

Information Retrieval

Spring 2024

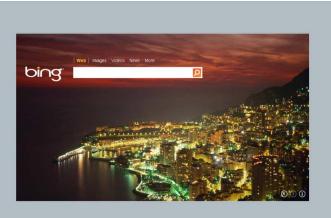
Prof. Oren Kurland

Web search



Web Images Video News Maps more ▾
Google Search I'm Feeling Lucky Advanced Search Help Language Tools

Advertising Programs - Business Solutions - About Google - Go to Google Israel
Make Google Your Homepage!



Modern search engines?



All Chat Images Videos News Maps Social More

what is the technion data science program

The Technion, also known as the Israel Institute of Technology, has several programs related to data science. One of its prestigious flagship programs is the Program in Data Science and Engineering [1](#), which was the first program of its kind in Israel. The Technion's Faculty of Data and Decision Sciences has world-class research in areas such as information retrieval [2](#), machine learning, and decision theory. Additionally, the Israel Data Science Initiative is the coordinating body for the seven Data Science Research Centers at Israeli universities [3](#), including the Technion. However, more specific information about the content and structure of the Technion's data science program would require additional research.

Interesting features

"More like this" "Query by example"

I saw an interesting page on the Web. Can you find me pages that are similar to this page?

What does it mean "similar"?

Personalization in search

- Personalize the web search page
 - What am I interested in?
 - Bias the ranking function according to my preferences
 - My query history

Desktop search

How is it different than Web search?

Image search

How does the search engine find pictures of NYC?

in response to a textual query “NYC”
in response to an example picture

In contrast to image generation

Search+Clustering

The screenshot shows a search results page for "Technion, Haifa". The top navigation bar includes links for "Suggested Site", "Suggested Site", "Web Site Gallery", "Search", and "Results 1-6 of 6 in Technion, Haifa". The search bar contains the query "Technion". Below the search bar is a "Did you mean" section with "Technion" highlighted. The main content area displays six search results, each with a title, a snippet of text, and a link to the full result. The results include various topics related to the Technion, such as Moshe Dayan, Christopher Polyn, and the Technion DRIVE Accelerator.

Federated (Distributed) retrieval

The screenshot shows the Metacrawler search engine interface. At the top is the Metacrawler logo with the tagline "SEARCH THE SEARCH ENGINES!®". Below the logo is a search bar with the placeholder "Search Results from: Google, Yahoo!, bing". Underneath the search bar are buttons for "Web", "Images", "Video", "News", "Yellow Pages", and "White Pages". To the right of these buttons is a red "SEARCH" button. At the bottom of the search bar are links for "Advanced Search" and "Preferences".

Classification

Yahoo! Directory

The screenshot shows the Yahoo! Directory classification tree. The root categories listed are: Arts & Humanities, Business & Economy, Computers & Internet, Education, Entertainment, Government, Health, News & Media, Recreation & Sports, Reference, and Regional. Each category has a brief description and a list of sub-categories.

Filtering

- Automatically detect spam
- Highlight only the news pieces that might interest me
- Novelty detection
 - Highlight only the news pieces that might interest me and that bear content that I haven't seen before

Question Answering: Watson (2011)



From New York Times

Question Answering with LLMs

what topics does the information retrieval course at the Technion cover

As an AI language model, I don't have access to specific information about courses offered at Technion at any given point in time, but I can provide you with a general overview of what an Information Retrieval course might cover:

An Information Retrieval (IR) course generally covers the principles and techniques used for searching, retrieving and organizing large collections of textual data. It usually includes topics such as:

1. Basic concepts of information retrieval, such as indexing, retrieval models, and relevance ranking.
2. Retrieval models, such as Boolean, vector space, and probabilistic models.
3. Text processing techniques, such as tokenization, stemming, and stopword removal.
4. Evaluation metrics for information retrieval systems.
5. Query expansion and relevance feedback.
6. Web search and link analysis.
7. Machine learning techniques for information retrieval, such as classification, clustering, and learning to rank.
8. Applications of information retrieval, such as search engines, recommender systems, and text mining.

Recommendation

From Ynet



Information retrieval

The task of finding/classifying/extracting information that is of interest to a user given a specification of her **information need**

IR applications

- Search
 - Basic search
 - Web retrieval (Google, Yahoo, Bing, A9, Alta Vista)
 - User feedback
 - Cross-lingual retrieval
 - Federated (distributed) search
 - Semi-structured-text retrieval (e.g., XML retrieval)
 - Multi-media retrieval
 - Visualization of retrieved results
- Classification
 - Topic-based classification
 - Sentiment analysis
- Clustering
- Filtering
 - Adaptive filtering
 - Topic detection and tracking
 - Novelty detection
- Question Answering (QA)
 - Fact-based vs. multi-perspective question answering

NLP applications

- Parsing
- Summarization
- Text generation
- Machine translation
- Text extraction
 - entity and relationship extraction
- Textual entailment
- and more ...

The two main (tentative) topics of the course

- Search
- Clustering

and their interactions ...

Text book

Introduction to Information Retrieval, Cambridge University Press. 2008. (Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze)

Book is available online at:

<http://nlp.stanford.edu/IR-book/>

Some of the lecture slides will be adapted from those accompanying the book

Additional text books

- **Information Retrieval in Practice.** W. Bruce Croft, Donald Metzler, and Trevor Strohman, 2010.
- **Modern Information Retrieval (second edition).** R. Baeza-Yates, B. Ribeiro-Neto. Addison-Wesley, 2011.
- **Information Retrieval: Algorithms and Heuristics.** D.A. Grossman, O. Frieder. Springer, 2004.
- **Managing Gigabytes.** I.H. Witten, A. Moffat, T.C. Bell. Morgan Kaufmann, 1999.
- **Language Modeling for Information Retrieval.** W.B. Croft, J. Lafferty. Springer, 2003

Administrative stuff

- Instructor
 - Oren Kurland, Bloomfield 411 kurland@technion.ac.il
 - Office hours: TBD
- Teacher assistant
 - Haya Nachimovsky
- Course web site
 - Moodle
 - You **must** register to Moodle to get course-related announcements
 - Includes lecture slides (the day before the lecture), recitation slides (the day before the recitation), etc.

Grading

- Final exam (80-90 pts, Takef)
 - You cannot pass the course without passing the exam
- HWs (10-20 pts, Takef; 2-4 HW assignments)
 - Can and should do in pairs
- Search engine competition
 - Bonus points
- Search engine optimization competition
 - Bonus points

Academic Integrity

- Any use of material which is not yours, without proper acknowledgment/reference, will result in **failing the course**
- Any use of generative AI tools (e.g., GPT) to solve HW problems is strictly forbidden and will result in automatic failure in the course.

Prerequisites

- Some basic linear algebra
- Some basic probability
- Some knowledge of statistics (and optimization) is a plus
- Know how to program
- Permission from the instructor

The goal

- Get acquainted with the basics of the information retrieval field
- Acquire tools and background required for reading scientific papers
- Understand how many of the tools you use nowadays might be designed
- Have fun!

Introduction to search engines

Slides adapted from the companion website to the "Introduction to IR" book by Manning, Raghavan and Schutze

IR vs. databases: Structured vs unstructured data

- Structured data tends to refer to information in "tables"

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Typically allows numerical range and exact match (for text) queries, e.g.,
Salary < 60000 AND Manager = Smith.

Unstructured data in 1650

- Which plays of Shakespeare contain the words **Brutus AND Caesar** but **NOT Calpurnia**?
- One could *grep* all of Shakespeare's plays for **Brutus** and **Caesar**, then strip out lines containing **Calpurnia**?
 - Slow (for large corpora)
 - **NOT Calpurnia** is non-trivial
 - Other operations (e.g., find the word **Romans** near **countrymen**) not feasible
 - Ranked retrieval (best documents to return)
 - later lectures

Term-document incidence

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Brutus AND Caesar but NOT Calpurnia

1 if play contains word, 0 otherwise

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for **Brutus**, **Caesar** and **Calpurnia** (complemented) → bitwise AND.
- $110100 \text{ AND } 110111 \text{ AND } 101111 = 100100$.

Answers to query

■ Antony and Cleopatra, Act III, Scene ii

- Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,
When Antony found Julius **Caesar** dead,
He cried almost to roaring; and he wept
When at Philippi he found **Brutus** slain.

■ Hamlet, Act III, Scene ii

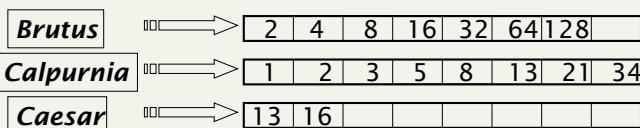
- Lord Polonius: I did enact Julius **Caesar** I was killed i' the Capitol; **Brutus** killed me.

Bigger corpora

- Say there are $m = 500K$ ***distinct*** terms and 1,000,000 documents
 - Huge matrix!
 - But, the matrix is extremely sparse
 - The majority of entries in the matrix are zeroes
 - What's a better representation?
 - We only record the 1 positions.

Inverted index

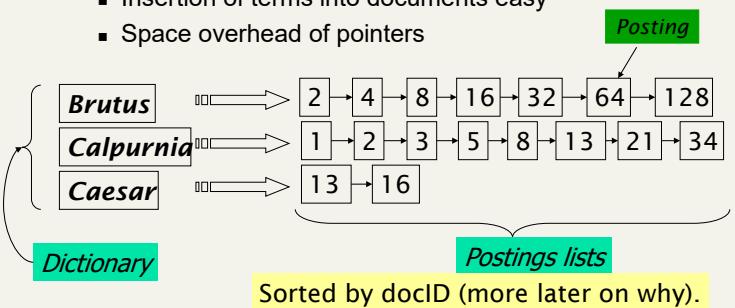
- For each term T , we must store a list of all documents that contain T .
 - Do we use an array or a list for this?



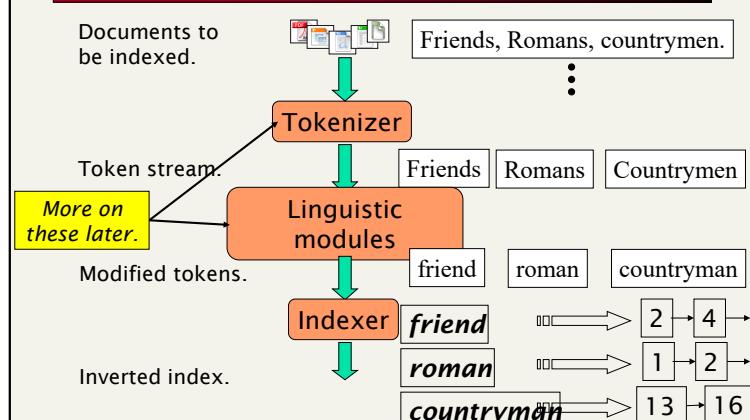
What happens if the word *Caesar* is added to document 14?

Inverted index

- Linked lists generally preferred to arrays
 - Dynamic space allocation
 - Insertion of terms into documents easy
 - Space overhead of pointers

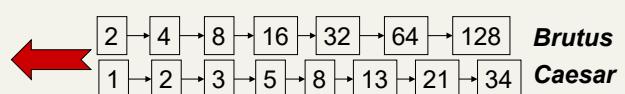


Inverted index construction



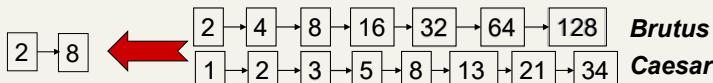
Query processing: AND

- Consider processing the query:
Brutus AND Caesar
 - Locate **Brutus** in the Dictionary;
 - Retrieve its postings.
 - Locate **Caesar** in the Dictionary;
 - Retrieve its postings.
 - “Merge” the two postings:



The merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries



If the list lengths are x and y , the merge takes $O(x+y)$ operations.

Crucial: postings sorted by docID.

Boolean queries: Exact match

- The Boolean Retrieval model can be used to ask a query that is a Boolean expression:
 - Boolean Queries are queries using **AND**, **OR** and **NOT** to join query terms
 - Views each document as a set of words
 - Is precise: document matches condition or not.
- Primary commercial retrieval tool for 3 decades.
- Professional searchers (e.g., lawyers) still like Boolean queries:
 - You know exactly what you're getting.

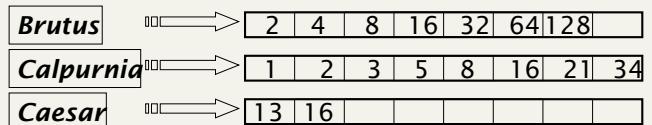
Example: WestLaw

<http://www.westlaw.com/>

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- Tens of terabytes of data; 700,000 users
- Majority of users *still* use boolean queries
 - Precision, transparency and control
- Long, precise queries; proximity operators; incrementally developed; not like web search
- Professional searchers often like Boolean search:
 - But that doesn't mean they actually work better....

Query optimization

- What is the best order for query processing?
- Consider a query that is an **AND** of t terms.
- For each of the t terms, get its postings, then **AND** them together.

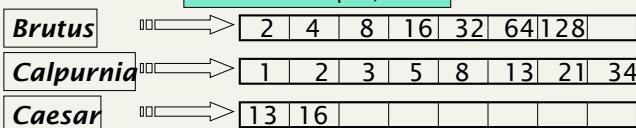


Query: **Brutus AND Calpurnia AND Caesar**

Query optimization example

- Process in order of increasing doc-freq:
 - start with smallest set, then keep cutting further.*

This is why we keep term df in the index (we also keep cf)



Execute the query as (**Caesar AND Brutus**) AND **Calpurnia**.

More general optimization

- e.g., (**hello OR goodbye**) AND (**morning OR evening**)
- Get freq's for all terms.
- Estimate the size of each **OR** by the sum of its freq's (conservative).
- Process in increasing order of **OR** sizes.

What's ahead in IR? Beyond term search

- What about phrases?
 - **Stanford University**
- Proximity: Find **Gates NEAR Microsoft**.
 - Need index to capture position information in docs. More later.

Evidence accumulation

- 1 vs. 0 occurrence of a search term
 - 2 vs. 1 occurrence
 - 3 vs. 2 occurrences, etc.
 - Usually more seems better
- Need term frequency information in docs

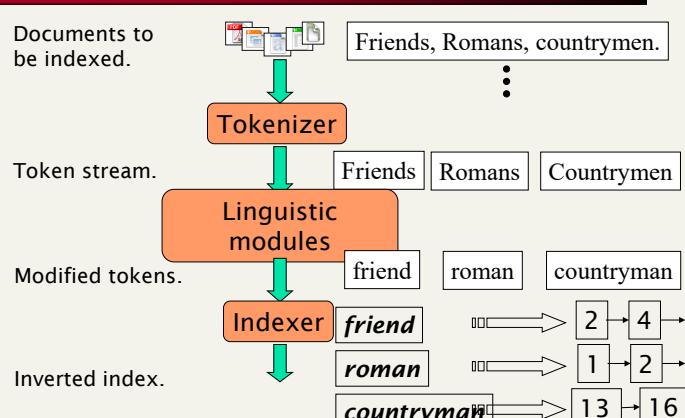
Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Often we want to rank/group results

The indexing pipeline

Slides adopted from the companion website to the "Introduction to IR" book by Manning, Raghavan and Schutze

Recall basic indexing pipeline



Parsing a document

- What format is it in?
 - pdf/word/excel/html?
- What language is it in?
- What character set is in use?

Each of these is a classification problem

But these tasks are often done heuristically ...

Complications: Format/language

- Documents being indexed can include docs from many different languages
 - A single index may have to contain terms of several languages.
 - Sometimes a document or its components can contain multiple languages/format
 - French email with a German pdf attachment.
 - What is a unit document?
 - A file?
 - An email? (Perhaps one of many in an mbox.)
 - An email with 5 attachments?
 - A group of files (PPT or LaTeX in HTML)

Tokenization

- Input: “*Friends, Romans and Countrymen*”
 - Output: Tokens
 - *Friends*
 - *Romans*
 - *Countrymen*
 - Each such token is now a candidate for an index entry, after further processing
 - Described later
 - But what are valid tokens to emit?

Tokenization

- Issues in tokenization:
 - *Finland's capital* →
Finland? *Finlands?* *Finland's?*
 - *Hewlett-Packard* →
Hewlett and *Packard* as two tokens?
 - *State-of-the-art*: break up hyphenated sequence.
 - *San Francisco*: one token or two? How do you decide it is one token?

Numbers

- 3/12/91 Mar. 12, 1991
 - 55 B.C.
 - B-52
 - My PGP key is 324a3df234cb23e
 - 100.2.86.144
 - Often, don't index as text.
 - But often very useful: think about things like looking up error codes/stacktraces on the web
 - (One answer is using n-grams: later in the semester)
 - Will often index "meta-data" separately
 - Creation date, format, etc.

