Department of Computer Science
The University of Melbourne
COMP90042 WEB SEARCH AND TEXT ANALYSIS (Semester 1, 2017)

Workshop exercises: Week 9

Discussion

1. Compare using a **term–document matrix** vs. an **inverted index**:
   (a) for resolving a **Boolean** query efficiently.
   (b) for resolving a **ranked** query efficiently.
2. Say we wished to resolve a ranked query over a document collection: we want a TF-IDF model which utilises the cosine similarity in the resulting vector space. What are the advantages and disadvantages, when applying this model using the following representation (in the inverted index):
   (a) Raw term–frequencies are recorded in the index;
   (b) TF-IDF weights are recorded in the index;
   (c) Documents are normalised to length 1, corresponding weights of term are recorded in the index?

Programming

1. Issue some queries using the small IR engine given in the iPython notebook `WSTA_N15_information_retrieval`. Read (some of) the documents that are returned: confirm that the keyword(s) is/are present, and adjudge whether you think these documents are relevant to your query.
2. What effect do the various preprocessing regimes have on the efficiency (time) and effectiveness (relevant results) of querying with the system (note: not building the index)? In particular, consider:
   (a) Stemming
   (b) Stopping
   (c) Tokenisation (e.g. of non-alphabetic tokens)
3. Extend the given IR engine to support disjunctive (OR) querying, and negation (NOT).

Catch-up

- What is **information retrieval**? What is an **information retrieval engine**?
- What is a **term–document matrix**? How is it different to an **inverted index**?
- What is **Boolean querying**? What is **ranked querying**?
- What does it mean for a document to be **relevant** to a query?
- What is a **vector space model**? How can we find **similarity** in a vector space?
- What is a **TF-IDF model**? What are some common examples of TF-IDF models?
- Confirm that you can find the **cosine similarity** between (the vectors which define) a document and a query. How do we use this value in ranked querying?

Get ahead

- For a collection of N documents, how large would you expect its inverted index to be? (Note that you will need to make some assumptions to estimate this.) What if the inverted index is also a **positional index**?
- In the ranked retrieval engine, try to alter the structure of the index according to the other values specified in Discussion Q2. Do you notice any differences in the query efficiency?