Query Languages and Retrieval Methods

Boolean Queries: Using operators like AND, OR, and BUT for combining keywords.

Natural Language Queries: Treating queries as bag-of-words for vector-space models.

Phrasal Queries: Retrieving documents containing specific ordered phrases.

Proximity Queries: Searching for words with specific distance constraints.

Pattern Matching: Queries that match strings rather than word tokens.

Text Properties and Languages

Word Frequency Distribution: Heavy-tailed distribution of word frequencies.

Zipf's Law: Frequency of a word is inversely proportional to its rank.

Markup Languages: HTML, XML for annotating documents with tags.

Metadata: Descriptive and semantic information about documents.

Semantic Web Technologies and Linked Data

RDF (Resource Description Framework): Describing things and their interrelations.

SPARQL: Query language for RDF data.

Linked Data: Principles for publishing and connecting data on the web.

Web Search and Spidering

Challenges in Web Search: Distributed data, large volume, unstructured data.

Web Spidering: Systematic navigation of the web to index pages.

PageRank: Algorithm for ranking web pages based on importance.

Deep Learning

Neural Network s: Artificial neurons, activation functions.

Recurrent Neural Networks (RNNs): For sequential data processing.

Convolutional Neural Networks (CNNs): Originally for images, adaptable to text.

Naïve Bayes Classifiers

Bayes' Theorem: Probabilistic approach to classification.

Naïve Bayes Assumption: Features are independent given the class.

TF-IDF (Term Frequency-Inverse Document Frequency): Weighting scheme for information retrieval.

- **Precision (P):** Proportion of retrieved documents that are relevant.
  $P = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}}$
- **Recall (R):** Proportion of relevant documents that are retrieved.
  $R = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}}$
- **Average Precision (AP):** Average of precision values at all relevant documents' ranks.
- **F-measure:** Harmonic mean of precision and recall.
  $F = \frac{2 \times P \times R}{P+R}$

The Boolean model used binary weights (a word is present or absent from a document) and it uses strict Boolean constraints (AND, OR, NOT or combinations) in order to choose relevant documents. Advantages:
- simple, easy to understand
- easy to implement, fast

Disadvantages:
- does not allow for partial matches
- does not produce a ranking
- users find it difficult to express complex queries
- difficult to perform relevance feedback

The Cosine Vector Space Model uses term frequencies (and inverse document frequencies) and computes similarity between query and documents using cosine. Advantages:
- allows for partial matching
- produces a ranking

Disadvantages:
can be implemented efficiently, but not as easy as the Boolean model, some documents can be retuned even if they do not match all the query terms.

	In	Out	It 0		It 1		It 2	
			a	h	a	h	a	h
A	B,D	B,C,D	1	1	2	3	3	6
B	A,C	A,D	1	1	2	2	5	5
C	A	B, D	1	1	1	2	3	5
D	A,B,C	A	1	1	3	1	7	2

###Step 1: Initialization
- Assign equal hub and authority scores to all pages.

Step 2: Iterative Computation
- Update authority scores:
  - Authority(A) = Hub(B) + Hub(D) = 2
  - Authority(B) = Hub(A) + Hub(C) = 2
  - Authority(C) = Hub(B) = 1
  - Authority(D) = Hub(A) + Hub(B) + Hub(C) = 3

- Update hub scores:
  - Hub(A) = Authority(B) + Authority(D) = 3
  - Hub(B) = Authority(A) + Authority(C) = 2
  - Hub(C) = Authority(D) = 2
  - Hub(D) = Authority(A) = 1

Step 3: Convergence
- Repeat Step 2 until convergence (scores stabilize).

Step 4: Ranking
- Rank pages based on their authority scores:

1. Page D (Authority score = 7)
2. Page B (Authority score = 5)
3. Pages A and C(Authority score = 3)

These rankings indicate the relative importance of pages based on their authority in the web graph.

PR(A) = (1-d) + d(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn)) where T1 ... Tn are the pages that point to a page A (the incoming links), d is damping factor (usually d = 0.85), C(A) is number of links going out of a page A and PR(A) is the PageRank of a page A. Normalize the scores by dividing by the sum of all scores at the end of each iteration.

P(A) = 0.15 + 0.85 * (P(B)/2 + P(D))
P(B) = 0.15 + 0.85 * (P(A)/3 + P(C)/2)
P(C) = 0.15 + 0.85 * (P(A)/3)
P(D) = 0.15 + 0.85 * (P(A)/3 + P(B)/2 + P(C)/2)

Initial values: P(A) = 0.25 P(B) = 0.25 P(C) = 0.25 P(D) = 0.25
Iteration 1
P(A) = 0.15 + 0.85 * (0.25/2 + 0.25) = 0.46875
P(B) = 0.15 + 0.85 * (0.25/3 + 0.25/2) = 0.327
P(C) = 0.15 + 0.85 * (0.25/3) = 0.2208
P(D) = 0.15 + 0.85 * (0.25/3 + 0.25/2 + 0.25/2) = 0.433

