Lecture 3: Information Retrieval

William Webber (`william@williamwebber.com`)

COMP90042, 2014, Semester 1, Lecture 3

What we'll learn today

- How to take a user query and return a ranked list of results
- How implement this operation in a reasonably efficient way
- How to automatically expand the query by adding synonyms and related words

Reviewing: document similarity in VSM

- Document is BOW
- Project into term space as vector, with dimension lengths given by TF*IDF
- Calculate document similarity as cosine of angle between their vectors
- Implement as dot product on unit-length vectors

Same process can be used to *rank* documents by decreasing similarity to given document.

Query processing in VSM

- Treat the query as a (short) (pseudo-)document
- Calculate (VSM cosine) similarity between query pseudo-document and each document in collection
- Rank documents by decreasing similarity with query
- Return to user in rank order (generally only top results initially)

Index

- In last week's worksheet, every time we ran a similarity computation, we recalculated unit-length TF*IDF vectors for all documents.
- Since these do not change from query to query, save processing by precalculating and store results in an index.
- But we still need to iterate through all documents to rank by similarity.
- This an $O(|D|)$ operation.

Term-wise processing

- In document similarity, only terms occurring in both documents contribute to cosine score (remember the dot-product!)
- In query processing by pseudo-document model, therefore, only documents that contain query terms need to be considered (which makes intuitive sense)
- Complexity reduced to $O(\max \mathrm{df}_t)$
  - Note: because Zipfian distribution, most frequent term dominates.
  - Very good reason to drop stop-words!
- Need an index that supports quickly finding which documents a term occurs in

Inverted index

Index designed to support query processing:

- Keys are terms
- Values are lists of $\langle d, w_{t,d} \rangle$ pairs
- Each $\langle d, w_{t_d} \rangle$ pair called a *posting*
- List of these called a *postings list*

Term		Postings list
tea	$\rightarrow$	1:1.4 ; 3:1.0 ; 6:1.7 ; ...
two	$\rightarrow$	2:2.3 ; 3:1.0 ; 4:1.7 ; ...
me	$\rightarrow$	1:1.0 ; 2:1.4 ; ...

Query processing on inverted index

- For each term t in query:
  - Load postings list for t
  - For each posting $\langle d, w_{t_d} \rangle$ in list:
    - $a_d \mathrel{+}= w_{t_d}$ [1]
- Sort documents by decreasing a_d
- Return sorted results to user

NOTE: there are a lot of efficiency optimizations that we won't go into here!

[1] a_d is called an "accumulator"

Tweaking the formula

- Previous algorithm does not precisely calculate cosine distance between pseudo-document and documents, as:
  - IDF
  - $\log(1 + f_{q,t})$
  - Unit-length normalization

  not applied
- Unit-length normalization doesn't matter to query processing (why not?), but other can
- In fact, many of formula component choices made here (e.g. $\text{TF} = \log(f_{d,t} + 1)$ vs. $\text{TF} = f_{d,t}$) are heuristic (as is the VSM model itself)
  - Zobel and Moffat, "Exploring the Similarity Space" (1998) identify $(8 \times 9 \times 2 \times 6 \times 14) = 12096$ possible different combinations of choices
- Once can try different variants to improve effectiveness
- (We'll talk next lecture about how to test success)

Alternative document length normalization

- To date, normalized document vectors to unit length
- But is this correct?
  - Very short documents will get high scores for term occurrences
  - Long documents may cover many topics, satisfy many queries

Empirical adjustment

Assume that we have:

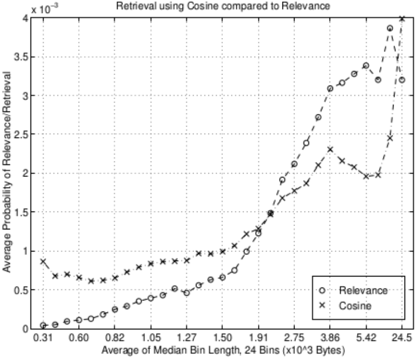
- Large number of queries
- Judgments of which documents are relevant to which queries

Then we can compare:

