Information Retrieval

WS 2015 / 2016

Lecture 2, Tuesday October 27th, 2015
(Ranking, Evaluation)

Prof. Dr. Hannah Bast
Chair of Algorithms and Data Structures
Department of Computer Science
University of Freiburg
Overview of this lecture

- Organizational
  - Your experiences with ES1: Inverted index
  - Choice of language: Python, Java, C++

- Contents
  - Ranking: tf.idf and BM25
  - Evaluation: Ground truth, Precision, ...

- Exercise Sheet #2: implement BM25, tune your ranking, and then evaluate on a small benchmark

  There will be small competition (table on the Wiki)
Experiences with ES1 1/3

Summary / excerpts

- Nice and interesting exercise
- Easy for those with sufficient programming experience
- More work than expected for those out of practice

"First I thought the task is very easy and I'm finished soon, but then I got stuck in small coding details."

Don't worry, this will get much better over the course

- Getting used to Daphne and SVN took some time

But many of you are used to it already
Results

- Queries with only specific words work very well
  matrix, hobbit, edward scissorhands, ...

- Already one unspecific word often leads to bad results
  the matrix, tom hanks, lord of the rings, ...

- Several of you implemented cool additional features
  Highlighting, better ranking, command history (wow), autocompletion (wow wow wow wow), ...
Experiences with ES1  3/3

- Choice of programming language
  - Let me briefly repeat what I said in Lecture 1
    Python will be the least work
    Feel absolutely free to also use Java or C++, but be prepared for more work
    For ES1, we provided equivalent code for all three languages, to give you a working example for each
    We will not do that for future lectures, except:
    For lectures about efficiency, we will provide code in both Java and C++, but not in Python
Motivation

- Queries often return many hits
- Typically more than one wants to (or even can) look at
  
  For web search: often millions of documents
  
  But even for less hits a proper ranking is **key** to usability
- So we want to have the most "relevant" hits first
- **Problem:** how to measure what is how "relevant"
Basic Idea

- In the inverted lists, for each doc id also have a score

uni 17 0.5 , 53 0.2 , 97 0.3 , 127 0.8
freiburg 23 0.1 , 34 0.8 , 53 0.1 , 127 0.7

- While merging, aggregate the scores, then sort by score

MERGED 17 0.5 , 23 0.1 , 34 0.8 , 53 0.3 , 97 0.3 , 127 1.5
SORTED 127 1.5 , 34 0.8 , 17 0.5 , 53 0.3 , 97 0.3 , 23 0.1

- The entries in the list are referred to as postings

Above, it's only doc id and score, but a posting can also contain more information, e.g. the position of a word
Getting the top-k results

- A full sort of the result list takes time $\Theta(n \cdot \log n)$, where $n$ is the number of postings in the list.
- Typically only the top-$k$ hits need to be displayed.

Then a **partial sort** is sufficient: get the $k$ largest elements, for a given $k$.

Can be computed in time $\Theta(n + k \cdot \log k)$.

$k$ rounds of HeapSort yield time $\Theta(n + k \cdot \log n)$.

For constant $k$ these are both $\Theta(n)$.

For ES2, you can ignore this issue.
Meaningful scores

- How do we precompute good scores
  
  uni: 17 0.5, 53 0.2, 97 0.3, 127 0.8
  
  freiburg: 23 0.1, 34 0.8, 53 0.1, 127 0.7

- Goal: the score for the posting for doc $D_i$ in the inverted list for word $w$ should reflect the relevance of $w$ in $D_i$
  
  In particular, the larger the score, the more relevant

- Problem: relevance is somewhat subjective
  
  But it has to be done somehow anyway!
Term frequency ($tf$)

- The number of times a word occurs in a document

- **Problem:** some words are frequent in many documents, regardless of the content

<table>
<thead>
<tr>
<th>Word</th>
<th>57</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td>university</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>of</td>
<td>14</td>
<td>23</td>
</tr>
<tr>
<td>freiburg</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SCORE SUM</th>
<th>57</th>
<th>123</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>22</td>
<td>26</td>
</tr>
</tbody>
</table>

A word like "of" should not count much for relevance

Many of you observed that already, working on ES1
Document frequency (df)

- The number of documents containing a particular word

\[ df_{university} = 16.384, \ df_{of} = 524.288, \ df_{freiburg} = 1.024 \]

