Machine Problem 2. Written by Christian Kümmerle.

Question 1: Implementation of different scoring functions

1 Boolean Dot Product

The implementation of the Boolean Dot Product is very simple, since every term that occurs in the query and the document is simply counted. The score function is:

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    return 1;
}
```

Resulting MAP: 0.20682027649769577.

2 Term Frequency-Inverse Document Frequency Dot Product (TFI-DF Dot Product)

The implementation of the unnormalized version of the TF-IDF dot product just uses for the terms $c(w, d)$, N and df the respective Lucene functions or Java values $termFreq$, $stats.getNumberOfDocuments()$ and $stats.getDocFreq()$.

```
protected float score(BasicStats stats, float termFreq, float docLength) {
   float score = (float) (termFreq*
   MathUtil.log(Math.E,(stats.getNumberOfDocuments()+1)/(stats.getDocFreq())));
   return score;
}
```

Resulting MAP: 0.1381063321385902.

3 Okapi BM25

For the implementation of Okapi BM25, I made use of the assumption that the query term frequency may be assumed as one. Because of that, I set the values of the query term frequency $c(w; q)$ to 1.

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    double k1 = 1.20;
    double k2 = 1000;
    double b = 0.75;
    float score = (float) (Math.log((stats.getNumberOfDocuments()-stats.getDocFreq()+0.5)/
    (stats.getDocFreq()+0.5))*((k1+1)*termFreq)/
    (k1*(1-b+b*docLength/stats.getAvgFieldLength())+termFreq)*(k2+1)*1/(k2+1));
    return score;
}
```

Resulting MAP: 0.26156255333674694.

4 Pivoted Length Normalization

As for Okapi BM25, we set the query term frequency to 1.

```
protected float score(BasicStats stats, float termFreq, float docLength) {
    double s = 0.13;
    float score = (float) ((1+Math.log(1+Math.log(termFreq)))/
    (1-s+s*docLength/stats.getAvgFieldLength())*1*
    Math.log((stats.getNumberOfDocuments()+1)/stats.getDocFreq()));
    return score;
}
```

Resulting MAP: 0.26977214541730665.

5 Language Model: Jelinek-Mercer

In order to implement the Jelinek-Mercer language model, I neglected the contribution of the term $|q| \log \alpha_d$, which is constant for a given query, because of

$$\alpha_d = \frac{1 - \sum_{w \in d} p_{seen}(w|d)}{\sum_{w \notin d} p(w|C)} = \frac{1 - \sum_{w \in d} \left[(1-\lambda) \frac{c(w,d)}{n} + \lambda p(w|C) \right]}{\sum_{w \notin d} p(w|C)} =$$

$$= \frac{\sum_w \lambda p(w|C) + \sum_w (1-\lambda) p(w|C) - \sum_{w \in d} \lambda p(w|C) - \sum_{w \in d} (1-\lambda) \frac{c(w,d)}{n}}{\sum_{w \notin d} p(w|C)}$$

$$= \frac{\sum_{w \notin d} \lambda p(w|C)}{\sum_{w \notin d} p(w|C)} + (1-\lambda) \frac{\overbrace{\sum_w p(w|C)}^{=1} - \frac{1}{n} \overbrace{\sum_{w \in d} c(w,d)}^{=n}}{\sum_{w \notin d} p(w|C)} = \lambda + (1-\lambda) \cdot 0 = \lambda,$$

which means that the term $|q| \log \alpha_d$ is not dependent on a certain document.

In our calculations, n denotes the length of document d and the maximum likelihood probability

$$p_{ml}(w|d)$$

was estimated as $\frac{c(w,d)}{n}$, exactly as it was done in the lecture.

The code of the score function looks as follows:

```
protected float score(BasicStats stats, float termFreq, float docLength) {
  float lambda = (float) 0.91;
  float score = float score = (float) (Math.log(((1-lambda)*termFreq/docLength +
  lambda*model.computeProbability(stats))/(lambda*model.computeProbability(stats))));
  return score;
}
```

Resulting MAP: 0.19298173749786665.

