Information Retrieval: Assignment 2 - Retrieval System

member1@student.ethz.ch
member2@student.ethz.ch
member3@student.ethz.ch

December 1, 2015

1 Retrieval System

In this project, we have build a system which retrieves a ranked list of documents relevant to a query. We have used the provided collection of documents and we have evaluated the system performance on 40 queries and relevant statements.

The system retrieves the ranked document lists for multiple queries simultaneously. The process flow consists of several steps:

1. extract queries and relevance judgements from the files provided.

2. scan the collection of documents in a streaming manner, using *TipsterCorpusIterator* and build up maps with term and collection frequencies for the query words, inverse document frequencies and document lengths. This will be used by the different models to evaluate the relevance of a document.

3. for every query, calculate the score of every document in the collection and keep the top 100 documents in a map. The score is calculated either using the term-based model or the language model depending on the value of the boolean parameter "languageModel".

4. print the resulting ranking to a text file

5. evaluate the system's performance in terms of P, R, F_1 and MAP for the training set.

Both the queries and the documents are tokenized by splitting on various standard characters, stop words are removed and then each word is stemmed using the *PorterStemmer*. In the case of query words which contain hyphens, we have decided to keep both the hyphenated word and the individual terms to increase the chance of finding the term in the collection as queries are usually very short.

2 Models

We have experimented with various models and here we describe the best performing ones with the optimal parameters based on the test queries.

2.1 Term-based model

Initially we implemented the standard TF-IDF model as described in the lecture slides. As the performance was not satisfactory, we experimented with *augmented TF-IDF* and the *OKAPI BM25*[1] model. Both of those improved the performance over the standard function, with the **OKAPI BM25** ranking function 1 performing best. This was used in our system.

$$score(Q, D) = \sum_{q \in Q} IDF(q) * \frac{tf(q, D) * (k + 1)}{k(1 - b + b\frac{|D|}{avgDL}) + tf(q, D)}, \tag{1}$$

where k and b are parameters of the model controlling the scaling of the term frequencies and the normalization of the term frequencies by the document length.

The inverse documented frequency is defined as:

$$IDF(q) = \log \frac{N - df(q) + 0.5}{df(q) + 0.5}, \tag{2}$$

N - documents in collection, $df(q)$ - #docs in which q occurs

The IDF has the property that if a word occurs in more than half of the documents, the log term becomes negative, thus a document is penalized for containing the query word. This should not happen as we have removed stop words from the documents and queries, but just in case we limit the IDF to 0 from the bottom but capping the document frequency to half of the number of documents. This method has been used in other similar projects.

We experimented with different values for the model parameters and found the optimal values to be 0.35 and 1 for b and k respectively.

2.2 Language model

For the language model, we used the standard probabilistic scheme, where $P(q|d)$ is the probability of the query q given the document d and documents are ranked by their probability to generate q according to the maximum likelihood principle. Independence of query terms is assumed given the relevant document, so that the likelihood of the query is the product of the likelihoods of the individual query words 3.

$$P(q|d) = \prod_{w_i \in q} P(w_i|d) \tag{3}$$

We used the observed term frequencies in the collection to estimate the posterior probabilities of the query words given a document - $\hat{P}(w_i|d)$. However, as not all query words occur in all documents, for the likelihood $P(w_i|d)$ we experimented with different types of collection smoothing - Jelinek-Mercer, Dirichlet and two-stage smoothing which combines both [2]. We ended up using the two-stage smoothing as defined in equation 4 because it proved to be the most effective on the training data.

$$logP(w_i|d) = log\Big[(1 - \lambda)\frac{\hat{P}(w_i|d) + \mu * P(w_i)}{|d| + \mu} + \lambda P(w_i)\Big], \tag{4}$$

[1] S. Robertson, S. Walker, M. M. Beaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In NIST Special Publication 500-236: The Fourth Text REtrieval Conference (TREC-4), pages 73 96, Gaithersburg, MD, 1995.

[2] Zhai, Chengxiang, and John Lafferty. "A study of smoothing methods for language models applied to information retrieval." ACM Transactions on Information Systems (TOIS) 22.2 (2004): 179-214.

The prior $P(w_i) = \frac{cf(w_i)}{|C|}$ is the collection frequency of the word normalized by the size of the collection (in terms of words) and λ and μ are model parameters.

In terms of interpretation, λ roughly controls the noise in the query, thus the bigger it is, the less weight is given to the query terms. On the other hand, μ is the standard Dirichlet prior smoothing parameter and controls the collection smoothing to account for words not present in the document.

The optimal values were determined to be 0.1 for λ, which is expected given the short queries, and 1000 for μ.

