

## Homework #3 (due in 3 weeks – 3/12/09)

### Computing Document Similarity (100=10+40+10+20+20 points)

In the first two assignments, you processed a text corpus and created key data structures: a lexicon and an inverted file. For a static collection, such indexing can occur once, and then many different queries can be issued against the collection. The goal of this assignment is to have you demonstrate the ability to score and rank documents in order of their presumed relevance to queries. This time you will work with a small test collection based on Time Magazine articles from 1963. Note that this collection has different characteristics from the *CAESAR-POLO-ESAU* collection: the documents were not scanned and the error rate is low, words are up-cased, docids don't start at 1 and have gaps. There are about 400 documents that take up about 1.5 MB of text.

#### Query Parsing (10 points)

Along with the documents a set of queries for the TIME collection has been placed on the course web page. Download this file. Your retrieval engine program must process this file and create a representation for each query (e.g., a list of strings (terms) and weights (counts)). Show me the query vector your system uses for the first query (only).

#### Dot-Product Scoring (40 points)

A simple way to score documents against a query is to use the dot product measure. Use term frequency information from the query and documents to compute a score for each document that contains one or more of the query terms<sup>1</sup>, but don't use IDF-weighting. For example, suppose a query was “cool sledding locations” and a document was “the best sledding locations in howard county are found at public schools that have big hills, suitable for sledding”. The term ‘cool’ doesn't appear. The term ‘sledding’ occurs 2 times and ‘locations’ appears once. The dot product for this query and document would be  $(1*0 + 1*2 + 1*1) = 3$ . Rank documents according to this metric and produce an output file (see description below). In ranking documents, you may split ties in any fashion. **Note:** If you implement cosine scoring (below), you should not produce a dot product ranking; just supply the cosine-based ranked list.

#### Cosine Scoring (10 points)

Use TFxIDF term weighting for both documents and the query, and compute the cosine similarity measure for documents in the collection containing at least one of the query terms. Produce an output file that contains ranked documents for all topics, according to the formatting instructions below. If you do not want to attempt cosine scoring, it's only worth 10% of the assignment.

#### Batch Processing (20 points)

To put all these pieces together, your program should process an input query file and produce an output file that indicates the ranking of scored documents. For each query, list no more than 30 documents. Your output file must follow precise formatting instructions. For each query (list them in the same order as the input file) include documents in ranked order (ascending, with 1 being the first rank). Each line should have six columns, separated by a single space. The first column should be the query id (1, 2, ..., 83); the second should be the string “Q0” (read as 'Q zero'); the third should be the document id, the fourth should be the rank (1, 2, ... 30); the fifth should be your numerical score (real or integral); and, the last column should be your last name (no spaces). You do not have to include 30 documents for each query, but do not include more than 30. This file should be named *yourlastname-a.txt* and be emailed to me at paulmac@apl.jhu.edu; do not submit a hardcopy printout of this file.

#### Stemming Experiment (20 points)

To investigate whether stemming might improve query performance, I want you to create a second, separate index of the collection that uses different tokenization rules. You should truncate any term longer than 5 characters (for example, “America” becomes “ameri”; “americans” also becomes “ameri”). I call this 5-stemming. Produce an output file named *yourlastname-b.txt* and email it to me. If you like, you may opt-out of doing this stemming experiment – the penalty is only 20% of the assignment. There are publicly available stemming algorithms (e.g., Porter's Snowball stemmer). You may use a different stemmer if you are interested in doing so, but if you do, then describe it.

---

<sup>1</sup> You may remove stop words from your documents and/or query vectors. If you do this, indicate so, and rank documents that contain at least one of the remaining (useful) terms.