CIS 833 – Information Retrieval and Text Mining

Lecture 21

Query Expansion

November 12, 2015

Credits for slides: Allan, Arms, Manning, Lund, Noble, Page.

Planning

- PageRank implementation: last assignment (due Dec 1st)
- Final exam: November 19th or December 3rd ?
- Project presentation: finals week (during the exam time)
- Project report: by the end of the finals week

How Do We Augment the User Query?

- A thesaurus provides information on synonyms and semantically related words and phrases.
- Manual thesaurus
  - E.g. MedLine: physician, syn: doc, doctor, MD, medico
- Global Analysis: (static; of all documents in collection)
  - Automatically derived thesaurus
    - (co-occurrence statistics)
  - Refinements based on query log mining
    - Common on the web
- Local Analysis: (dynamic)
  - Analysis of documents in result set

Statistical Thesaurus

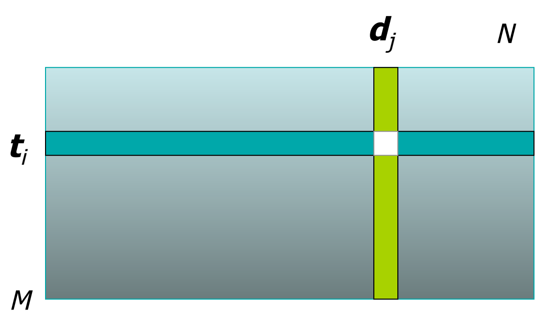
- Existing human-developed thesauri are not easily available in all languages.
- Human thesuari are limited in the type and range of synonymy and semantic relations they represent.
- Semantically related terms can be discovered from statistical analysis of corpora.

Automatic Thesaurus Generation

- Attempt to generate a thesaurus automatically by analyzing the collection of documents
- Fundamental notion: similarity between two words
- Definition 1: Two words are similar if they co-occur with similar words.
- Definition 2: Two words are similar if they occur in a given grammatical relation with the same words.
  - You can harvest, peel, eat, prepare, etc. apples and pears, so apples and pears must be similar.
- Co-occurrence based is more robust, grammatical relations are more accurate.

Co-occurrence Thesaurus

- Simplest way to compute a thesaurus is based on term-term similarities in $C = AA^T$ where A is term-document matrix.
- $w_{i,j}$ = (normalized) weight for (t_i, $\mathbf{d}_j$) – simplest case could be frequency



Co-occurrence Matrix

$$
\begin{array}{c|cccc|}
 & t_1 & t_2 & t_3 \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots & t_n \\
\hline
t_1 & c_{11} & c_{12} & c_{13} \ldots\ldots\ldots\ldots\ldots\ldots\ldots & c_{1n} \\
t_2 & c_{21} & & & \\
t_3 & c_{31} & & & \\
. & . & & & \\
. & . & & & \\
t_n & c_{n1} & & & \\
\hline
\end{array}
$$

What does C contain if A is a term-doc incidence (0/1) matrix?

c_{ij}: Correlation factor between term i and term j

$$c_{ij} = \sum_{d_k \in D} w_{ik} \times w_{jk}$$

For each t_i, pick terms with high values in C

Automatic Thesaurus Generation Example

word	ten nearest neighbors
absolutely	absurd whatsoever totally exactly nothing
bottomed	dip copper drops topped slide trimmed slig
captivating	shimmer stunningly superbly plucky witty
doghouse	dog porch crawling beside downstairs gazec
Makeup	repellent lotion glossy sunscreen Skin gel p
mediating	reconciliation negotiate cease conciliation p
keeping	hoping bring wiping could some would othe
lithographs	drawings Picasso Dali sculptures Gauguin
pathogens	toxins bacteria organisms bacterial parasite
senses	grasp psyche truly clumsy naive innate awl

Normalized Association Matrix

- Frequency based correlation factor favors more frequent terms.
- Normalize association scores:

$$s_{ij} = \frac{c_{ij}}{c_{ii} + c_{jj} - c_{ij}}$$

- Normalized score is 1 if two terms have the same frequency in all documents.