- Probability of document being retrieved given length
- Probability of document being relevant given length

and adjust if these two probabilities are out of line

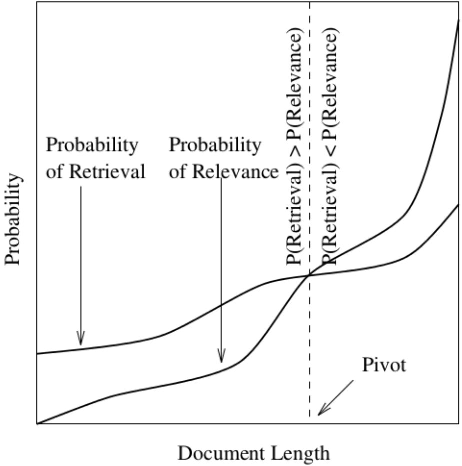
Probability retrieved v. relevant given length



- Look at mean empirical relation

Amit Singhal, Chris Buckley, and Mandar Mitra, "Pivoted Document Length Normalization", SIGIR 1996

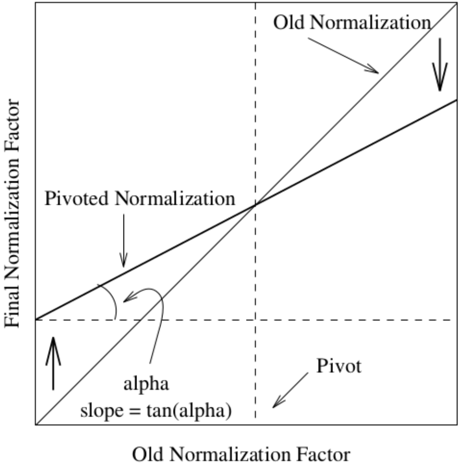
Probability retrieved v. relevant given length



- Look at mean empirical relation
- Simplify and identify "pivot". Lengths greater than pivot point should be boosted; less, decreased

Amit Singhal, Chris Buckley, and Mandar Mitra, "Pivoted Document Length Normalization", SIGIR 1996

Probability retrieved v. relevant given length



- Look at mean empirical relation
- Simplify and identify "pivot". Lengths greater than pivot point should be boosted; less, decreased
- Linearly approximate to "slope"

Amit Singhal, Chris Buckley, and Mandar Mitra, "Pivoted Document Length Normalization", SIGIR 1996

Pivoted document length normalization

- w weight of term in document (e.g. TF*IDF)
- n_u original normalization (e.g. unit-length normalization by length of document vector)
- p pivot point for pivot normalization
- s slope of pivot normalization
- w_p pivot-normalized weight of term

$$w_p = \frac{w}{(1.0 - s) \cdot p + s \cdot n_u} \tag{1}$$

- Various approximations and factors (see Singhal et al.)
- Note that we are no longer calculating cosine distance, but pseudo-cosine distance!
- Require dataset to tune on, and will be tuned to that dataset
- Gives significant improvement in effectiveness

Looking back and forward

Back

- Queries can be processing in VSM by treating query as (pseudo-)document
- Inverted index supports efficient query processing
- Various tweaks to VSM formulae possible, of which pivoted document length normalization empirically the most effective

Looking back and forward

Forward

- Queries short, possible ambiguous; can be expanded by finding similar terms (next lecture)
- In following lecture, will look at evaluation of IR methods, for selecting methods and tuning parameters
- Later, we will look at probabilistic methods, that present themselves as more theoretically grounded, requiring fewer heuristic "hacks"

Further reading

- Chapter 2, "The term vocabulary and postings lists"[2], from Section 2 onwards, of Manning, Raghavan, and Schutze, *Introduction to Information Retrieval* (more advanced methods for postings lists)
- Justin Zobel and Alistair Moffat, "Inverted files for text search engines"[3], ACM Computing Surveys, 2006 (authoritative survey paper on inverted indexes by pioneers in their optimization, who happen to be UniMelb professors)
- Singhal, Buckley, and Mitra, "Pivoted document length normalization"[4], SIGIR, 1996 (introduced pivoted DLN; first author is now head of search engineering at Google)

[2] http://nlp.stanford.edu/IR-book/pdf/02voc.pdf
[3] http://www.cs.mu.oz.au/~jz/fulltext/compsurv06.pdf
[4] http://singhal.info/pivoted-dln.pdf