For simplicity, number are powers of 2, see below why

- Inverse document frequency (idf)

\[ idf = \log_2 \left( \frac{N}{df} \right) \quad N = \text{total number of documents} \]

For the example df scores above and \( N = 1,048,576 = 2^{20} \)

\[ idf_{university} = 6, \ idf_{of} = 1, \ idf_{freiburg} = 10 \]

Understand: without the \( \log_2 \), small differences in df would have too much of an effect; why exactly \( \log_2 \) → later slide
Combining the two (tf.idf)

- Reconsider our earlier tf only example

\[
\begin{align*}
\text{university} & : \ldots, 57 5, \ldots \ldots, 123 2, \ldots \\
\text{of} & : \ldots, 57 14, \ldots \ldots, 123 23, \ldots \\
\text{freiburg} & : \ldots, 57 3, \ldots \ldots, 123 1, \ldots \\
\text{SCORE SUM} & : \ldots, 57 22, \ldots \ldots, 123 26, \ldots 
\end{align*}
\]

- Now combined with idf scores from previous slide

\[
\begin{align*}
\text{university} & : \ldots, 57 30, \ldots \ldots, 123 12, \ldots \\
\text{of} & : \ldots, 57 14, \ldots \ldots, 123 23, \ldots \\
\text{freiburg} & : \ldots, 57 30, \ldots \ldots, 123 10, \ldots \\
\text{SCORE SUM} & : \ldots, 57 74, \ldots \ldots, 123 45, \ldots 
\end{align*}
\]
Problems with tf.idf in practice

- The idf part is fine, but the tf part has several problems
- Let $w$ be a word, and $D_1$ and $D_2$ be two documents
- **Problem 1:** assume that $D_1$ is longer than $D_2$
  Then tf for $w$ in $D_1$ tends to be larger than tf for $w$ in $D_2$, because $D_1$ is longer, not because it's more "about" $w$
- **Problem 2:** assume that $D_1$ and $D_2$ have the same length, and that the tf of $w$ in $D_1$ is twice the tf of $w$ in $D_2$
  Then it is reasonable to assume that $D_1$ is more "about" $w$ than $D_2$, but just a little more, and not twice more
The BM25 (best match) formula

- This tf.idf style formula has consistently outperformed other formulas in standard benchmarks over the years

**BM25 score** = \( tf^* \cdot \log_2 (N / df) \), where

\[
\begin{align*}
tf^* &= tf \cdot \frac{(k + 1)}{(k \cdot (1 - b + b \cdot \frac{DL}{AVDL}) + tf)} \quad (\equiv: K) \\
tf &= \text{term frequency, } DL = \text{document length, } AVDL = \text{average document length}
\end{align*}
\]

- Standard setting for BM25: \( k = 1.75 \) and \( b = 0.75 \)

  Binary: \( k = 0, \ b = 0 \); Normal tf.idf: \( k = \infty, \ b = 0 \) \( \Rightarrow \ K = 1 \)

\[
\begin{align*}
K^* &= K \cdot (0 + 1) / (0 + \ldots + K) \\
&= \frac{1}{K} \\
\end{align*}
\]

\[
\lim_\infty K^* = \frac{1}{e} \approx 0.3679
\]
Plausibility argument for BM25, part 1

- Start with the simple formula $tf \cdot idf$
- Replace $tf$ by $tf^*$ such that the following properties hold:
  - $tf^* = 0$ if and only if $tf = 0$
  - $tf^*$ increases as $tf$ increases
  - $tf^* \to$ some limit as $tf \to \infty$
- The simplest formula with these properties is
  $$tf^* = tf \cdot \frac{k + 1}{k + tf}$$
Plausibility argument for BM25, part 2

- So far, we have $tf^* = tf \cdot \frac{k + 1}{k + tf}$
- Normalize by the length of the document
  - full normalization: $\alpha = \frac{DL}{AVDL}$ ... too extreme
  - some normalization: $\alpha = (1 - b) + b \cdot \frac{DL}{AVDL}$
  - replace $tf$ by $tf / \alpha$
- This gives us $tf^* = \frac{tf}{\alpha} \cdot \frac{k + 1}{k + \frac{tf}{\alpha}}$
- And hence $tf^* = \frac{tf \cdot (k + 1)}{k \cdot \alpha + tf}$

Lots of "theory" behind this formula, but to me not really more convincing than these simple plausibility arguments
Implementation advice