3 Evaluation on training data

To evaluate our model, we have used the provided *qrels* statements. Any document not explicitly stated as relevant in the relevance judgements is assumed to be irrelevant, thus the scores represent worst-case behaviour in terms of accuracy. For each query we report the precision (P), recall (R), the harmonic mean F_1 and the AP, as well as the MAP for the whole system. All reported values are at rank 100 and the total number of relevant documents is assumed to be $min(TP + FN, 100)$ for the average precision calculations.

3.1 Performance values of OKAPI BM25 Model

Query: 51 Precision: 0.8 - Recall: 0.5797101449275363 - F1: 0.6722689075630253 Average Precision: 0.7146126238923501

Query: 52 Precision: 0.52 - Recall: 0.09719626168224299 - F1: 0.1637795275590551 Average Precision: 0.29455456562708354

Query: 53 Precision: 0.58 - Recall: 0.10157618213660245 - F1: 0.17287630402384502 Average Precision: 0.31497881614972956

Query: 54 Precision: 0.4 - Recall: 0.23391812865497075 - F1: 0.2952029520295203 Average Precision: 0.22937511090976187

Query: 55 Precision: 0.67 - Recall: 0.08271604938271605 - F1: 0.14725274725274726 Average Precision: 0.47059454279770024

Query: 56 Precision: 0.79 - Recall: 0.08997722095671981 - F1: 0.16155419222903886 Average Precision: 0.6394523522327801

Query: 57 Precision: 0.59 - Recall: 0.1279826464208243 - F1: 0.2103386809269162 Average Precision: 0.399094266998191

Query: 58 Precision: 0.67 - Recall: 0.42138364779874216 - F1: 0.5173745173745173 Average Precision: 0.47781502216670146

Query: 59 Precision: 0.25 - Recall: 0.04317789291882556 - F1: 0.07363770250368189 Average Precision: 0.09644209311471354

Query: 60 Precision: 0.05 - Recall: 0.08333333333333333 - F1: 0.0625 Average Precision: 0.030939639469882544

Query: 61 Precision: 0.56 - Recall: 0.27184466019417475 - F1: 0.36601307189542487 Average Precision: 0.45844065409895424

Query: 62 Precision: 0.34 - Recall: 0.11447811447811448 - F1: 0.17128463476070532 Average Precision: 0.16825827286767056

Query: 63 Precision: 0.22 - Recall: 0.10576923076923077 - F1: 0.14285714285714285 Average Precision: 0.07732793596436555

Query: 64 Precision: 0.12 - Recall: 0.032 - F1: 0.05052631578947369 Average Precision: 0.016788141481460586

Query: 65 Precision: 0.35 - Recall: 0.09067357512953368 - F1: 0.1440329218106996 Average Precision: 0.15436576238256985

Query: 66 Precision: 0.33 - Recall: 0.16751269035532995 - F1: 0.22222222222222224 Average Precision: 0.17906936753946223

Query: 67 Precision: 0.0 - Recall: 0.0 - F1: 0.0 Average Precision: 0.0

Query: 68 Precision: 0.1 - Recall: 0.05128205128205128 - F1: 0.06779661016949151 Average Precision: 0.027580874494638006

Query: 69 Precision: 0.23 - Recall: 0.4423076923076923 - F1: 0.3026315789473685 Average Precision: 0.19004873322895843

Query: 70 Precision: 0.26 - Recall: 0.4727272727272727 - F1: 0.33548387096774196 Average Precision: 0.1954595650918126

Query: 71 Precision: 0.72 - Recall: 0.18947368421052632 - F1: 0.3 Average Precision: 0.6203094200090942

Query: 72 Precision: 0.04 - Recall: 0.03361344537815126 - F1: 0.0365296803652968 Average Precision: 0.002711484943627801

Query: 73 Precision: 0.01 - Recall: 0.00546448087431694 - F1: 0.007067137809187279 Average Precision: 2.2222222222222223E-4

Query: 74 Precision: 0.05 - Recall: 0.01002004008016032 - F1: 0.01669449081803005 Average Precision: 0.0036018937409717927

Query: 75 Precision: 0.16 - Recall: 0.043835616438356165 - F1: 0.06881720430107527 Average Precision: 0.02938972199378294

Query: 76 Precision: 0.04 - Recall: 0.013605442176870748 - F1: 0.02030456852791878 Average Precision: 0.012077691811734364

Query: 77 Precision: 0.37 - Recall: 0.26811594202898553 - F1: 0.3109243697478992 Average Precision: 0.1668433646727741

Query: 78 Precision: 0.6 - Recall: 0.37037037037037035 - F1: 0.4580152671755725 Average Precision: 0.3853030453188929

