15-853: Algorithms in the Virtual World

Indexing and Searching II

Vector Space Model
Model each document as a vector in \( n \) dimensional space: \((1, 0, 0, 0, \ldots, 0)\)

- Aardvark
- Ant
- Zebra

Query can also be modeled as a vector.

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

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:

Selecting Weights

\[ f_{d,t} = \text{number of times } t \text{ appears in document } d \]

\[ w_{d,t} = \text{weight of } t \text{ in } d \]

Accounting for frequency within a document

\[ w_{d,t} = f_{d,t} \quad \text{frequency} \]

\[ w_{d,t} = \log(1 + f_{d,t}) \quad \text{log frequency} \]

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:**

\[
\cos \theta = \frac{w_q \cdot w_d}{||w_q|| \cdot ||w_d||}
\]

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

### Frequencies and Inverted Lists

Each inverted list:

\[
\{ (t; w_1; f_{d_1, t}), (d_{1}, f_{d_1, t}), \ldots, (d_{m}, f_{d_m, t}) \}
\]

(aardvark: (2,3),(5,4))

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

### Queries (cosine measure)

**Algorithm:**

1. **Query** (Q)
   - \( A = \emptyset \) (Accumulators for documents)
   - For each term \( t \in Q \)
     - \( t; w_1; P_t \) = Search lexicon for \( t \)
     - \( P_t = \text{uncompress}(P_t) \)
     - For each \( (d; f_{d, t}) \) in \( P_t \)
       - if \( a_d \in A \)
         - \( a_d = a_d + w_1 \cdot \log(f_{d, t}) \)
       - else
         - \( a_d = w_1 \cdot \log(f_{d, t}) \)
       - \( A = A + (a_d) \)
   - For each \( a_d \in A \)
     - \( a_d \) = \( a_d/|d| \)
   - Select \( k \) documents from \( A \) with largest \( a_d \)

### Relevance Feedback

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

We can generate each query from previous queries:

\[
Q_{i+1} = \pi Q_0 + \alpha Q_i + \alpha \sum_{d \in R_i} D_d + \beta \sum_{d \in I_i} D_d
\]

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 Valued Decompositions
- Applications to indexing and searching

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

**Duplicate Removal**: 