Models in IR

Lecture 3
CS 410/510
Information Retrieval on the Internet

I have met with but one or two persons in the course of my life who understood the art of Walking, that is, of taking walks,—who had a genius, so to speak, for *sauntering:* which word is beautifully derived from "idle people who roved about the country, in the Middle Ages, and asked charity, under pretence of going *à la Sainte Terre,"* to the Holy Land, till the children exclaimed, "There goes a *Sainte-Terrer,"* a Saunterer, a Holy-Lander. They who never go to the Holy Land in their walks, as they pretend, are indeed mere idlers and vagabonds; but they who do go there are saunterers in the good sense, such as I mean. Some, however, would derive the word form *sans terre,* without land or a home, which, therefore, in the good sense, will mean, having no particular home, but equally at home everywhere. For this is the secret of successful sauntering. He who sits still in a house all the time may be the greatest vagrant of all; but the saunterer, in the good sense, is no more vagrant than the meandering river, which is all the while sedulously seeking the shortest course to the sea. But I prefer the first, which, indeed, is the most probable derivation. For every walk is a sort of crusade, preached by some Peter the Hermit in us, to go forth and reconquer this Holy Land from the hands of the Infidels.

- from an essay by Henry David Thoreau

What is this essay about? Justify your answer.

Taxonomy of IR models

- Basic taxonomy:
  - Boolean (set theoretic)
  - Vector (algebraic)
  - Probabilistic (probabilistic)
- Note, will only consider "ad hoc" retrieval tasks, not filtering or routing tasks

Note: See chapter 2 of Baeza-Yates text for more complete treatment of definitions and formalisms

Formal characterization of IR models

An IR model is a quadruple $[D, Q, F, R(q_i, d_j)]$ where:

1. D is a set of logical views (representations) for the documents in the collection
2. Q is a set of logical views (representations) for the user information needs (queries)
3. F is a framework for modeling document representations, queries, and their relationships
4. $R(q_i, d_j)$ is a ranking function that associates a real number with $q_i \in Q$ and a $d_j \in D$ that defines an ordering among documents wrt the query q_i.

Basic concepts

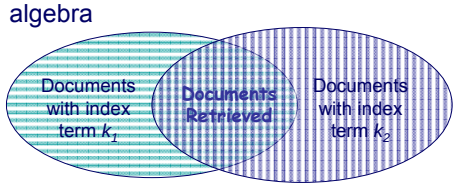
- Documents described by representative keywords called index terms
  - Could be assigned, extracted, selected
- May want to assign numerical weights to indicate importance, reflecting ability to
  - Summarize document contents
  - Discriminate this document from others
- Ranking function generally predicts the relevance of query q_i to document d_j

Weighted index terms

- Let k_i be an index term and d_j be a document
  - $w_{i,j} \geq 0$ is the weight associated with (k_i, d_j)
  - Quantifies the importance of k_i for describing d_j or for discriminating d_j from other documents
- Let $\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{t,j})$ be a vector of weighted index terms to describe d_j
  - where *t* is the number of index terms
- Usually make a simplifying assumption that the index terms weights are independent
  - They are not independent
  - Some systems try to exploit co-occurrence data

Boolean model

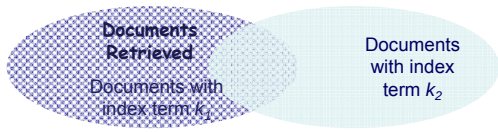
- Queries are Boolean expressions
- Retrieval based on set theory & Boolean algebra



Query: k_1 **AND** k_2

Boolean model

Query: k_1 **OR** k_2



Query: k_1 **NOT** k_2

Boolean model

- Index term weights all = 1
  - Index terms either present or absent
- similarity (d_j, q) = 1 if any of the conjunctive components of the query is satisfied*; 0 otherwise

Note similarity to data retrieval and DB query language

Where query is written as a Boolean expression in CNF

Boolean model

- Prediction of relevance is binary
  - relevant or nonrelevant
- No inherent ranking function; need some way to order results (but may not be useful)
  - Publication date
  - Alphabetically, e.g. by title or author
  - Random
- Boolean operators difficult for many users
- Result set often too small or too large