Normalization Sum = 0.46875 + 0.327 + 0.2208 + 0.433 = 1.44955
P(A) = 0.46875 / 1.44955 = 0.323
P(B) = 0.327 / 1.44955 = 0.225
P(C) = 0.2208 / 1.44955 = 0.152
P(D) = 0.433 / 1.44955 = 0.298

Iteration 2
P(A) = 0.15 + 0.85 * (0.225/2 + 0.298) = 0.498
P(B) = 0.15 + 0.85 * (0.323/3 + 0.152/2) = 0.306
P(C) = 0.15 + 0.85 * (0.323/3) = 0.241
P(D) = 0.15 + 0.85 * (0.323/3 + 0.225/2 + 0.152/2) = 0.401
Normalization Sum = 1.445
P(A) = 0.344
P(B) = 0.211
P(C) = 0.166
P(D) = 0.277

Page A points to pages B, C, and D.
Page B points to pages A and D.
Page C points to pages B and D.
Page D points to page A.

Here is a simplified example of the vector space retrieval model. Consider a very small collection C that consists in the following three documents:

d1: "new york times"
d2: "new york post"
d3: "los angeles times"

Some terms appear in two documents, some appear only in one document. The total number of documents is N=3. Therefore, the idf values for the terms are:

angles	$log_2(3/1)=1.584$
los	$log_2(3/1)=1.584$
new	$log_2(3/2)=0.584$
post	$log_2(3/1)=1.584$
times	$log_2(3/2)=0.584$
york	$log_2(3/2)=0.584$

For all the documents, we calculate the tf scores for all the terms in C. We assume the words in the vectors are ordered alphabetically.

	angeles	los	new	post	times	york
d1	0	0	1	0	1	1
d2	0	0	1	1	0	1
d3	1	1	0	0	1	0

Now we multiply the tf scores by the idf values of each term, obtaining the following matrix of documents-by-terms: (All the terms appeared only once in each document in our small collection, so the maximum value for normalization is 1.)

	angeles	los	new	post	times	york
d1	0	0	0.584	0	0.584	0.584
d2	0	0	0.584	1.584	0	0.584
d3	1.584	1.584	0	0	0.584	0

Given the following query: "new new times", we calculate the tf-idf vector for the query and compute the score of each document in C relative to this query, using the cosine similarity measure. When computing the tf-idf values for the query terms we divide the frequency by the maximum frequency (2) and multiply with the idf values.

q	0	0	(2/2)*0.584=0.584	0	(1/2)*0.584=0.292	0

We calculate the length of each document and of the query:

Length of d1 = sqrt(0.584^2+0.584^2+0.584^2)=1.011
Length of d2 = sqrt(0.584^2+1.584^2+0.584^2)=1.786
Length of d3 = sqrt(1.584^2+1.584^2+0.584^2)=2.316
Length of q = sqrt(0.584^2+0.292^2)=0.652

Then the similarity values are:

cosSim(d1,q) = (0*0+0*0+0.584*0.584+0*0+0.584*0.292+0.584*0) / (1.011*0.652) = 0.776
cosSim(d2,q) = (0*0+0*0+0.584*0.584+1.584*0+0*0.292+0.584*0) / (1.786*0.652) = 0.292
cosSim(d3,q) = (1.584*0+1.584*0+0*0.584+0*0+0.584*0.292+0*0) / (2.316*0.652) = 0.112

According to the similarity values, the final order in which the documents are presented as result to the query will be: d1, d2, d3.

1. Calculate Term Frequency (TF) for a Term in a Document:

- Term Frequency (TF) measures how often a term occurs in a document.
- Formula:
  $\text{TF}_{t,d} - \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$

2. Calculate Inverse Document Frequency (IDF) for a Term:

- Inverse Document Frequency (IDF) measures how important a term is across a collection of documents. Rare terms are weighted more heavily than common terms.
- Formula:
  $\text{IDF}_t - \log \frac{\text{Total number of documents}}{\text{Number of documents containing term } t}$

3. Compute TF-IDF for a Term in a Document:

- TF-IDF is the product of TF and IDF.
- Formula:
  $\text{TF-IDF}_{t,d} - \text{TF}_{t,d} \times \text{IDF}_t$

Formula for Cosine Similarity (cosSim):

$$\text{cosSim}(d_1, d_2) = \frac{\sum_{i=1}^{n} \text{TF-IDF}_{i,d_1} \times \text{TF-IDF}_{i,d_2}}{\sqrt{\sum_{i=1}^{n} (\text{TF-IDF}_{i,d_1})^2} \times \sqrt{\sum_{i=1}^{n} (\text{TF-IDF}_{i,d_2})^2}}$$

Where:

- n is the total number of unique terms in the documents.
- $\text{TF-IDF}_{i,d_1}$ and $\text{TF-IDF}_{i,d_2}$ are the TF-IDF values for term i in documents d_1 and d_2 respectively.

$$\text{Coverage Ratio} = \frac{\text{Number of Relevant Documents Retrieved}}{\text{Total Number of Relevant Documents}}$$

$$\text{Novelty Ratio} = \frac{\text{Number of Unknown Relevant Documents Retrieved}}{\text{Total Number of Relevant Documents Retrieved}}$$