Tokenization: Language issues

- *L'ensemble* → one token or two?
 - *L?* *L'?* *Le?*
 - Want *L'ensemble* to match with *un ensemble*
 - German noun compounds are not segmented
 - Lebensversicherungsgesellschaftsangestellter

Normalization

- Need to “normalize” terms in indexed text as well as query terms into the same form
 - We want to match ***U.S.A.*** and ***USA***
 - We most commonly implicitly define equivalence classes of terms
 - e.g., by deleting periods in a term
 - Alternative is to do asymmetric expansion:
 - Enter: ***window*** Search: ***window, windows***
 - Enter: ***windows*** Search: ***Windows, windows***
 - Enter: ***Windows*** Search: ***Windows***
 - Potentially more powerful, but less efficient

Case folding

- Reduce all letters to lower case
 - exception: upper case (in mid-sentence?)
 - e.g., **General Motors**
 - **Fed** vs. **fed**
 - **SAIL** vs. **sail**
 - Often best to lower case everything, since users will use lowercase regardless of 'correct' capitalization...

Stop words

- With a stop list, you exclude from dictionary entirely the commonest words. Intuition:
 - They have little semantic content: *the, a, and, to, be*
 - They take a lot of space: ~30% of postings
- But the trend is away from doing this:
 - Good compression techniques means the space for including stopwords in a system is very small
 - Good query optimization techniques mean you pay little at query time for including stop words.
 - You need them for:
 - Phrase queries: "King of Denmark"
 - Various titles, etc.: "Let it be", "To be or not to be"
 - "Relational" queries: "flights to London"

Thesauri

- Handle synonyms and homonyms
 - Hand-constructed equivalence classes
 - e.g., **car** = **automobile**
 - **color** = **colour**
- Rewrite to form equivalence classes
- Index such equivalences
 - When the document contains **automobile**, index it under **car** as well (usually, also vice-versa)
- Or expand query?
 - When the query contains **automobile**, look under **car** as well

Lemmatization

- Reduce variant forms to base form
- E.g.,
 - *am, are, is* → *be*
 - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form

Stemming (e.g., Porter's algorithm)

- Reduce terms to their "roots" before indexing
- "Stemming" suggest crude affix chopping
 - language dependent
 - e.g., **automate(s), automatic, automation** all reduced to **automat**.

for example compressed and compression are both accepted as equivalent to compress.

for example compress and compress ar both accept as equival to compress

Language-specificity

- Many of the above features embody transformations that are
 - Language-specific and
 - Often, application-specific
- These are "plug-ins" to the indexing process
- Both open source and commercial plug-ins available for handling these

Phrase queries

- Want to answer queries such as “**stanford university**” – as a phrase
- Thus the sentence “*I went to university at Stanford*” is not a match.
 - The concept of phrase queries has proven easily understood by users; about 10% of web queries are phrase queries
- No longer suffices to store only $\langle term : docs \rangle$ entries

A first attempt: Biword indexes

- Index every consecutive pair of terms in the text as a phrase
- For example the text “Friends, Romans, Countrymen” would generate the biwords
 - friends romans**
 - romans countrymen**
- Each of these biwords is now a dictionary term
- Two-word phrase query-processing is now immediate.

Longer phrase queries

- Longer phrases are processed as we did with wild-cards:
- stanford university palo alto** can be broken into the Boolean query on biwords:
stanford university AND university palo AND palo alto

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.

Can have false positives!

Solution 2: Positional indexes

- Store, for each **term**, entries of the form:
 \langle number of docs containing **term**:
doc1: position1, position2 ... ;
doc2: position1, position2 ... ;
etc. \rangle

Positional index example

$\langle be : 993427;$
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, ...>

Which of docs **1,2,4,5** could contain “**to be or not to be**”?

- Can compress position values/offsets
- This expands postings storage *substantially*
- Nevertheless, it is now commonly used because of the power and usefulness of phrase and proximity queries ... whether used explicitly or implicitly in a ranking retrieval system.

Processing a phrase query

- Extract inverted index entries for each distinct term: **to**, **be**, **or**, **not**.
- Merge their **doc:position** lists to enumerate all positions with “**to be or not to be**”.
 - to**:
 - 2:1,17,74,222,551; **4:8,16,190,429,433**; 7:13,23,191; ...
 - be**:
 - 1:17,19; **4:17,191,291,430,434**; 5:14,19,101; ...
- Same general method for proximity searches

Ranking, term weighting

Scoring

- Thus far, our queries have all been Boolean
 - Docs either match or not
 - Exact match approach
- Good for expert users with precise understanding of their needs and the corpus
- Not good for (the majority of) users with poor Boolean formulation of their needs
- Most users don't want to wade through 1000's of results – cf. use of web search engines

Scoring

- We wish to return in order the documents most likely to be useful to the searcher
 - Best match approach
- How can we rank order the docs in the corpus with respect to a query?
- Assign a score – say in [0,1]
 - for each doc on each query
- Begin with a perfect world – no spammers
 - Nobody stuffing keywords into a doc to make it match queries

Full text queries

- Think of the Boolean query **bill OR rights**
- Most users more likely to type **bill rights** or **bill of rights**
 - How do we interpret these *full text* queries?
 - No Boolean connectives
 - Of several query terms some may be missing in a doc
 - Only some query terms may occur in the title, etc.

Incidence matrices

- Recall: Document is binary vector X in $\{0,1\}^V$
 - Query is a vector
- Score: Overlap measure: (between the query and the document)

$$|X \cap Y|$$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Example

- On the query **Antony Brutus Calpumia**, Shakespeare's *Julius Caesar* has a score of 3
- All other Shakespeare plays have a score of maximum 2
- Thus in a rank order, *Julius Caesar* would come out top

Overlap matching

- Doesn't consider:
 - Term frequency in document
 - Term scarcity in collection (document mention frequency)
 - recall stopwords (a, and, or, then)
 - Length of documents
 - score not normalized

Overlap matching

- One can normalize in various ways:
 - Jaccard coefficient:
$$|X \cap Y| / |X \cup Y|$$
 - Cosine measure:
$$|X \cap Y| / \sqrt{|X| \times |Y|}$$

Scoring: query term occurrences in documents

- Next idea: if a document talks about a topic *more*, then it is a better match
- This applies even when we only have a single query term.
- A document is relevant if it contains many occurrences of the query terms
- This leads to the idea of term weighting.

Term weighting

Term-document count matrices

- Consider the number of occurrences of a term in a document:
 - Bag of words model
 - Document is a vector in \mathbb{N}^v : a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words view of a doc

- Thus the doc
 - **John is quicker than Mary.**
- is indistinguishable from the doc
- **Mary is quicker than John.**

Which of the indexes discussed so far distinguish these two docs?

Digression: terminology

- **WARNING:** In a lot of IR literature, “frequency” is used to mean “count”
 - Thus *term frequency* in IR literature is used to mean *number of occurrences* in a doc
 - Not divided by document length (which would actually make it a frequency)
- We will conform to this misnomer
 - In saying term frequency we mean the number of occurrences of a term in a document.

Term frequency *tf*

- Long docs are favored because they’re more likely to contain query terms
- Can fix this to some extent by normalizing for document length
- But is raw *tf* the right measure?

Weighting term frequency: *tf*

- What is the relative importance of
 - 0 vs. 1 occurrence of a term in a doc
 - 1 vs. 2 occurrences
 - 2 vs. 3 occurrences ...
- Unclear: while it seems that more is better, a lot isn’t proportionally better than a few
 - Can just use raw *tf*
 - Another option commonly used in practice:
 $wf_{t,d} = \log(tf_{t,d} + 1)$

Retrieval score computation

- Score for a query q = sum over terms t in q :
$$score(d; q) = \sum_{t \in q} wf_{t,d}$$
- [Note: 0 if no query terms in document]
- Still doesn’t consider term scarcity in collection (**Antony** is rarer than **and**)

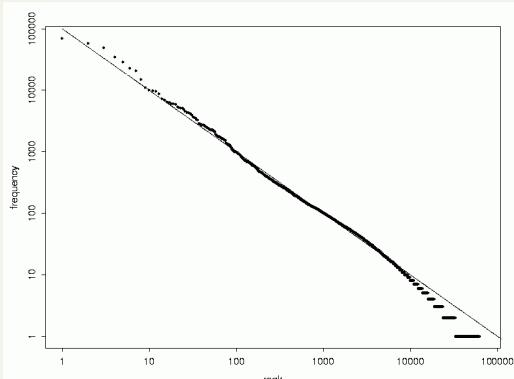
Weighting should depend on the term overall

- Which of these tells you more about a doc?
 - 10 occurrences of *hernia*?
 - 10 occurrences of *and*?
- Would like to attenuate the weight of a common term
 - But what is “common”?
- Suggest looking at collection frequency (*cf*)
 - The total number of occurrences of the term in the entire collection of documents

Zipf’s law

- The i th most frequent term has frequency proportional to $1/i$.
 - This is a power law
- Let this frequency be c/i .
- Then $\sum_{i=1}^{m=500,000} c/i = 1$.
- The k th Harmonic number is $H_k = \sum_{i=1}^k 1/i$.
- Thus $c = 1/H_m$, which is $\sim 1/\ln m = 1/\ln(500k) \sim 1/13$.
- So the i th most frequent term has frequency roughly $1/13i$.
- Content is in the rare terms ...

Zipf's law log-log plot



Back to Document frequency

- df = number of docs in the corpus containing the term

Word	cf	df
ferrari	10422	17
insurance	10440	3997

- Document/collection frequency weighting is only possible in known (static) collection.
- So how do we make use of df ?

tf x idf term weights

- tf x idf measure combines:
 - term frequency (tf)
 - or wf , some measure of term density in a doc
 - inverse document frequency (idf)
 - measure of informativeness of a term: its rarity across the whole corpus
 - could just be raw count of number of documents the term occurs in ($idf_i = 1/df_i$)
 - but by far the most commonly used version is:

$$idf_i = \log\left(\frac{n}{df_i}\right)$$

Summary: tf x idf (or tf.idf)

- Assign a tf.idf weight to each term i in each document d

$$w_{i,d} = tf_{i,d} \times \log(n / df_i)$$

What is the wt of a term that occurs in all of the docs?

$tf_{i,d}$ = frequency of term i in document j

n = total number of documents

df_i = the number of documents that contain term i

- Increases with the number of occurrences *within* a doc
- Increases with the rarity of the term *across* the whole corpus

Real-valued term-document matrices

- Function (scaling) of count of a word in a document:
 - Bag of words model
 - Each is a vector in \mathbb{R}^v
 - Here log-scaled $tf.idf$

Note can be >1!

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	13.1	11.4	0.0	0.0	0.0	0.0
Brutus	3.0	8.3	0.0	1.0	0.0	0.0
Caesar	2.3	2.3	0.0	0.5	0.3	0.3
Calpurnia	0.0	11.2	0.0	0.0	0.0	0.0
Cleopatra	17.7	0.0	0.0	0.0	0.0	0.0
mercy	0.5	0.0	0.7	0.9	0.9	0.3
worser	1.2	0.0	0.6	0.6	0.6	0.0

Documents as vectors

- Each doc j can now be viewed as a vector of $wf \times idf$ values, one component for each term
- So we have a vector space
 - terms are axes
 - docs live in this space
 - even with stemming, may have 20,000+ dimensions

The vector space model

Some slides are adapted from the companion website to the "Introduction to IR" book by Manning, Raghavan and Schütze

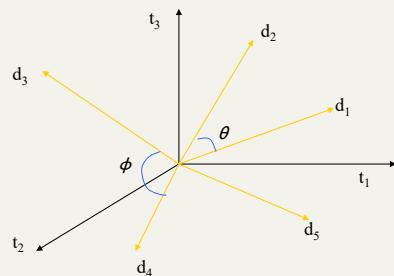
Documents as vectors

- Each doc d can now be viewed as a vector of $wf \times idf$ values, one component for each term
- So we have a vector space
 - terms are axes
 - docs live in this space
 - even with stemming, may have 50,000+ dimensions

Why turn docs into vectors?

- First application: Query-by-example
 - Given a doc d , find others "like" it.
- Now that d is a vector, find vectors (docs) "near" it.
- If I like a few documents, how do I use the idea of query-by-example ?

Intuition



Postulate: Documents that are "close together" in the vector space talk about the same things.

Desiderata for proximity

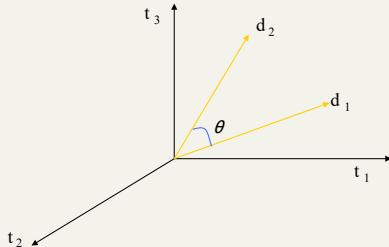
- If d_1 is near d_2 , then d_2 is near d_1 .
- If d_1 is near d_2 , and d_2 is near d_3 , then d_1 is not far from d_3 .
- No doc is closer to d than d itself.

First cut

- Idea: Distance between d_1 and d_2 is the length of the vector $|d_1 - d_2|$.
 - Euclidean distance
- Why is this not a great idea?

Cosine similarity

- Distance between vectors d_j and d_k captured by the cosine of the angle between them.
- Note – this is *similarity*, not distance
 - No triangle inequality for similarity.



Cosine similarity

$$\text{sim}(d_j, d_k) = \frac{\vec{d}_j \cdot \vec{d}_k}{\|d_j\| \|d_k\|} = \frac{\sum_{i=1}^n w_{i,j} w_{i,k}}{\sqrt{\sum_{i=1}^n w_{i,j}^2} \sqrt{\sum_{i=1}^n w_{i,k}^2}}$$

- Cosine of angle between two vectors
- The denominator involves the lengths of the vectors.

Normalization

Normalized vectors

- For normalized vectors, the cosine is simply the dot product:

$$\cos(\vec{d}_j, \vec{d}_k) = \vec{d}_j \cdot \vec{d}_k$$

Example

- Docs: Austen's *Sense and Sensibility*, *Pride and Prejudice*; Bronte's *Wuthering Heights*. tf weights

	SaS	PaP	WH
<i>affection</i>	115	58	20
<i>jealous</i>	10	7	11
<i>gossip</i>	2	0	6
	SaS	PaP	WH
<i>affection</i>	0.996	0.993	0.847
<i>jealous</i>	0.087	0.120	0.466
<i>gossip</i>	0.017	0.000	0.254

- $\cos(\text{SAS}, \text{PAP}) = .996 \times .993 + .087 \times .120 + .017 \times 0.0 = 0.999$
- $\cos(\text{SAS}, \text{WH}) = .996 \times .847 + .087 \times .466 + .017 \times .254 = 0.889$

Exercise

- Euclidean distance between vectors:

$$|d_j - d_k| = \sqrt{\sum_{i=1}^n (d_{i,j} - d_{i,k})^2}$$

- Show that, for normalized vectors, Euclidean distance gives the same proximity ordering as the cosine measure

Queries in the vector space model

Central idea: the query as a vector:

- We regard the query as short document
- We return the documents ranked by the closeness of their vectors to the query, also represented as a vector.

$$\text{sim}(d_j, d_q) = \frac{\vec{d}_j \cdot \vec{d}_q}{\|\vec{d}_j\| \|\vec{d}_q\|} = \frac{\sum_{i=1}^n w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^n w_{i,j}^2} \sqrt{\sum_{i=1}^n w_{i,q}^2}}$$

- Note that d_q is very sparse!

Who invented the vector space model ?

- Most references are to
 - **A vector space model for automatic indexing**, G. Salton, A. Wong and C. S. Yang., 1975.
- But, this has prior mentioning
 - **Automatic information organization and retrieval**, Gerard Salton, 1968.

Summary: What's the point of using vector spaces?

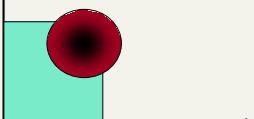
- A well-formed algebraic space for retrieval
- Key: A user's query can be viewed as a (very) short document.
- Query becomes a vector in the same space as the docs.
- Can measure each doc's proximity to it.
- Natural measure of scores/ranking – no longer Boolean.
 - Queries are expressed as bags of words

Interaction: vectors and phrases

- Scoring phrases doesn't fit naturally into the vector space world:
 - **"tangerine trees" "marmalade skies"**
 - Positional indexes don't calculate or store tf.idf information for **"tangerine trees"**
- Biword indexes treat certain phrases as terms
 - For these, we can pre-compute tf.idf.
- Problem: we cannot expect end-user formulating queries to know what phrases are indexed
- Theoretical problem: dimension correlation
- We can use a positional index to boost or ensure phrase occurrence

Vectors and Boolean queries

- Vectors and Boolean queries really don't work together very well
- In the space of terms, vector proximity selects by spheres: e.g., all docs having cosine similarity ≥ 0.5 to the query
- Boolean queries on the other hand, select by (hyper-)rectangles and their unions/intersections



Query language vs. scoring

- May allow user a certain query language, say
 - Free text basic queries
 - Phrase, wildcard etc. in Advanced Queries.
- For scoring (oblivious to user) may use all of the above, e.g. for a free text query
 - Highest-ranked hits have query as a phrase
 - Next, docs that have all query terms near each other
 - Then, docs that have some query terms, or all of them spread out, with tf x idf weights for scoring

Pivoted document length normalization

- Singhal Buckley and Mitra SIGIR '96

Why do we need normalization for term weights ?

