Comparing Word Occurrences across Documents: Information Retrieval, Terminology Extraction, etc.

Adam Meyers
New York University
2015
Outline

• Classifying Documents
  – Viewing “subject” of a document as a function of the set of words contained in the document
  – Similar documents → similar word distribution

• Search Query
  – Find document that is similar to query

• Terminology Extraction
  – Find words and word sequences that are significant, i.e., are valid search terms

• Other areas:
  – Cluster “similar documents”: topic modeling, sublanguage identification, …
Ad Hoc Information Retrieval

- Model of document = unordered set of terms contained in that document (ignore word order)
  - Term = word, bigram, trigram, noun group, or other small unit of consecutive items
- Query = user input, typically a set of terms
- Collection = set of documents that system
- Goal find documents that are “closest” to query
Vector Model

- Model documents and queries as vectors
- Feature values filled by the weight of terms
  - Values also called dimensions
- Example:
  - Terms: potato chip, chicken, sesame seed, coconut milk
  - Vector for query about Thai soups $\vec{S} = (0, 20, 2, 100)$
  - Vector for chicken and coconut soup recipe $\vec{S} = (0, 40, 0, 100)$
  - Vector for chicken noodle soup recipe $\vec{S} = (0, 20, 0, 0)$
- IR task: find documents that most closely “match” query
  - Matching via similarity metric defined on pairs of vectors
- Weights and Similarity Scores need to be defined
TFIDF = Common Weight for Vector

- Term Frequency – number of times term $t$ occurs in document
- Inverse Document Frequency: Reciprocal of portion of large document set that contain term $t$, normalized with log function:
  \[
  \log\left(\frac{\text{NumberOfDocuments}}{\text{NumberOfDocumentsContaining}(t)}\right)
  \]

- $\text{TFIDF}(t) = \text{TF}(t) \times \text{IDF}(t)$
  - Scores terms highly that occur frequently in a document or query
  - Scores terms highly that are infrequent in collection
Example: *coconut milk* vs. *tablespoon*

- **coconut milk**
  - occurs ~ 3 times in chicken and coconut soup recipe
    - Term frequency = 3
  - occurs in 4 out of 10,000 documents in collection
  - inverse document frequency = \(\log(10000/4) = \log(2500) = 7.82\)
  - TFIDF = \(3 \times 7.82 = 23.46\)

- **tablespoon**
  - occurs 4 times in chicken and coconut soup recipe
    - Term frequency = 4
  - occurs in 1200 out of 10,000 documents in corpus
  - inverse document frequency = \(\log(10000/1200) = \log(8.33) = 2.12\)
  - TFIDF = \(4 \times 2.12 = 8.48\)

- *coconut milk* is more highly weighted for Thai Soup recipes than *tablespoon*

- Note: Suitability of query term may depend on the nature of the collection
  - Is this a collection of recipes? – *tablespoon* not good search term
  - Is collection diverse: instructions, news, …? – *tablespoon* may be good search term
Cosine Similarity: Common Similarity Score

\[ \text{Similarity}(A, B) = \frac{\sum_i a_i \times b_i}{\sqrt{\sum_i a_i^2 \times \sum_i b_i^2}} \]

- Cosine of the Angle Between the Vectors
- Numerator is Dot Product, Denominator is a normalizing factor, based on lengths of vectors
- If a query is A and a document if B
  - Cosine similarity high if values of a and b are similar
Example

- Vectors have values corresponding to terms:
  - potato chip, chicken, sesame seed, coconut milk, ground beef

- 2 Queries
  - Q1 chicken, coconut milk: (0,5,0,5,0)
  - Q2 ground beef, potato chip: (4,0,0,0,7)

- 2 Documents
  - D1 Chicken and Coconut Soup Recipe: (0,7,0,9,0)
  - D2 Hamburger Recipe: (3,0,2,0,9)

- Cosign similarities
  - |   | Q1  | Q2  |
  - |---|-----|-----|
  - D1 | 99.2 | 0   |
  - D2 | 0   | 95.9 |
Other Factors

- Many more terms (possibly thousands) represented in each vector
- More weights, normalizations, etc.
- Other similarity measures and weighting functions
- Lists of “stop words”, e.g., the, a, in, to, does, …
- Stemming procedures that consider some terms to be the same, e.g., [cat, cats], [analyze, analyzes, analyzed, analysis, analyse, …]
- Identifying other similar words, e.g., synonyms
  – query expansion, term clustering, …
- Systems identify word sequences as terms: N-grams or chunking
Evaluation of Doc Extraction

- Output = A Ranked List of Documents
  - Some high-ranked errors “worse” than low-ranked
  - Ranking makes relevant/irrelevant distinction subtle
  - Mean Average Precision (MAP): average precision weighted by rank

- Too Expensive to Create Gold Standard Manually
  - Collections can be millions or billions of documents
  - Precision can be approximated by taking samples of the text or evaluating the top N ranked terms manually.
  - Recall can also be approximated by some sort of sampling, e.g., only manually evaluating a subset of the collection

- Precision/Recall tradeoff curves based on numbers in the ranking
  - Typically, precision goes down and recall goes up as more documents in the ranking are considered
Sample Precision/Recall Tradeoff Based on Number of Search Results

Precision vs. Recall graph with number of search results on the x-axis and precision/recall on the y-axis.
Precision/Recall Curve
Final Remarks about Document Retrieval

• TFIDF weighting + Cosine similarity
  – standard in IR document retrieval for over 50 years

• Web Search Engines
  – use these methods to identify relevant documents
  – they use other metrics, e.g., PageRank, to rank documents by their “importance”

• Some areas of Opinion/Sentiment Extraction
  – Similar methods applied to differentiating positive/negative opinions in documents
  – More Difficult
  – Same terms linked to positive/negative in different contexts
  • low, high, small, large, thin, thick, visible, loud, soft, …
    – high/low quality, high/low interest, high/low resolution
Terminology Talk

• Do Terminology Talk Now
Homework

- Jurafsky and Martin Chapter 23.1
- Meyers, et. al. 2015 paper
  - Code Available from github:
    - https://github.com/AdamMeyers/The_Termolator
    - https://github.com/ivanhe/termolator/
- Implement a simple system that:
  - Computes inverse document frequency for each token occurring at least 2 times in all-OANC.txt
  - optionally remove stop words from
    - http://cs.nyu.edu/courses/fall15/CSCI-UA.0480-006/stop_list.py
  - for tokenization, you can use nltk if you want
  - For a given new file, will find the top N ranked terms according to the IDF scores from OANC