Louis Eisenberg, 10-22-04

CS 276A midterm review session notes

Postings → vector space

Say we have n docs and m terms in the lexicon.

For each term w_i:

compute idf, e.g. $idf_i = \log(n/df_i)$.

for each document d_j:

count $tf_{i,j}$ or compute wf, e.g. $wf_{i,j} = 0$ if $tf_{i,j} = 0$ or $1 + \log tf_{i,j}$ if $tf_{i,j} > 0$.

$w_{i,j} = tf_{i,j} * idf_i$ or $wf_{i,j} * idf_i$

Now each document is a vector d in m dimensions, where $d_i = w_{i,j}$.

Often want to normalize: divide each component by the vector's length, e.g. the L_2 norm:

$$d_i = \frac{w_{i,j}}{\sqrt{\sum_{k=1}^{m} d_k^2}}$$

We can do the same with queries (treating them as small documents).

Matching queries to docs (or docs to docs): cosine similarity. For normalized vectors, just the dot product.

Recall, precision, F measure

Precision: % of retrieved docs that are relevant
Recall: % of relevant docs that are retrieved

How do you maximize precision?

How do you maximize recall?

If a system is doing a decent job of ranking its results, you would generally expect precision to decrease as the number of docs retrieved increases. Recall can only go up as numDocs increases.

Interpolated precision: $precision_i = \max precision_k$ such that $k \geq i$, where $precision_i$ means the precision when i docs are retrieved.

F measure combines precision and recall.

F = $\frac{(\beta^2+1)PR}{B^2P+R}$. Balanced F is just $F_1 = \frac{PR}{P+R}$.

So the numDocs for which you have peak F measure value probably represents a compromise between precision and recall – you increase numDocs to increase recall, but at the expense of precision.

Random projection

Reduce vector space from m dimensions down to k.
Not choosing k terms randomly and eliminating all the other axes – rather, choosing k random directions (i.e. linear combinations of the m axes) that are all orthogonal to each other.

kxm projection matrix R, mxn term-doc matrix A, random projection W = RxA.

Other topics to think about

- types of indices, their strengths and weaknesses, what they can and can't do:
  basic inverted, n-word, n-gram, positional, permuterm
- data structures for indices: trees (binary trees, B-trees) vs. hashtables
- linear zone combinations
- spell correction (edit distance, context-sensitive, n-grams, soundex…)
- index mergesort
- Zipf's law and block estimates for postings lists
- consequences of stemming – potential benefits and drawbacks