdot idf$ measure results naturally from either purely probabilistic or information theoretic arguments, and although they differ in the details, in the end, all of these authors conclude that the theoretical arguments presented fit nicely with the well-known experimental success.

Our investigation considers the role of relevance information in retrieval systems based on $tf \cdot idf$ ranking functions. Exploiting relevance information to revise the importance weights of given query terms and/or to expand the query with new terms usually improves the retrieval quality significantly, i.e., leads to a better ranking of documents than the one obtained by the initial provided by the user.

The main goal of this paper is to investigate how the estimate of term discriminativeness, represented by the term's *idf*, should be updated after relevance information becomes available. From an information theoretic argument, adding relevance information to a retrieval system corresponds to a loss of entropy. Without relevance information, the corpus of all documents is the foundation for estimating the discriminativeness (informativeness) of each term. With relevance information, we should remove the documents for which relevance information is available (let it be positive or negative relevance feedback) from the corpus of all documents.

We view *idf* primarily as a *summary statistic* of a set of documents. Church and Gale have demonstrated that *idf* is an attractive summary statistic of term sets, because it is more stable across collections of different years and different sources than alternatives such as, e.g., the variance of term occurrence in documents [2].

The classical binary independent retrieval (BIR) model [8] explains *idf* as the ranking that results from the situation where we have no relevance information. We have found that considering *idf* as a statistic of the sets of relevant and non-relevant documents provides an even stronger connection to the BIR model. We observed that the relevance-based term weights in the weighting schemes known as F1–F4 can be expressed conveniently as summations of the *idf* values of the different sets involved.

An open question remains how the discriminativeness of

a term should be measured. The classical *idf* is based on the occurrence probability $P(t|c) = n_D(t,c)/N_D(c)$, where $n_D(t,c)$ is the number of documents in collection c in which term t occurs and N_D is the number of documents in the collection. For the within-document term frequency, it is known that alternative estimates, such as those based on 'lifting' [13] and those based on Poisson approximations [6, 7], lead to improved retrieval results. The question that follows directly is whether the arguments motivating these alternative estimations apply equally to the estimate used for term discriminativeness.

The paper is structured as follows. Section 2 gives a concise overview of related work. Section 3 presents the main results of our research, where we relate the *idf* summary statistic over sets of relevant and non-relevant documents to the term weighting schemes of the Binary Independent Retrieval (BIR) model. Section 4 discusses the impact of our results for retrieval systems using *tf*$\cdot$ *idf* ranking, and Section 5 presents an experimental evaluation. We summarise the contributions of this work in the final Section 6.

2. BACKGROUND

Robertson recently surveyed research attempting to find plausible theoretical models explaining why *tf*$\cdot$ *idf* is a good approach to ranking [9]. The same work also presents the BM25 ranking function as another instantiation of *tf*$\cdot$ *idf* for ranked retrieval. Although we did not set out to provide a theoretical explanation for *tf*$\cdot$ *idf* approaches, the results obtained are related to these works, especially where we contribute into revealing a closer relationship of *idf* measures to the Binary Independent Retrieval (BIR) model [8].

From the start, our primary goal has been to investigate the role of the *idf* summary statistic in the incorporation of relevance information. Ruthven and Lalmas' review of approaches to relevance feedback [12] pointed us into the direction of updating *idf* according to the classic probabilistic model. Like [2], we approach *idf* as a robust statistic of term occurrence that helps identify 'good' keywords. Other related work in information retrieval theory includes the discussion of disjoint term spaces in [15], and in particular their proposal to represent discriminativeness probabilistically.

Our research can also be viewed as an extension of [10], where inverse document frequency has been related to the probability of being informative. Especially the results obtained on the TREC collections presented in Section 5 can be considered an experimental investigation into the applicability of the idea proposed there to define the probability of a term being informative based on the assumption of documents as independent rather than disjoint events, which motivates the use of Poisson probabilities.

3. IDF AND THE PROBABILITY OF RELEVANCE

This section discusses the relationship between the IDF and the probability of relevance. We review briefly the *idf* definition (section 3.1), and the probability of relevance (section 3.2), and the BIR model (section 3.3). Then, section 3.4 presents the *idf*-based formulation of the BIR model. Section 3.5 gives an intuitive explanation, based on the idea of 'virtual documents', for the parameter smoothing applied for dealing with singularities of the BIR model. Finally, we propose and investigate in section 3.6 Poisson-based estimates for probabilities such as $P(t|r)$ and $P(t|c)$ (probability that t occurs in relevant documents and probability that t occurs in the documents of the collection).

3.1 IDF

The *idf* is defined as the logarithm of the probability that a term occurs in the collection. Let c be the collection, let t be a term, let $N_D(c)$ be the number of documents in c, and let $n_D(t,c)$ be the number of documents in c in which t occurs. Then, the *idf* is defined as follows:

$$\begin{aligned} P(t|c) &:= \frac{n_D(t,c)}{N_D(c)} \\ idf(t,c) &:= -\log P(t|c) \end{aligned}$$

It reflects the discriminativeness of term t in collection c.

3.2 Probability of Relevance

The probability of relevance is denoted as $P(r|d,q)$, where r is the relevance event, d is the document event, and q is the query event.[1]

Bayes' theorem gives:

$$P(r|d,q) = \frac{P(r,d,q)}{P(d,q)}$$

Next, we split the conjunction r,d,q. Once we let d depend on q, and once we let q depend on d.

$$\begin{aligned} P(r|d,q) &= \frac{P(d|q,r)\cdot P(r|q)\cdot P(q)}{P(d,q)} \quad (1) \\ &= \frac{P(q|d,r)\cdot P(r|d)\cdot P(d)}{P(d,q)} \quad (2) \end{aligned}$$

Equation 1 is the foundation of the binary independent retrieval (BIR) model and equation 2 is the foundation of the language modelling (LM) approaches.

Using the odds $O(r|d,q) = P(r|d,q)/P(\bar{r}|d,q)$ instead of $P(r|d,q)$ leads to the same ranking, but $P(d,q)$, $P(d)$ and $P(q)$ drop out.

$$\begin{aligned} O(r|d,q) = \frac{P(r|d,q)}{P(\bar{r}|d,q)} &= \frac{P(d|q,r)\cdot P(r|q)}{P(d|q,\bar{r})\cdot P(\bar{r}|q)} \\ &= \frac{P(q|d,r)\cdot P(r|d)}{P(q|d,\bar{r})\cdot P(\bar{r}|d)} \end{aligned}$$

This elegant formulation of the BIR model, the LM approaches, and their differences, can be found in [5].

Now, to instantiate the model, documents and queries are considered as conjunctions of features, denoted with x_t. If we assume the features to be independent, we obtain:

$$\begin{aligned} P(d|q,r) &= \prod_t P(x_t|q,r) \\ P(q|d,r) &= \prod_t P(x_t|d,r) \end{aligned}$$

Usually, the words (terms) occurring in documents and queries are considered as features.

3.3 Binary Independent Retrieval (BIR) Model

The BIR model estimates the probability $P(d|q,r)$ based on the presence and absence of terms. The probability

[1]Note that $P(R|d,q)$ is also commonly used in literature. We prefer however a consistent use of lower case to denote events (as opposed to random variables).

$P(X_t = 1|q, r)$ is the probability that term t occurs in relevant documents. Analogously, $P(X_t = 0|q, r)$ is the probability of its absence.

As usual in formulations of the BIR model, we define abbreviations for those probabilities. We set $a_t := P(X_t = 1|q, r)$ and $b_t := P(X_t = 1|q, \bar{r})$. (We deviate from the more common choice of p_t and q_t because this frequently causes confusion with probabilities and queries).

Since X_t is a binary random variable, we can rewrite the probability for an event x_t as follows [14]:

$$\begin{aligned} P(x_t|q, r) &= a_t^{x_t} \cdot (1 - a_t)^{1-x_t} \\ P(x_t|q, \bar{r}) &= b_t^{x_t} \cdot (1 - b_t)^{1-x_t} \end{aligned}$$

We obtain:

$$\begin{aligned} O(r|d, q) &= \frac{P(d|q, r)}{P(d|q, \bar{r})} \cdot O(r|q) \\ &= \prod_t \left(\frac{P(x_t|q, r)}{P(x_t|q, \bar{r})} \right) \cdot O(r|q) \\ &= \prod_t \left(\frac{a_t^{x_t} \cdot (1 - a_t)^{1-x_t}}{b_t^{x_t} \cdot (1 - b_t)^{1-x_t}} \right) \cdot O(r|q) \end{aligned}$$

The log transformation yields:

$$\begin{aligned} &\log O(r|d, q) = \\ &= \sum_t \log \frac{a_t^{x_t} \cdot (1 - a_t)^{1-x_t}}{b_t^{x_t} \cdot (1 - b_t)^{1-x_t}} + \log O(r|q) \\ &= \sum_t \log \frac{a_t^{x_t} \cdot (1 - b_t)^{x_t} \cdot (1 - a_t)}{b_t^{x_t} \cdot (1 - a_t)^{x_t} \cdot (1 - b_t)} + \log O(r|q) \\ &= \sum_t \left(x_t \cdot \log \frac{a_t \cdot (1 - b_t)}{b_t \cdot (1 - a_t)} \right) + \\ &\qquad \sum_t \log \frac{1 - a_t}{1 - b_t} + \log O(r|q) \\ &= \sum_t (x_t \cdot h_t) + Q \end{aligned}$$

Q is a constant depending only on the query. Since it has no effect on the ranking of the documents, we do not consider it further.

