15-499: Algorithms and Applications

Indexing and Searching II

Indexing and Searching Outline

Introduction: model, query types
Inverted Indices: Compression, Lexicon, Merging

Vector Models:
- Selecting weights
- Cosine measure
- Relevance feedback

Latent Semantic Indexing:
Link Analysis: PageRank (Google), HITS
Duplicate Removal:

Vector Space Model

Model each document as a vector in \( n \) dimensional space: \((1, 0, 0, .9, .5, ... , 0)\)

Aardvark Ant Zebra

Query can also be modeled as a vector.

Uses:
- Ranked keyword search
- Relevance feedback
- Semantic indexing
- Clustering

Selecting Weights

\( f_{d,t} \) = number of times \( t \) appears in document \( d \)
\( w_{d,t} \) = weight of \( t \) in \( d \)

Accounting for frequency within a document
\( w_{d,t} = f_{d,t} \) frequency
\( w_{d,t} = \log(1 + f_{d,t}) \) log frequency

Accounting for information content of a term
\( w_t = \log(1/p) = \log(N/f_t) \)
giving: \( w_{d,t} = \log (N/ f_t ) \log(1 + f_{d,t}) \)
Similarity between vectors

*Dot product:* \( w_q \cdot W_d \)

*Inverse Euclidean distance:* \( 1/||w_q - W_d|| \)

*Problem:* they weight longer documents more heavily.

*Cosine metric:* \( \frac{w_q \cdot W_d}{||w_q|| \cdot ||W_d||} \)

Based on \( X \cdot Y = ||X|| \cdot ||Y|| \cdot \cos \theta \)

---

Queries (cosine measure)

Algorithm: *Query*(Q)

\[ A = \emptyset \] (Accumulators for documents)

For each term \( t \in Q \)

\( (t; w_t; P_t) = \) Search lexicon for \( t \)

\( P_t = \) uncompress\( (P_t) \)

For each \( (d, f_{d,t}) \) in \( P_t \)

if \( a_d \in A \)

\( a_d = a_d + w_t \cdot \log(f_{d,t}) \)

else

\( a_d = w_t \cdot \log(f_{d,t}) \)

\( A = A + \{a_d\} \)

For each \( a_d \in A \)

\( a_d = a_d/|d| \)

Select \( k \) documents from \( A \) with largest \( a_d \)

---

Frequencies and Inverted Lists

Each inverted list:

\[ \{t; w_t; (f_{d,1}, f_{d,2}, \ldots, f_{d,m})| (d,m, f_{d,m}) \} \]

\[ \{aardvark; 1; [(2,3), (5,4)] \} \]

Frequency counts can typically be compressed at least as well as distances. (2 bits/pointer in TREC.)

---

Relevance Feedback

Consider a sequence of queries \( Q_1, Q_2, \ldots, Q_n \) in which \( R_i \) are the relevant and irrelevant documents returned by query \( i \) (typically marked by the user)

We can generate each query from previous queries:

\[ Q_{i+1} = \alpha Q_i + \omega \sum_{d \in R_i} D_d + \beta \sum_{d \in I} D_d \]

What is the efficiency problem with these queries?
**Clustering**

**Goals:**
1. Speed up searches for complicated queries
2. Find documents which are similar

There are many techniques for clustering, as well as many other applications of clustering.

**Indexing and Searching Outline**

**Introduction:** model, query types

**Inverted Indices:** Compression, Lexicon, Merging

**Vector Models:** Weights, cosine distance

**Latent Semantic Indexing:**
- Singular Value Decompositions
- Applications to indexing and searching

**Link Analysis:** PageRank (Google), HITS

**Duplicate Removal:**