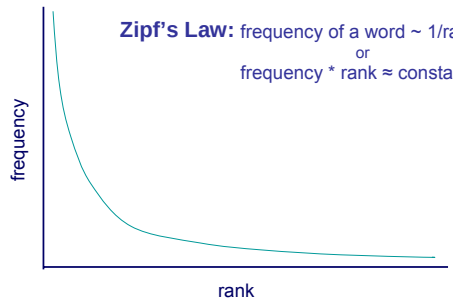
Doc 1:	We had so much fun at the Kohler factory that Kaye suggested we check out the GM plant in Janesville. It's a huge plant, 3.5 million square feet, with 3 assembly lines. Two of them make trucks and Bluebird bus frames, but the line we saw makes Chevy Suburbans and similar light trucks, at the rate of one every 67 seconds.
Doc 2:	The Janesville facility was built by GM in 1919 as the Sampson Tractor Plant, and started making trucks as well the next year. In 1922 they started making Chevrolet passenger cars there.
Doc 3:	A few overall comments on the Suburban line. Janesville is an assembly plant, so all the parts are made elsewhere, and come to the plant by truck and rail.
Doc 4:	There is very little inventory of parts on site. Basically, enough parts for one shift arrive at one time by train or truck.

Query	Doc 1	Doc 2	Doc 3	Doc 4
janesville AND parts				
frames OR parts				
(truck OR trucks) NOT cars				
(plant NOT parts) OR (truck AND train)				

Vector space model

- Start with a little history

A Little History



A Little History: Hans-Peter Luhn

- Luhn was an early proponent (1950s) of using statistical properties of words in documents for automated retrieval and abstracting
- Proposed word frequency as a measure of significance
  - Repetition of words by author indicates significance
  - Words that are too common constitute "noise"
  - Medium frequency words are best discriminators, have best "resolving power"
  - Described early stemming algorithm to consolidate words before calculating frequency

A Little History: Hans-Peter Luhn

- Used significance of words and proximity to other significant words to calculate "significance factor" of sentences
  - to select sentences for auto-abstracts
- Proposed a coefficient of similarity between documents
  - Based on relative word frequencies
  - $sim(X,Y) = \sum_i \min(f_i, g_i)$
  - Where f_i is the relative frequency of word *i* in document *X* and g_i is the relative frequency of *i* in *Y*
    - Rel freq of *i* in *X* = freq of *i* in X/total number of words in *X*

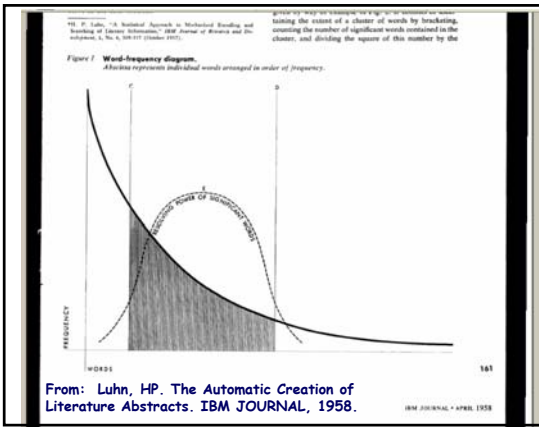


Figure 1 Word-frequency diagram.

From: Luhn, HP. The Automatic Creation of Literature Abstracts. IBM JOURNAL, 1958.

A little history: Gerard Salton

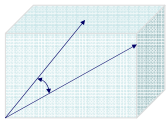
- Developed and extensively studied the vector space model for retrieval (with colleagues and students)
- Implemented the model
  - SMART experimental IR system
- Studied effect of many parameters and improvements
  - e.g. stemming, similarity measures, term-weighting approaches
- Provided evidence that terms with best discriminating value were those of medium to rare (but not too rare) frequency

Vector space model

- Indexing terms are coordinates in a multidimensional information space
- Documents and queries represented as *n*-dimensional vectors
  - *n* = total number of terms
  - *i*-th element in vector is weight of the *i*-th term
    - Weight derived from a term-weighting algorithm
    - Term-weighting most often uses some form of word frequency calculation

Vector space model

- Allows assignment of non-binary weights to index terms
- Allows computation of similarity between documents and queries
  - Usually calculated as the cosine of the angle between two vectors $\vec{d_j}$ and $\vec{q}$ (or a variation on that calculation)
  - Natural to return ranked list of documents



Vector space model

Cosine of angle between two vectors:

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \times \left|\vec{q}\right|}$$

$$= \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

$\left|\vec{q}\right|$ is the same for all docs; does not affect ranking

$\left|\vec{d_j}\right|$ allows normalization for length of the document

$sim(d_j, q)$ ranges from 0 to +1

cosine coefficient for similarity can be used with either binary or real-valued term weights

Vector space model

- Assignment of term weights is typically based on word frequencies
  - Frequency of term in document
    - High frequency indicates term reflects document content
  - Frequency of term in the entire collection
    - Document frequency is # documents with term
    - Low frequency in collection suggests good discriminator
  - TF*IDF
    - TF is frequency of term in document (possibly normalized)
    - IDF is inverse of document frequency (usually calculated as log (N/n_i) where N = #docs in collection, n_i = #docs with term i)

