Lecture 4: Query Expansion

William Webber (william@williamwebber.com)

COMP90042, 2014, Semester 1, Lecture 4

What we'll learn today

- How to find "similar" terms to a given term in a collection
- How to expand a query to overcome the ambiguity in human language

Query narrowness and ambiguity

"motorbike"	Will miss references to "motorcycles"
"java"	Island, coffee, or programming language?
"vehicle theft"	Motorbike, car, truck theft?

Inexactness of human language

Human language is inexact:

- **synonym** Different words, one concept ("cab" vs "taxi")
- **homonym** One word, different concepts ("jaguar" car, animal)
- **hyponym** Generalization ("cat" $\rightarrow$ "animal")
- **hypernym** Specialization ("athlete" $\rightarrow$ "sprinter")
- **meronym** Part of whole ("car" $\rightarrow$ "wheel")
- **holonym** Whole for part ("Germany" $\rightarrow$ "Europe")

Also mispellings, foreign languages, slang etc..

Clarifying queries: possibilities

- Suggest additional or clarifying terms to user
  - [java] $\rightarrow$ ([java indonesia] | [java coffee] | [java programming])?
  - Often done by finding clarifying co-occurring terms or phrases
- Add synonyms and other -nyms directly to query:
  - [cat] $\rightarrow$ [cat feline jaguar animal puss ...]
- Add associated non-nyms to help weight results:
  - [olympics] $\rightarrow$ [medal record sochi champion torch ...]
- Allow user to "explore the term space", discover vocabulary of collection

Clarifying queries: manual thesaurus

Could use external, curated thesaurus (Roget's, WordNet 3.1)

car $\rightarrow$ vehicle; automobile, truck, trailer, bus, taxi ...

java $\rightarrow$ coffee; chicken; island

crimea $\rightarrow$???

- Reasonable for generic concept words
- Quickly outdated; poor for names; poor for associated words
- Expensive to maintain (huge effort for Wordnet, now obsolete)

(If you're going down this route, use Wikipedia!)

Automatic thesaurus

- Build an automatic thesaurus by finding "similar" terms in collection
- Term similarity can be defined analogously to document similarity using the Term-Document Matrix:

  Document similarity Two documents are similar if they are close to each other in term space

  Term similarity Two terms are similar if they are close to each other in document space

Question

What does it mean for two terms to be "near" each other in document space?

Transformations for term frequency calculations

- What is the equivalent of "inverse document frequency"? Is it a useful transformation?
- What is the equivalent of "document-length normalization"? Do we want to do this?

Unit-normalized term similarity formula

$f_{d,t}$ frequency of term t in document d

D set of documents

$$
\begin{aligned}
n_t &= \Big(\sum_{d \in D} f_{d,t}^2\Big)^{1/2} && (1) \\
w_{d,t} &= \frac{f_{d,t}}{n_t} && (2) \\
\mathrm{sim}_u(t_1, t_2) &= \sum_{d \in D} w_{d,t_1} \cdot w_{d,t_2} && (3)
\end{aligned}
$$

- Calculate distance between terms as cosine
- With unit-normalized vectors

Unit-normalized (cosine) distance

Term	"Similar" terms
socc	*tabulat*, match, cup, goal, club, play, leag, halftim, goalkeep, *internazional*, divid, draw, scor, stand, *turnstil* ...
jaguar	seahawk, rotons, precip, luckey, touchdown, dolphin, quarterback, redskin, harbaugh, chevrolet, porsch, xk8, throwaway, terrel ...
najibullah	lafrai, murtaz, ivgin, seh, darulam, tajik, kart, arghand, sarmad, mikhailov, tajikist, rocket, afgh, frontlin, invit ...

- Tends to through up very rare suggestions, especially for rare terms
- Why?