Metric Correlation Matrix

- Association correlation does not account for the proximity of terms in documents, just co-occurrence frequencies within documents.
- Metric correlations account for term proximity.

$$c_{ij} = \sum_{k_u \in V_i} \sum_{k_v \in V_j} \frac{1}{r(k_u, k_v)}$$

V_i: Set of all occurrences of term *i* in any document.
$r(k_u,k_v)$: Distance in words between word occurrences k_u and k_v
(∞ if k_u and k_v are occurrences in different documents).

Normalized Metric Correlation Matrix

- Normalize scores to account for term frequencies:

$$s_{ij} = \frac{c_{ij}}{|V_i| \times |V_j|}$$

Query Expansion with Correlation Matrix

- For each term *i* in query, expand query with the *n* terms, *j*, with the highest value of c_{ij} (s_{ij}).
- This adds semantically related terms in the “neighborhood” of the query terms.

Automatic Thesaurus Generation Discussion

- Quality of associations is usually a problem.
- Term ambiguity may introduce irrelevant statistically correlated terms.
  - "Apple computer" → "Apple red fruit computer"
- Problems:
  - False positives: Words deemed similar that are not
  - False negatives: Words deemed dissimilar that are similar
- Since terms are highly correlated anyway, expansion may not retrieve many additional documents.

Automatic Local Analysis

- At query time, dynamically determine similar terms based on analysis of top-ranked retrieved documents.
- Base correlation analysis on only the "local" set of retrieved documents for a specific query.
- Avoids ambiguity by determining similar (correlated) terms only within relevant documents.
  - "Apple computer" → "Apple computer Powerbook laptop"

Global vs. Local Analysis

- Global analysis requires intensive term correlation computation only once at system development time.
- Local analysis requires intensive term correlation computation for every query at run time (although number of terms and documents is less than in global analysis).
- But local analysis gives better results.

Global Analysis Refinements

- Only expand query with terms that are similar to *all* terms in the query.

$$sim(k_i, Q) = \sum_{k_j \in Q} c_{ij}$$

  - "fruit" not added to "Apple computer" since it is far from "computer."
  - "fruit" added to "apple pie" since "fruit" close to both "apple" and "pie."
- Use more sophisticated term weights (instead of just frequency) when computing term correlations.

Query Expansion Conclusions

- Expansion of queries with related terms can improve performance, particularly recall.
- However, must select similar terms very carefully to avoid problems, such as loss of precision.

Text Classification

Textbook Material

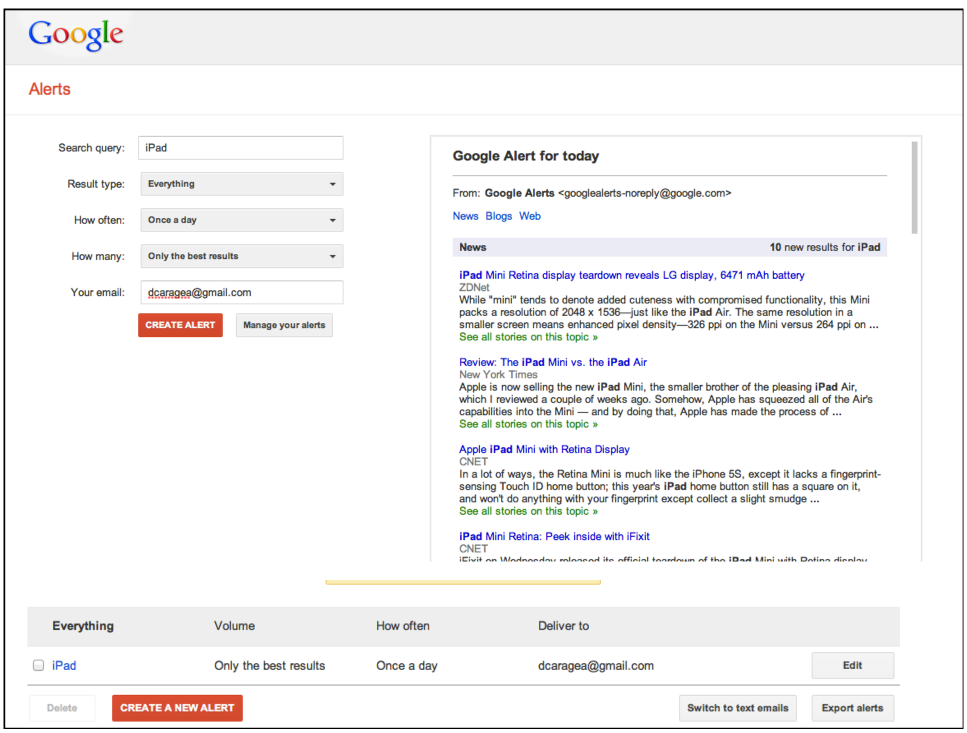
- Next – Text Classification
  - Chapter 13: Text Classification and Naïve Bayes
  - Chapter 14: Vector Space Classification
  - Chapter 15: Support Vector Machines