Vector space model

- Many variations on term-weighting have been tried, e.g.
  - Logarithmic term frequencies
  - Term frequencies normalized to max term frequency and scaled to fall in range 0.5 – 1
  - Salton and Buckley experimented extensively with various permutations in the SMART system using multiple test collections
- Query terms may be weighted or binary
  - Weighted may be useful for long queries, such as documents or long descriptions of information needs

Vector space model

- Advantages
  - Term weighting improves performance compared to term overlap (weights = 0 or 1)
  - Allows partial matching
  - Allows ranked retrieval
- Drawbacks
  - Assumes indexing terms are independent

Doc1	Doc2	Doc3
Chevy assembly occurs in Janesville at the Chevy factory.	Assembly of cars in Janesville is interesting.	Factory assembly of Chevy cars is interesting.

Raw term freq

IDF

Query: Chevy assembly Janesville

Term	TF_1	TF_2	TF_3	DF	$\log(N/n_i)$	$w_{i,d1}$	$w_{i,d2}$	$w_{i,d3}$	$w_{i,q}$
chevy									
assembly									
occurs									
janesville									
factory									
cars									
interesting									

Doc	Similarity to query:
1	
2	
3	

Probabilistic approach

- Answer the "Basic Question":
  - "What is the probability that *this* document is relevant to *this* query?"*
- Rank documents by probability of relevance
  - estimate P (Relevance|Document)
- Follows from Probability Ranking Principle
  - "If retrieved documents are ordered by decreasing probability of relevance on the data available, then the system's effectiveness is the best to be gotten for the data." *

*K. Sparck Jones, S Walker, S.E. Robertson. A Probabilistic Model of Information Retrieval: Development and Status. TR 446, Cambridge University Computer Laboratory, September 1998.

Probabilistic model

- Ranking is based on probability of relevance to the query, not similarity
- Relevance assumed to be binary
- Relevance of one document assumed to be independent of relevance of other documents
- Relevance assumed to be an attribute of the relationship between document and query, independent of user situation

A little history: Probabilistic model

- Maron and Kuhns (1960)
  - Proposed calculation of a relevance number
    - A measure of the probable relevance of a document for a requestor
    - A number used to rank documents
  - Proposed probabilistic indexing
    - Indexer assigns terms with a probability
      - probability that the document will be relevant to a user who is interested in the subject designated by the term
    - Weighted index terms will characterize content more accurately

A little history: Probabilistic model

- Maron and Kuhns (1960)
  - Proposed techniques for finding the "closest" index terms to the original query terms in order to retrieve more documents
    - Query expansion
  - Proposed expanding the result set using a distance function (based on weighted index terms) to find documents similar to the original retrieved documents
    - Relevance feedback

Probabilistic model

- Goal: rank documents according to probability of being in the relevant set
  - estimate P(Relevance|Document) for a query
- Based on Bayes theorem

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B \mid A)P(A) + P(B \mid \overline{A})P(\overline{A})}$$

Probabilistic model

- Bayes theorem $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$
  - P(A) is prior probability
    - Assumes no knowledge of P(B)
  - P(A|B) is conditional probability
    - Is the probability of A given a known value for B
- Conditional probability defined:
  $P(A \mid B) = \frac{P(A \cap B)}{P(B)}$
- Combine with corresponding equation for P(A|B) and rearrange:
  $P(A \mid B)P(B) = P(A \cap B) = P(B \mid A)P(A)$

Example

- Suppose my friend recommends a movie. I don't like many movies, in fact I only like about 10% of them. My friend likes 90% of the movies I like, and about 50% of the ones I don't like. What's the probability that I'll like this one?

Probabilistic model

- Initial model
  - Used binary term weights
  - Assumed term independence
  - Formulated so as to compute probabilities of relevance based on user feedback
    - Following initial simple retrieval
  - Also inspired work to predict relevance weighting with little or no relevance information

Probabilistic model

- Like vector space model
  - Uses vector representation of terms in document and query with term weights
  - Uses a ranking function to order retrieved documents
  - May use term frequency data to estimate probability
- Unlike vector space model
  - Ranking based on calculation of probability, not similarity

Probabilistic model

- Where do all those equations come from?