LYRL 30k collection

Normalized cosine term similarity

Term	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	n_t
	Document ($f_{t,d}$)										
ivgin	0	1	0	0	0	0	0	0	0	0	1.0
najibullah	0	2	0	1	1	0	1	0	1	1	3.0
afghanist	2	0	1	1	0	1	1	0	0	1	3.0

Term	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
	Document ($w_{t,d}$)									
ivgin	0	1	0	0	0	0	0	0	0	0
najibullah	0	2/3	0	1/3	1/3	0	1/3	0	1/3	1/3
afghanist	2/3	0	1/3	1/3	0	1/3	1/3	0	0	1/3

$$\mathrm{sim}_u(\mathrm{najibullah}, \mathrm{ivgin}) = 0.66 \tag{4}$$

$$\mathrm{sim}_u(\mathrm{najibullah}, \mathrm{afghanist}) = 0.33^1 \tag{5}$$

- Length norm places heavy weight on singleton occurrences
- Why is this not (so bad) a problem with documents?

[1] Incorrectly 0.44 in original

Term similarity: raw frequencies

- "Try" working with raw frequencies, instead of normalized ones
- Note: though the computation is similar (dot product), we are not calculated cosine or any direct geometric distance
- (Anyone know the geometric interpretation of the dot product of two unnormalized vectors?)

Term similarity: frequency formula

$t_1, \ t_2$	The terms we wish to compare
$f_{d,t}$	Number of occurences of term t in document d
D	Set of all documents in collection

$$\mathrm{sim}_f(t_1, t_2) = \sum_{d \in D} f_{d,t_1} \cdot f_{d,t_2} \tag{6}$$

Implementation note

- Only need to consider documents that both terms occur in.
- Can be computed on inverted index postings list
- Finding "most similar" term requires traversing full vocabulary

Question

What types of terms are we biasing towards by using "raw" TF scores?

Term similarities with frequency formulae

Term	"Similar" terms
socc	play, match, goal, cup, leagu, club, scor, divid, minut, result, game, *year*, win, team, champ ...
jaguar	car, *percent*, sale, *year*, yard, touchdown, quart, motor, vehicl, unit, britain, pass, august, *million*, market ...
najibullah	govern, taleb, kabul, afgh, minist, rocket, foreign, tajik, kill, invit, radio, islam, fight, confer, afghanist ...

- Can throw up words that are globally frequent, but not topical
- More tweaking needs to be done ...

What is "similar"

Term	"Similar" terms
socc	play, match, goal, cup, leagu, club, scor, divid, minut, result, game, *year*, win, team, champ ...

- What sort of "similar" terms are being found? And not found?

What is "similar"

Term	"Similar" terms
socc	play, match, goal, cup, leagu, club, scor, divid, minut, result, game, *year*, win, team, champ ...

- What sort of "similar" terms are being found? And not found?
- Obvious synonym of "soccer" not found
- Why is this "similarity" bad at finding synonyms?

What is “similar”

Term	“Similar” terms
socc	play, match, goal, cup, leagu, club, scor, divid, minut, result, game, *year*, win, team, champ . . .

- What sort of “similar” terms are being found? And not found?
- Obvious synonym of “soccer” not found
- Why is this “similarity” bad at finding synonyms?
- Because synonyms rarely appear in same document (why?)
- Will expanding this way still help find documents with synonyms?

What is "similar"

Term	"Similar" terms
socc	play, match, goal, cup, leagu, club, scor, divid, minut, result, game, *year*, win, team, champ ...

- What sort of "similar" terms are being found? And not found?
- Obvious synonym of "soccer" not found
- Why is this "similarity" bad at finding synonyms?
- Because synonyms rarely appear in same document (why?)
- Will expanding this way still help find documents with synonyms?
- Yes, because co-occurring words will tend to occur with synonym

Individually expanding query terms

- Say query is [swing buttons]
- We might add [slide playground child kindergarten] for "swing"
- We might add [sewing repair shirt trouser] for "buttons"
- Would query [swing buttons slide playground child kindergarten sewing repair shirt trouser] help user find what they want?