Term weight h_t, the *term discriminativeness* of t, equals:

$$h_t := \log \frac{a_t \cdot (1 - b_t)}{b_t \cdot (1 - a_t)}$$

We write $h(t, c)$ to refer to the discriminativeness of term t in collection c.

Estimation of probabilities a_t and b_t uses the proportions of relevant and non-relevant documents in the collection. First, we introduce the classical and our alternative notation (also used in [11]). As the reader will notice, the classical formulation makes use of r, which also denotes the relevance event introduced before. We apologise for this overloading, but apply the classical notation where it improves the readability.

r_t	$n_D(t, r)$	number of relevant documents with t
R	$N_D(r)$	number of relevant documents
n_t	$n_D(t, c)$	number of documents with t
N	$N_D(c)$	number of documents

Review that, in our notation (given in the second column), r and c are sets of documents: r is the set of relevant documents and c is the set of all documents (the collection). This notation is better suited to clarify dualities between estimates involving all documents, the relevant documents, or the non-relevant documents, and the document or (within-document) term frequencies.

Next, consider the four alternatives for estimation, classically denoted by F1 to F4 [8].

F1: $h_t := \log \frac{r_t/R}{n_t/N}$: term occurrence is independent in *all* documents; consider occurrence only, not absence

F2: $h_t := \log \frac{r_t/R}{(n_t - r_t)/(N - R)}$: term occurrence is independent in *the non-relevant* documents; consider occurrence only, not absence

F3: $h_t := \log \frac{r_t/R \cdot (1 - n_t/N)}{n_t/N \cdot (1 - r_t/R)}$: term occurrence is independent in *all* documents; consider occurrence and absence

F4: $h_t := \log \frac{r_t/R \cdot (1 - (n_t - r_t)/(N - R))}{(n_t - r_t)/(N - R) \cdot (1 - r_t/R)} = \log \frac{r_t \cdot ((N - R) - (n_t - r_t))}{(n_t - r_t) \cdot (R - r_t)}$: term occurrence is independent in *the non-relevant* documents; consider occurrence and absence

3.4 IDF and the BIR Model

From the formulation in the previous section, the term weights h_t can be viewed as sums of *idf*-values.

Reconsider the definition of *idf*:

$$idf(t, c) := -\log \frac{n_D(t, c)}{N_D(c)} = -\log \frac{n_t}{N}$$

Using the set relevant documents instead of the full collection, we obtain:

$$idf(t, r) := -\log \frac{n_D(t, r)}{N_D(r)} = -\log \frac{r_t}{R}$$

This dual statistic reflects the discriminativeness of term t in the set of relevant documents r.

Consider the sum of *idf*-values we obtain for each of the four settings of h_t:

F1: $h_t = -idf(t, r) + idf(t, c)$: Here, the BIR model can be explained as follows: Given relevance information, the *idf* of the terms based on the relevant documents is subtracted from the standard *idf*. That means: If an overall rare term occurs rarely among the relevant documents, then $idf(t, r)$ is large, thus, h_t is significantly smaller than $idf(t, c)$. It is smaller, since, although the term is discriminative overall, for this particular query, the term is not leading to relevant documents. If an overall rare term is frequent among the relevant documents, then $idf(t, r)$ is small, and we obtain $h_t \approx idf(t, c)$. We obtain $h_t = idf(t, c)$ if t occurs in all relevant documents. Without relevance information, each term occurs in all relevant documents, because the set of relevant documents is empty.

F2: $h_t = -idf(t, r) + idf(t, \bar{r})$: Here, $\bar{r}$ denotes the set of non-relevant documents, $\bar{r} = c \setminus r$ (also known as the 'complement method'). Positive and negative relevance feedback is exploited in this second possible setting of h_t. The discriminativeness of a term is now based on the set of non-relevant documents, rather than on the set of all relevant as it was the case for the first setting.

F3: $h_t = -idf(t,r) - idf(\bar{t},c) + idf(\bar{t},r) + idf(t,c)$: The third setting shows all four combinations of term occurrence or absence, and relevant or all documents.

$-idf(t,r)$ is the discriminativeness of t in relevant documents. Reduces h_t only slightly for terms frequent in r, but strongly for non-frequent terms.

$-idf(\bar{t},c)$ reduces h_t only slightly if the event "t is absent in the collection" is frequent. Then, t occurs rarely, i.e., $idf(t,c)$ is large.

$idf(\bar{t},r)$ is the discriminativeness of the absence of t in r. If the event "t is absent in relevant documents" is rare, then t occurs frequently in r, and this leads to a high h_t. Thus, $idf(\bar{t},r)$ supports the message of $-idf(t,r)$; i.e., "rare absence of terms in relevant documents is good" means "frequent occurrence in relevant documents is good".

F4: $h_t = -idf(t,r) - idf(\bar{t},\bar{r}) + idf(t,\bar{r}) + idf(\bar{t},r)$: The fourth setting shows all possible four combinations of term occurrence or absence, and relevant or non-relevant documents.

$-idf(t,r)$ and $idf(\bar{t},\bar{r})$ have been discussed for the previous cases.

$-idf(\bar{t},\bar{r})$ supports $-idf(t,\bar{r})$, i.e., if the event "t is absent in non-relevant document" is frequent (means, t hardly occurs in non-relevant documents), $-idf(\bar{t},\bar{r})$ diminishes, i.e., it leads to little reduction for h_t.

$idf(t,\bar{r})$ is the discriminativeness of t in non-relevant documents. Without relevance information, $idf(t,\bar{r}) = idf(t,c)$.

This representation of the term weights of the BIR model based on a sum of *idf*-values gives a clear justification of the *idf* through the BIR model. It generalises the results and claim in [9], page 512: "It is now apparent that we can regard IDF as a simple version of the RSJ weight applicable when we have no relevance information."

When no relevance information is available (maximum entropy), the classical *idf* corresponds to the measure of discriminativeness. The more relevance information becomes available, the more the term's discriminativeness (in the *remainder* of the collection) will decrease (a loss of entropy). It decreases because the corpus of documents for which no relevance information is available is smaller, and, more importantly, because the discriminativeness of the query term in the relevant documents is *subtracted* from the discriminativeness obtained for all (or all non-relevant) documents.

3.5 Dealing with the Singularities

The estimates for the discriminativeness weights h_t derived above have to be adapted for dealing with singularities. The following cases require adaptation of the estimates:

- $R = N_D(r) = 0$: No relevance information given.
- $r_t = n_D(t,r) = 0$: Term t does not occur in any relevant document. Then, $\log P(t|r)$ is not defined.
- $N = N_D(c) = 0$: Empty collection.
- $n_t = n_D(t,c) = 0$: Term t does not occur in any document.
- $R = r_t = N_D(r) = n_D(t,r)$: Term t occurs in all relevant documents. Singularity for F3 and F4.
- $N = n_t = N_D(c) = n_D(t,c)$: Term t occurs in all documents. Singularity for F3 and F4.
- $n_t = r_t = n_D(t,c) = n_D(t,r)$: All documents in which term t occurs are relevant. Singularity for F4.
- $N - R = n_r - r_t = N_D(c) - N_D(r) = n_D(t,c) - n_D(t,r)$: Term t occurs in all non-relevant documents. Singularity for F4.

A common reformulation (smoothing) of the estimates is to add constants to the expressions in numerator and denominator.

The classic smoothing proposed for F4 (see [9]) adds a constant value 0.5 to each document count:

$$h_t := \log \frac{(r_t + 0.5) \cdot ((N-R) - (n_t - r_t) + 0.5)}{(R - r_t + 0.5) \cdot (n_t - r_t + 0.5)}$$

Although the literature has given various mathematical motivations for this constant, we like to offer a particularly intuitive explanation. Consider that we add two virtual documents to each collection, one of which is relevant. Assuming that each term occurs in half of the virtual and relevant documents gives the following probability estimates:

$$P(t|c) := \frac{n_D(t,c) + 1}{N_D(c) + 2} \quad P(t|r) := \frac{n_D(t,r) + 0.5}{N_D(r) + 1}$$

Based on these adapted estimates, we obtain the smoothed F4:

$$\begin{aligned} h_t = & \log \frac{(r_t + 0.5)/(R+1) \cdot (1 - (n_t - r_t + 0.5)/(N-R+1))}{(1 - (r_t+0.5)/(R+1)) \cdot (n_t - r_t + 0.5)/(N-R+1)} \\ = & \log \frac{(r_t + 0.5) \cdot ((N-R) - (n_t - r_t) + 0.5)}{(R - r_t + 0.5) \cdot (n_t - r_t + 0.5)} \end{aligned}$$

So, extending the document sets with two virtual documents is sufficient to explain the classical parameter adaptation. Next, consider four virtual documents; two of which are relevant, all terms occur in half of the documents, and all terms occur in half of the relevant documents.