- Higher term frequencies
 - Long documents usually use the same terms repeatedly
- More terms
 - Long documents have numerous different terms

Normalization schemes

- Cosine normalization

$$\sqrt{\sum_{i=1}^n tf_{i,j}^2}$$

- Attacks both reasons for normalization (higher *tf*s, more terms)

Normalization schemes (2)

- Maximum tf normalization
- Smart system (Cornell University)

$$\text{newTF}_{i,j} = 0.5 + 0.5 * \frac{tf_{i,j}}{\max_k tf_{k,j}}$$

- INQUERY system (UMASS)

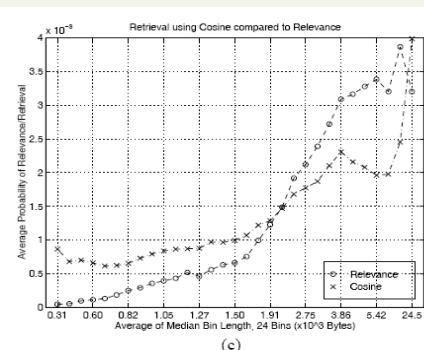
$$\text{newTF}_{i,j} = 0.4 + 0.6 * \frac{tf_{i,j}}{\max_k tf_{k,j}}$$

but only compensates for high *tf*s

Goal

- Learn how a retrieval pattern deviates from the relevance pattern for a given normalization scheme

Likelihood of relevance/retrieval



Observation from the plot

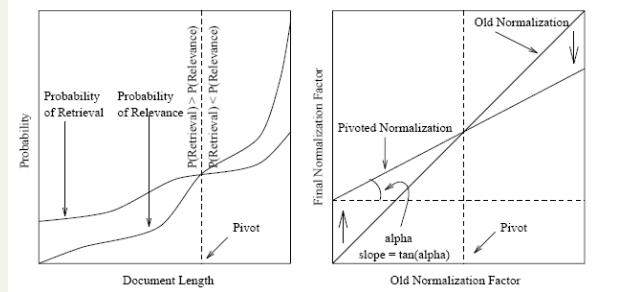
- Short documents are retrieved with higher probability than their probability of relevance
- It is less likely to retrieve longer documents as compared to the likelihood of their relevance

Cosine normalization tends to favor short documents in retrieval !

Pivoted normalization scheme

- Insight: to boost the chances of retrieval for documents of a certain length, we should lower the value of the normalization factor for those documents, and vice versa

Pivoted normalization



The math

pivoted_normalization =

$$(1.0 - \text{slope}) * \text{pivot} + \text{slope} * \text{old_normalization}$$

Removing a parameter

$$\frac{\text{tf.idf}}{(1.0 - \text{slope}) * \text{pivot} + \text{slope} * \text{old_normalization}}$$

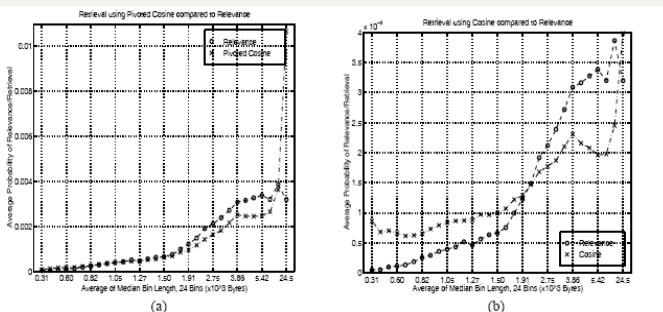
is rank equivalent (using the inner product) to

$$\frac{\text{tf.idf} * (1.0 - \text{slope}) * \text{pivot}}{(1.0 - \text{slope}) * \text{pivot} + \text{slope} * \text{old_normalization}} =$$

$$\frac{\text{tf.idf}}{1 + \frac{\text{slope}}{(1.0 - \text{slope}) * \text{pivot}} * \text{old_normalization}} =$$

$$\frac{\text{tf.idf}}{1 + c * \text{old_normalization}}$$

(Pivoted) Cosine normalization



Retrieval performance

- Improvements are obtained if pivoted normalization is used
- But how do we measure performance ?
 - Next lecture ...

Evaluation

Some slides are adapted from the companion website to the "Introduction to IR" book by Manning, Raghavan and Schütze

This lecture

- Results summaries:
 - Making our good results usable to a user
- How do we know if our results are any good?
 - Evaluating a search engine
 - Benchmarks
 - Precision and recall

Results summaries

Summaries

- Having ranked the documents matching a query, we wish to present a results list
- Most commonly, the document title plus a short summary
- The title is typically automatically extracted from document metadata
- What about the summaries?

Summaries

- Two basic kinds:
 - Static
 - Dynamic
- A static summary of a document is always the same, regardless of the query that hit the doc
- Dynamic summaries are *query-dependent*: attempt to explain why the document was retrieved for the query at hand

Static summaries

- In typical systems, the static summary is a subset of the document
- Simplest heuristic: the first 50 (or so – this can be varied) words of the document
 - Summary cached at indexing time
- More sophisticated: extract from each document a set of “key” sentences
 - Simple NLP heuristics to score each sentence
 - Summary is made up of top-scoring sentences.
- Most sophisticated: NLP used to synthesize a summary
 - Seldom used in IR; cf. text summarization work

Dynamic summaries

- Present one or more “windows” within the document that contain several of the query terms
 - “KWIC” snippets: Keyword in Context presentation
- Generated in conjunction with scoring
 - If query found as a phrase, the/some occurrences of the phrase in the doc
 - If not, windows within the doc that contain multiple query terms

Generating dynamic summaries

- If we have only a positional index, we cannot (easily) reconstruct context surrounding hits
 - Why ?
- If we *cache the documents* at index time, we can run the window through it, cueing to hits found in the positional index
 - E.g., positional index says “the query is a phrase in position 4378” so we go to this position in the cached document and stream out the content
- Most often, cache a fixed-size prefix of the doc
 - Note: Cached copy can be outdated

Dynamic summaries

- Producing good dynamic summaries is a tricky optimization problem
 - The real estate for the summary is normally small and fixed
 - Want short item, so show as many KWIC matches as possible, and perhaps other things like title
 - Want snippets to be long enough to be useful
 - Want linguistically well-formed snippets: users prefer snippets that contain complete phrases
 - Want snippets maximally informative about doc
- But users really like snippets, even if they complicate IR system design

Evaluating search engines

Measures for a search engine

- How fast does it index
 - Number of documents/hour
 - (Average document size)
- How fast does it search
 - Latency as a function of index size
 - Efficiency
- Expressiveness of query language
 - Ability to express complex information needs
 - Speed on complex queries

Measures for a search engine

- All of the preceding criteria are *measurable*: we can quantify speed/size; we can make expressiveness precise
- The key measure: user happiness
 - What is this?
 - Speed of response/size of index are factors
 - But blindingly fast, useless answers won’t make a user happy
- Need a way of quantifying user happiness
 - Effectiveness

Measuring user happiness

- Issue: who is the user we are trying to make happy?
 - Depends on the setting
- Web engine: user finds what they want and return to the engine
 - Can measure rate of return users
- eCommerce site: user finds what they want and make a purchase
 - Is it the end-user, or the eCommerce site, whose happiness we measure?
 - Measure time to purchase, or fraction of searchers who become buyers?

Measuring user happiness

- Enterprise (company/govt/academic): Care about "user productivity"
 - How much time do my users save when looking for information?
 - Many other criteria having to do with breadth of access, secure access, etc.

Happiness: elusive to measure

- Commonest proxy: *relevance* of search results
- But how do you measure relevance?
- We will detail a methodology here, then examine its issues
- Relevant measurement requires 3 elements:
 1. A benchmark document collection
 2. A benchmark suite of queries
 3. A binary assessment of either Relevant or Irrelevant for each query-doc pair
 - Some work on more-than-binary, but not the standard

Evaluating an IR system

- Note: the **information need** is translated into a **query**
- Relevance is assessed relative to the **information need** *not* the **query**
- E.g., Information need: *I'm looking for information on whether drinking red wine is more effective at reducing your risk of heart attacks than white wine.*
- Query: **wine red white heart attack effective**
- You evaluate whether the doc addresses the information need, not whether it has those words

Standard relevance benchmarks

- TREC - National Institute of Standards and Testing (NIST) has run a large IR test bed for many years
- Reuters and other benchmark doc collections used
- "Retrieval tasks" specified
 - sometimes as queries
- Human experts mark, for each query and for each doc, Relevant or Irrelevant
 - or at least for subset of docs that some system returned for that query ...

The pooling method of TREC

- Look at the top-1000 retrieved documents of each participating engine, pool them, and have humans assess them
 - Advantages ?
 - Disadvantages ?

Unranked retrieval evaluation: Precision and Recall

- **Precision:** fraction of retrieved docs that are relevant = $P(\text{relevant}|\text{retrieved})$
- **Recall:** fraction of relevant docs that are retrieved = $P(\text{retrieved}|\text{relevant})$

	Relevant	Not Relevant
Retrieved	tp	fp
Not Retrieved	fn	tn

- Precision $P = tp/(tp + fp)$
- Recall $R = tp/(tp + fn)$

Accuracy

- Given a query an engine classifies each doc as "Relevant" or "Irrelevant".
- Accuracy of an engine: the fraction of these classifications that is correct.
- Why is this not a very useful evaluation measure in IR?

Why not just use accuracy?

- How to build a 99.9999% accurate search engine on a low budget....



- People doing information retrieval *want to find something* and have a certain tolerance for junk.

Precision/Recall

- You can get high recall (but low precision) by retrieving all docs for all queries!
- Recall is a non-decreasing function of the number of docs retrieved
- In a good system, precision decreases as either number of docs retrieved or recall increases
 - A fact with strong empirical confirmation

Difficulties in using precision/recall

- Should average over large corpus/query ensembles
- Need human relevance assessments
 - People aren't reliable assessors
- Assessments have to be binary
 - Nuanced assessments?
- Heavily skewed by corpus/authorship
 - Results may not translate from one domain to another

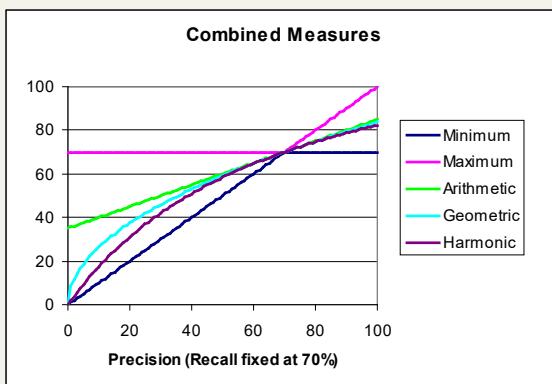
A combined measure: F

- Combined measure that assesses this tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
 - i.e., with $\beta = 1$ or $\alpha = \frac{1}{2}$
- Harmonic mean is a conservative average
 - See CJ van Rijsbergen, *Information Retrieval*

F_1 and other averages

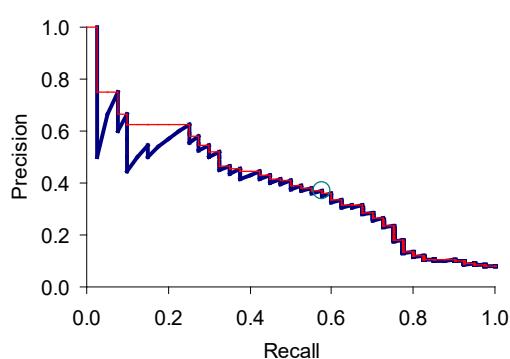


Evaluating ranked results

- Evaluation of ranked results:

- The system can return any number of results
- By taking various numbers of the top returned documents (levels of recall), the evaluator can produce a *precision-recall curve*

A precision-recall curve

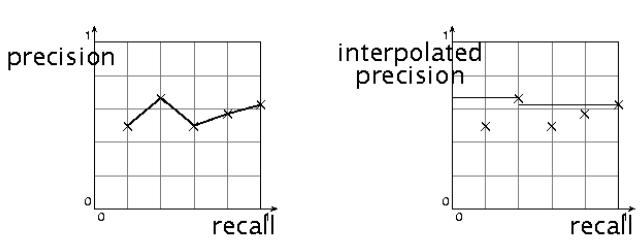


Averaging over queries

- A precision-recall graph for one query isn't a very sensible thing to look at
- You need to average performance over a whole bunch of queries.
- But there's a technical issue:
 - Precision-recall calculations place some points on the graph
 - How do you determine a value (interpolate) between the points?

Interpolated precision

- Idea: if locally precision increases with increasing recall, then you should get to count that...
- So you max of precisions to right of value

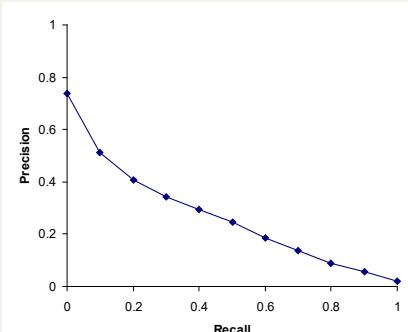


Average interpolated precision

- 11-point interpolated average precision
 - The standard measure in the TREC competitions: you take the precision at 11 levels of recall varying from 0 to 1 by tenths of the documents, using interpolation (the value for 0 is always interpolated!), and average them over queries
 - Evaluates performance at all recall levels

Typical (good) 11 point precisions

- SabIR/Cornell 8A1 11pt precision from TREC 8 (1999)



Precision at top ranks

- Graphs are good, but people want summary measures!
- Precision at fixed retrieval level (prec@k)
 - Perhaps most appropriate for web search: all people want are good matches on the first one or two results pages
 - But has an arbitrary parameter of k

Yet more evaluation measures...

- Mean average precision (MAP)
 - Average of the precision value obtained for the top k documents, each time a relevant doc is retrieved
 - Avoids interpolation, use of fixed recall levels
 - MAP for query collection is arithmetic average.
 - Macro-averaging: each query counts equally
 - GMAP for query collection is geometric average.
 - Macro-averaging: each query counts equally
- R-precision
 - If have known (though perhaps incomplete) set of relevant documents of size Rel , then calculate precision of top Rel docs returned
 - Perfect system could score 1.0.

Variance

- For a test collection, it is usual that a system does crummily on some information needs (e.g., MAP = 0.1) and excellently on others (e.g., MAP = 0.7)
- Indeed, it is usually the case that the variance in performance of the same system across queries is much greater than the variance of different systems on the same query.
- That is, there are easy information needs and hard ones!

Creating Test Collections for IR Evaluation

Test Corpora

TABLE 4.3 Common Test Corpora

Collection	NDocs	NQrys	Size (MB)	Term/Doc	Q-D RelAss
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2		24.5
CISI	1460	112	2		46.5
Cranfield	1400	225	2		53.1
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	> 100,000

From corpora to test collections

- Still need
 - Test queries
 - Relevance assessments
- Test queries
 - Must be germane to docs available
 - Best designed by domain experts
 - Random query terms generally not a good idea
- Relevance assessments
 - Human judges, time-consuming
 - Are human panels perfect?

TREC

- TREC Ad Hoc task from first 8 TRECs is standard IR task
 - 50 detailed information needs a year
 - Human evaluation of pooled results returned
 - More recently other related things: Web track, HARD
- A TREC query (TREC 5)

```
<top>
<num> Number: 225
<desc> Description:
What is the main function of the Federal Emergency
Management Agency (FEMA) and the funding level provided
to meet emergencies? Also, what resources are available to
FEMA such as people, equipment, facilities?
</top>
```

Impact of Inter-judge Agreement

- Impact on [absolute](#) performance measure can be significant (0.32 vs 0.39)
- Little impact on ranking of different systems or [relative](#) performance

Critique of pure relevance

- Relevance vs [Marginal Relevance](#) (Carbonell and Goldstein)
 - A document can be redundant even if it is highly relevant
 - Duplicates
 - The same information from different sources
 - Marginal relevance is a better measure of utility for the user.
- Using facts/entities as evaluation units more directly measures true relevance.
- But harder to create evaluation set

Can we avoid human judgment?