Individually expanding query terms

- Say query is [swing buttons]
- We might add [slide playground child kindergarten] for "swing"
- We might add [sewing repair shirt trouser] for "buttons"
- Would query [swing buttons slide playground child kindergarten sewing repair shirt trouser] help user find what they want?
- Expanding terms independently, irrespective of their joint connotation, is dangerous!

Local expansion through automatic feedback

- How do we find co-occurring terms in important documents that query terms co-occur in?
- Well, query processing itself finds (hopefully) important documents that query terms co-occur in
- So we can look in the query results themselves for expansion terms
- This known as "pseudo-relevance feedback" (PRF)
  - In "true relevance feedback", the user marks retrieved documents as relevant or irrelevant
  - Terms in relevant documents get positive weight, in irrelevant negative
  - This akin to text classification (which we'll talk about later)
  - PRF is "pseudo" because we "assume all results are relevant"

Query expansion through automatic feedback

- Run original query against index
- Take top-ranking result documents
- Extract (weighted) terms from results and add them to query
  - (or enhance the query pseudo-document vector)
- Run expanded query against index
- Return results to user

Several algorithms for doing this; we'll look at one from 1970 (!)

Rocchio's algorithm for PRF

$$d_e = \alpha q_0 + \beta \frac{1}{|D_r|} \sum_{d_i \in D_r} d_i \qquad (7)$$

q_0	Original query vector
D_r	Set of result documents
α, β	Weights
q_e	Expanded query vector

- α, β set by "intuition"
- ...or tuned by experimentation

Rocchio's PRF algorithm illustrated

	Document ($w_{t,d}$)					
(Ps-)doc	"taxi"	"cab"	"hail"	"tea"	"two"	Total
d1	0.7	0.0	0.7	0.0	0.0	
d2	0.0	0.7	0.7	0.0	0.0	
d3	0.05	0.0	0.0	0.65	0.7	
(qry)	1.0	0.0	0.0	0.0	0.0	
$q \bullet$ d3	0.05	0.0	0.0	0.0	0.0	0.05
(exp)	0.85	0.0	0.35	0.0	0.0	
$e \bullet$ d3	0.04	0.0	0.0	0.0	0.0	0.04
$e \bullet$ d2	0.0	0.0	0.25	0.0	0.0	0.25

- Query [taxi]
- Result ranking: $\langle$ d1, d3, d2 $\rangle$
- Expand with top result, $\alpha = \beta = 0.5$
- Submit expanded query
- Result ranking: $\langle$ d1, d2, d3 $\rangle$

Query expansion in practice

- Suggestion / expansion by raw term similarity not widely used
  - Latent Semantic Analysis a preferred method (see later)
  - Co-occurring noun phrases can give better suggestions
- Pseudo-relevance feedback:
  - Gives moderate average gain (but makes some queries worse)
  - Quite expensive (involves processing large expanded queries)
  - Cost–benefit tradeoff not justified for web-scale search
- Query suggestion actually done by search log mining:
  - See how people reformulate queries
  - . . . and suggest these reformulations to others
  - (Also how spelling correction is done)
  - (Hopefully, will have time to look at automatic user-feedback methods later)

Looking back and forward

Back

- Queries processed in VSM by treating query as (pseudo-)document
- Inverted index for efficient processing
- Tweaks to VSM formulae, including pivoted document length normalization
- Query expansion:
  - Global, by looking at co-occurring terms throughout collection
  - Local, by looking for terms in query results
- Rocchio's algorithm (PRF) by adding result document vectors to query vector, resubmitting

Looking back and forward

Forward

- A lot of heuristic alternatives introduced here.
- How do we know which one to pick?
- In next 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”
- Pseudo-relevance feedback generalizes to true relevance feedback, which is a form of text classification, to be looked at in a couple of weeks.

Further reading

- Chapter 9, "Relevance feedback and query expansion"[2], of Manning, Raghavan, and Schutze, *Introduction to Information Retrieval* (on query expansion, also discusses semi-curated methods using thesauri)

[2] http://nlp.stanford.edu/IR-book/pdf/09expand.pdf