$$P(t|c) := \frac{n_D(t,c) + 2}{N_D(c) + 4} \quad P(t|r) := \frac{n_D(t,r) + 1}{N_D(r) + 2}$$

This solution feels more comfortable, because all numbers are integers; they can be interpreted as document counts. We obtain:

$$\begin{aligned} h_t = & \log \frac{(r_t + 1)/(R+2) \cdot (1 - (n_t - r_t + 1)/(N - R + 2))}{(1 - (r_t + 1)/(R+2)) \cdot (n_t - r_t + 1)/(N - R + 2)} \\ = & \log \frac{(r_t + 1) \cdot ((N-R) - (n_t - r_t) + 1)}{(R - r_t + 1) \cdot (n_t - r_t + 1)} \end{aligned}$$

As a general explanation, we derive:

$$P(t|c) := \frac{n_D(t,c) + 2\epsilon}{N_D(c) + 4\epsilon} \quad P(t|r) := \frac{n_D(t,r) + 1\epsilon}{N_D(r) + 2\epsilon}$$

The number of virtual documents that is added to the collection reflects the uncertainty about relevance. We assume that each term t occurs in half of the virtual documents (since occurrence is a binary event), that half of the virtual documents are relevant (since relevance is a binary event), and that each term t occurs in half of the relevant documents. As demonstrated, these assumptions explain the classical smoothing proposed for F4 by defining $\epsilon = 0.5$.

3.6 Poisson-based Probability Estimation

Probability estimation for $P(t|c)$ and $P(t|r)$ and related probabilities have so far been estimated on what we refer to as n/N estimates and their adaptations. Consider a set of N trials, and let n_t be the number of trials in which the event t is true. Then, $P(t) = n_t/N$ is the probability that the event is true among N trials. We refer to this estimate as the disjointness-based estimate, since it can be explained by the theorem of total probability as $P(t) := \sum_d P(t|d) \cdot P(d)$, where the events d are disjoint and exhaustive.

The appropriateness of these estimates can however be questioned, in particular in the light of the summation of *idf*-values as follows from the BIR model. For, the sets involved (set of relevant documents, set of non-relevant documents, set of all documents) are different in cardinality: usually, only a small fraction of all documents forms the set of relevant documents.

Compare this to estimates for $P(t|d)$, i.e., the probability that a term describes (occurs in) a document. While the disjointness-based probability corresponds to the probability that t occurs, the Poisson-based probability of term occurrence is the probability that t occurs at least once in n_t trials. The usefulness of a Poisson-based probability is well-known for the within-document term frequency (referred to as *tf* or *lf*) of a term t in a document d [6, 7].

Let $n_L(t, d)$ be the number of locations at which t occurs in d. Then, the Poisson-based estimate of term (location) frequency is defined as follows:

$$P(t|d) := tf(t, d) = lf(t, d) = \frac{n_L(t, d)}{K_d + n_L(t, d)}$$

Here, $P(t|d)$ is the probability that term t is representative for document d. The steep rise of the within-document term frequency for small occurrences ($n_L(t, d)$ relatively small) leads to a superior retrieval quality. For example, let K be the average occurrence frequency ($K := 1/N_T \cdot \sum_t n_t$, where N_T is the number of events/terms). We obtain $P(t) = 0.5$ if n_t is the average occurrence, $P(t) < 0.5$ for n_t less than the average occurrence, and $P(t) > 0.5$ for n_t greater than the average occurrence.

This effect of a steep rise is also reflected in the well-known lifting function with lifting (zooming) factor z_r ([13] and related publications):

$$tf(t, d) = z_d + (1 - z_d) \cdot \frac{n_L(t, d)}{N_L(d)}$$

For $z_d := 0.5$, the lifting yields $tf(t, d) \approx 0.5$, if $n_L(t, d) = 1$. Compare this to the Poisson approximation, where for $K_d = 1$, we obtain $tf(t, d) = 0.5$, if $n_L(t, d) = 1$.

Looking at the sum of *idf*-values as presented in section 3.3, we deal with sets of significantly different cardinality. The set of relevant documents is much smaller than the set of non-relevant and the set of all documents. This different cardinality and the success of the lifting of the probability $P(t|d)$ are the motivations for investigating Poisson-based estimates for $P(t|c)$ and $P(t|r)$.

Review the Poisson-based probability estimation for $P(t|c)$, and $P(t|r)$:

$$P(t|c) := df(t, c) := \frac{n_D(t, c)}{K_c + n_D(t, c)}$$

$$P(t|r) := df(t, r) := \frac{n_D(t, r)}{K_r + n_D(t, r)}$$

Compare these also to the formulation of $P(t|d)$ given before. The notation shows the strong analogy of the probabilities involved.

Section 5 reports the experimental results obtained for Poisson-based estimates. Before we started the experimental investigation, we looked at an analytical experiment to investigate the nature of the Poisson-based estimate. Consider the analytical experiment in Table 1. Let a collection with $N_D(\bar{r}) = 10^6$, one million documents, be given, and let a query with $N_D(r) = 10$, i.e., ten relevant documents, be given. For ease of reading, we use logarithm base 10 for the numerical illustration (the base of the logarithm does not matter for ranking purpose, as the base is a constant, term-independent factor).

The table shows some numerical values for the discriminativeness $h_t = idf(t, \bar{r}) - idf(t, r)$, earlier also referred to as the F2 estimate. The table shows six cases of possible term distributions, and the cases (a), (d), (e) and (f) (bold face) illustrate the difference between the $n(t, x)/N$ and the $n(t, x)/(K_x + n(t, x))$ estimates. Here, x represents a set of documents: the set r of relevant documents, or the set $\bar{r}$ of non-relevant documents. The main findings from this analytical investigation are:

1. The classical estimate n_t/N leaves h_t to be high for terms rare in the collection; the effect of the occurrence of the term in the relevant document is minor (see (a), (b), and (c)). h_t is mainly determined by $idf(t, \bar{r})$.

2. The Poisson-based estimate leads to a dramatic impact on h_t for terms that occur rarely in the relevant documents (see (a) and (d)). For a rare term, the sum of idf_P-values is zero (case (a))! For a more frequent term, $idf_P(t, \bar{r})$ remains the main impact of t on the RSV (cases (b) and (c)).

3. The classical estimate n_t/N leaves $idf(t, \bar{r})$ to be the dominant factor even for terms that are relatively frequent in $\bar{r}$, and the frequency of t in r has hardly an effect (see (d), (e), and (f)).

4. The Poisson-based estimate leads to small $idf_P(t, \bar{r})$ values for terms that are relatively frequent. The impact of $idf_P(t, r)$ is strong (see (d)) for terms rare among the relevant. For a term relatively frequent in relevant and non-relevant documents, the effect on the RSV is minimal (see (e) and (f)). As the analytical investigation shows, this cancelling out of terms is already happening for terms that are little to medium frequent, if the parameter K of the Poisson-approximations is small (for the analytical experiment, $K_{\bar{r}} = K_r = 1$).

This analytical experiment illustrates the different nature of the classical n_t/N (idf) estimate and the Poisson-based estimate $n/(K+n)$ (idf_P). As our TREC experiments show, the setting of K has a major impact on the retrieval quality. Before we report on the experiments, we first summarise the impact of our results on $tf \cdot idf$-based retrieval functions.

4. IMPACT ON TF-IDF-BASED RANKING

One result of our investigation is that any retrieval system based on $tf \cdot idf$ ranking may incorporate relevance information by replacing its *idf*-component by h_t, where h_t is a

Table 1: Analytical Experiment: Comparison of h_t for $n(t,x)/N$ and $n(t,x)/(K_x + n(t,x))$

			$n(t,x)/N$			$n(t,x)/(1+n(t,x))$		
	$n_D(t,r)$	$n_D(t,\bar{r})$	$idf(t,r)$	$idf(t,\bar{r})$	$h_t = -idf(t,r) + idf(t,\bar{r})$	$idf_P(t,r)$	$idf_P(t,\bar{r})$	$h_t = -idf_P(t,r) + idf_P(t,\bar{r})$
a	1	1	$-\log\frac{1}{10}$	$-\log\frac{1}{10^6}$	$-1+6=5 \approx idf(t,\bar{r})$	$-\log\frac{1}{2}$	$-\log\frac{1}{2}$	0
b	5	1	$-\log\frac{5}{10}$	$-\log\frac{1}{10^6}$	$> 5 \approx idf(t,\bar{r})$	$-\log\frac{5}{6}$	$-\log\frac{1}{2}$	$\approx idf_P(t,\bar{r})$
c	10	1	$-\log\frac{10}{10}$	$-\log\frac{1}{10^6}$	$6 = idf(t,\bar{r})$	$-\log\frac{10}{11}$	$-\log\frac{1}{2}$	$\approx idf_P(t,\bar{r})$
d	1	100	$-\log\frac{1}{10}$	$-\log\frac{100}{10^6}$	$-1+4=3 \approx idf(t,\bar{r})$	$-\log\frac{1}{2}$	$-\log\frac{100}{101}$	$\approx -idf_P(t,r)$
e	5	100	$-\log\frac{5}{10}$	$-\log\frac{100}{10^6}$	$> 3 \approx idf(t,\bar{r})$	$-\log\frac{5}{6}$	$-\log\frac{100}{101}$	≈ 0
f	10	100	$-\log\frac{10}{10}$	$-\log\frac{100}{10^6}$	$4 = idf(t,\bar{r})$	$-\log\frac{10}{11}$	$-\log\frac{100}{101}$	≈ 0

sum of *idf*-values derived from the sets of relevant and non-relevant documents as presented in section 3.4. This result coincides with the BM25 ranking formula, and extends the relationship of *idf* and BIR model as presented in [9].