- Not really
- Makes experimental work hard
 - Especially on a large scale
- In some very specific settings, can use proxies
- Example below, approximate vector space retrieval
- But once we have test collections, we can reuse them (so long as we don't overtrain too badly)

Query expansion

Some slides are adapted from the companion website to the "Introduction to IR" book by Manning, Raghavan and Schutze

This lecture

- Improving results
 - For high recall. E.g., searching for *aircraft* doesn't match with *plane*; nor *thermodynamic* with *heat*
- Options for improving results...
 - Focus on relevance feedback
 - The complete landscape
 - Global methods
 - Query expansion
 - Thesauri
 - Automatic thesaurus generation
 - Local methods
 - Relevance feedback
 - Pseudo relevance feedback

Relevance Feedback

- Relevance feedback: user feedback on relevance of docs in initial set of results
 - User issues a (short, simple) query
 - The **user** marks returned documents as relevant or non-relevant.
 - The **system** computes a better representation of the information need based on feedback.
 - Relevance feedback can go through one or more **iterations**.
- Idea: it may be difficult to formulate a good query when you don't know the collection well, so iterate

Relevance Feedback: Example

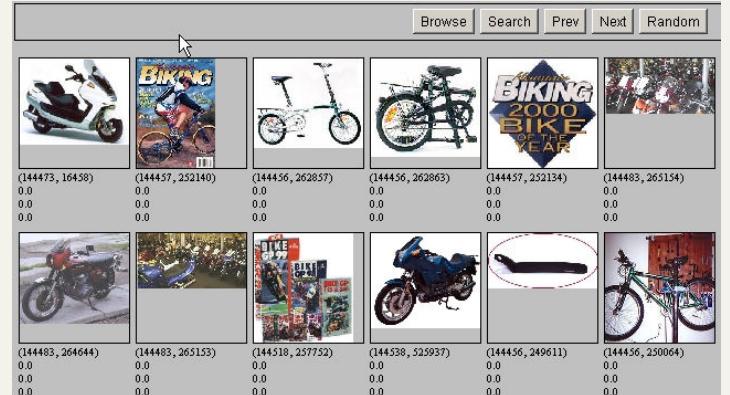
- Image search engine
<http://nayana.ece.ucsb.edu/imsearch/imsearch.html>

The screenshot shows a Netscape browser window with the URL <http://nayana.ece.ucsb.edu/imsearch/imsearch.html>. The page displays a grid of 10 image thumbnails related to bicycles. Below the thumbnails, there is a table with two rows of data for each image, showing numerical values. At the bottom of the page, there is a note about the database size and a search bar with the word 'Search'.

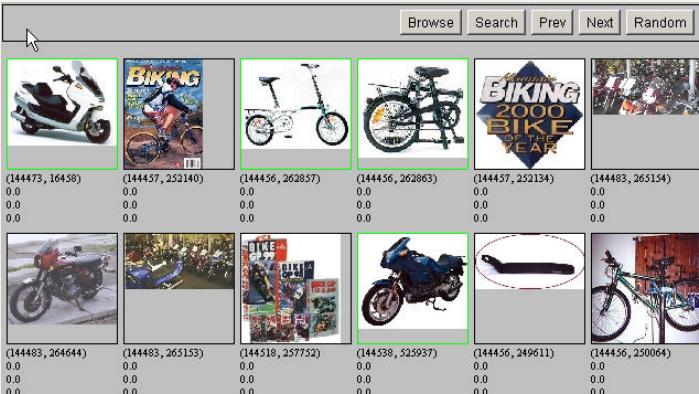
	(144473, 16438)	(144457, 252140)	(144456, 262857)	(144456, 262863)	(144457, 252134)	(144483, 265154)
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0

Designed by [Baris Sumengen](#) and [Shawn Newsam](#)
Powered by JЛАMP2000 (Java, Linux, Apache, Mysql, Perl, Windows2000)

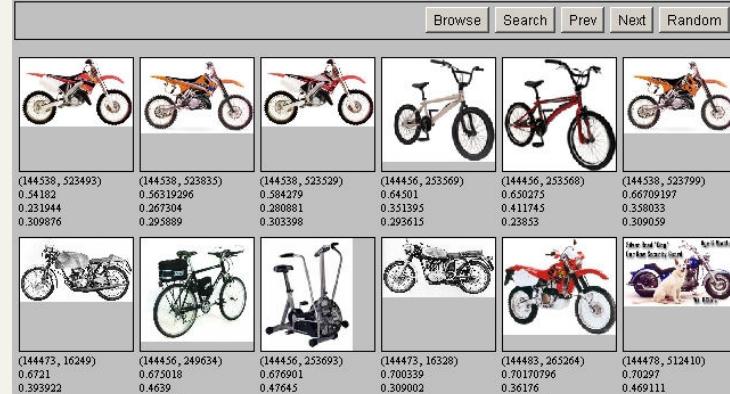
Results for Initial Query



Relevance Feedback



Results after Relevance Feedback



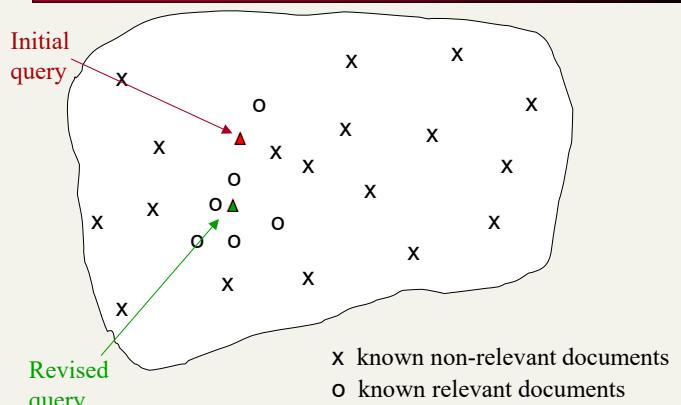
Rocchio 1971 Algorithm (SMART)

- Used in practice:

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

- q_m = modified query vector; q_0 = original query vector; α, β, γ : weights (hand-chosen or set empirically); D_r = set of known relevant doc vectors; D_{nr} = set of known irrelevant doc vectors
- New query moves toward relevant documents and away from irrelevant documents
- Tradeoff α vs. β/γ : If we have a lot of judged documents, we want a higher β/γ .
- Term weight can go negative
 - Negative term weights are ignored (set to 0)

Relevance feedback on initial query



Relevance Feedback in vector spaces

- We can modify the query based on relevance feedback and apply standard vector space model.
- Use only the docs that were marked.
- Relevance feedback can improve recall and precision
- Relevance feedback is most useful for increasing *recall* in situations where recall is important
 - Users can be expected to review results and to take time to iterate

Positive vs Negative Feedback

- Positive feedback is more valuable than negative feedback (so, set $\gamma < \beta$; e.g. $\gamma = 0.25$, $\beta = 0.75$).
- Many systems only allow positive feedback ($\gamma=0$).



Relevance Feedback: Assumptions

- A1: User has sufficient knowledge for initial query.
- A2: Relevance prototypes are “well-behaved”.
 - Term distribution in relevant documents will be similar
 - Term distribution in non-relevant documents will be different from those in relevant documents
 - Either: All relevant documents are tightly clustered around a single prototype.
 - Or: There are different prototypes, but they have significant vocabulary overlap.
 - Similarities between relevant and irrelevant documents are small

Violation of A1

- User does not have sufficient initial knowledge.
- Examples:
 - Misspellings (Brittany Spears).
 - Cross-language information retrieval (hígado).
 - Mismatch of searcher’s vocabulary vs. collection vocabulary
 - Cosmonaut/astronaut

Violation of A2

- There are several relevance prototypes.
- Examples:
 - Pop stars that worked at Burger King
- Often: instances of a general concept

Relevance Feedback: Problems

- Why do most search engines not use relevance feedback?

Relevance Feedback: Problems

- Long queries are inefficient for typical IR engine.
 - Long response times for user.
 - High cost for retrieval system.
 - Partial solution:
 - Only reweight certain prominent terms
 - Perhaps top 20 by term frequency
- Users are often reluctant to provide explicit feedback
- It's often harder to understand why a particular document was retrieved after applying relevance feedback



Relevance Feedback Example: Initial Query and Top 8 Results

- Query: New space satellite applications
- Note: want high recall
- + 1. 0.539, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
 - + 2. 0.533, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
 - 3. 0.528, 04/04/90, Science Panel Backs NASA Satellite Plan, But Urges Launches of Smaller Probes
 - 4. 0.526, 09/09/91, A NASA Satellite Project Accomplishes Incredible Feat: Staying Within Budget
 - 5. 0.525, 07/24/90, Scientist Who Exposed Global Warming Proposes Satellites for Climate Research
 - 6. 0.524, 08/22/90, Report Provides Support for the Critics Of Using Big Satellites to Study Climate
 - 7. 0.516, 04/13/87, Arianespace Receives Satellite Launch Pact From Telesat Canada
 - + 8. 0.509, 12/02/87, Telecommunications Tale of Two Companies

Relevance Feedback Example: Expanded Query

■ 2.074 new	15.106 space
■ 30.816 satellite	5.660 application
■ 5.991 nasa	5.196 eos
■ 4.196 launch	3.972 aster
■ 3.516 instrument	3.446 arianespace
■ 3.004 bundespost	2.806 ss
■ 2.790 rocket	2.053 scientist
■ 2.003 broadcast	1.172 earth
■ 0.836 oil	0.646 measure

Top 8 Results After Relevance Feedback

- + 1. 0.513, 07/09/91, NASA Scratches Environment Gear From Satellite Plan
- + 2. 0.500, 08/13/91, NASA Hasn't Scrapped Imaging Spectrometer
- 3. 0.493, 08/07/89, When the Pentagon Launches a Secret Satellite, Space Sleuths Do Some Spy Work of Their Own
- 4. 0.493, 07/31/89, NASA Uses 'Warm' Superconductors For Fast Circuit
- + 5. 0.492, 12/02/87, Telecommunications Tale of Two Companies
- 6. 0.491, 07/09/91, Soviets May Adapt Parts of SS-20 Missile For Commercial Use
- 7. 0.490, 07/12/88, Gaping Gap: Pentagon Lags in Race To Match the Soviets In Rocket Launchers
- 8. 0.490, 06/14/90, Rescue of Satellite By Space Agency To Cost \$90 Million

Evaluation of relevance feedback strategies

- Use q_0 and compute precision and recall graph
- Use q_m and compute precision recall graph
 - Assess on all documents in the collection
 - Spectacular improvements, but ... it's cheating!
 - Partly due to known relevant documents ranked higher
 - Must evaluate with respect to documents not seen by user
 - Use documents in residual collection (set of documents minus those assessed relevant)
 - Measures usually then lower than for original query
 - But a more realistic evaluation
 - Relative performance can be validly compared
- Empirically, one round of relevance feedback is often very useful. Two rounds is sometimes marginally useful.

Relevance Feedback Summary

- Relevance feedback has been shown to be very effective at improving relevance of results.
 - Requires enough judged documents, otherwise it's unstable (≥ 5 recommended)
 - Requires queries for which the set of relevant documents is medium to large
- Full relevance feedback is painful for the user.
- Full relevance feedback is not very efficient in most IR systems.
- Other types of interactive retrieval may improve relevance by as much with less work.

The complete landscape

- Global methods
 - Query expansion/reformulation
 - Thesauri (or WordNet)
 - Automatic thesaurus generation
- Local methods
 - Relevance feedback
 - Pseudo relevance feedback

Query Expansion

- In relevance feedback, users give additional input (relevant/non-relevant) on **documents**, which is used to reweight terms in the documents
- In query expansion, users give additional input (good/bad search term) on **words or phrases**.

Query Expansion: Example

YOU ARE HERE > Home > My InfoSpace > Meta-Search > Web Search Results

Web Search Results

Your Search

jaguar Select: Web

Yellow Pages White Pages Classifieds

Are you looking for?

Jacksonville Jaguars Jaguar Car Black Jaguar Jaguar XK8
Wild Jaguars Jaguar Jaguar Accessories Jaguar Automobile

Also: see www.altavista.com, www.teoma.com

Types of Query Expansion

- **Global Analysis:** (static; of **all documents** in collection)
 - Controlled vocabulary
 - Maintained by editors (e.g., medline)
 - Manual thesaurus
 - E.g. MedLine: physician, syn: doc, doctor, MD, medico
 - Automatically derived thesaurus
 - (co-occurrence statistics)
 - Refinements based on query log mining
 - Common on the web
- **Local Analysis:** (dynamic)
 - Analysis of documents in **result set**

Controlled Vocabulary

The screenshot shows the PubMed search interface. The search bar contains the query "cancer". Below the search bar, the PubMed Query is displayed as: ("neoplasms"[MeSH Terms] OR cancer[Text Word]). The interface includes various navigation links like About Entrez, Text Version, Entrez PubMed Overview, Help | FAQ, Tutorial, New/Noteworthy, E-Utilities, and PubMed Services.

Thesaurus-based Query Expansion

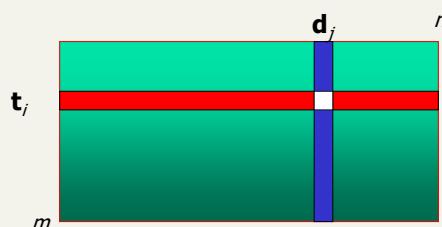
- This doesn't require user input
- For each term, t , in a query, expand the query with synonyms and related words of t from the thesaurus
 - shipment → shipment cargo freight
- May weight added terms less than original query terms.
- Generally increases recall.
- Widely used in many science/engineering fields
- May significantly decrease precision, particularly with ambiguous terms.
 - "interest rate" → "interest rate fascinate evaluate"
- There is a high cost of manually producing a thesaurus
 - And for updating it for scientific changes

Automatic Thesaurus Generation

- Attempt to generate a thesaurus automatically by analyzing the collection of documents
- Two main approaches
 - Co-occurrence based (co-occurring words are more likely to be similar)
 - Shallow analysis of grammatical relations
 - Entities that are grown, cooked, eaten, and digested are more likely to be food items.

Co-occurrence Thesaurus

- Simplest way to compute one is based on term-term similarities in $C = AA^T$ where A is term-document matrix.
- $w_{ij} = (\text{normalized}) \text{ weighted count } (t_i, d_j)$



Automatic Thesaurus Generation Example

word	ten nearest neighbors
absolutely	absurd whatsoever totally exactly nothing
bottomed	dip copper drops topped slide trimmed slig
captivating	shimmer stunningly superbly plucky witty
doghouse	dog porch crawling beside downstairs gazed
Makeup	repellent lotion glossy sunscreen Skin gel p
mediating	reconciliation negotiate cease conciliation p
keeping	hoping bring wiping could some would othe
lithographs	drawings Picasso Dali sculptures Gauguin I
pathogens	toxins bacteria organisms bacterial parasite
senses	grasp psyche truly clumsy naive innate awl

Automatic Thesaurus Generation Discussion

- Quality of associations is usually a problem.
- Term ambiguity may introduce irrelevant statistically correlated terms.
 - "Apple computer" → "Apple red fruit computer"
- Problems:
 - False positives: Words deemed similar that are not
 - False negatives: Words deemed dissimilar that are similar
- Since terms are highly correlated anyway, expansion may not retrieve many additional documents.

Query Expansion: Summary

- Query expansion is often effective in increasing recall.
 - Not always with general thesauri
 - Fairly successful for subject-specific collections
- In most cases, precision is decreased, often significantly.
- Overall, not as useful as relevance feedback; *may* be as good as pseudo-relevance feedback

Pseudo Relevance Feedback

- Automatic local analysis
- Pseudo relevance feedback attempts to **automate** the manual part of **relevance feedback**.
- Retrieve an initial set of relevant documents.
- *Assume* that top m ranked documents are relevant.
- Do relevance feedback
- Mostly works (better than global analysis!)
 - Found to improve performance in TREC ad-hoc task
 - Danger of query drift

Pseudo relevance feedback: Cornell SMART at TREC 4 (Buckley et al. '96)

precision@50:

- Inc.Itc (no PsRF) 64.2%
- Inc.Itc (with PsRF) 72.7%
- Lnu.Itu (no PsRF) 74.2%
- Lnu.Itu (with PsRF) 87%

PsRF: adding 20 terms to original query

Introduction to Information Retrieval

Probabilistic Retrieval

Okapi BM25

Some slides are adapted from the "Introduction to information retrieval" book (Manning et al.) and some are based on Robertson&Zaragoza's FNTIR manuscript.