6 Language Model: Dirchlet Prior

First, I justify the value of α_d,

$$\alpha_d = \frac{\mu}{\mu + n},$$

which is now *document-dependent*:

$$\alpha_d = \frac{1 - \sum_{w \in d} p_{seen}(w|d)}{\sum_{w \notin d} p(w|C)} = \frac{1 - \frac{1}{\mu+n} \sum_{w \in d} [c(w,d) + \mu p(w|C)]}{\sum_{w \notin d} p(w|C)} =$$

$$= \frac{1}{\mu + n} \frac{(\mu + n) - \overbrace{\sum_{w \in d} c(w,d)}^{=n} - \sum_{w \in d} \mu p(w|C)}{\sum_{w \notin d} p(w|C)} = \frac{1}{\mu + n} \frac{\mu - \mu \sum_{w \in d} p(w|C)}{\sum_{w \notin d} p(w|C)} =$$

$$= \frac{1}{\mu + n} \frac{\mu \cdot (1 - \sum_{w \in d} p(w|C))}{\sum_{w \notin d} p(w|C)} = \frac{1}{\mu + n} \frac{\mu \cdot \sum_{w \notin d} p(w|C)}{\sum_{w \notin d} p(w|C)} = \frac{\mu}{\mu + n}.$$

As an approximation of the actual formula

$$r(q,d) = \sum_{w \in q,d} \log \frac{p_{seen}(w|d)}{\alpha_d p(w|C)} + |q| \log \alpha_d,$$

due to the structure of Lucene, I only counted the term $\log \alpha_d$ for each term that co-occurs in *the query and the document*, and not for all terms that occur in a query. This leads to the following implementation:

```
protected float score(BasicStats stats, float termFreq, float docLength) {
   float mu = (float) 19.5;
   float score = (float) (Math.log((termFreq+ mu*model.computeProbability(stats))/
   ((docLength+mu)*(mu/(mu+docLength))*model.computeProbability(stats)))
   +Math.log(mu/(mu+docLength)));
   return score;
}
```

Resulting MAP: 0.27570489844683393.

Question 2: Parameter tuning

For the Okapi BM25 model, I tried different values for the parameters k_1 and b inside of the permitted range $[1.2; 2]$ resp. $[0.75; 1.2]$. It turned out that the values

$$k_1 = 1.2$$

and

$$b = 0.75$$

lead to the highest MAP, namely $MAP = 0.26156255333674694$. The value of k_2 can be chosen arbitrarily, since the value $c(w; q)$ is always 1 by our simplifying assumption and there fore, the last factor

$$\frac{(k_2 + 1) \cdot c(w; q)}{k_2 + c(w; q)}$$

is always 1 and does not depend on k_2.

In the Dirichlet Prior language model, I found (by trying different values) that the parameter value

$$\mu = 19.5$$

leads to the best MAP value, namely $MAP = 0.27570489844683393$.

Question 3: Document analyzer

Disabling the lower case filter and/ or the length filter doesn't change the resulting MAP at all for the Okapi BM25. This is no surprise, since all the queries that are contained in our test procedure have only lower case letters and none of the terms in the queries has more than 35 letters.

docs are all in lower case as well!!

Disabling the stop word filter reduces the MAP by about 0.5 percentage points. This is not that much, but especially for queries with a lot of stop words and few "meaningful" query terms as "please supply information on the theory and use of parametric

very good analysis!!

amplifiers", we have a significant reduction of the performance of BM25 (for the mentioned query, the performance drops from 0.4716666666666667 to 0.30964285714285716, for example). This makes sense, since a stop word filter is more likely to be effective if the query contains a lot of stop words.

If you disable the stemmer in the document analyzer, there is a massive drop in the MAP, from about 0.262 to 0.087. This is can be easily explained: If we consider the query "synthesis of filters having high attenuation characteristics using a block diagram approach", we can observe that, for example, that the relevant document (in line 7304)

> filter design data for communication engineers a design procedure is outlined which permits rapid computation of the element values of the optimum zobel filter having a given attenuation characteristic

is not considered within the 10 most relevant documents after disabling the stemmer, whereas it appears at position 2 if we use the stemmer. This is probably due to the fact that the obvious semantical connection between the terms "filter" and "characteristic" in the document and "filters" resp. "characteristics" in the query cannot be made by the ranking function, if we do not use a stemmer.