Consider the definition of a $tf \cdot idf$-based retrieval function, and we illustrate the replacement of *idf* by the discriminativeness measure h_t:

$$\begin{aligned} RSV_{tf-idf} &:= \sum_t idf(t,c) \cdot tf(t,d) \cdot tf(t,q) \\ &= \sum_t h(t,c) \cdot tf(t,d) \cdot tf(t,q) \end{aligned}$$

The probabilities based on the within-document and within-query term frequencies are estimated as follows:

$$P(t|d) := tf(t,d) := \frac{n_L(t,d)}{K_d + n_L(t,d)}$$

$$P(t|q) := tf(t,q) := \frac{n_L(t,q)}{K_q + n_L(t,q)}$$

The factors K_d and K_q serve for normalisation purposes. The BM25 (as the currently most successful $tf \cdot idf$-based retrieval function) proposes $h(t,c)$ in place of *idf*, and adds various constants to the retrieval function. The motivation for the constants is to leverage the effect of $h(t,c)$, $tf(t,d)$, and $tf(t,q)$, and to fit the retrieval function to the nature of the collection for which it is applied.

To illustrate, we substitute $h(t,c)$ with the *idf*-based formulation of F1:

$$RSV_{tf-idf} = \sum_t [-idf(t,r) + idf(t,c)] \cdot tf(t,d) \cdot tf(t,q)$$

The factor $idf(t,c)$ is the classical *idf*. The factor $-idf(t,r)$ represents the relevance information. This factor is negative, since $h(t,c)$ is smaller than $idf(t,c)$ for terms that occur rarely in relevant documents. The example shows that for $h(t,c)$ based on F1, $RSV_{tf \cdot idf}$ corresponds to a classic $tf \cdot idf$ if no relevance information is available, and, the formulation also shows that with $idf(t,c)$ only we actually assume that t occurs in all relevant documents ($idf(t,r) = 0$).

This framework makes the relationship between *idf* and relevance information fully explicit.

5. EXPERIMENTS

So far, the results of this paper have been of theoretical nature: relevance feedback leads to a revised *idf* in $tf \cdot idf$-based retrieval functions. In the derivation of this result, we proposed Poisson-based probability estimates instead of the classic (n/N-based) ones. As the weighting schemes corresponding to the BIR model combine *idf*-values of sets with highly different cardinalities, we investigate experimentally whether a Poisson-based estimation makes sense for probabilities $P(t|c)$ and $P(t|r)$.

5.1 Experimental Setup

The experimental evaluation consists of a first series of *adhoc* retrieval experiments to investigate the effect of Poisson-based estimates for *idf* in a setting in absence of relevance information, and a subsequent series of *routing* experiments to investigate its effect on processing relevance information using the four variants of the BIR model to exploit relevance information.

We have used the TREC test collections developed at TREC-7 and TREC-8 (using the data of 'disks 4 and 5', and topic sets 351–400 and 401–450 respectively). Pre-processing of the documents included neither stop-word removal nor stemming, resulting in the same experimental setup as the one described in [4]. However, terms in the query that occur on Van Rijsbergen's stop-word list have been removed from the query text.

Like the adhoc experiment, the routing experiment follows the setup described in [4], where the training data consists of the Los Angeles Times articles and the test data consists of the remaining data. This division of the TREC data in training and testing is a natural one for a routing task, because the training articles date from 1989 and 1990, in time preceding the rest of the collection which contains publications from 1991 until 1994.

The retrieval system has been developed by extending the open-source main memory database system MonetDB[2] with some minimal extra functionality to support information retrieval. To validate the basic workings of this retrieval system implementation, we first reproduced the baseline adhoc retrieval approach presented in [4], and obtained indeed the mean average precision (MAP) reported there.[3] We then proceeded to implement the BM25 ranking, where we choose $k_1 = 1.2$ and $k_3 = 1000$ following the Okapi TREC papers. We have set the length normalisation parameter b to 0.7627, as derived analytically in [1].

5.1.1 Adhoc experiments

Table 2 presents an overview of the results on the TREC-7 and TREC-8 adhoc test collections. First, compare the bottom row displaying the baseline results of a standard BM25 ranking using *idf* estimation, to the other rows, showing the results for BM25 using idf_P instead of *idf*. On the short

[2]See `http://monetdb.cwi.nl/`.

[3]Notice that these results are lower than those reported in [3], which we attribute to differences in pre-processing.

Table 2: Mean Average Precision of adhoc retrieval experiments for topics created from title (T), description (D) and narrative (N). The bottom line is the normal *idf* estimate; the others are Poisson-estimated *idf* with varying K, where $N = 528,024$, and $\widehat{n_t} = \frac{1}{N}\sum_t n_t \approx 260$.

	TREC–7			TREC–8		
K	T	TD	TDN	T	TD	TDN
1	0.143	0.146	0.153	0.161	0.160	0.168
$\widehat{n_t}$	0.144	0.149	0.161	0.164	0.166	0.184
$N/100$	0.153	0.164	0.193	0.171	0.194	0.214
$N/50$	0.157	0.168	0.198	0.175	0.198	0.218
$N/10$	**0.163**	**0.176**	**0.204**	0.185	0.209	**0.225**
$N/3$	0.161	0.175	0.195	0.188	**0.211**	0.223
$N/2$	0.159	0.173	0.192	0.188	0.211	0.221
N	0.157	0.170	0.187	**0.188**	0.210	0.218
-	0.138	0.148	0.154	0.183	0.194	0.194

Table 3: Mean Average Precision of adhoc retrieval experiments for topics created from title (T), description (D) and narrative (N), using *idf* only.

	T	TD	TDN
TREC-7	0.124	0.095	0.041
TREC-8	0.136	0.111	0.064

TREC-7 queries, all of the idf_P results outperform the baseline. If K is sufficiently high, the mean average precision of runs using idf_P is better than the baseline regardless the topic length; $K = N/10$ gives near-best results for all of the experimental conditions. From these experiments, it seems fair to conclude that it makes sense to apply the Poisson-estimation to the occurrence frequencies underlying the inverse document frequency as a model of the discriminative power of query terms.

Tables 3 and 4 present the results obtained using $idf(t, c)$ weighting *only*, using *idf* and idf_P respectively. Performance of the title-only queries is still reasonably good when compared to the results in Table 2. Using *idf* on its own degrades however for longer queries, whereas idf_P results are impressive regardless the topic length.

5.1.2 Routing experiments

The routing experiments use the relevance assessments on the LA Times articles in the estimation of $idf(t, r)$. The baseline results are those obtained on the test data (not including the training data) without using any relevance information. We then apply the four weighting schemes of the BIR model, comparing the effect of relevance information obtained with the standard *idf* to that using the Poisson-based idf_P.

Table 4: Mean Average Precision of adhoc retrieval experiments for topics created from title (T), description (D) and narrative (N), using idf_P only ($K = N/10$).

	T	TD	TDN
TREC-7	0.133	0.143	0.127
TREC-8	0.136	0.158	0.137

Table 5: Mean Average Precision of routing experiments with weighting strategies, for topics created from title (T), description (D) and narrative (N). The Poisson-based *idf*, denoted idf_P, is parameterised with $K = N/10$.

		TREC–7			TREC–8		
Method		T	TD	TDN	T	TD	TDN
-	*idf*	0.125	0.135	**0.147**	**0.181**	**0.193**	0.194
F1	*idf*	**0.135**	**0.138**	0.144	0.154	0.180	**0.201**
F2	*idf*	**0.135**	**0.138**	0.144	0.154	0.180	**0.201**
F3	*idf*	0.135	0.137	0.139	0.153	0.177	0.195
F4	*idf*	0.135	0.137	0.139	0.153	0.177	0.195
-	idf_P	**0.161**	**0.178**	**0.206**	**0.181**	**0.213**	**0.237**
F1	idf_P	0.152	0.165	0.191	0.168	0.205	0.229
F2	idf_P	0.152	0.165	0.191	0.168	0.205	0.229
F3	idf_P	0.151	0.165	0.190	0.168	0.205	0.228
F4	idf_P	0.151	0.165	0.190	0.168	0.205	0.228

Table 6: Mean Average Precision of routing experiments with idf_P for $K = 1$, for weighting strategies, topics created from title (T), description (D) and narrative (N).