- First compute the inverted lists with $tf$ scores
  
  You already did that (implicitly or explicitly) for ES1

- Along with that compute the document length (DL) for each document, and the average document length (AVDL)

- Make a second pass over the inverted lists and replace the $tf$ scores by $tf \cdot idf$ scores

$$tf \cdot (k + 1) / (k \cdot (1 - b + b \cdot DL / AVDL) + tf) \cdot \log_2 (N / df)$$

Note that the $df$ of a word is just the length (number of postings) in its inverted list
Further refinements

- For ES2, play around with the BM25 parameters $k$ and $b$
- Boost results that match each query word at least once

Warning: when you restrict your results to such matches, you might miss some relevant results

For example: steven spielberg movies

- Somehow take the popularity of a movie into account

In the file on the Wiki, movies are sorted by popularity

Popularity scores also have a Zipf distribution, so you might take $\sim N^{-\alpha}$ as popularity score for the N-th movie in the list

- Anything else that comes to your mind and might help ...
Advanced methods

- There is a multitude of other sources / approaches for improving the quality of the ranking, for example:

  Using query logs and click-through data

  Who searches what and clicked on what ... main pillar for the result quality of big search engines like Google

  Learning to rank

  Using machine learning (more about that in a later lecture) to find the best parameter setting
Ground truth

- For a given query, the ids of all documents relevant for that query

Query: matrix movies
Relevant: 10, 582, 877, 10003

- For ES2, we have built a ground truth for 10 queries

That was a lot of work, mostly Björn's, thanks!

Building a good and large enough ground truth is a common (and time-consuming) part in research in IR
Precision (P@k)

- The P@k for a given result list for a given query is the percentage of relevant documents among the top-k

Query: matrix movies
Relevant: 10, 582, 877, 10003
Result list: 582, 17, 5666, 10003, 10, 37, ...

P@1: 1/1 = 100%
P@2: 1/2 = 50%
P@3: 1/3 = 33%
P@4: 2/4 = 50%
P@5: 3/5 = 60%
Average Precision (AP)

- Let $R_1, \ldots, R_k$ be the sorted list of positions of the relevant document in the result list of a given query.

- Then AP is the average of the $k$ P@$R_i$ values.

Query: matrix movies
Relevant: 10, 582, 877, 10003
Result list: 582, 17, 5666, 10003, 10, ..., 877
$R_1, \ldots, R_4$: 1, 4, 5, 40
P@$R_1, \ldots, P@$R_4$: 100%, 50%, 60%, 10%
AP: 55%

Note: for docs not in result list, just take P@$R_i = 0
Mean Precisions (MP@k, MP@R, MAP)

- Given a benchmark with several queries + ground truth
- Then one can capture the quality of a system by taking the mean (average) of a given measure over all queries

\[ \text{MP@k} = \text{mean of the P@k values over all queries} \]

\[ \text{MP@R} = \text{mean of the P@R values over all queries} \]

\[ \text{MAP} = \text{mean of the AP values over all queries} \]

These are very common measures, which you will find in a lot of research papers.
Other measures

- There is a BIG variety of other evaluation measures, e.g.

- **nDCG** = normalized discounted cumulative growth
  Takes into account that documents can have varying degrees of relevance, e.g. primary and secondary
  Gives credit if primary is ranked before secondary

- **BPref** = binary relevance ... preference relation
  Takes into accounts that some documents are unjudged
  This is a frequent problem in benchmarks for huge text corpora, where complete judgment is impossible
  E.g. all relevant document for "tom hanks" on the web
Overfitting

- Tuning parameters / methods to achieve good results on a given benchmark is called **overfitting**

  In an extreme case: for each query from the benchmark, output the list of relevant docs from the ground truth

- In a realistic environment (real search engine or competition), one is given a **training** set for development

  The actual evaluation is on a **test** set, which must not be used / was not available during development

For ES2, do the development / tuning on some queries on your choice, then evaluate without further changes
References

- In the Manning/Raghavan/Schütze textbook
  - Section 6: Scoring and Term Weighting
  - Section 8: Evaluation in Information Retrieval

- Relevant Papers
  - Probabilistic Relevance: BM25 and Beyond  [FnTIR 2009]
  - Test Collection Based Evaluation of IR Systems  [FnTIR 2010]

- Relevant Wikipedia articles
    - #Performance_and_correctness_measures