As a conclusion, it can be observed that the "preprocessing" of the content of the queries and documents by removing stop words and stemming the terms increases the effectiveness of the ranking algorithm sharply. Here, the most important part of the preprocessing is the stemming.

thanks for your careful explanation, but it's partially correct: during indexing time, the docs are stemmed, so just turn on/off analyzer (w/ stem) could be totally wrong.

Question 4: Comparison between different ranking models

1 TF-IDF vs. Boolean dot product

The query "secondary emission of electrons by positive ion bombardment of the cathode" is one example where TF-IDF performed significantly better that the Boolean dot product model, resulting in an average precision of 0.3226984126984127 for TF-IDF and 0.1375 for the Boolean dot product. In order to analyze the reason for this, we have a look on the document on line 6569,

> method of investigating secondary emission in conductors bombarded by ions several oscillographic methods are discussed an the double modulation method is described in detail the ion beam is modulated by a rectangular pulse and the secondary emission electrode potential by a lf sawtooth pulse the principal advantage of this method is the . possibility of separating the true secondary ions from those produced by the heating due to the bombardment oscillograms of the current voltage characteristics of the secondary emission are shown for ta and w specimens bombarded by rb ions

very good analysis!!

This document is labeled as relevant for the query and TF-IDF ranks it indeed on position 1, while the Boolean dot product does not rank it within the 10 first positions at all. One important reason for that is the fact that one main term of the query, "bombardment", occurs multiple times in the document, indicating high relevance with respect to the query. The formula of the TF-IDF dot product takes this into account by counting the word multiple times, whereas the Boolean dot product does not do that. Additionally, "bombardment" respectively its stem is a term that probably doesn't occur in many documents, which means that the IDF contribution in the TF-IDF dot product assigns a higher value to the term. This is not the case for the Boolean dot product.

2 TF-IDF vs. Okapi BM25

A query where Okapi BM25 performs significantly better that the TF-IDF dot product is "systems of data coding for information transfer". The document on line 7985,

```
the use of a reflected code in digital control systems
```

ranks on position 3 with BM25, but is not in the top 10 for the TF-IDF dot product. In my point of view, the most important reason for that is the short document length: There are only two (stemmed) terms which co-occur in the query and the document, "coding/code" and "systems". But two matches for such a short document is a strong indicator for relevance, and this is well reflected in the formula of BM25, since there the denominator

$$k_1(1 - b + b\frac{n}{n_{avg}}) + c(w; d),$$

where n is the document length and n_{avg} the average document length, favors terms co-occurring in *short* documents and the query compared to those in long documents.

3 Okapi BM25 vs. Dirichlet Prior smoothed Language Model

The general performance of the two models is quite similar in average, scoring a MAP of 0.2616 for the Okapi BM25 model and a MAP of 0.2757 for the Dirichlet Prior smoothed language model. But there are visible differences in the performance for specific queries:

First, we compare the performance for both ranking models for the query "variable capacitance amplifiers": Here, the Okapi BM25 model scores an average precision of 0.6365 with 7 of 10 correctly recognized relevant documents, and the Dirichlet Prior model scores an average precision of 1.0 with 10 of 10 correctly recognized relevant documents. In order to analyze the reason for that, we examine the document (line 8835)

```
pulse shaping with variable capacitance diodes small signal and analysis is carried out
for variable capacitance integrating and differentiating circuits
```

.

It is ranked erroneously on position 4 with Okapi BM25 as it is not labeled as relevant, and not in the top 10 for the Dirichlet Prior language model. Apparently, BM25 values highly the double occurrence of the terms "variable" and "capacitance", but fails to reflect the missing of the (important) query term "amplifier(s)". If you examine the formula of BM25, it can be seen that the factor

$$\frac{(k_1+1)\cdot c(w;d)}{k_1\cdot(1-b+b\frac{n}{n_{avg}})+c(w;d)}$$

gives probably a too high weight for terms that occur in the query and multiple times in the document, especially for longer documents. This becomes clearer if we compare it with the formula of the Dirichlet Prior smoothed language model, where the count of the occurrences of the word w in the document d, $c(w;d)$, appears only logarithmically.