Relevance Feedback

- In relevance feedback, the user marks a number of documents as relevant/nonrelevant.
- We then try to use this information to return better search results.
- Suppose we just tried to learn a filter for nonrelevant documents.
- This is an instance of a text classification problem:
  - Two "classes": **relevant**, **nonrelevant**
  - For each document, decide whether it is relevant or nonrelevant
- The notion of classification is very general and has many applications within and beyond information retrieval.

Standing Queries

- The path from information retrieval to text classification:
  - You have an information need, say:
    - *MacBook Pro*
    - You want to rerun an appropriate query periodically to find new news items on this topic
  - You will be sent new documents that are found
    - i.e., it's classification not ranking
- Such queries are called ***standing queries***
  - Long used by "information professionals"
  - A modern mass instantiation is **Google Alerts**



Other Text Classification Examples:

Many search engine functionalities use classification

Assign labels to each document or web-page:

- Labels are most often topics such as Yahoo-categories
  e.g., "finance," "sports," "news>world>asia>business"
- Labels may be genres
  e.g., "editorials" "movie-reviews" "news"
- Labels may be opinion on a person/product
  e.g., "like", "hate", "neutral"
- Labels may be domain-specific
  e.g., "interesting-to-me" : "not-interesting-to-me"
  e.g., "contains adult language" : "doesn't"
  e.g., language identification: English, French, Chinese, …
  e.g., search vertical: about Linux versus not
  e.g., "link spam" : "not link spam"

Categorization/Classification

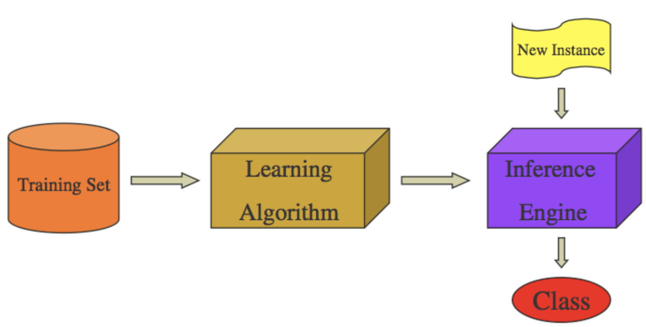
- Given:
  - A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
    - Issue: how to represent text documents.
  - A fixed set of classes:
    $C = \{c_1, c_2, \ldots, c_J\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a *classification function* whose domain is X and whose range is C.
    - We want to know how to build classification functions ("classifiers").