		TREC–7			TREC–8		
Method		T	TD	TDN	T	TD	TDN
-	idf_P	0.141	0.148	0.158	0.149	0.153	0.175
F1	idf_P	0.121	0.089	0.048	0.129	0.104	0.086
F2	idf_P	0.121	0.089	0.048	0.129	0.104	0.086
F3	idf_P	0.089	0.064	0.050	0.132	0.112	0.095
F4	idf_P	0.089	0.064	0.050	0.132	0.112	0.095

Table 5 summarises the routing results. Whereas with classical *idf*, relevance feedback obtained from the assessments on the Los Angeles Times articles improves results over the baseline, applying the BIR relevance weighting using the Poisson-based idf_P does not lead to results that improve upon *its* baseline. Notice however that *all* experimental results using idf_P result in higher mean average precision scores than all of those using *idf*, with or without relevance information. While for long queries, a marginal performance improvement is observed with standard *idf* weighting, the results after feedback remain significantly below the idf_P baseline. The differences in mean average precision between F1 and F2 seem negligible, as well as those between F3 and F4. Surprisingly, the weighting schemes based on term presence only (F1 and F2) outperform consistently those on term presence and absence (F3 and F4).

We presented only the experimental results for $K = N/10$, but the observations made hold for other choices of K as well. Table 6 shows an adverse effect of the Poisson weighting on the routing task performance (especially on the longer queries). Again, these are representative for all settings with small K.

5.2 Results and Analysis

The difference in performance between short and long queries in tables 3 and 4 can be partially explained by assuming that users stating a short 'title' query select very carefully the most discriminative terms with respect to relevance, while long queries like the description and narrative also contain noisy terms that do not directly relate to the underlying information need.

The *idf* values do not differentiate between discriminative

yet non-relevant terms and non-discriminative yet relevant terms in the query. The Poisson-based *idf* yields the same order of terms with respect to their discriminativeness as the classical *idf* does, but reduces the influence of non-discriminative terms on the ranking. Therefore, the effect of using one or the other approach is stronger for long queries than for short queries.

Setting the K-parameters of the Poisson-estimates to small values corresponds to weakening the effect of *idf* on the RSV, whereas a large K strengthens the effect of *idf*. Consistently, we can observe that a strong *idf* leads to better retrieval results.

A Poisson-based *idf* with large K is consistently superior to the classical *idf*. This experimental result can be explained as follows from a theoretical point of view. The Poisson-based *idf* distinguishes stronger between discriminative terms, and distinguishes less between non-discriminative terms than the classical *idf*. As our results show, this nature of the Poisson-based *idf* of being stronger for rare terms and less strong for frequent terms leads to better retrieval quality. This result perfectly coincides with the superiority of the Poisson-based probability reflecting the within-document frequency.

Further in-depth analysis is needed to give an explanation as to why the ample training data in the routing experiment did not lead to a considerable improvement on the test data; though we like to point out that [4] also reports disappointing results with feedback in the routing experiment performed; so, maybe in the specific experimental setup chosen it is especially hard to gain improvements, and we should repeat the experiment on another (TREC) routing or filtering task.

6. SUMMARY AND CONCLUSIONS

This paper makes explicit the relationship between *idf* and the binary independent retrieval (BIR) model. The impact of this result is that relevance information can be incorporated into any $tf \cdot idf$-based retrieval function by revising the *idf* according to the relevance information available. Our result makes explicit that relevance information can be viewed as a loss of entropy. The discriminativeness of terms, initially high because of maximal entropy, decreases for two reasons. First, the event space for estimating term discriminativeness after feedback is smaller. Second, the discriminativeness of terms in relevant documents is subtracted from the original *idf*.

In addition to this theoretical framework for relating *idf* and the BIR model, and for justifying the incorporation of relevance information into a revised *idf*-component, we investigated Poisson-based probability estimations. Motivated by the success of Poisson-based estimates for within-document frequencies, we applied Poisson-estimates for the occurrence probabilities $P(t|r)$ (term occurrence in relevant documents) and $P(t|c)$ (term occurrence in all documents), which form the basis for the discriminativeness measures. The overall result is that the Poisson-based *idf* is superior to the classical *idf*, in particular for long queries. Unfortunately, we could not prove experimentally a positive effect in improving retrieval by relevance feedback.

Our next research steps include the transformation of *idf*-values for measuring discriminativeness (informativeness, respectively) into probabilities. This is a key issue for probabilistic retrieval models and probabilistic reasoning. For example, the framework described in [15] is based on a disjoint space of concepts, where we could apply an *idf*-based probability as an estimate of the probability of term informativeness. The refinement of those term space probabilities according to relevance feedback data has been an open problem. Though using *idf*-values as a basis for probability estimation poses a number of problems, the result of this paper on how to manage relevance information in an *idf*-based space might lead to new insights in how information theory and probability theory interrelate.

Acknowledgement

We thank Djoerd Hiemstra for providing pre-processed TREC data.

7. REFERENCES

[1] G. Amati. *Probability Models for Information Retrieval based on Divergence from Randomness*. PhD thesis, Glasgow University, June 2003.

[2] K.W. Church and W.A. Gale. Inverse document frequency: A measure of deviations from poisson. In *Third Workshop on Very Large Corpora, ACL Anthology*, 1995.

[3] A.P. de Vries and D. Hiemstra. The Mirror DBMS at TREC-8. In *Proceedings of the Eighth Text Retrieval Conference TREC-8*, pages 725–734, Gaithersburg, Maryland, November 1999.

[4] D. Hiemstra, S.E. Robertson, and H. Zaragoza. Parsimonious Language Models for Information Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, July 2004.

[5] J. Lafferty and Ch. Zhai. *Probabilistic Relevance Models Based on Document and Query Generation*, chapter 1. Kluwer, 2002.

[6] S.E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, London, et al., 1994. Springer-Verlag.

[7] S.E. Robertson, S. Walker, and M.M. Hancock-Beaulieu. Large test collection experiments on an operational interactive system: Okapi at TREC. *Information Processing and Management*, 31:345–360, 1995.

[8] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[9] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments. *Journal of Documentation*, 60(5):503–520, 2004.

[10] T. Roelleke. A frequency-based and a poisson-based probability of being informative. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada*, pages 227–234, 2003.

[11] T. Roelleke, T. Tsikrika, and G. Kazai. A general matrix framework for modelling information retrieval. *Journal on Information Processing & Management (IP&M)*, 2005. To appear.

[12] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145, 2003.

[13] G. Salton and C. Buckley. On the use of spreading activation methods in automatic information retrieval. In *11th International Conference on Research & Development in Information Retrieval*, pages 147–160, Grenoble, France, June 1988.

[14] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2. edition, 1979.

[15] S.K.M. Wong and Y.Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68, 1995.

Exercise 1.3 - TF-IDF, BM25 & BM25VA, and evaluation (100/100 points)

188.980 Advanced Information Retrieval 2018S

March 13, 2018

Abstract

At the end of this exercise, you should have an understanding of the practical issues of creating an inverted index for information retrieval, being able to implement basic scoring, as well as being able to comprehend and apply advance scoring functions from research papers. The exercise should be done in pairs, but each of the members has to be able to answer questions about the implementation provided. **Make sure you go through the lecture slides and relevant book chapters at least once before starting on this exercise.**

Task

In Exercise 1.3, your task is to index and search **without** the use of any library (like: Lucene, Terrier, etc.[1]), to create a scorer that implements both TF-IDF and BM25 scoring functions, and to implement an improved version of BM25 based on the article published by Aldo Lipani et al. 2015 at ICTIR and available online at `https://www.researchgate.net/publication/280599788_Verboseness_Fission_for_BM25_Document_Length_Normalization?ev=prf_pub` as well as at the end of this Exercise sheet. You also have to evaluate your results by using the `trec_eval` program. Finally, you need to write a short report, upload everything to TUWEL, and present your results in person.

Warning

About a week before the hand-in of this assignment we offer the possibility to do a pre-hand-in (i.e. an optional checkpoint) in order to solve any sort of technical and non-technical problems you may encounter during the execution of the assignment, so please take advantage of it. **Excuses about technical problems at the presentation session will not be tolerated**.

Functionality (70%)

Index

Build an inverted index from the document collection CD4&5 of the TIPSTER collection that is available in TUWEL: TREC8Adhoc.tar.bz2 in TREC8all.zip.

Each file in the provided dataset contains several documents (indicated by the `<DOC>...</DOC>` tags). Single documents contain an ID (`<DOCNO>...</DOCNO>`) and text (`<TEXT>...</TEXT>`). Documents that do not have a `<TEXT>` tag can be ignored.

If you encounter encoding problems reading the dataset, make sure that you use the correct file encoding `ISO 8859-1`.

[1] However, you may use a library for stemming or lemmatizing

Before indexing, apply any combination of the techniques described in Chapter 2 of the *Introduction to Information Retrieval* book (case folding, removing stop-words, stemming, lemmatization). These options should be exposed as parameters in the index creation phase. For the stemming and lemmatization components, you can use a library at your choice.

You should provide an executable file `indexer.sh` (*.bat for windows) that given the necessary parameters generates the required index.

You may use any format for storing the index.

Search

Implement a basic search functionality and provide it as a command line interface (CLI). No GUI is required. The CLI allows the user to pass search parameters and a file that contains the search topics (i.e. `topicsTREC8Adhoc.txt`). The system then returns a list of documents ranked by a similarity function of your choice (in Exercise 1.3 either: TF-IDF, BM25 or BM25VA). The scoring functions and their individual parameters need to be exposed as parameters to the CLI.