Query: 79 Precision: 0.02 - Recall: 0.008620689655172414 - F1: 0.012048192771084338 Average Precision: 0.0013

Query: 80 Precision: 0.19 - Recall: 0.05080213903743316 - F1: 0.08016877637130802 Average Precision: 0.04914780607961647

Query: 81 Precision: 0.19 - Recall: 0.3064516129032258 - F1: 0.23456790123456792 Average Precision: 0.13217721698168614

Query: 82 Precision: 0.88 - Recall: 0.14691151919866444 - F1: 0.25178826895565093 Average Precision: 0.7961039218085947

Query: 83 Precision: 0.15 - Recall: 0.023734177215189875 - F1: 0.04098360655737705 Average Precision: 0.03422501795708256

Query: 84 Precision: 0.12 - Recall: 0.030379746835443037 - F1: 0.048484848484848485 Average Precision: 0.015873622515920078

Query: 85 Precision: 0.74 - Recall: 0.08277404921700224 - F1: 0.1488933601609658 Average Precision: 0.6072964264442896

Query: 86 Precision: 0.21 - Recall: 0.09859154929577464 - F1: 0.134185303514377 Average Precision: 0.05768460362419949

Query: 87 Precision: 0.07 - Recall: 0.03723404255319149 - F1: 0.04861111111111111 Average Precision: 0.007254861588380954

Query: 88 Precision: 0.03 - Recall: 0.01818181818181818 - F1: 0.022641509433962263 Average Precision: 0.0014748316199929103

Query: 89 Precision: 0.11 - Recall: 0.06321839080459771 - F1: 0.08029197080291971 Average Precision: 0.08076415302963291

Query: 90 Precision: 0.5 - Recall: 0.18796992481203006 - F1: 0.27322404371584696 Average Precision: 0.2970042420617517

TOTAL VALUES

Precision: 0.32575 - **Recall:** 0.098704643587607 **F1:** 0.17184688781854043 -**Mean Average Precision:** 0.21089909722332584

3.2 Performance values of MLE Model with 2-stage smoothing

Query: 51 Precision: 0.85 - Recall: 0.6159420289855072 - F1: 0.7142857142857143 Average Precision: 0.7650476610626161

Query: 52 Precision: 0.58 - Recall: 0.10841121495327102 - F1: 0.1826771653543307 Average Precision: 0.3505273486955493

Query: 53 Precision: 0.59 - Recall: 0.10332749562171628 - F1: 0.1758569299552906 Average Precision: 0.3350223140384919

Query: 54 Precision: 0.42 - Recall: 0.24561403508771928 - F1: 0.3099630996309963 Average Precision: 0.22520739250822275

Query: 55 Precision: 0.69 - Recall: 0.08518518518518518 - F1: 0.15164835164835164 Average Precision: 0.5032396152090693

Query: 56 Precision: 0.83 - Recall: 0.09453302961275627 - F1: 0.16973415132924338 Average Precision: 0.7129519416902302

Query: 57 Precision: 0.57 - Recall: 0.12364425162689804 - F1: 0.20320855614973263 Average Precision: 0.3932100386158303

Query: 58 Precision: 0.64 - Recall: 0.4025157232704403 - F1: 0.4942084942084943 Average Precision: 0.44929373382548216

Query: 59 Precision: 0.35 - Recall: 0.06044905008635579 - F1: 0.10309278350515463 Average Precision: 0.1403662581510984

Query: 60 Precision: 0.08 - Recall: 0.13333333333333333 - F1: 0.1 Average Precision: 0.0444802285528092

Query: 61 Precision: 0.66 - Recall: 0.32038834951456313 - F1: 0.4313725490196079 Average Precision: 0.5640599907956146

Query: 62 Precision: 0.33 - Recall: 0.1111111111111111 - F1: 0.16624685138539042 Average Precision: 0.1746772441689847

Query: 63 Precision: 0.2 - Recall: 0.09615384615384616 - F1: 0.12987012987012989 Average Precision: 0.05329207475308931

Query: 64 Precision: 0.15 - Recall: 0.04 - F1: 0.06315789473684211 Average Precision: 0.03790941527086509

Query: 65 Precision: 0.18 - Recall: 0.046632124352331605 - F1: 0.07407407407407407 Average Precision: 0.027215455622701494

Query: 66 Precision: 0.28 - Recall: 0.14213197969543148 - F1: 0.18855218855218858 Average Precision: 0.15655983867610979

Query: 67 Precision: 0.01 - Recall: 0.0018726591760299626 - F1: 0.003154574132492114 Average Precision: 2.2727272727272727E-4