Classification Methods

- Supervised learning of a document-label assignment function
  - Many systems partly rely on machine learning
    - Relevance Feedback (Rocchio)
    - k-Nearest Neighbors (simple, powerful)
    - Naive Bayes (simple, common method)
    - Support-vector machines (newer, more powerful)
    - … plus many other methods
    - *No free lunch: requires hand-classified training data*
    - But data can be built up (and refined) by amateurs
      - CrowdSource
        - Amazon Mechanical Turk:https://www.mturk.com/mturk/welcome
        - CrowdFlower: http://crowdflower.com/
        - Herd It: http://herdit.org/blog/
- Note that many commercial systems use a mixture of methods

Closer Look at Classification

- **Classification task**: Learning how to label correctly new instances from a domain based on a set of previously labeled instances



Learning to Classify

- A training example is an instance $x \in X$, paired with its correct category $c(x)$:

  $<x, c(x)>$ for an unknown classification function, c.
- Given a set of training examples, D,
- Find a hypothesized classification function, $h(x)$, such that:

$$\forall < x, c(x) > \in D : h(x) = c(x)$$

Consistency

Sample Category Learning Problem

- Instance language: <size, color, shape>
  - size ∈ {small, medium, large}
  - color ∈ {red, blue, green}
  - shape ∈ {square, circle, triangle}
- C = {positive, negative}
- D:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

General Learning Issues

- Many hypotheses are usually consistent with the training data.
- Bias
  - Any criteria other than consistency with the training data that is used to select a hypothesis.
- Classification accuracy (% of instances classified correctly).
  - Measured on independent test data.
- Training time (efficiency of training algorithm).
- Testing time (efficiency of subsequent classification).

Generalization

- Hypotheses must generalize to correctly classify instances not in the training data.
- Simply memorizing training examples is a consistent hypothesis that does not generalize.
- *Occam's razor*:
  - Finding a *simple* hypothesis helps ensure generalization.

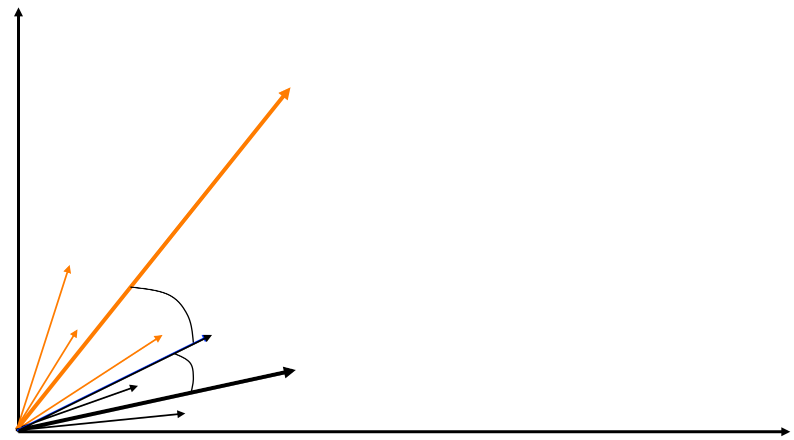
Learning Algorithms

- Relevance Feedback (Rocchio)
- k-Nearest Neighbors (simple, powerful)
- Naive Bayes (simple, common method)
- Support-vector machines (new, more powerful)
- … plus many other methods

Using Relevance Feedback (Rocchio)

- Use standard TF/IDF weighted vectors to represent text documents (normalized by maximum term frequency).
- For each category, compute a *prototype* vector by summing the vectors of the training documents in the category.
- Assign test documents to the category with the closest prototype vector based on cosine similarity.

Illustration of Rocchio Text Categorization



Rocchio Properties

- Does not guarantee a consistent hypothesis.
- Forms a simple generalization of the examples in each class (a *prototype*).
- Prototype vector does not need to be averaged or otherwise normalized for length since cosine similarity is insensitive to vector length.
- Classification is based on similarity to class prototypes.

Learning Algorithms for Classification Tasks

- Relevance Feedback (Rocchio)
- k-Nearest Neighbors (simple, powerful)
- Naive Bayes (simple, common method)
- Support-vector machines (new, more powerful)
- … plus many other methods