Details regarding the BM25 modification: Your task is to modify your implementation of the BM25 scoring function and create a new similarity class called BM25VA as in the paper, which introduces the concept of document repetitiveness and infers through an heuristic the parameter b. This scoring function requires unusual statistics of the document that can be found in the direct index, e.g., the number of unique terms per document and, (to be computed once) the mean average document repetitiveness.

You are provided with a set of 50 topics to search for. The topic file is contained in the dataset and has a simple SGML structure. Each topic is delimited by `<TOP>` tags and you are free to use all or part of their content in your search.

As a recommendation, try to implement topic processing and the actual scoring/ranking as two separate tasks. By topic processing, we understand here simply getting the terms that will be searched in the index. Consider the use or not use of the different elements in the topic document.

Your search should be (moderately) efficient. It is, for example, not OK to perform your collection indexing only at search time, and also loading the inverted index should be reasonably in time, or at least not happen before each search.

The output of the CLI given the previously described parameters should be a ranked list of up to 1000 documents, where each line is in the following format:

$$\{\textit{topic-id}\}\ \textit{Q0}\ \{\textit{document-id}\}\ \{\textit{rank}\}\ \{\textit{score}\}\ \{\textit{run-name}\}$$

where

topic-id is an integer number between 401 and 450;

document-id is an identifier for the document (e.g. LA122690-0033);

Q0 this field is unnecessary to our purposes but must be present;

rank is an integer indicating the rank of the object in the sorted list (normally, this should be ascending from the first line to the last line)

score the similarity score calculated by (it has to not exceed the float precision and normally, this should be descending from the first line to the last line)

run-name a name you give to your experiment (free for you to choose)

Here is a short example of a potential output for all the topics:

```
401 Q0 FBIS3-10899 1 2.813525 grp5-exp1
401 Q0 FBIS3-13322 2 1.0114759 grp5-exp1
...
450 Q0 LA043089-0083 1000 0.08848727 grp5-exp1
```

You should provide an executable `searcher.sh` (*.bat for windows) file that given the necessary parameters (i.e. `[scoring function] [scoring function's paramters] [topic file]`) generates a ranked list as above.

Evaluation

You must use `trec_eval` (available at: `https://github.com/usnistgov/trec_eval`) to calculate the Mean Average Precision (MAP) of your result lists over the 50 topics provided. `trec_eval` is a small C application, which means that you have to compile it for your own machine. In principle, you can use any C compiler (most Linux distributions already come with one), and just run the `make` command in the folder[2].

```
$trec_eval -q -m map -c qrels_file your_results_file
```

where the parameters are[3]:

-m measure: Add 'measure' to the lists of measures to calculate and print. If 'measure' contains a '.', then the name of the measure is everything preceding the period, and everything to the right of the period is assumed to be a list of parameters for the measure, separated by ','. There can be multiple occurrences of the -m flag. 'measure' can also be a nickname for a set of measures. Current nicknames include

'official' : the main measures often used by TREC

'all_trec' : all measures calculated with the standard TREC results and rel_info format files.

'set' : subset of all_trec that calculates unranked values.

'prefs' : Measures not in all_trec that calculate preference measures.

-c: Average over the complete set of queries in the relevance judgements instead of the queries in the intersection of relevance judgements and results. Missing queries will contribute a value of 0 to all evaluation measures (which may or may not be reasonable for a particular evaluation measure, but is reasonable for standard TREC measures.) Default is off.

-q: In addition to summary evaluation, give evaluation for each query/topic

and where

`qrels_file` is the file available on the resources on TUWEL.

`your_results_file` is your results file using the format indicated above, where all topics are present.

In the end, your results should be in a table like:

	TF-IDF	BM25	BM25VA
401	0.0301	0.0252	0.0251
...	...	...	...
450	0.0230	0.0350	0.0241
AVG	0.0210	0.0220	0.0151

[2]Note that last year some students could not compile the latest trec_eval on Windows, but managed to compile version 8.1 (available here `http://trec.nist.gov/trec_eval/`). If that happens to you, you may use 8.1 as well.

[3]taken from the help menu of trec_eval. Type `trec_eval -h` to see the full list of options

Point allocation

The 70 percentual points are allocated as follows:

Document processing	
basic tokenizer	12
case folding	2
special strings	2
stemming	3
lemmatization	3
Index creation	
Simple posting list, Hash or B-Tree dictionary	13
Single Pass In Memory Indexing[4]	6
Map-Reduce	6
Search	
TF-IDF	3
BM25	3
Paper scoring function (BM25VA)	7
Evaluation	
simple	12
calculate statistical significance for all pairings (=3 pairs) of scoring functions	2
Total	**74**

Report (15%)

Describe your prototype in a report. Describe briefly your index, in particular which additional information is required to be saved, if any. Show how the score is calculated for two documents adjacent in the ranked list. For this, you may use a query of your choosing (the simpler the query, the simpler the explanation, but you must have at least two terms in the query). Finally, show your MAP results calculated using `trec_eval`. **Try to find a justification of why some of your retrieval models performed better than the others.** The report must explain how to run the prototype. Maximum size: 4 pages or 2000 words.

Hand-in Presentation (15%)

As part of the presentation, general knowledge about the system you have used will be tested (e.g. what a tokenizer is, which stemmer have you used, how to change it, how to change the scoring function, what are the language-specific features).

- All the source code must be uploaded on a public Git repository, and the coordinators invited (markus.zlabinger@tuwien.ac.at and lipani@ifs.tuwien.ac.at);
- Prototypes are presented to the coordinator in one of the labs;
- Final deadline is **April 18th - 23:59** . For that, you must upload a zipped file to TUWEL (report, source code, and corresponding executable files incl. all dependencies);
- Your submission has to be self-contained;
- You must book a time on TUWEL for the presentation;
- You present on your own notebook to the coordinator in the lab;
- Lab locations are on TISS.

Note: You must have 35 points for this exercise in order to pass the course.

[4]Depending on your hardware, you will probably have enough memory to index everything without the creation of additional blocks. Despite of having enough memory, make sure that you create at least 2 blocks when using SPIMI.

Verboseness Fission for BM25 Document Length Normalization

Aldo Lipani[1] Mihai Lupu[2] Allan Hanbury[2] Akiko Aizawa[1]

[1]National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku
Tokyo, Japan
{surname}@nii.ac.jp

[2]Inst. of Software Technology & Interactive Systems
Vienna University of Technology
Vienna, Austria
{surname}@ifs.tuwien.ac.at

ABSTRACT

BM25 is probably the most well known term weighting model in Information Retrieval. It has, depending on the formula variant at hand, 2 or 3 parameters (k_1, b, and k_3). This paper addresses b—the document length normalization parameter. Based on the observation that the two cases previously discussed for length normalization (multi-topicality and verboseness) are actually three: multi-topicality, verboseness with word repetition (repetitiveness) and verboseness with synonyms, we propose and test a new length normalization method that removes the need for a b parameter in BM25. Testing the new method on a set of purposefully varied test collections, we observe that we can obtain results statistically indistinguishable from the optimal results, therefore removing the need for ground-truth based optimization.

1. INTRODUCTION

BM25 is the most longevous weighting schema in Information Retrieval (IR), still widely used in industry and studied in research. The peculiarity of this weighting schema is its probabilistic root that is based on the 2-Poisson model of term frequencies in documents [13]. In its classic version, a document d is scored by the function:

$$S(q,d) = \sum_{t \in T_q \cap T_d} \frac{(k_3+1)tf_q}{k_3+tf_q} \frac{(k_1+1)\overline{tf_d}}{k_1+\overline{tf_d}} \log \frac{|D|+0.5}{df_t+0.5}$$

with

$$\overline{tf_d}(t) = \frac{tf_d(t)}{B} \quad B = (1-b) + b\frac{L_d}{avgdl}$$

where q is the query, D is the set of documents, $d \in D$ is a document, T_d and T_q are the sets of document terms and query terms, $\overline{tf_d}$ is the normalized term frequency of the term t within the document d, tf_q is the term frequency of the term t within the query, df_t is the document frequency of the term t, L_d the length of the document d, $avgdl$ the average document length over the collection D of documents, and k_1, b and k_3 the three parameters with domains $[0,\infty[$, $[0,1]$ and $[0,\infty[$.

This semi-parametric retrieval function [10] has 3 degrees of freedom, each with a specific meaning: k_1 and k_3 tune how fast the respective tf component saturates, expressing the importance of the presence of an additional occurrence of the term t in the document or query. The parameter b controls the normalization of the tf component, varying between the two extremes of non normalized, when $b = 0$, and fully normalized by the coefficient of variation of the document length, when $b = 1$.

The tuning of the three parameters is not an easy problem, nor a resource free task, due to the required development of a test collection. Hence, in most cases, the suggested values are used: $k_1 = 1.2$, $b = 0.75$ [14] and $k_3 = 8$ [12] . Still, as shown by Chowdhury et al. [2], tuning the parameters can lead to a considerable improvement in the effectiveness of the retrieval system. However, tuning is only possible if ground truth is available, and another, more analytical approach can be taken. This consists in trying to better understand the geometry of the information space, in order to extend and improve the current model.