Introduction to Information Retrieval

The Probability Ranking Principle

- If retrieved documents are ordered by decreasing probability of relevance on the data available, then the system effectiveness is the best that can be obtained for the data

Introduction to Information Retrieval

Notation

- R is a random variable that takes 1 (relevant) and 0 (non-relevant) as values
 - R depends on q (the query) and d (the document)
- We will often write rel as a shortcut for $R=1$ and \overline{rel} as a shortcut for $R=0$

The basic model

$$\boxed{\bullet P(\text{rel} | d, q) \propto_q \frac{P(\text{rel} | d, q)}{P(\overline{\text{rel}} | d, q)}} \quad (2.1)$$

$$\approx \frac{P(d | \text{rel}, q) P(\text{rel} | q)}{P(d | \overline{\text{rel}}, q) P(\overline{\text{rel}} | q)} \quad (2.2)$$

$$\propto_q \frac{P(d | \text{rel}, q)}{P(d | \overline{\text{rel}}, q)} \quad (2.3)$$

Conditional independence
=> Bag of terms model

Confined to query terms

$$\approx \prod_{i \in V} \frac{P(TF_i = tf_i | \text{rel}, q)}{P(TF_i = tf_i | \overline{\text{rel}}, q)} \quad (2.4)$$

$$\approx \prod_{i \in q} \frac{P(TF_i = tf_i | \text{rel})}{P(TF_i = tf_i | \overline{\text{rel}})} \quad (2.5)$$

The basic model (continued.)

$$\boxed{\text{Confined to query terms} \approx \prod_{i \in q} \frac{P(TF_i = tf_i | \text{rel})}{P(TF_i = tf_i | \overline{\text{rel}})}} \quad (2.5)$$

$$\approx_q \sum_q \log \frac{P(TF_i = tf_i | \text{rel})}{P(TF_i = tf_i | \overline{\text{rel}})} \quad (2.6)$$

$$\begin{aligned} \text{Definition } U_i(x) &:= \log \frac{P(TF_i = x | \text{rel})}{P(TF_i = x | \overline{\text{rel}})} \\ &= \sum_q U_i(tf_i) \quad (2.7) \\ &= \sum_{q, tf_i > 0} U_i(tf_i) + \sum_{q, tf_i = 0} U_i(0) \\ &\quad - \sum_{q, tf_i > 0} U_i(0) + \sum_{q, tf_i > 0} U_i(0) \end{aligned} \quad (2.8)$$

The basic model (continued.)

$$\begin{aligned} &= \sum_{q, tf_i > 0} U_i(tf_i) + \sum_{q, tf_i = 0} U_i(0) \\ &\quad - \sum_{q, tf_i > 0} U_i(0) + \sum_{q, tf_i > 0} U_i(0) \end{aligned} \quad (2.8)$$

$$= \sum_{q, tf_i > 0} (U_i(tf_i) - U_i(0)) + \sum_q U_i(0) \quad (2.9)$$

$$\propto_q \sum_{q, tf_i > 0} (U_i(tf_i) - U_i(0)) \quad (2.10)$$

Final scoring function:

$$RSV(d; q) = \sum_{q, tf_i > 0} c_i$$

Special case: BIM (Binary Independence Model)

- TFi = 0 or TFi = 1

$$RSV^{BIM} = \sum_{x_i=q_i=1} c_i^{BIM}; c_i^{BIM} = \log \frac{p_i(1-r_i)}{(1-p_i)r_i} \leftarrow \begin{array}{l} \text{Log odds ratio} \\ \text{where} \end{array}$$

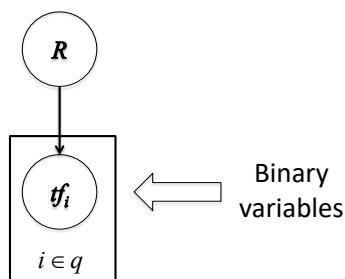
where

	document	relevant (R=1)	not relevant (R=0)
term present	$x_i = 1$	p_i	r_i
term absent	$x_i = 0$	$(1 - p_i)$	$(1 - r_i)$

- Simplifies to (with constant $p_i = 0.5$)

$$RSV = \sum_{x_i=q_i=1} \log \frac{N}{n_i}$$

Graphical model for BIM



A key limitation of the BIM

- BIM – like much of original IR – was designed for titles or abstracts, and not for modern full text search
- We want to pay attention to term frequency and document lengths, just like in other models we've discussed.

Okapi BM25

- BM25 “Best Match 25” (they had a bunch of tries!)
 - Developed in the context of the Okapi system
 - Started to be increasingly adopted by other teams during the TREC competitions
 - It works well
- Goal: be sensitive to these quantities while not adding too many parameters
 - (Robertson and Zaragoza 2009; Spärck Jones et al. 2000)

Eliteness

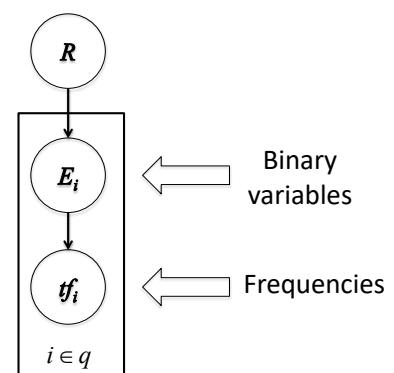
- Model term frequencies using *eliteness*
- What is eliteness?
 - Hidden variable for each document-term pair, denoted as E_i for term i
 - Represents *aboutness*: a term is elite in a document if, in some sense, the document is about the concept denoted by the term
 - Eliteness is binary
 - Term occurrences depend only on eliteness...
 - ... but eliteness depends on relevance

Elite terms

Text from the Wikipedia page on the NFL draft showing **elite terms**

The **National Football League Draft** is an annual event in which the **National Football League (NFL)** teams select eligible college football players. It serves as the league's most common source of player recruitment. The basic design of the **draft** is that each **team** is given a position in the **draft order** in reverse order relative to its **record** ...

Graphical model with eliteness



Retrieval Status Value

- Similar to the BIM derivation, we have

$$RSV^{\text{elite}} = \sum_{i \in q, tf_i > 0} c_i^{\text{elite}}(tf_i);$$

where

$$c_i^{\text{elite}}(tf_i) = \log \frac{p(TF_i = tf_i | R=1)p(TF_i = 0 | R=0)}{p(TF_i = 0 | R=1)p(TF_i = tf_i | R=0)}$$

and using eliteness, we have:

$$\begin{aligned} p(TF_i = tf_i | R) &= p(TF_i = tf_i | E_i = \text{elite})p(E_i = \text{elite} | R) \\ &\quad + p(TF_i = tf_i | E_i = \overline{\text{elite}})(1 - p(E_i = \text{elite} | R)) \end{aligned}$$

Generative model for documents

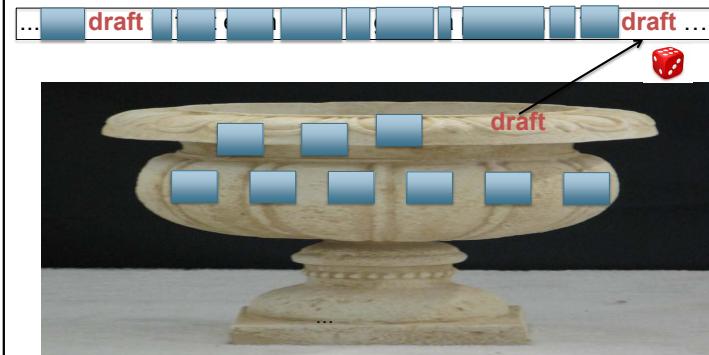
- Words are drawn independently from the vocabulary using a multinomial distribution

... the **draft** is that each **team** is given a position in the **draft** ...



Generative model for documents

- Distribution of term frequencies (tf) follows a binomial distribution --- approximates to a Poisson



Poisson distribution

- The Poisson distribution models the probability of the number of events occurring in a fixed interval of time/space with known average rate and independently of the last event

$$p(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

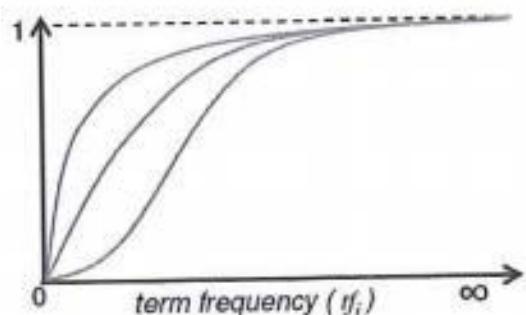
- Examples

- Number of cars arriving at the toll booth per minute
- Number of typos in a sentence

2-Poisson model

- Assume that term frequencies in a document (tf_i) follow a Poisson distribution
 - Approximates a binomial distribution
 - “Fixed interval” implies fixed document length ...
 - ... will fix later
- “2-Poisson” because the distribution is different depending on whether the term is elite or not

Graphing $c_i^{elite}(tf_i)$ for different parameter values of the 2-Poisson



Qualitative properties

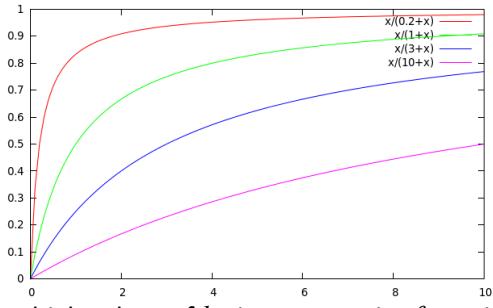
- $c_i^{elite}(0) = 0$ (This is a definition)
- $c_i^{elite}(tf_i)$ increases monotonically with tf_i
- ... but asymptotically approaches a maximum value as $tf_i \rightarrow \infty$
- ... with the asymptotic limit being c_i^{BIM} ← Weight of eliteness feature

Approximating the saturation function

- Estimating parameters for the 2-Poisson model is not easy
- ... So approximate it with a simple parametric curve that has the same qualitative properties

$$\frac{tf}{k_1 + tf}$$

Saturation function



- For high values of k_1 , increments in tf_i continue to contribute significantly to the score
- Contributions tail off quickly for low values of k_1

“Early” versions of BM25

- Version 1: using the saturation function

$$c_i^{BM25v1}(tf_i) = c_i^{BIM} \frac{tf_i}{k_1 + tf_i}$$

- Version 2: BIM simplification to IDF

$$c_i^{BM25v2}(tf_i) = \log \frac{N}{df_i} \times \frac{(k_1 + 1)tf_i}{k_1 + tf_i}$$

- $(k_1 + 1)$ factor doesn't change ranking, but makes term score 1 when $tf_i = 1$
- Similar to $tf-idf$, but terms scores are bounded

Document length normalization

- Longer documents are likely to have larger tf_i values
- Why might documents be longer?
 - Verbosity: suggests observed tf_i too high
 - Larger scope: suggests observed tf_i may be right
- A real document collection probably has both effects
- ... so should apply some kind of normalization

Document length normalization

- Document length:

$$dl = \sum_{i \in l} tf_i$$

- $avdl$: Average document length over collection
- Length normalization component

$$B = \left((1 - b) + b \frac{dl}{avdl} \right), \quad 0 \leq b \leq 1$$

- $b = 1$ full document length normalization
- $b = 0$ no document length normalization

Okapi BM25

- Normalize tf using document length

$$tf'_i = \frac{tf_i}{B}$$

$$\begin{aligned} c_i^{BM25}(tf_i) &= \log \frac{N}{df_i} \times \frac{(k_1 + 1)tf'_i}{k_1 + tf'_i} \\ &= \log \frac{N}{df_i} \times \frac{(k_1 + 1)tf_i}{k_1((1 - b) + b \frac{dl}{avdl}) + tf_i} \end{aligned}$$

- BM25 ranking function

$$RSV^{BM25} = \sum_{i \in q} c_i^{BM25}(tf_i);$$

Okapi BM25

$$RSV^{BM25} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1)tf_i}{k_1((1 - b) + b \frac{dl}{avdl}) + tf_i}$$

- k_1 controls term frequency scaling
 - $k_1 = 0$ is binary model; $k_1 = \text{large}$ is raw term frequency
- b controls document length normalization
 - $b = 0$ is no length normalization; $b = 1$ is relative frequency (fully scale by document length)
- Typically, k_1 is set around 1.2–2 and b around 0.75

Ranking with features

- Textual features
 - Zones: Title, author, abstract, body, anchors, ...
 - Proximity
 - ...
- Non-textual features
 - File type
 - File age
 - Page rank
 - ...

Ranking with zones

- Straightforward idea:
 - Apply your favorite ranking function (BM25) to each zone separately
 - Combine zone scores using a weighted linear combination
- But that seems to imply that the eliteness properties of different zones are different and independent of each other
 - ...which seems unreasonable

Ranking with zones

- Alternate idea
 - Assume eliteness is a term/document property shared across zones
 - ... but the relationship between eliteness and term frequencies are zone-dependent
- Consequence
 - First combine evidence across zones for each term
 - Then combine evidence across terms

BM25F with zones

- Calculate a weighted variant of total term frequency
 - ... and a weighted variant of document length
- $$\tilde{tf}_i = \sum_{z=1}^Z v_z tf_{zi} \quad \tilde{dl} = \sum_{z=1}^Z v_z len_z \quad avd\tilde{l} = \text{Average } \tilde{dl} \text{ across all documents}$$
- where
- v_z is zone weight
 - tf_{zi} is term frequency in zone z
 - len_z is length of zone z
 - Z is the number of zones

Simple BM25F with zones

$$RSV^{\text{SimpleBM25F}} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1)\tilde{tf}_i}{k_1((1 - b) + b \frac{\tilde{dl}}{avd\tilde{l}}) + \tilde{tf}_i}$$

- Simple interpretation: zone z is “replicated” v_z times
- But we may want zone-specific parameters (k_1 , b , IDF)

BM25F

- Empirically, zone-specific length normalization (i.e., zone-specific b) has been found to be useful
- $$\tilde{tf}_i = \sum_{z=1}^Z v_z \frac{tf_{zi}}{B_z}$$
- $$B_z = \left((1 - b_z) + b_z \frac{len_z}{avlen_z} \right), \quad 0 \leq b_z \leq 1$$

$$RSV^{\text{BM25F}} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1)\tilde{tf}_i}{k_1 + \tilde{tf}_i}$$

Ranking with non-textual features

- Assumptions
 - Usual independence assumption
 - Independent of each other and of the textual features
 - Allows us to factor out $\frac{p(F_j = f_j | R=1)}{p(F_j = f_j | R=0)}$ in BIM-style derivation
 - Relevance information is *query independent*
 - Usually true for features like page rank, age, type, ...
 - Allows us to keep all non-textual features in the BIM-style derivation where we drop non-query terms

Ranking with non-textual features

$$RSV = \sum_{i \in q} c_i(t_f_i) + \sum_{j=1}^F \lambda_j V_j(f_j)$$

where

$$V_j(f_j) = \log \frac{p(F_j = f_j | R=1)}{p(F_j = f_j | R=0)}$$

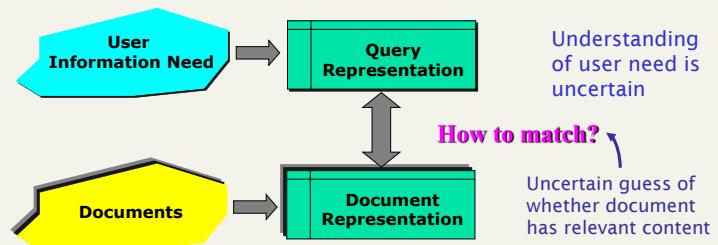
and λ_j is an artificially added free parameter to account for rescalings in the approximations

- Care must be taken in selecting V_j . Some possibilities:
- | | | |
|--------------------------|--------------------------------|---|
| $\log(\lambda'_i + f_j)$ | $\frac{f_j}{\lambda'_i + f_j}$ | $\frac{1}{\lambda'_i + \exp(-f_j \lambda''_j)}$ |
|--------------------------|--------------------------------|---|
- $RSV^{BM25} + \log(pagerank)$ has worked well in practice

Probabilistic IR

Some slides are adapted from the companion website to the "Introduction to IR" book by Manning, Raghavan and Schutze

Why probabilities in IR?



In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our uncertainties?

The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is core of an IR system:**
 - In what order do we present documents to the user?
 - We want the "best" document to be first, second best second, etc....
- **Idea: Rank by probability of relevance of the document w.r.t. information need**
 - $P(\text{relevant} | \text{document}_i, \text{query})$

Recall a few probability basics

- For events a and b :

- Bayes' Rule

$$p(a, b) = p(a \cap b) = p(a | b)p(b) = p(b | a)p(a)$$

$$p(\bar{a} | b)p(b) = p(b | \bar{a})p(\bar{a})$$

$$p(a | b) = \frac{p(b | a)p(a)}{p(b)} = \frac{p(b | a)p(a)}{\sum_{x=a, \bar{a}} p(b | x)p(x)}$$

Prior
Posterior

$$\text{■ Odds: } O(a) = \frac{p(a)}{p(\bar{a})} = \frac{p(a)}{1 - p(a)}$$

The Probability Ranking Principle

"If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data."

Probability Ranking Principle (PRP)

- Theorem:
 - Using the PRP is optimal, in that it minimizes the loss (Bayes risk) loss

Probability Ranking Principle

- How do we compute relevance probabilities?
 - Do not know exact probabilities, have to use estimates
 - Binary Independence Retrieval (BIR) is the simplest model
- Questionable assumptions
 - "Relevance" of each document is independent of relevance of other documents.
 - Really, it's bad to keep on returning **duplicates**
 - Boolean model of relevance
 - That one has a single step information need
 - Seeing a range of results might let user refine query

Binary Independence Model (Robertson and Sparck Jones 1976)

- Traditionally used in conjunction with PRP
- "**Binary**" = **Boolean**: documents are represented as binary incidence vectors of terms :
 - $\vec{x} = (x_1, \dots, x_n)$
 - $x_i = 1$ iff term i is present in document x .
- "**Independence**": terms occur in documents independently
- Different documents can be modeled as same vector
- Bernoulli Naive Bayes model (cf. text categorization!)

