Lecture 8: Query Expansion and Relevance Feedback

Trevor Cohn (tcohn@unimelb.edu.au)
Slide credits: William Webber

COMP90042, 2015, Semester 1
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
- Learning from relevance feedback
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

Two broad types of method for refining the query

**Global methods** reformulate query terms independent of the query results

- using a thesaurus or WordNet
- via automatic thesaurus generation

**Local methods** reformulate query based on initial results for the query

- using relevance feedback
- using pseudo relevance feedback
- using indirect feedback, e.g., click-through data
Global methods: possibilities

- Suggest additional or clarifying terms to user
  - [java] → ([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] → [cat feline jaguar animal puss . . . ]
- Add associated terms (automatic thesaurus) to help weight results:
  - [olympics] → [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 → vehicle; automobile, truck, trailer, bus, taxi . . .
- java → coffee; island; programming language
- carlton → ???

- 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- Quickly outdated; poor for names; poor for associated words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

\[
n_t = \sqrt{\sum_{d \in D} f_{d,t}^2}
\]

\[
w_{d,t} = \frac{f_{d,t}}{n_t}
\]

\[
sim_u(t_1, t_2) = \sum_{d \in D} w_{d,t_1} \cdot w_{d,t_2}
\]

- Calculate similarity between terms using cosine
- With unit-normalized vectors
## Unit-normalized (cosine) distance

<table>
<thead>
<tr>
<th>Term</th>
<th>“Similar” terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>jaguar</td>
<td>seahawk, rotons, precip, luckey, touchdown, dolphin, quarterback, redskin, harbaugh, chevrolet, porsch, xk8, throwaway, terrel . . .</td>
</tr>
<tr>
<td>najibullah</td>
<td>lafrai, murtaz, ivgin, seh, darulam, tajik, kart, arghand, sarmad, mikhailov, tajikist, rocket, afgh, frontlin, invit . . .</td>
</tr>
</tbody>
</table>

- Tends to through up very rare suggestions, especially for rare terms
- Why?
## Normalized cosine term similarity

<table>
<thead>
<tr>
<th>Term</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
<th>d7</th>
<th>d8</th>
<th>d9</th>
<th>d10</th>
<th>n_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>ivgin</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>najibullah</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3.0</td>
</tr>
<tr>
<td>afghanist</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3.0</td>
</tr>
</tbody>
</table>

\[ \text{Document } (f_{t,d}) \]

<table>
<thead>
<tr>
<th>Term</th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
<th>d7</th>
<th>d8</th>
<th>d9</th>
<th>d10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ivgin</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>najibullah</td>
<td>0</td>
<td>2/3</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>0</td>
<td>1/3</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>afghanist</td>
<td>2/3</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
</tr>
</tbody>
</table>

\[ \text{Document } (w_{t,d}) \]

\[ \text{sim}_u(najibullah, ivgin) = 0.66 \quad (1) \]
\[ \text{sim}_u(najibullah, afghanist) = 0.33 \quad (2) \]

- Length norm places heavy weight on singleton occurrences
- Why is this not (so bad) a problem with documents?
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 occurrences of term \( t \) in document \( d \)

\( D \) Set of all documents in collection

\[
\text{sim}_f(t_1, t_2) = \sum_{d \in D} f_{d,t_1} \cdot f_{d,t_2}
\]  \hspace{1cm} (3)

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

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

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

<table>
<thead>
<tr>
<th>Term</th>
<th>“Similar” terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>socc</td>
<td>play, match, goal, cup, leagu, club, scor, divid, minut, result, game, year, win, team, champ . . .</td>
</tr>
</tbody>
</table>

What sort of “similar” terms are being found? And not found?
What is “similar”

<table>
<thead>
<tr>
<th>Term</th>
<th>“Similar” terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>socc</td>
<td>play, match, goal, cup, leagu, club, scor, divid, minut, result, game, year, win, team, champ ...</td>
</tr>
</tbody>
</table>

- 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”

<table>
<thead>
<tr>
<th>Term</th>
<th>“Similar” terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>socc</td>
<td>play, match, goal, cup, leagu, club, scor, divid, minut, result, game, year, win, team, champ . . .</td>
</tr>
</tbody>
</table>

- 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”

<table>
<thead>
<tr>
<th>Term</th>
<th>“Similar” terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>socc</td>
<td>play, match, goal, cup, leagu, club, scor, divid, minut, result, game, year, win, team, champ</td>
</tr>
</tbody>
</table>

- 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
An Alternative “Distributional Representation”

Instead of terms and documents in which they occur

▶ use the local context in which the terms occur
▶ (no longer bag-of-words)

E.g., for soccer

| ...at the world cup | soccer | tournament ... |
| ...right before the world cup | soccer | tournament was about to ... |
| ...the creative inborn talent for the soccer team | held matchless superiority |
| ...germany ’s group a league, bayer leverkusen ... |

What other words occur in similar contexts?
Other team sport terms, such as cricket, rugby, ...
Distributional Representations

Use of local context typically limited to a 1 – 3 words to either side.