In this paper, we focus on the term frequency normalization, reopening the discussion described by Robertson and Zaragoza [13] about the verboseness and scope hypotheses. We propose a new parameter-free normalization, based on the features of the document collection. We test this model using a sample of five test collections from TREC and CLEF, selected on purpose from different domains: Web, News, Medical, and Patent, in order to verify experimentally the dependency between the normalization factor and the features of the document collection.

The remainder of the paper is structured as follows: in Section 2 we provide a very brief summary of the extensive work already done on the study and understanding of the term frequency normalization. Section 3 provides the intuition of our method and introduces the required concepts and the method itself. In Section 4 we present and discuss our experimental results. We conclude in Section 5.

2. RELATED WORK

The initiators of the discussion about the term frequency normalization are the early participants in TREC, with first insights appearing after TREC-3, and the first efforts on document length normalization showing improved results in TREC-4 [3]. To understand why a document is long, Robertson and Zaragoza [13, p. 358] describe two hypotheses: a) verboseness, to convey the same information using

ICTIR'15, September 27 - 30, 2015, Northampton, MA, USA.

ACM 978-1-4503-3833-2/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2808194.2809486 .

more words than needed; and b) scope, to convey information containing more topics, details, or aspects. These hypotheses have a conflicting effect when treating the normalization in terms of length, because while the first suggests to normalize the *tf* by the length, the second suggests the opposite. Hence, the introduction of a soft normalization based on the coefficient of variation of the document length and the introduction of the b parameter that controls the slope of the normalization factor. This is of course not the only way for length normalization. Among others, Singhal et al. [17] studied it extensively for the TF-IDF model.

Not much work has been done on the scope hypothesis, except perhaps the effort spent in passage retrieval. Here, document length is circumvented by viewing the document as a collection of concatenated shorter documents to be retrieved individually.

More work has been done to tackle the verboseness issue. Na et al. [11] briefly introduce the concept of verboseness given by repetitiveness of terms. They compare it with multi-topicality under the language modeling framework. The normalization factors are corrected based on the assumption that the vocabulary size can be used to estimate the number of topics contained in the document. He and Ounis [5] introduced a new term frequency normalization following the idea of Amati [1], who introduced the use of Dirichlet Priors. He and Ounis point out the relationship between test collection features on term frequency normalization, and introduce a new parameter, learned from the test collection. They defined the normalization effect and hypothesized that the optimal parameter is the value that makes the normalization factor give similar normalization results across different corpora [4, 6]. Lv and Zhai pointed out that the retrieval pattern of BM25 does not follow the relevancy pattern, biasing the system against long documents, and introduced a boosting parameter δ that summed to the normalized term frequency in a first version [9] and then summed to the term frequency component in a second version [8] to correct the pattern discrepancy.

Rousseau and Varzirgiannis [15] analyze the problem in terms of function composition, comparing BM25 with TF-IDF and combining the two works previously mentioned, to gain a better understanding of the similarity across the models. Some efforts have been directed towards understanding and removing the parameters of BM25: Lv and Zhai [7] pointed out that it is more effective to use a term-specific k_1, and that it is possible to estimate it using an information gain measure to quantify the contributions of repeated term occurrences. They do not address b.

Overall, a criticism of all of these works is that the studies of and experiments with new models of the term frequency normalization always use the same kind of test collection, News and Web corpora.

3. THE NORMALIZATION HYPOTHESES

Document length normalization is based on the observation that documents are long either because they are verbose, or because they cover more aspects, as discussed.

The insight at the base of this study is that we can distinguish two kinds of verboseness: a) *repetitiveness*, in which the same terms are repeated many times (e.g. legal or patent texts); and b) *non-repetitiveness*, where the writer uses different terms to describe the same thing (e.g. over-descriptive narration in Balzac's novels). In the first case, *tf* is expected to be higher, so it should be normalized more than in the second case, where it is naturally low because of the use of different terms. While non-repetitiveness implies a more semantic analysis of the text, repetitiveness can be easily identified by counting the number of times terms are repeated on average. We define this in an obvious way as the average term frequency:

$$avgtf_d = \frac{1}{|T_d|} \sum_{t \in T_d} tf_d(t) = \frac{L_d}{|T_d|} \tag{1}$$

However, we should observe that while a high $avgtf_d$ is indicative of repetitiveness, a low $avgtf_d$ would indicate either a broader document that would fall into the scope hypothesis, or a verbose, non-repetitive document. From the observation that *tf* is expected to be higher, we have the chance to better discern the sets of elite and non-elite documents described in the 2-Poisson model. Embedding this new knowledge in the model would take into account the fact that observing a high *tf* can be due either to its relevance in the document, as an elite term, or because of its repetitiveness, as boilerplate, non-elite term.

Our intuition is that it is possible to differentiate the two kinds of verboseness for a specific document based in part on collection statistics. We can then diminish the effect of the *tf* for each document by comparing the average *tf* of a document with $mavgtf$, the mean average term frequency of the collection:

$$mavgtf = \frac{1}{|D|} \sum_{d \in D} avgtf_d \tag{2}$$

First, a few observations on these new indicators. The average term frequency of a document d ($avgtf_d$) is an indicator of how many times the same term is repeated in the document. If a document does not have any repetitions, $avgtf_d$ is equal to 1. The average term frequency is simply document length over number of unique terms, and this makes it easy to verify that $avgtf$ has domain $[1, \infty[$, assuming documents of finite length $[1, maxL]$, where $maxL$ is the length of the longest document. Since $mavgtf$ is the average of the $avgtf$, it has the same domain. If the test collection is made on average of documents with a low level of repetition, then the $mavgtf$ is very close to 1. The $mavgtf$ is a collection specific value that summarizes the repetitiveness of the language in a specific corpus.

From the intuition above we infer first that if the language of a collection is repetitive (high $mavgtf$), we expect to need more length normalization. Therefore we define the BM25 document length normalization factor b as:

$$b = 1 - mavgtf^{-1} \tag{3}$$

This definition of b has the required domain $[0, 1[$ and increases monotonically with $mavgtf$.

With this normalization parameter, we can define a new normalization factor, B_{-b} to be used in BM25:

$$B_{-b} = mavgtf^{-1} + (1 - mavgtf^{-1}) \frac{L_d}{avgdl} \tag{4}$$

However, the reader may have already noticed a potential issue with this new b: while in theory it has domain $[0, 1[$, it will only reach 1 as $mavgtf \to \infty$. In practice the $mavgtf$ is actually small, in the range of the low single digits, and this will limit the values of b to the lower half of the normalization spectrum. The normalization factor B_{-b}

Corpus	EC	Challenge	$\|D\|$	$mavgtf$
Aquaint	TREC	Hard 2005	1,033,461	1.519
Disks 4&5	TREC	Ad Hoc 8	528,155	1.574
eHealth'13	CLEF	eHealth 2013	1,102,848	2.205
.GOV	TREC	Web 2002	1,247,753	2.481
CLEF-IP'10	CLEF	CLEF-IP 2010	2,670,678	3.008

Table 1: Corpora used, with information about the challenge and evaluation campaign (EC) to which it belongs, number of documents, and mean average term frequency.

will therefore tend to be very conservative in its document length normalization. This is partially by design: as we said before – we do not want to do strong length normalization if the collection is not using repetitive language. Even more, at this point we are still not making a distinction between repetitiveness and non-repetitiveness at document level. To do so, and at the same time to control the $(1-b)$ component in B, we need to introduce another factor in B:

$$B_{\text{VA}} = (1-b)\frac{avgtf_d}{mavgtf} + b\frac{L_d}{avgdl} \quad (5)$$

This new factor, $\frac{avgtf_d}{mavgtf}$, boosts the normalization factor B when the document at hand is repetitive.

The new formulation for B can also be seen as a reinterpretation of document length normalization: it is now no longer a linear combination between doing or not doing length normalization, but rather a linear combination between normalizing for repetitiveness or length (non-repetitiveness), controlled by a parameter b bound to the general repetitiveness of the language of the collection. For collections that are generally repetitive, it will tend to do length normalization, and the newly added factor will reduce this normalization only for those documents that are not repetitive. For collections that are generally non-repetitive, it will tend to not do length normalization, and the newly added factor will increase this normalization only for those documents that are repetitive. Intuitively, the method compares the repetitiveness of the document with that of the collection. The proposed variant makes the repetitiveness of the document no longer a good indicator of verboseness if the collection is generally repetitive.

Finally, using our b from Eq. 3, our variant of BM25 normalization factor is:

$$B_{\text{VA}} = mavgtf^{-2}\frac{L_d}{|T_d|} + (1 - mavgtf^{-1})\frac{L_d}{avgdl} \quad (6)$$

4. EXPERIMENTS

To test our predictions we selected five ad hoc test collections from TREC and CLEF, with the aim to observe differences in the use of language, in different domains. We selected from News, Web, Medical, and Patent corpora, listed in Table 1, where we can observe how the average term frequency varies across the corpora. To assess the different experiments, we used the condensed version [16] of mean average precision (MAP') and precision at 10 (P@10') because of their better stability in case of incomplete judgments. We tested the new normalization factors, B_{-b} and B_{VA}, against two different configurations of the classic BM25: standard and ideal. The BM25 standard is characterized by having the suggested configuration of the parameters, k_1 and b. In the ideal BM25, the two parameters have been optimized using as training set and test set the same set of topics, which of course makes it an unrealistic scenario, but an interesting upper limit. In this case, k_1 varies between 0.5 and 2.5. In all experiments we set $k_3 = 0$ to avoid any potential interferences of the tf_q in the scoring of the document.