Binary Independence Model

- Queries: binary term incidence vectors
- Given query q ,
 - for each document d need to compute $p(R|q,d)$.
 - replace with computing $p(R|q,x)$ where x is binary term incidence vector representing d
 - Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R|q, \vec{x}) = \frac{p(R|q)p(\vec{x}|R,q)}{p(NR|q, \vec{x})} = \frac{\frac{p(R|q)p(\vec{x}|R,q)}{p(\vec{x}|q)}}{\frac{p(NR|q)p(\vec{x}|NR,q)}{p(\vec{x}|q)}} = \frac{p(R|q)}{p(NR|q)} \cdot \frac{p(\vec{x}|R,q)}{p(\vec{x}|NR,q)}$$

Binary Independence Model

$$O(R|q, \vec{x}) = \frac{p(R|q, \vec{x})}{p(NR|q, \vec{x})} = \frac{p(R|q)}{p(NR|q)} \cdot \frac{p(\vec{x}|R,q)}{p(\vec{x}|NR,q)}$$

Constant for a given query

—

Needs estimation

- Using **Independence Assumption**:

$$\frac{p(\vec{x}|R,q)}{p(\vec{x}|NR,q)} = \prod_{i=1}^n \frac{p(x_i|R,q)}{p(x_i|NR,q)}$$

$$\text{So : } O(R|q, d) = O(R|q) \cdot \prod_{i=1}^n \frac{p(x_i|R,q)}{p(x_i|NR,q)}$$

Binary Independence Model

$$O(R | q, d) = O(R | q) \cdot \prod_{i=1}^n \frac{p(x_i | R, q)}{p(x_i | NR, q)}$$

- Since x_i is either 0 or 1:

$$O(R | q, d) = O(R | q) \cdot \prod_{x_i=1} \frac{p(x_i=1 | R, q)}{p(x_i=1 | NR, q)} \cdot \prod_{x_i=0} \frac{p(x_i=0 | R, q)}{p(x_i=0 | NR, q)}$$

- Let $p_i = p(x_i=1 | R, q)$; $r_i = p(x_i=1 | NR, q)$;

- Assume, for all terms not occurring in the query ($q_i=0$) $p_i = r_i$

Then...

This can be changed (e.g., in relevance feedback)

Binary Independence Model

$$\begin{aligned} O(R | q, \vec{x}) &= O(R | q) \cdot \underbrace{\prod_{\substack{x_i=q_i=1 \\ i}} \frac{p_i}{r_i}}_{\text{All matching terms}} \cdot \underbrace{\prod_{\substack{x_i=0 \\ q_i=1}} \frac{1-p_i}{1-r_i}}_{\text{Non-matching query terms}} \\ &= O(R | q) \cdot \underbrace{\prod_{\substack{x_i=q_i=1 \\ i}} \frac{p_i(1-r_i)}{r_i(1-p_i)}}_{\text{All matching terms}} \cdot \underbrace{\prod_{q_i=1} \frac{1-p_i}{1-r_i}}_{\text{All query terms}} \end{aligned}$$

Binary Independence Model

$$O(R | q, \vec{x}) = O(R | q) \cdot \underbrace{\prod_{x_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)}}_{\text{Constant for each query}} \cdot \underbrace{\prod_{q_i=1} \frac{1-p_i}{1-r_i}}_{\text{Only quantity to be estimated for rankings}}$$

- Retrieval Status Value:

$$RSV = \log \prod_{x_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

Binary Independence Model

- All boils down to computing RSV.

$$\begin{aligned} RSV &= \log \prod_{x_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)} \\ RSV &= \sum_{x_i=q_i=1} c_i; \quad c_i = \log \frac{p_i(1-r_i)}{r_i(1-p_i)} \end{aligned}$$

So, how do we compute c_i 's from our data ?

Binary Independence Model

- Estimating RSV coefficients.
- For each term i look at this table of document counts:

Documents	Relevant	Non-Relevant	Total
$X_i=1$	s	$n-s$	n
$X_i=0$	$S-s$	$N-n-S+s$	$N-n$
Total	S	$N-S$	N

- Estimates: $p_i \approx \frac{s}{S}$ $r_i \approx \frac{(n-s)}{(N-S)}$

$$c_i \approx K(N, n, S, s) = \log \frac{s/(S-s)}{(n-s)/(N-n-S+s)}$$

For now, assume no zero terms. More next lecture.

Estimation – key challenge

- If non-relevant documents are approximated by the whole collection, then r_i (prob. of occurrence in non-relevant documents for query term) is n/N and $\log (1-r_i)/r_i = \log (N-n)/n \approx \log N/n = IDF!$
- p_i (probability of occurrence in relevant documents) can be estimated in various ways:
 - from relevant documents if know some
 - Relevance weighting can be used in feedback loop
 - proportional to prob. of occurrence in collection
 - more accurately, to log of this (Greiff, SIGIR 1998)

Iteratively estimating p_i

1. Assume that p_i constant over all x_i in query
 - $p_i = 0.5$ (even odds) for any given doc
2. Determine guess of relevant document set:
 - V is fixed size set of highest ranked documents on this model
3. We need to improve our guesses for p_i and r_i , so
 - Use distribution of x_i in docs in V . Let V_i be set of documents containing x_i
 - $p_i = |V_i| / |V|$
 - Assume if not retrieved then not relevant
 - $r_i = (n_i - |V_i|) / (N - |V|)$
4. Go to 2. until converges then return ranking

17

PRP and BIR

- Getting reasonable approximations of probabilities is possible.
- Requires restrictive assumptions:
 - **term independence**
 - **terms not in query don't affect the outcome**
 - **boolean representation of documents/queries/relevance**
 - **document relevance values are independent**
- Some of these assumptions can be removed
- Problem: either require partial relevance information or only can derive somewhat inferior term weights

Removing term independence

- In general, index terms aren't independent
- Dependencies can be complex
- van Rijsbergen (1979) proposed model of simple tree dependencies
- Each term dependent on one other
- In 1970s, estimation problems held back success of this model

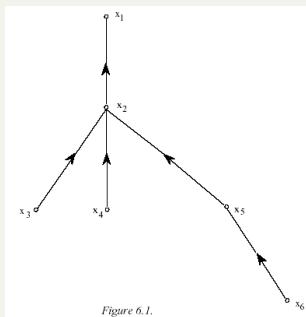


Figure 6.1.

Good and Bad News

- Standard Vector Space Model
 - Empirical for the most part; success measured by results
 - Few properties provable
- Probabilistic Model Advantages
 - Based on a firm theoretical foundation
 - Theoretically justified optimal ranking scheme
- Disadvantages
 - Making the initial guess to get V
 - Binary word-in-doc weights (not using term frequencies)
 - Independence of terms (can be alleviated)
 - Amount of computation
 - Has never worked convincingly better in practice

Okapi BM25 (Robertson et al.)

$$RSV^{BM25} = \sum_{i \in q} \log \frac{N}{df_i} \cdot \frac{(k_1 + 1)tf_i}{k_1((1-b) + b \frac{dl}{avdl}) + tf_i}$$

The LM approach to IR (contd.)

Comparison to vector-space representation (tf.idf)

- LM probabilities are estimated using relative term frequencies
 - reminiscent of tf
- Probabilities are based on term-frequencies that represent length normalization
- Where is IDF ?
 - in the smoothing ...

The connection between smoothing and IDF (Zhai&Lafferty '01)

For notational convinience, we will write $p(w|d)$ for $p(w|M_d)$
 w : term, d : document, q : query with query-terms $q = q_1 \dots q_n$

$p_s(w|d)$ probability of word w that is seen in d
 $p_u(w|d)$ probability of word w that is not seen in d
 $tf(w \in d)$ the count of word w in document d

The query likelihood approach: score document d by

$$p(q|d) = \prod_i p(q_i|d)$$

$$\log p(q|d) = \sum_i \log p(q_i|d)$$

$$\begin{aligned} \log p(q|d) &= \sum_i \log p(q_i|d) = \\ &= \sum_{i: tf(q_i \in d) > 0} \log p_s(q_i|d) + \sum_{i: tf(q_i \in d) = 0} \log p_u(q_i|d) \\ &= \sum_{i: tf(q_i \in d) > 0} \log p_s(q_i|d) - \boxed{\sum_{i: tf(q_i \in d) > 0} \log p_u(q_i|d)} + \boxed{\sum_{i: tf(q_i \in d) > 0} \log p_u(q_i|d)} \\ &+ \sum_{i: tf(q_i \in d) = 0} \log p_u(q_i|d) \\ &= \sum_{i: tf(q_i \in d) > 0} \log \frac{p_s(q_i|d)}{p_u(q_i|d)} + \sum_i \log p_u(q_i|d) \end{aligned}$$

The probability of unseen words

Smoothing with the corpus C :

$$p_u(q_i|d) = \alpha_d p_{MLE}(q_i|C) = \alpha_d \frac{tf(q_i \in C)}{\sum_{w'} tf(w' \in C)}$$

α_d is a constant that depends on d

$$\begin{aligned} \log p(q|d) &= \sum_i \log p(q_i|d) = \\ &= \sum_{i: tf(q_i \in d) > 0} \log \frac{p_s(q_i|d)}{p_u(q_i|d)} + \sum_i \log p_u(q_i|d) \\ &= \sum_{i: tf(q_i \in d) > 0} \log \frac{p_s(q_i|d)}{\alpha_d p_{MLE}(q_i|C)} + n \log \alpha_d + \boxed{\sum_i \log p_{MLE}(q_i|C)} \end{aligned}$$

↑
plays the "IDF" role
↑
document independent

Efficient implementation with inverted index !

Where is the relevance in the LM approach ?

- Recall that in the probabilistic approach we looked at $p(R|q,d)$
- In the LM approach we look at $p(q|d)$
- Where did relevance go ?

A unified derivation for the probabilistic and LM approaches (Lafferty&Zhai '03)

- Robertson-Sparck Jones probabilistic model

Start the derivation from the basic question:
What is the probability that *this* document is relevant to *this* query ?

R, D, Q are random variables
 r relevant, \bar{r} not relevant (values of R)

$$p(R = r | D, Q) = 1 - p(R = \bar{r} | D, Q)$$

probability ranking principle: (Robertson '77)
rank by

$$\log \frac{p(r | D, Q)}{p(\bar{r} | D, Q)} = \log \frac{p(D, Q | r) p(r)}{p(D, Q | \bar{r}) p(\bar{r})}$$

The Robertson-Sparck Jones Model

$$\begin{aligned} \log \frac{p(r | D, Q)}{p(\bar{r} | D, Q)} &= \log \frac{p(D, Q | r) p(r)}{p(D, Q | \bar{r}) p(\bar{r})} \\ &= \log \frac{p(D | Q, r) p(Q | r) p(r)}{p(D | Q, \bar{r}) p(Q | \bar{r}) p(\bar{r})} \\ &= \log \frac{p(D | Q, r) p(r | Q)}{p(D | Q, \bar{r}) p(\bar{r} | Q)} \\ &= \log \frac{p(D | Q, r)}{p(D | Q, \bar{r})} + \log \frac{p(r | Q)}{p(\bar{r} | Q)} \\ &\stackrel{\text{rank}}{=} \log \frac{p(D | Q, r)}{p(D | Q, \bar{r})} \end{aligned}$$

The Robertson-Sparck Jones Model (BIR model)

A document is made up of a collection of attributes

$$D = (x_1, \dots, x_n)$$

The attributes are independent given R and Q

$$p(D | Q, r) = \prod_{i=1}^n p(x_i | Q, r)$$

$$p(D | Q, \bar{r}) = \prod_{i=1}^n p(x_i | Q, \bar{r})$$

The LM approach

- Assumption 1:

– Condition on the event $R = \bar{r}$ document D is independent of the query Q; that is:

$$p(D, Q | R = \bar{r}) = p(D | R = \bar{r}) p(Q | R = \bar{r})$$

- Assumption 2:

$$p(D, R) = p(D)p(R)$$

LM approach

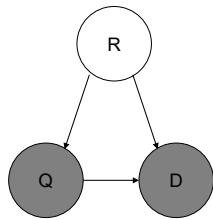
$$\begin{aligned}
 \log \frac{p(r | D, Q)}{p(\bar{r} | D, Q)} &= \log \frac{p(D, Q | r)p(r)}{p(D, Q | \bar{r})p(\bar{r})} \\
 &= \log \frac{p(Q | D, r)}{p(Q | \bar{r})} + \log \frac{p(r | D)}{p(\bar{r} | D)} \quad \text{Use assumption 1} \\
 &\stackrel{\text{rank}}{=} \log p(Q | D, r) + \log \frac{p(r | D)}{p(\bar{r} | D)} \\
 &\stackrel{\text{rank}}{=} \log p(Q | D, r) + \log \frac{p(r)}{p(\bar{r})} \quad \text{Assumption 2} \\
 &\stackrel{\text{rank}}{=} \log p(Q | D, r)
 \end{aligned}$$

LM approach

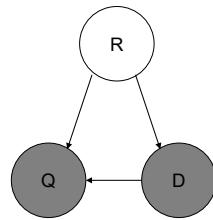
$Q = (A_1, \dots, A_m)$ query terms

$$\log \frac{p(r | D, Q)}{p(\bar{r} | D, Q)} \stackrel{\text{rank}}{=} \sum_{i=1}^m \log p(A_i | D, r) + \log \frac{p(r | D)}{p(\bar{r} | D)}$$

Probabilistic model

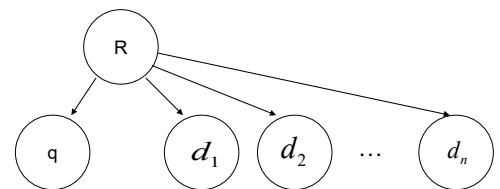


LM approach



Relevance models (Lavrenko&Croft '01)

Given query q and relevant documents d_1, d_2, \dots, d_n there is a relevance language model R that generates the terms both in q and in d_1, \dots, d_n



All we have to do is ..

Estimate the probability assigned by the relevance language model R to term w in the vocabulary: $p(w | R)$

Why is this enough?

Because following our generative assumption (and assuming term independence), we can now rank document d in the corpus by

$$\frac{p(d | R)}{p(d | NR)} \stackrel{\text{def}}{=} \prod_{w \in d} \frac{p(w | R)}{p(w | NR)}$$

where NR is a language model "generating" non-relevant documents

How do we estimate $p(w | NR)$?

More importantly, how do we estimate $p(w | R)$?

Estimating R

Play a game of generating terms by a black box R

$$p(w | R) \approx p(w | q_1, \dots, q_k)$$

$$p(w | R) \approx \frac{p(w, q_1, \dots, q_k)}{p(q_1, \dots, q_k)}$$

I.I.D. sampling

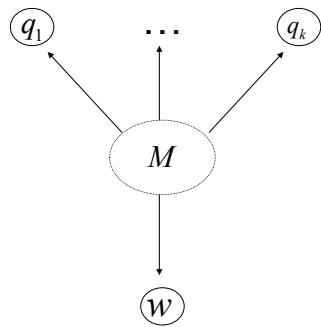
Assume that the query words q_1, \dots, q_k and the words w in relevant documents are sampled identically and independently from a unigram distribution M_R .

Let M represent some finite universe of unigram distributions from which we could sample.

Sampling process: pick a distribution $M \in M$ with probability $p(M)$ and sample from it $k+1$ times. Then

$$p(w, q_1, \dots, q_k) = \sum_{M \in M} p(M) p(w, q_1, \dots, q_k | M)$$

I.I.D Sampling



I.I.D. sampling (contd.)

$$p(w, q_1, \dots, q_k) = \sum_{M \in M} p(M) p(w, q_1, \dots, q_k | M)$$

Using the I.I.D. assumption we get that the conditional probability is

$$p(w, q_1, \dots, q_k | M) = p(w | M) \prod_{i=1}^k p(q_i | M)$$

Now we can estimate the joint probability

$$p(w, q_1, \dots, q_k) = \sum_{M \in M} p(M) p(w | M) \prod_{i=1}^k p(q_i | M)$$

Final estimation details

For the universe M use documents that are the top-retrieved ones with respect to a simple basic language model approach. (D_{init})

$$p(w | M_d) = (1 - \lambda) \frac{tf(w \in d)}{dl_d} + \lambda p_{MLE}(w | C)$$

$$p(q_1, \dots, q_k) = \sum_w p(w, q_1, \dots, q_k)$$

Shortcut for the estimation

$$\begin{aligned} p(w, q_1, \dots, q_k) &= \sum_{d \in D_{init}} p(M_d) p(w | M_d) \prod_{i=1}^k p(q_i | M_d) \\ &= \sum_{d \in D_{init}} p(M_d) p(w | M_d) p(q | M_d) = \sum_{d \in D_{init}} p(w | M_d) p(M_d | q) p(q) \end{aligned}$$

$$\text{Recall that } p(w | R) \approx \frac{p(w, q_1, \dots, q_k)}{p(q)}$$

Therefore,

$$p(w | R) \approx \sum_{d \in D_{init}} p(w | M_d) p(M_d | q)$$

$$p(M_d | q) = \frac{p(q | M_d) p(M_d)}{\sum_{d \in D_{init}} p(M_d) p(q | M_d)}$$

Summary

Basic relevance model: (RM1)

$$p(w | R) \approx \sum_{d \in D_{init}} p(w | M_d) p(M_d | q)$$

$$p(M_d | q) = \frac{p(q | M_d) p(M_d)}{\sum_{d \in D_{init}} p(M_d) p(q | M_d)}$$

Interpolated relevance model: (RM3, Abdul Jaleel et al. '04)

$$p(w | IR) \approx \beta p_{MLE}(w | q) + (1 - \beta) \sum_{d \in D_{init}} p(w | M_d) p(M_d | q)$$

Ranking

- Lavrenko&Croft '03 showed that the preferred way to use R for ranking documents is to compute the KL divergence between a document language model and R

$$Score(d) = D(p(\bullet | R) \| p(\bullet | M_d))$$

$$Score(d) = \sum_w p(w | R) \log \frac{p(w | R)}{p(w | M_d)}$$

Empirical performance

TREC queries 101-150 (title)				TREC queries 151-200 (title)					
Rel	LM	Rel.M	%chg	Wilc.	Rel	LM	Rel.M	%chg	Wilc.
Rel	4805	4805			Rel	4933	4933		
Rret	2981	3733	+25.23	0.0156*	Rret	3288	3222	-2.01	0.0367*
0.00	0.6132	0.6161	+0.5	0.4524	0.00	0.7699	0.7248	-5.9	0.1697
0.10	0.4090	0.4686	+14.6	0.0651	0.10	0.5669	0.5913	+4.3	0.1524
0.20	0.3267	0.4066	+24.5	0.0042*	0.20	0.4494	0.5201	+15.7	0.0084*
0.30	0.2815	0.3562	+26.6	0.0035*	0.30	0.3628	0.4797	+32.2	0.0005*
0.40	0.2277	0.3171	+39.3	0.0001*	0.40	0.3239	0.4090	+26.3	0.0096*
0.50	0.1922	0.2803	+45.8	0.0000*	0.50	0.2596	0.3258	+25.5	0.0256*
0.60	0.1579	0.2393	+51.6	0.0001*	0.60	0.2187	0.2649	+21.1	0.0400*
0.70	0.1094	0.1799	+64.5	0.0027*	0.70	0.1772	0.1852	+4.5	0.2976
0.80	0.0693	0.1205	+74.0	0.0411*	0.80	0.1436	0.1134	-21.0	0.1327
0.90	0.0441	0.0578	+30.8	0.3576	0.90	0.1048	0.0561	-46.5	0.0172*
1.00	0.0267	0.0113	-57.7	0.0372*	1.00	0.0319	0.0165	-48.2	0.0571
Avg	0.2021	0.2617	+29.50	0.0017*	Avg	0.2878	0.3182	+10.55	0.0971

Empirical performance (contd.)