On the other hand, it is more difficult to find an easy explanation for the cases where BM25 performs better. Generally spoken, it can be observed that for those queries where the Dirichlet Prior model performs badly (say, it has an average precision of 0.2 or less), the BM25 model performs a little better.

good analysis, and it will be even better if you can propose something new based on your analysis and understanding of the paper.

Question 5: Problems of scoring functions and outlook

The scoring model Okapi BM25 performs particularly bad for queries that are longer, contain a lot of non-stop words and that contain a subset of words which transport the fundamental information, but also a some words whose meaning is not central for the query. For example, the query "the phenomenon of radiation caused by charged particles moving in varying electric and magnetic fields" results in an average precision of only 0.025.[1] Okapi BM25 ranks the document

> charged particle orbits in varying magnetic fields a magnetic field with time variation but with azimuthal symmetry about an axis is considered the movement of the charged particle is found by considering it to move nearly in a circle about a guiding centre whose motion can also be calculated

on position 1, which is not labeled as relevant. A reason for that is that the probably most important term in the query, "radiation", doesn't appear in the document, but this is not penalized since the algorithm has a hard time distinguishing important query terms from unimportant.

[1] It has to be noted that, for this query, all of the 6 tested ranking models don't perform very well. The maximal average precision was attained by the TF-IDF dot product, with a value of 0.2.

In the paper [1], an axiomatic approach to the problem of finding a "good" scoring function is taken. Several reasonable "axioms" that scoring functions should fulfill are mentioned, looking at the retrieval problem for given queries and documents from a term-to-term point of view. For example, it is assumed that, a one-term document $D = \{d\}$ and a one-term query $Q = \{q\}$ being given, a retrieval function should attain a strictly greater value if $q = d$ than if $q \neq d$. Because of the structure of the Okapi BM25 function, more precisely, because of the logarithm term

it would be better if you could relate this with your previous example query.

$$\ln \frac{N - df + 0.5}{df + 0.5}, \tag{1.1}$$

the value $f(q, q) = weight(q)$ (for the explanation of the notation, refer to the paper) can be negative[2], whereas $f(q, d) = 0$ if $q \neq d$. Therefore, Okapi BM25 doesn't fulfill the very reasonable axiom mentioned above. Fixing this contradiction should give rise to a better retrieval function, at least for the problematic cases as in the example above. This can be done by replacing the term (1.1) simply by an IDF term, giving rise to the Modified Okapi BM25 formula

$$S(q, d) = \sum_{w \in q, d} \ln \frac{N + 1}{df} \cdot \frac{(k_1 + 1)c(w; d)}{k_1(1 - b + b\frac{n}{n_{avg}}) + c(w; d)} \cdot \frac{(k_2 + 1)c(w; q)}{k_2 + c(w; q)}.$$

Table 3 of [1] gives some evidence that this modification indeed enhances the performance, at least for the bottom 25-percentile, which fits well to the theoretical explanation by the axiomatic approach.

Writing down the primitive weighting function, the query growth function and the document growth function for Okapi BM25 and generalizing some of of the appearing terms under the constraint that they still fulfill the axioms of the paper, paragraph 3.3.2 leads to a new scoring function called F2-EXP(s,k), $0 \leq s \leq 1$, $0 \leq k \leq 1$,

$$S(q, d) = \sum_{w \in q, d} c(w; q) \cdot \left(\frac{N}{df}\right)^k \cdot \frac{c(w; d)}{s + \frac{s \cdot n}{n_{avg}} + c(w; d)}.$$

For an optimal choice of the parameters s and k, this new model performs significantly better than Okapi BM25 and also than the modified Okapi BM25 for the bottom 25-percentile, on the data sets used in the paper, which can be seen in Table 3 of [1].

References

[1] H. Fang and C. Zhai. An exploration of axiomatic approaches to information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research*

[2] To be precise, this is the case if $df > \frac{N}{2}$.

and Development in Information Retrieval, SIGIR '05, pages 480–487, New York, NY, USA, 2005. ACM.