Query: 68 Precision: 0.11 - Recall: 0.05641025641025641 - F1: 0.07457627118644067 Average Precision: 0.02951663365031152

Query: 69 Precision: 0.18 - Recall: 0.34615384615384615 - F1: 0.23684210526315788 Average Precision: 0.13217016245583968

Query: 70 Precision: 0.16 - Recall: 0.2909090909090909 - F1: 0.20645161290322578 Average Precision: 0.13183331755580166

Query: 71 Precision: 0.76 - Recall: 0.2 - F1: 0.3166666666666667 Average Precision: 0.6578158023517631

Query: 72 Precision: 0.06 - Recall: 0.05042016806722689 - F1: 0.0547945205479452 Average Precision: 0.006385709827697658

Query: 73 Precision: 0.01 - Recall: 0.00546448087431694 - F1: 0.007067137809187279 Average Precision: 1.4705882352941175E-4

Query: 74 Precision: 0.03 - Recall: 0.006012024048096192 - F1: 0.01001669449081803 Average Precision: 0.0045

Query: 75 Precision: 0.14 - Recall: 0.038356164383561646 - F1: 0.06021505376344086 Average Precision: 0.0313141288849357

Query: 76 Precision: 0.17 - Recall: 0.05782312925170068 - F1: 0.08629441624365482 Average Precision: 0.08884608118040757

Query: 77 Precision: 0.38 - Recall: 0.2753623188405797 - F1: 0.319327731092437 Average Precision: 0.16588301012262985

Query: 78 Precision: 0.58 - Recall: 0.35802469135802467 - F1: 0.44274809160305345 Average Precision: 0.36677670503501986

Query: 79 Precision: 0.01 - Recall: 0.004310344827586207 - F1: 0.006024096385542169 Average Precision: 0.0011111111111111111

Query: 80 Precision: 0.19 - Recall: 0.05080213903743316 - F1: 0.08016877637130802 Average Precision: 0.05901166249784437

Query: 81 Precision: 0.17 - Recall: 0.27419354838709675 - F1: 0.20987654320987656 Average Precision: 0.0928925025237254

Query: 82 Precision: 0.87 - Recall: 0.14524207011686144 - F1: 0.24892703862660948 Average Precision: 0.7892735081352352

Query: 83 Precision: 0.18 - Recall: 0.028481012658227847 - F1: 0.049180327868852465 Average Precision: 0.05305551216404572

Query: 84 Precision: 0.07 - Recall: 0.017721518987341773 - F1: 0.028282828282828285 Average Precision: 0.007212480805854301

Query: 85 Precision: 0.66 - Recall: 0.0738255033557047 - F1: 0.13279678068410464 Average Precision: 0.5390522394589317

Query: 86 Precision: 0.2 - Recall: 0.09389671361502347 - F1: 0.12779552715654954 Average Precision: 0.05044473362638858

Query: 87 Precision: 0.08 - Recall: 0.0425531914893617 - F1: 0.05555555555555556 Average Precision: 0.006845435434577033

Query: 88 Precision: 0.07 - Recall: 0.04242424242424243 - F1: 0.05283018867924529 Average Precision: 0.0054705228819152876

Query: 89 Precision: 0.11 - Recall: 0.06321839080459771 - F1: 0.08029197080291971 Average Precision: 0.07895958860083621

Query: 90 Precision: 0.52 - Recall: 0.19548872180451127 - F1: 0.28415300546448086 Average Precision: 0.31189912467965397

TOTAL VALUES Precision: 0.328 - **Recall:** 0.09938641012044543 **F1:** 0.17079966121239837 - **Mean Average Precision:** 0.2135926214043023

4 The project

4.1 Run the code

The project consists of a main class, *Retrieval.scala*, that contains the main method, and of tinyir library sources. We have added them to the project because we have modified some classes, which are therefore different from the original ones. The program can be launched with the command

```
scala Retrieval zippath
```

where *zippath* is the only command parameter and contains the path of the folder where the zips of the collection are located.

4.2 languageModel and trainingSet variables

The *Retrieval* object has 2 member variables that must be set to false or true in order to decide which model to use for the score of the queries and whether the queries that have to be processed are part of a training or of a test set. The explanation of the values to use for both variables is included in the code.

4.3 Log and result

During the execution the program prints to the console one line every 1000 processed documents. We introduced this counter to have a perception of the speed of the process and we found useful to leave it in the code. When the program finishes, in case of an execution on a training set, the performance values are displayed in the console. In both cases (training or test set), the rank of the first 100 documents for each query is saved in the file *result.txt*, created in the execution folder.