Probabilistic model

- Starting point: want to calculate a score based on $P(L \mid D) = \frac{P(D \mid L)P(L)}{P(D)}$
  - P(L) is probability user will ***like*** document (i.e. relevance)
- For convenience, use log-odds which is order preserving

$$\log \frac{P(L \mid D)}{P(\overline{L} \mid D)} = \log \frac{P(D \mid L)P(L)}{P(D \mid \overline{L})P(\overline{L})} = \log \frac{P(D \mid L)}{P(D \mid \overline{L})} + \log \frac{P(L)}{P(\overline{L})}$$

- Since $\log \frac{P(L)}{P(\overline{L})}$ will not affect ranking order, we ignore it

Probabilistic model

- We want to find a score based on: $\log \frac{P(D \mid L)}{P(D \mid \overline{L})}$
- We can consider document D as a set of attributes, or term weights

$$P(D \mid L) = \prod_i P(A_i = a_i \mid L)$$

  - Where A_i is the *i-th* attribute and a_i is its value for a given document
- Substitute; now our score is calculated as:

$$\sum_i \log \frac{P(A_i = a_i \mid L)}{P(A_i = a_i \mid \overline{L})}$$

Probabilistic model

- Problem with $\sum_i \log \frac{P(A_i = a_i \mid L)}{P(A_i = a_i \mid \bar{L})}$
  - Requires a component for each term that is absent ($A_i = 0$); a zero value would be natural
  - Subtract the "natural zero" component from every document, only consider terms present
  - Order-preserving transformation:

$$\sum_i (\log \frac{P(A_i = a_i \mid L)}{P(A_i = a_i \mid \bar{L})} - \log \frac{P(A_i = 0 \mid L)}{P(A_i = 0 \mid \bar{L})}) = \sum_i \log \frac{P(A_i = a_i \mid L) P(A_i = 0 \mid \bar{L})}{P(A_i = a_i \mid \bar{L}) P(A_i = 0 \mid L)}$$

  - Let W be a function that assigns a weight for each value of each attribute:
    - $W(A_i = a_i) = \log \frac{P(A_i = a_i \mid L) P(A_i = 0 \mid \bar{L})}{P(A_i = a_i \mid \bar{L}) P(A_i = 0 \mid L)}$
    - $W(A_i = 0) = 0$
    - Score = $\sum_i W(A_i = a_i)$

Probabilistic model

- Score is based on: $\sum_i \log \frac{P(A_i = a_i \mid L) P(A_i = 0 \mid \bar{L})}{P(A_i = a_i \mid \bar{L}) P(A_i = 0 \mid L)}$
- Weight for term presence: $w_i = \log \frac{p_i (1 - q_i)}{q_i (1 - p_i)}$
  - where p_i = P(term$_i$ present|L)
  - and q_i = P(term$_i$ present|L)
- So, how to assign w$_i$?
  - i.e. how to assign p_i and q_i?

Probabilistic model

- Consider possible ways to assign values to p_i
  - Unweighted: presence/absence of term
    - (p_i, q_i = 0 or 1)
  - Collection frequency weights
  - Incorporate relevance information from user
  - Term frequency (within document) weights

Contingency table

	Relevant	Non-relevant	
Contains term	r	n – r	n
Does not contain term	R – r	N – n – R + r	N – n
	R	N – R	N

Probabilistic model

- To use contingency table to assign weights:
  - For any i, $p = \frac{r}{R}$ $q = \frac{n - r}{N - R}$
- So $w_i = \log \frac{p_i (1 - q_i)}{q_i (1 - p_i)} = \log \frac{r(N - n - R + r)}{(R - r)(n - r)}$

	Relevant	Non-relevant	
Contains term	r	n – r	n
Does not contain term	R – r	N – n – R + r	N – n
	R	N – R	N

Probabilistic model

- Must estimate values for p and q
- To get a well-behaved formula, add 0.5 to all central cells in contingency table
- If no relevance info available, using collection frequencies for weighting:
  - Assume R is small relative to N
  - Estimate q as n/N (proportion of docs that contain term)
  - Assume p = a constant
  - Transform equation, get CFW = $\sum_i \log \frac{N}{n_i}$

Probabilistic model

- If relevance info available
  - Use term frequency information from relevant documents to estimate *p* and *q* using the contingency table (after addition of 0.5 to central cells)
- RW = $\sum_i \log \frac{(r_i + 0.5)(N - n_i - R + r_i + 0.5)}{(R - r_i + 0.5)(n_i - r_i + 0.5)}$

Probabilistic model

- No relevance data: using within document term frequency data
- Early work modeled as two types of occurrences
  - Document is about the topic of the term
    - document is *elite* for that term
  - Document not about the topic of the term
  - Modeled as a mixture of two Poisson distributions
  - Very complex formula, difficult to estimate parameters, little performance benefit
  - Simplified equation (with tuning parameters) has similar desired behavior; has been very successful

Probabilistic model

- Summary
  - Based on estimating the probability that a document is relevant to a query
  - Requires various assumptions
  - Like the vector model, calculates a score to be used for relevance and uses a weighted vector to represent the terms in queries and documents
  - Underlies some very successful research prototypes