	ROBUST			WT10G		
	MAP	p@5	NDCG@5	MAP	p@5	NDCG@5
Initial	24.8	47.9	50.0	19.6	33.4	32.8
RM1 (OptAQ)	23.4	48.6	50.9	16.6 [*]	33.6	31.2
RM3 (OptAQ)	28.1 ^r	49.8	51.9	21.7 ^r	34.2	33.0
RM1 (OptPQ)	28.7 ⁱ	56.5 ⁱ	59.5 ⁱ	23.2 ⁱ	43.1 ⁱ	41.4 ⁱ
RM3 (OptPQ)	34.4 ^r	59.6 ^r	62.5 ^r	30.3 ^r	49.1 ^r	47.6 ^r
RMI (LOO)	23.4	48.6	50.9	16.6 [*]	33.6	31.2
RM3 (LOO)	27.3 ^r	48.5	50.5	21.2 ^r	33.8	32.2
RMI (10Fold)	23.4	48.6	50.9	15.7 ^r	32.0	29.4
RM3 (10Fold)	27.9 ^r	49.2	51.3	20.4 _r	34.0	32.1

Table 1: OptAQ– optimizing MAP over all queries, OptPQ– optimizing MAP per query, LOO– leave-one-out using MAP for optimization, 10Fold– 10-fold cross validation. i,r - statistically significant differences with Initial and RM1.

Pseudo-Feedback-Based Query Models

A feedback document set F

The goal:

Inducing a query language model from F (and q):

$$p(w | M_F) = ?$$

The relevance model was our first example for a pseudo-feedback-based query model

A divergence minimization approach (Zhai&Laferty '01)

$$M_F = \arg \min \frac{1}{|F|} \sum_{d_i \in F} D(p(\bullet | M_F) \| (p(\bullet | M_{d_i}))$$

Minimize the average divergence between the feedback model and the language models of documents in the feedback set

Issue: Dominance of general corpus terms

Approach: apply corpus-based regularization

$$M_F = \arg \min \frac{1}{|F|} \sum_{d_i \in F} D(p(\bullet | M_F) \| (p(\bullet | M_{d_i})) - \lambda D(p(\bullet | M_F) \| p(\bullet | M_C))$$

Divergence minimization

$$\begin{aligned} M_F &= \arg \min \frac{1}{|F|} \sum_{d_i \in F} D(p(\bullet | M_F) \| (p(\bullet | M_{d_i})) - \lambda D(p(\bullet | M_F) \| p(\bullet | M_C)) \\ A(w) &= \frac{1}{|F|} \sum_{d_i \in F} p(w | M_F) \log \frac{p(w | M_F)}{p(w | M_{d_i})} - \lambda p(w | M_F) \log \frac{p(w | M_F)}{p(w | M_C)} \end{aligned}$$

Divergence minimization

$$A = \frac{1}{|F|} \sum_{d_i \in F} p(w|M_F) \log \frac{p(w|M_F)}{p(w|M_{d_i})} - \lambda p(w|M_F) \log \frac{p(w|M_F)}{p(w|M_C)}$$

$$0 = \frac{\partial A}{\partial p(w|M_F)} = \frac{1}{|F|} \sum_{d_i \in F} (\log p(w|M_F) + 1) - \frac{1}{|F|} \sum_{d_i \in F} \log p(w|M_{d_i}) - \lambda (\log p(w|M_F) + 1) + \lambda \log p(w|M_C)$$

$$p(w|M_F) = \exp \left[\frac{1}{1-\lambda} \frac{1}{|F|} \sum_{d_i \in F} \log p(w|M_{d_i}) - \frac{\lambda}{1-\lambda} \log p(w|M_C) - 1 \right]$$

Query anchoring:

$$p_{DM}(w|M_q) = \alpha p_{MLE}(w|q) + (1-\alpha)p(w|M_F)$$

A mixture model (Zhai&Lafferty '01)

- Assume that the feedback documents $F = \{d_1, \dots, d_k\}$ are generated by a mixture of two multinomial distributions:

- The corpus language model: $p(w|M_C)$
- The “feedback” language model: $p(w|M_F)$

For convenience of notation we will write: $\theta = M_F$

Our goal is to estimate $p(w|\theta)$ for every term w

Note

For retrieval we will use M_q , where

$$p(w|M_q) = \beta p_{MLE}(w|q) + (1-\beta)p(w|\theta)$$

The score of document d with respect to query q will be based on the KL divergence:

$$D(p(\bullet|M_q) \| p(\bullet|M_d))$$

The goal is to avoid query drift.

(We will come back to this point later on.)

The data likelihood

d_j is the j'th term in document d_i

$$p(F|\theta) = \prod_{d_i \in F} \prod_{d_j \in d_i} [(1-\lambda)p(d_j|\theta) + \lambda p(d_j|M_C)]$$

$$L(F) = \log p(F|\theta)$$

$$M_F^* = \arg \max_{\theta} L(F) =$$

$$\arg \max_{\theta} \sum_{d_i \in F} \sum_{d_j \in d_i} \log [(1-\lambda)p(d_j|\theta) + \lambda p(d_j|M_C)]$$

Difficult to obtain an analytical solution using the Lagrange multipliers method

- Solution: the EM (expectation maximization) algorithm
- Idea: augment the data with some latent variables so that the “complete” data likelihood is easy to maximize

Conceptual approach

- The given data is the “incomplete” data
- The augmented data (with latent variables) is the “complete” data
- We will maximize the incomplete data likelihood by maximizing the expected likelihood of the complete data wherein the expectation is over all possible assignments of the latent variables

Translation model (Berger and Lafferty)

- Basic LMs do not address issues of synonymy.
 - Or any deviation in expression of information need from language of documents
- A translation model lets you generate query words not in document via “translation” to synonyms etc.
 - Or to do cross-language IR, or multimedia IR

$$P(q|M_d) = \prod_{w \in q} \sum_{v \in \text{Lexicon}} P(v|M_d) T(w|v)$$

Basic LM Translation

- Need to learn a translation model (using a dictionary or via statistical machine translation)

Using term proximity information in ad hoc retrieval

Background

- All retrieval methods we have seen so far are based on a bag-of-terms representation
 - The assumption is that terms are independent of each other
- We now turn to consider retrieval methods that use information induced from term proximities

Passage-based document retrieval

- A document can be topically heterogeneous
- A document relevant to a query can contain a very short (segment) with relevant information
 - TREC's relevance judgment regime
- Devise document ranking functions that utilize information from passages in the document

Passage types

- Discourse passages
 - Sentences, paragraphs, HTML markups
- Semantic passages
 - Marked based on presumed topic shifts in the document
 - e.g., TextTiling (Hearst&Plaunt '93)
- Fixed/variable length windows of terms

Passage-based document ranking functions (Salton et al. '93, Callan '94, Wilkinson '94)

d : document

q : query

$g \in d$: a passage in d

$score(d; q) \stackrel{\text{def}}{=} \max_{g_i \in d} score(g_i; q)$

$score(d; q) \stackrel{\text{def}}{=} \lambda score(d; q) + (1 - \lambda) \max_{g_i \in d} score(g_i; q)$

A general framework for passage-based document ranking (Bendersky&Kurland '08)

$$p(q|d) = \sum_{g_i} p(q|d, g_i)p(g_i|d)$$

$$\hat{p}(g_i|d) \stackrel{\text{def}}{=} \begin{cases} \frac{p(g_i|d)}{\sum_{g_j \in d} p(g_j|d)} & \text{if } g_i \in d, \\ 0 & \text{otherwise;} \end{cases}$$

$$\text{Score}(d) \stackrel{\text{def}}{=} \sum_{g_i \in d} p(q|d, g_i)\hat{p}(g_i|d)$$

Estimating $p(q|d, g_i)$

$$h^{[M]}(d)p(q|d) + (1 - h^{[M]}(d))p(q|g_i)$$

$h^{[M]}(d)$ is d 's homogeneity level according to model M

$$\text{Score}(d) \stackrel{\text{def}}{=} h^{[M]}(d)p(q|d) + (1 - h^{[M]}(d)) \sum_{g_i \in d} p(q|g_i)\hat{p}(g_i|d)$$

Specific ranking functions

Setting $h^{[M]}(d) = 0$ and assuming a uniform distribution for $\hat{p}(g_i | d) \Rightarrow$

$$\text{Score}_{\text{mean}}(d) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{g_i \in d} p(q|g_i)$$

Alternatively, we can use a bound:

$$\text{Score}_{\text{inter-max}}(d) \stackrel{\text{def}}{=} h^{[M]}(d)p(q|d) + (1 - h^{[M]}(d)) \max_{g_i \in d} p(q|g_i)$$

with the special case:

$$\text{Score}_{\text{max}}(d) \stackrel{\text{def}}{=} \max_{g_i \in d} p(q|g_i)$$

Comparing mean&max

Table 2 Performance comparison of a standard document-based retrieval approach (Doc[base]) with the MaxPsg (MaxPsg[base]) and MeanPsg (MeanPsg[base]) algorithms

	PsgLength 150			PsgLength 50			PsgLength 25		
	MAP	p@5	p@10	MAP	p@5	p@10	MAP	p@5	p@10
FR12									
Doc[base]	22.0	19.1	13.3	22.0	19.1	13.3	22.0	19.1	13.3
MaxPsg[base]	28.4	19.1	14.8	30.1 ^d	19.1	14.8	18.1	13.3	13.8
MeanPsg[base]	13.9 ^d _p	10.5 ^d _p	9.5 _p	13.4 ^d _p	7.6 ^d _p	6.7 ^d _p	9.3 ^d _p	5.7 ^d	6.2 ^d _p
LA+FR45									
Doc[base]	22.7	30.6	26.4	22.7	30.6	26.4	22.7	30.6	26.4
MaxPsg[base]	21.9	27.7	25.5	21.7	28.5	25.7	21.7	29.4	26.8
MeanPsg[base]	20.2 ^d _p	26.0 ^d	23.0 ^d	16.3 ^d _p	19.2 ^d _p	18.1 ^d _p	14.6 ^d _p	17.9 ^d _p	17.2 ^d _p
WSJ									
Doc[base]	28.4	42.4	39.6	28.4	42.4	39.6	28.4	42.4	39.6
MaxPsg[base]	28.8	46.0	41.8	26.1 ^d	44.0	40.4	25.0 ^d	40.8	37.2
MeanPsg[base]	20.4 ^d _p	32.0 ^d _p	32.6 ^d _p	14.0 ^d _p	21.2 ^d _p	19.6 ^d _p	10.7 ^d _p	14.4 ^d _p	14.2 ^d _p
AP89									
Doc[base]	20.0	25.7	24.1	20.0	25.7	24.1	20.0	26.0	24.1
MaxPsg[base]	18.8 ^d _p	27.8	23.0	17.7 ^d	26.1	22.4	16.3 ^d	20.4	16.5 ^d
MeanPsg[base]	17.6 ^d _p	21.7 _p	22.4	14.3 ^d _p	14.8 ^d _p	13.5 ^d _p	12.0 ^d _p	12.2 ^d _p	10.9 ^d _p

Document homogeneity measures

$$h^{[length]}(d) \stackrel{\text{def}}{=} 1 - \frac{\log |d| - \min_{d_i \in \mathcal{C}} \log |d_i|}{\max_{d_i \in \mathcal{C}} \log |d_i| - \min_{d_i \in \mathcal{C}} \log |d_i|}.$$

$$H(d) \stackrel{\text{def}}{=} - \sum_{w \in d} \tilde{p}_d^{\text{MLE}}(w) \log \tilde{p}_d^{\text{MLE}}(w);$$

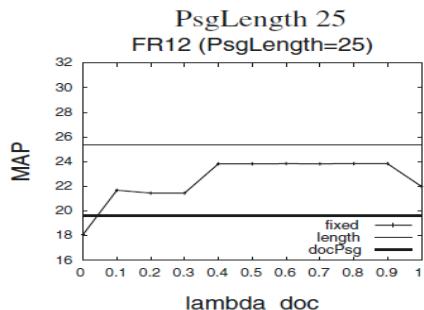
$$h^{[\text{ent}]}(d) \stackrel{\text{def}}{=} \begin{cases} 1 - \frac{H(d)}{\log |d|} & |d| > 1; \\ 1 & \text{otherwise.} \end{cases}$$

$$h^{[\text{interPsg}]}(d) \stackrel{\text{def}}{=} \begin{cases} \frac{2}{m(m-1)} \sum_{i < j, g_i, g_j \in d} \cos(g_i, g_j) & \text{if } m > 1 \\ 1 & \text{otherwise;} \end{cases}$$

$$h^{[\text{docPsg}]}(d) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{g_i \in d} \cos(\mathbf{d}, \mathbf{g}_i)$$

Inter-Max

$$\text{Score}(d) \stackrel{\text{def}}{=} h^{[M]}(d)p(q|d) + (1 - h^{[M]}(d)) \sum_{g_i \in d} p(q|g_i)\hat{p}(g_i|d)$$



Passage-based language models

- A passage g is a span of text => we can induce a language model (Liu&Croft '02)

$$\text{score}(d; q) \stackrel{\text{def}}{=} \max_{g_i \in d} p(q | M_{g_i})$$

Directly utilizing term proximity information

- Most passage-based document retrieval methods do not account for term proximities within the passages and do not let terms from different passages to (directly) affect each other
- We now turn to examine methods that address these issues

Bigram language models

$$P(w_1, w_2, \dots, w_n) = P(w_1)P(w_2|w_1) \dots P(w_n|w_{n-1})$$

$$P(w_i|w_{i-1}) = \lambda_1 P_2(w_i|w_{i-1}) + (1 - \lambda_1) P_1(w_i)$$

$$P_2(w_i|w_{i-1}, d) \approx \frac{C(w_{i-1}, w_i|d)}{C(w_{i-1}|d)}$$

Song&Croft '99

Table 3. Experimental Results on the WSJ Data Set

Retrieval Methods	11-pt Average	%Change	%Change
INQUERY	0.2172	-	
LM	0.2027	- 6.68%	-
GLM(40)	0.2198	+ 1.20%	+ 8.44%
GLM2(40+90)	0.2359	+ 8.61%	+ 16.38%

Table 4. Experimental Results on the TREC4 Data Set

Retrieval Methods	11-pt Average	%Change	%Change
INQUERY	0.1917	-	
LM	0.1890	- 1.41%	-
GLM(40)	0.1905	- 0.63%	+ 0.79%
GLM2(40+90)	0.1923	+ 0.31%	+ 1.75%

Biterm language models

(Srikanth&Srihari '02)

- Relaxing the constraint of ordered pairs of terms

$$P_{BT1}(w_{i-1}, w_i|d) \approx \frac{1}{2}[P_{bg}(w_{i-1}, w_i|d) + P_{bg}(w_i, w_{i-1}|d)]$$