Track	P.	k1	b	MAP'	P@10'
Standard Case					
Hard 2005	CL	1.20	0.75	0.2144‡	0.3600‡
	CL-b	1.20	-	0.2325†	0.4360†
	VA	1.20	-	0.2318†	0.4360†
Ad Hoc 8	CL	1.20	0.75	0.2504‡	0.4720
	CL-b	1.20	-	0.2578	0.4600
	VA	1.20	-	0.2677†	0.4940
eHealth'14	CL	1.20	0.75	0.5565	0.7694
	CL-b	1.20	-	0.5636‡	0.7878‡
	VA	1.20	-	0.5718‡	0.7694
Web'02	CL	1.20	0.75	0.2022	0.2460
	CL-b	1.20	-	0.1972	0.2440
	VA	1.20	-	0.2010	0.2520
CLEF-IP'10	CL	1.20	0.75	0.3562‡	0.6423‡
	CL-b	1.20	-	0.3537‡	0.6371‡
	VA	1.20	-	0.3556‡	0.6371‡
Ideal Case					
Hard 2005	CL	1.65	0.25	0.2346†	0.4440†
	CL-b	1.70	-	0.2335†	0.4360†
	VA	1.80	-	0.2332†	0.4140†‡
Ad Hoc 8	CL	0.45	0.40	0.2715†	0.4600
	CL-b	0.45	-	0.2713†	0.4520
	VA	0.55	-	0.2744†	0.4900
eHealth'14	CL	2.30	0.55	0.5849	0.8143
	CL-b	2.30	-	0.5844†	0.8143
	VA	2.40	-	0.5922	0.7959†
Web'02	CL	2.40	0.70	0.2062	0.2460
	CL-b	2.05	-	0.2012	0.2420
	VA	2.50	-	0.2056	0.2360
CLEF-IP'10	CL	2.50	1.0	0.3713†	0.6540†
	CL-b	2.50	-	0.3615	0.6536
	VA	2.25	-	0.3643	0.6567

Table 2: Scores obtained with the classic BM25 (CL), classic BM25 with b as in Eq. 3 (CL-b), and our variant (VA). † indicates statistical significance (t-test, p<0.05) against the standard classic BM25 (CL) and ‡ against the ideal classic BM25 (CL).

We used the search engine Terrier[1] 4.0 for the classic BM25 and developed and integrated in it our BM25 variants[2]. All the documents have been preprocessed using the English tokenizer and Porter stemmer of the Terrier search engine. The queries are extracted from the title only, except for the CLEF-IP 2010 where the abstract has been included.

Table 2 shows the performance of each weighting scheme in the two configurations mentioned above. In only two of the five collections (eHealth and CLEF-IP) the standard VA is lower and statistically significantly different from the ideal classic BM25 (CL). This can be explained by the combination of two effects: the large influence k_1 has on the results, as shown in Fig. 1 by the size of the gray areas, and the large difference between the standard k_1 and the ideal k_1.

[1] `http://www.terrier.org`
[2] Code available on the website of the first author

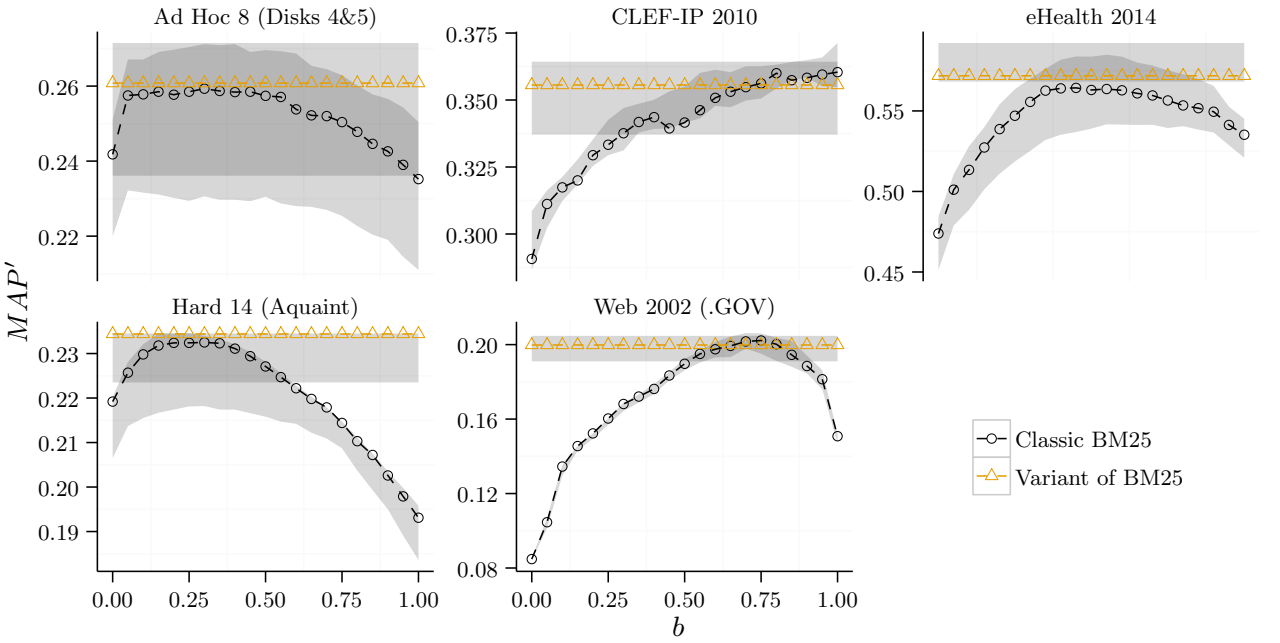

Figure 1: Performance sensitivity of the classic BM25 (CL) and our variant (VA) across all the test collections, with standard $k_1 = 1.2$. The gray area represents the range of values obtainable varying k_1 in the range $[0.5, 2.5]$.

5. CONCLUSION

We continued a discussion started 20 years ago in the context of TREC about the need for document length normalization and the nature of the document length itself. Previous studies, working on test collections of web and news corpora, failed to observe what in legal and patent collections is patently obvious: document length verbosity, in a bag-of-words model, can be expressed via repetition or via synonyms. We proposed a new factor B, including a specific value for the parameter b, and showed that, across different domains, the results are generally statistically indistinguishable from those obtained with ideal b values, without having to identify these ideal values. Together with previous works on estimation of the k_1 parameter, this brings us a step closer to a parameter-free, stable, BM25.

6. REFERENCES

[1] G. Amati and J. C. C. Van Rijsbergen. Probabilistic models for information retrieval based on divergence from randomness. *TOIS*, 20(4), 2002.

[2] A. Chowdhury, M. C. McCabe, D. Grossman, and O. Frieder. Document Normalization Revisited. In *Proc. of SIGIR*, 2002.

[3] D. Harman. Overview of the Fourth Text REtrieval Conference (TREC-4). In *Proc. of TREC 4*, 1995.

[4] B. He and I. Ounis. A Study of Parameter Tuning for Term Frequency Normalization. In *Proc. of CIKM*, 2003.

[5] B. He and I. Ounis. A Study of the Dirichlet Priors for Term Frequency Normalisation. In *Proc. of SIGIR*, 2005.

[6] B. He and I. Ounis. Term Frequency Normalisation Tuning for BM25 and DFR Models. In *Proc. of ECIR*, 2005.

[7] Y. Lv and C. Zhai. Adaptive Term Frequency Normalization for BM25. In *Proc. of CIKM*, 2011.

[8] Y. Lv and C. Zhai. Lower-bounding Term Frequency Normalization. In *Proc. of CIKM*, 2011.

[9] Y. Lv and C. Zhai. When Documents Are Very Long, BM25 Fails! In *Proc. of SIGIR*, 2011.

[10] D. Metzler and H. Zaragoza. Semi-parametric and non-parametric term weighting for information retrieval. In *Proc. of ICTIR*, 2009.

[11] S.-H. Na, I.-S. Kang, and J.-H. Lee. Improving term frequency normalization for multi-topical documents and application to language modeling approaches. In *Proc. of ECIR*, 2008.

[12] S. Robertson, S. Walker, M. Beaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In *Proc. of TREC 4*, 1995.

[13] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 2009.

[14] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. of TREC-3*, 1994.

[15] F. Rousseau and M. Vazirgiannis. Composition of TF Normalizations: New Insights on Scoring Functions for Ad Hoc IR. In *Proc. of SIGIR*, 2013.

[16] T. Sakai. Alternatives to Bpref. In *Proc. of SIGIR*, 2007.

[17] A. Singhal, C. Buckley, and M. Mitra. Pivoted Document Length Normalization. In *Proc. of SIGIR*, 1996.