- each column of matrix encodes word-word cooccurrence counts
- use cosine similarity between word vectors
- captures both semantic similarity and syntactic similarity

<table>
<thead>
<tr>
<th></th>
<th>cos(t, france)</th>
</tr>
</thead>
<tbody>
<tr>
<td>spain</td>
<td>0.678515</td>
</tr>
<tr>
<td>belgium</td>
<td>0.665923</td>
</tr>
<tr>
<td>netherlands</td>
<td>0.652428</td>
</tr>
<tr>
<td>italy</td>
<td>0.633130</td>
</tr>
</tbody>
</table>

Mikolov et al. 2013, Distributed Representations of Words and Phrases and their Compositionality. NIPS.  
https://code.google.com/p/word2vec/
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!
- Generally helps to increase recall, but at the expense of precision
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 top results are relevant
Query expansion through relevance feedback

- Run original query against index
- Take top-ranking result documents
  - PseudoRF assume all these documents are relevant
  - RF user input to annotate which documents are relevant cf. non-relevant
- Extract (weighted) terms from results and add (subtract) them to query
  - 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 relevance feedback

\[ q_e = \alpha q_0 + \beta \frac{1}{|D_r|} \sum_{d_i \in D_r} d_i - \gamma \frac{1}{|D_{nr}|} \sum_{d_i \in D_{nr}} d_i \]

- \( q_0 \): Original query vector
- \( D_r \): Set of relevant result documents (top results in PRF)
- \( D_{nr} \): Set of non-relevant result documents (empty in PRF)
- \( \alpha, \beta, \gamma \): Weights
- \( q_e \): Expanded query vector

- \( \alpha, \beta, \gamma \) set by “intuition”
- …or tuned by experimentation
### Rocchio’s PRF algorithm illustrated

<table>
<thead>
<tr>
<th>(Ps-)doc</th>
<th>“taxi”</th>
<th>“cab”</th>
<th>“hail”</th>
<th>“tea”</th>
<th>“two”</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>0.7</td>
<td>0.0</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
</tr>
<tr>
<td>d2</td>
<td>0.0</td>
<td>0.7</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
</tr>
<tr>
<td>d3</td>
<td>0.05</td>
<td>0.0</td>
<td>0.0</td>
<td>0.65</td>
<td>0.7</td>
<td>0.7</td>
</tr>
<tr>
<td>(qry)</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>q • d1</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.7</td>
</tr>
<tr>
<td>q • d2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>q • d3</td>
<td>0.05</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.05</td>
</tr>
<tr>
<td>(exp)</td>
<td>0.85</td>
<td>0.0</td>
<td>0.35</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>qe • d1</td>
<td>0.595</td>
<td>0.0</td>
<td>0.25</td>
<td>0.0</td>
<td>0.0</td>
<td>0.845</td>
</tr>
<tr>
<td>qe • d2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.25</td>
<td>0.0</td>
<td>0.0</td>
<td>0.25</td>
</tr>
<tr>
<td>qe • d3</td>
<td>0.04</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.04</td>
</tr>
</tbody>
</table>

- Query [ taxi ], initial result ranking: ⟨ d1, d3, d2 ⟩
- Expand with top result, $\alpha = \beta = 0.5, \gamma = 0$
- Final ranking: ⟨ d1, d2, d3 ⟩
Analysing Rocchio’s algorithm
Using explicit (user) relevance feedback

- Rocchio’s algorithm uses both positive and negative feedback
- But positive feedback found to be more useful, $\gamma \ll \beta, \alpha$
- More problematic, algorithm assumes documents form clusters

Clustering assumptions
Relevant documents are grouped together, so their centroid is meaningful

$$\frac{1}{|D_r|} \sum_{d_i \in D_r} d_i$$

The same applies for non-relevant documents

$$\frac{1}{|D_{nr}|} \sum_{d_i \in D_{nr}} d_i$$

Can (partially) address this by setting $D_{nr}$ to be the single highest ranked non-relevant document
Query expansion in practice

- Suggestion / expansion by raw term similarity not widely used
  - Latent Semantic Analysis a preferred method
- Relevance feedback:
  - Need for explicit feedback can annoy users
  - More indirect measures like click throughs used instead
  - Not typically used in commercial systems, bar “More like this”
- 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 more typically done by search log mining:
  - See how people reformulate queries
  - . . . and suggest these reformulations to others
  - Also how spelling correction is done
Looking back and forward

Back

- Query expansion and reformulation
- Global, by looking at co-occurring terms throughout collection
- Local, by looking for terms in query results
- Rocchio’s algorithm (P/RF) 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?
- Later, we will look at probabilistic methods, that present themselves as more theoretically grounded, requiring fewer heuristic “hacks”
- Relevance feedback is a form of text classification, to be looked at in a few weeks.
Further reading

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

- James R. Curran and Marc Moens, *Scaling Context Space*, ACL’02,