Week 8 – Information Retrieval WS 15/16

Johannes Jasper(755292)
Norman Rzepka(754644)

December 9, 2015

Assignment 1: Result Fusion

Result fusion is the technique which combines rankings from dierent search engines, to enrich the result set for a given user query and improve its quality.

a) Why should one use result fusion to combine evidence? Which are the advantages of this approach?

Using result fusion one can combine the results of multiple search/ranking algorithms. This is useful if the search engines focus on specific aspects of the query and/or the document at hand. The search engines could also focus on different domains (e.g./ google + google scholar for research papers). The impact of each search engine can be tuned specifically to the users needs.

b) Can you give any example of such *metasearch* web engines?

Many metasearch engines exist for price comparison (such as opodo or skyscanner looking for cheap flights). Federated search also exists for groups of specific search interfaces such as libraries where books can be searched for across multiple libraries.

Assignment 2: Rank aggregation

Compute the *reciprocal rank fusion* score for the documents in the following setting. Use the RRF formula from the lecture slides.

- Consider a collection D of 5 documents (a, b, c, d and e) and 3 different retrieval models.
- The rankings for given query `q` for the three models are the following.
  ModelA : [a:0.4, c: 0.35, d: 0.31, b: 0.29, e: 0.15],
  ModelB : [a:0.5, b: 0.35, d: 0.25, c: 0.21],
  ModelC : [c:0.4, b: 0.3, d: 0.28, a: 0.27, e: 0.09]

- Compute r_i for each document. In case a document does not appear in a ranking, you should use a reasonably large constant instead. Compute the RRF scores for all documents and write down the final ranking.

We used $c = |D|$ as constant for documents that do no occur in the result list.

$$RRFscore(a) = \frac{1}{60+1} + \frac{1}{60+1} + \frac{1}{60+4} = 0.04841188524590164$$
$$RRFscore(b) = \frac{1}{60+4} + \frac{1}{60+2} + \frac{1}{60+2} = 0.04788306451612903$$
$$RRFscore(c) = \frac{1}{60+2} + \frac{1}{60+4} + \frac{1}{60+1} = 0.04814747488101534$$
$$RRFscore(d) = \frac{1}{60+3} + \frac{1}{60+3} + \frac{1}{60+3} = 0.047619047619047616$$
$$RRFscore(e) = \frac{1}{60+5} + \frac{1}{60+5} + \frac{1}{60+5} = 0.046153846153846156$$

Final ranking: [a: 0.04841, c: 0.04814, b: 0.04788, d: 0.04761, e: 0.04615]

Assignment 3: Inference Networks

A user issues the query: *New shops in York, UK*. How would a web search engine handle this query internally to retrieve relevant results? E.g. not finding only documents about New York. Hints:

- Handling stopwords, stemming, capitalization
- Weighting web page elements
- Automatically generating additional queries

As mentioned, in a preprocessing step, stop words such as *in* would be removed. Further on terms such as *UK* would be extended by synonymes like *United Kingdom* or *Great Britain*. This would reduce the impact of documents about shops in New York. Furthermore inference Networks can use various features, one of which could be collocation of query terms (i.e. differentiating between *New [...] York* and *New York*). Stemming would probably not be of great use in this example since neither *New* not *York* have stems. Also capitalization is problematic in the example because *New* is not the part of a name here, but it is capitalized nonetheless because it is the first word of the sentence. weighting the title and the body of a document differently might help, since such documents would most likely talk about York or shops in York, but hardly use the form *New shops in York*. Thus the confusion with those terms might be reduced. Pseudo relevance feedback might also help extending the query further with terms related to the topic such as names of famous shopping streets or names of popular shops themselves.

Assignment 4: (Programming)

We modified the ranking process in such a way, that query tokens that appear in the title of a document get a higher weight. We created two language models, one for the title, one for the entire document, and combined the resulting scores linearly with a ratio of 0.7 to 0.3. For the computation of our title language model query terms that appear in the title were counted multiple times (5 in our case).

For the results please check the attached files `original.txt` and `improved.txt`.