P_{bg} is the standard bigram probability

$$P_{BT2}(w_i|w_{i-1}, d) \approx \frac{C(w_{i-1}, w_i|d) + C(w_i, w_{i-1}|d)}{C(w_{i-1}|d)}$$

$$P_{BT3}(w_i|w_{i-1}, d) \approx \frac{C(w_{i-1}, w_i|d) + C(w_i, w_{i-1}|d)}{\min\{C(w_{i-1}|d), C(w_i|d)\}}$$

The language models are then smoothed

Biterm language models

(Srikanth&Srihari '02)

WSJ

Method	AveP	%Change	%Change	%Change
SMLE(40)	0.2023	-	-	-
PCLM	0.2070	2.32	-	-
BG(40+10)	0.2229	10.18	7.68	-
BT1(40+10)	0.2176	7.56	5.12	-2.38
BT2(40+10)	0.2247	11.07	8.55	0.81
BT3(40+10)	0.2272	12.31	9.76	1.93

Caveats

- Smoothing weights are fixed

Adding proximity operators on top of existing retrieval methods

(Tao&Zhai '07)

$$R_1(Q, D) = KL(Q, D) + \pi(Q, D)$$

$$R_2(Q, D) = BM25(Q, D) + \pi(Q, D)$$

$$KL(Q, D) = \sum_{w \in q \cap d} c(w, q) \cdot \ln\left(1 + \frac{c(w, d)}{\mu \cdot p(w|C)}\right) + |q| \cdot \ln\frac{\mu}{|d| + \mu}$$

$$BM25(Q, D) = \sum_{w \in q \cap d} \left(\ln \frac{N - df(w) + 0.5}{df(w) + 0.5} \times \frac{(k_1 + 1) \times c(w, d)}{k_1((1 - b) + b \frac{|d|}{avdl}) + c(w, d)} \times \frac{(k_3 + 1) \times c(w, q)}{k_3 + c(w, q)} \right)$$

The effect of term proximity on the retrieval score (Tao&Zhai '07)

$$\begin{aligned} R_1(Q, D) &= KL(Q, D) + \pi(Q, D) \\ R_2(Q, D) &= BM25(Q, D) + \pi(Q, D) \end{aligned}$$

$$\pi(Q, D) = \log(\alpha + \exp(-\delta(Q, D)))$$

$\delta(Q, D)$ is a function of the distance between Q's terms in D

Empirical evaluation

	method/data	AP	DOE	FR	TREC8	WEB2g
LM	KL	0.2220	0.1803	0.2442	0.2509	0.3008
	Span	0.2203	0.1717	0.2436	0.2511	0.2992
	MinCover	0.2200	0.1685	0.2659	0.2455	0.2947
	MinDist	0.2265*	0.2018*	0.2718	0.2573*	0.3276*
	AveDist	0.2244	0.1922	0.2683	0.2538	0.3079
BM25	MaxDist	0.2247	0.1913	0.2687	0.2536	0.2966
	BM25	0.2302	0.1840	0.3089	0.2512	0.3094
	Span	0.2292	0.1808	0.3101	0.2468	0.3073
	MinCover	0.2260	0.1815	0.2881	0.2260	0.2966
	MinDist	0.2368*	0.2023*	0.3135	0.2585*	0.3395*
R_2	AveDist	0.2314	0.1960	0.3115	0.2506	0.3148
	MaxDist	0.2323	0.1942	0.3115	0.2492	0.3144

Proximity language model (Zhao&Yun '09)

The central idea:

- Use proximity information (of query terms in a document) in the Dirichlet prior for the document language model
 - Problematic in terms of theory!

Proximity language model (Zhao&Yun '09)

$$p_{Dir}(w | M_d) = \frac{tf(w \in d) + \mu P_{MLE}(w | M_C)}{dl_d + \mu}$$

$$p_{prox}(w | M_d) = \frac{tf(w \in d) + \lambda prox(w) + \mu P_{MLE}(w | M_C)}{dl_d + \sum_{w'} \lambda prox(w') + \mu}$$

Assign $prox(w) = 0$ for w which is not a query term

Computing the proximity function

$$prox(w) = a^{-b}$$

a is a free parameter

b is the min/average/sum (minimal) distance between w (which is a query term) and another query term in the document

Proximity LMs

Table 3: Performance Comparison of Different Ranking Models

Model/Measure	MAP	PR@5	PR@10	MAP	PR@5	PR@10
	AP88			WSJ90-92		
LM	0.2070	0.3120	0.2620	0.1534	0.2160	0.1860
LLM	0.2123	0.3280	0.2720	0.1589	0.2280	0.1940
PLM(P_MinDist)	0.2178	0.3440	0.2780	0.1751*	0.2320	0.1940
PLM(P_AveDist)	0.2167	0.3360	0.2740	0.1603	0.2320	0.1900
PLM(P_SumProx)	0.2203	0.3440	0.2840	0.1735*	0.2360*	0.2000*
WSJ87-92						
LM	0.3351	0.5440	0.4960	0.2704	0.4889	0.4698
LLM	0.3457	0.5640	0.5120	0.2651	0.4857	0.4587
PLM(P_MinDist)	0.3474	0.5680	0.5100	0.2983*	0.5365*	0.5095*
PLM(P_AveDist)	0.3446	0.5600	0.5060	0.2954*	0.5238*	0.5095*
PLM(P_SumProx)	0.3493	0.5720	0.5120	0.2984*	0.5397*	0.5154*

Positional LM (Lv&Zhai '09)

$c(w, i)$: the count of term w at position i (0/1)
 $k(i, j)$: the propagated count at position i from a term at position j

$$c'(w, i) = \sum_{j=1}^N c(w, j)k(i, j)$$

$$p(w|D, i) = \frac{c'(w, i)}{\sum_{w' \in V} c'(w', i)}$$

$$S(Q, D, i) = - \sum_{w \in V} p(w|Q) \log \frac{p(w|Q)}{p(w|D, i)}$$

Smoothing the positional LM

$$Z_i = \sum_{w \in V} c'(w, i) \quad \text{Virtual document length}$$

$$p_\mu(w|D, i) = \frac{c'(w, i) + \mu p(w|\mathcal{C})}{Z_i + \mu}$$

$$p_\lambda(w|D, i) = (1 - \lambda)p(w|D, i) + \lambda p(w|\mathcal{C})$$

Propagation functions

1. Gaussian kernel

$$k(i, j) = \exp \left[\frac{-(i-j)^2}{2\sigma^2} \right]$$

2. Triangle kernel

$$k(i, j) = \begin{cases} 1 - \frac{|i-j|}{\sigma} & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

3. Cosine (Hamming) kernel

$$k(i, j) = \begin{cases} \frac{1}{2} \left[1 + \cos \left(\frac{|i-j|\cdot\pi}{\sigma} \right) \right] & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

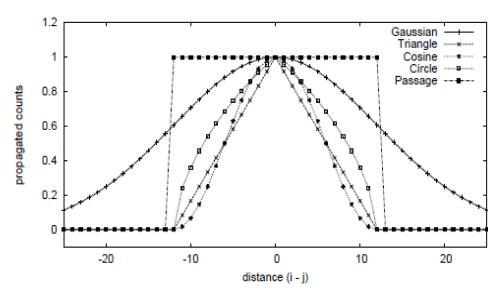
4. Circle kernel

$$k(i, j) = \begin{cases} \sqrt{1 - \left(\frac{|i-j|}{\sigma} \right)^2} & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

5. Passage kernel:

$$k(i, j) = \begin{cases} 1 & \text{if } |i-j| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

Propagation functions



Ranking documents using positional LMs

Best position strategy

$$S(Q, D) = \max_{i \in [1, N]} \{S(Q, D, i)\}$$

Multi position strategy

$$S(Q, D) = \frac{1}{k} \sum_{i \in TopK} S(Q, D, i)$$

Multi- σ strategy

$$S(Q, D) = \gamma \cdot \max_{i \in [1, N]} \{S_{\sigma_0}(Q, D, i)\} + (1 - \gamma) \cdot [-D(\theta_Q || \theta_D)]$$

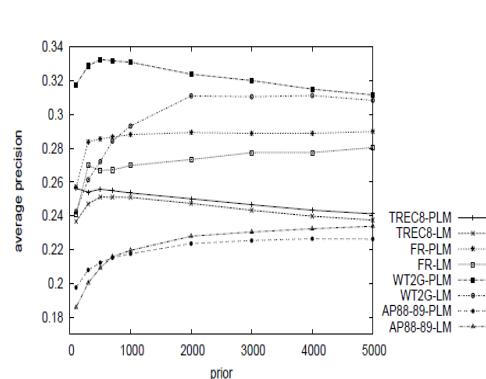
Comparing propagation functions (Best position strategy)

WT2G						TREC8					
kernel \ \sigma	25	75	125	175	275	kernel \ \sigma	25	75	125	175	275
Gaussian	0.2989	0.3213	0.3286	0.3307	0.3285	Gaussian	0.2364	0.2465	0.2503	0.2535	0.2550
Triangle	0.2661	0.3028	0.3149	0.3211	0.3288	Triangle	0.2244	0.2379	0.2438	0.2475	0.2500
Cosine	0.2621	0.3007	0.3128	0.3181	0.3243	Cosine	0.2257	0.2390	0.2430	0.2457	0.2486
Circle	0.2797	0.3140	0.3225	0.3273	0.3267	Circle	0.2315	0.2401	0.2464	0.2492	0.2523

FR						AP88-89					
kernel \ \sigma	25	75	125	175	275	kernel \ \sigma	25	75	125	175	275
Gaussian	0.2913	0.2679	0.2895	0.2880	0.2846	Gaussian	0.1926	0.2112	0.2162	0.2177	0.2198
Triangle	0.2585	0.2898	0.2858	0.2682	0.2897	Triangle	0.1709	0.1987	0.2077	0.2117	0.2173
Cosine	0.2603	0.2910	0.3000	0.2948	0.2858	Cosine	0.1682	0.1969	0.2063	0.2107	0.2144
Circle	0.2685	0.2754	0.2673	0.2877	0.2873	Circle	0.1801	0.2034	0.2093	0.2135	0.2159

Table 2: MAP Results of different kernel functions with Dirichlet smoothing method.

Comparison with standard document LMs (Best position strategy)



Multi-position strategy

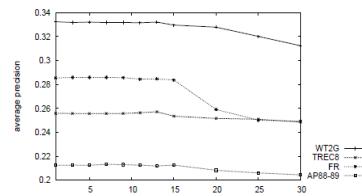


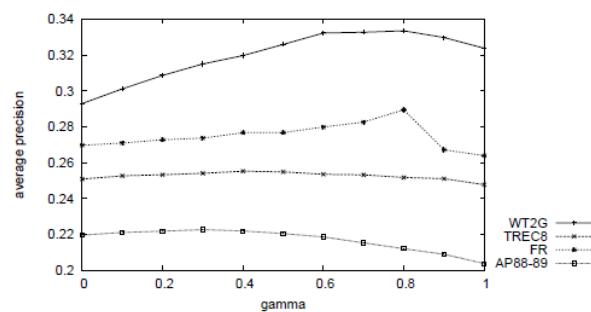
Figure 5: Sensitivity to the parameter k of multi-position strategy

Multi- σ strategy

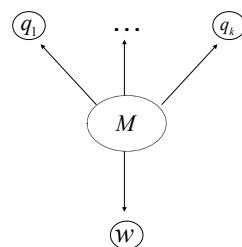
method\dataset	WT2G	TREC8	FR	AP88-89
KL	0.2931	0.2509	0.2697	0.2196
$\sigma = 25$	0.3247 ⁺	0.2562 ⁺	0.2936	0.2237⁺
$\sigma = 75$	0.3336⁺	0.2553 ⁺	0.2896 ⁺	0.2227
$\sigma = 125$	0.3330 ⁺	0.2559 ⁺	0.2885	0.2201
$\sigma = 175$	0.3324 ⁺	0.2574⁺	0.2858	0.2196
$\sigma = 275$	0.3255 ⁺	0.2561 ⁺	0.2852	0.2193

Table 5: The best performance of multi- σ strategy for different σ . '+' means that improvements over the baseline KL method are statistically significant.

Multi- σ strategy



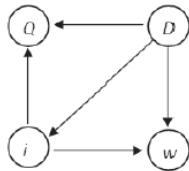
Relevance model: I.I.D Sampling



Pseudo-Feedback-Based Query Expansion using Term Proximity Information

Positional relevance model

(Lv&Zhai '10)



Positional relevance model

$$P(w|Q) = \frac{P(w, Q)}{P(Q)} \propto P(w, Q) = \sum_{D \in \mathcal{F}} \sum_{i=1}^{|D|} P(w, Q, D, i)$$

$$P(w, Q, D, i) = P(D)P(i|D)P(w, Q|D, i)$$

$$P(w, Q, D, i) \propto \frac{P(w, Q|D, i)}{|D|} = \frac{P(Q|D, i)P(w|D, i)}{|D|}$$

$$P(w|Q) \propto P(w, Q) \propto \sum_{D \in \mathcal{F}} \sum_{i=1}^{|D|} \underbrace{\frac{P(Q|D, i)P(w|D, i)}{|D|}}$$

$$P(w|D, i) = \begin{cases} 1.0 & \text{if } w \text{ occurs at position } i \text{ in } D \\ 0.0 & \text{otherwise} \end{cases}$$

Estimating $p(Q|D,i)$

$$P(w|D, i) = \frac{c'(w, i)}{\sum_{w' \in V} c'(w', i)}$$

$$c'(w, i) = \sum_{j=1}^{|D|} c(w, j) \exp \left[\frac{-(i-j)^2}{2\sigma^2} \right]$$

$$P_\lambda(w|D, i) = (1 - \lambda)P(w|D, i) + \lambda P(w|C)$$

$$P(Q|D, i) = \prod_{j=1}^m P_\lambda(q_j|D, i)$$

Empirical performance

Collection	Metric	NoFB	RM3	PRM1
Terabyte06	MAP	0.3047	0.3131	0.3322**+
	Pr@10	0.5367	0.5041	0.5306
	Pr@30	0.4653	0.4660	0.4884**+
	Pr@100	0.3547	0.3576	0.3671**+
ClueWeb09	eMAP	0.0713	0.0755	0.0809**+
	eP@10	0.2371	0.2307	0.2418**+
	eP@30	0.2433	0.2486	0.2536**+
	eP@100	0.2216	0.2283	0.2356**+

A Markov Random Field Model for Term Dependencies

Metzler&Croft SIGIR 2005

A Markov
Random Field
Model for
Term
Dependencies

The ad hoc retrieval task

Estimate $P(D|Q)$ which is rank equivalent to $P(D, Q)$

A Markov
Random Field
Model for
Term
Dependencies

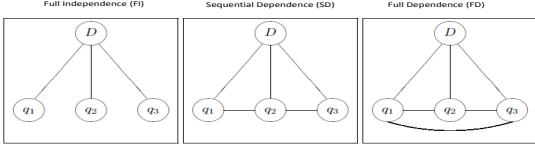
Markov Random Fields (MRFs)

A Markov Random Field Model for Term Dependencies

- Graph $G = (V, E)$
 - vertices: random variables
 - edges: dependencies
- A random variable is independent of its non-neighbors given observed values for its neighbors

Dependence modes

A Markov Random Field Model for Term Dependencies



Using MRFs to model term dependencies

A Markov Random Field Model for Term Dependencies

$$P_{\Lambda}(D, Q) = \frac{1}{Z_{\Lambda}} \prod_{c \in C(G)} \psi(c; \Lambda)$$

- $C(G)$: set of cliques in G
- $\psi(\cdot; \Lambda)$: non-negative potential function
- $Z_{\Lambda} = \sum_{Q,D} \prod_{c \in C(G)} \psi(c; \Lambda)$

Using MRFs to model term dependencies

A Markov Random Field Model for Term Dependencies

$$P_{\Lambda}(D|Q) \stackrel{\text{rank}}{=} P_{\Lambda}(D, Q) \stackrel{\text{rank}}{=} \log P_{\Lambda}(D, Q) \stackrel{\text{rank}}{=} \sum_{c \in C(G)} \log \psi(c; \Lambda)$$

$$\psi(c; \Lambda) \stackrel{\text{def}}{=} \exp[\lambda_c f_c(c)]$$

$$P_{\Lambda}(D|Q) \stackrel{\text{rank}}{=} \sum_{c \in C(G)} \lambda_c f_c(c)$$

Unigrams

A Markov Random Field Model for Term Dependencies

$$\begin{aligned} \psi_T(c) &= \lambda_T \log P(q_i|D) \\ &= \lambda_T \log \left[(1 - \alpha_D) \frac{tf_{q_i,D}}{|D|} + \alpha_D \frac{cf_{q_i}}{|C|} \right] \end{aligned}$$

Ordered sequences

A Markov Random Field Model for Term Dependencies

$$\begin{aligned} \psi_O(c) &= \lambda_O \log P(\#1(q_i, \dots, q_{i+k})|D) \\ &= \lambda_O \log \left[(1 - \alpha_D) \frac{tf_{\#1(q_i, \dots, q_{i+k}), D}}{|D|} + \alpha_D \frac{cf_{\#1(q_i, \dots, q_{i+k})}}{|C|} \right] \end{aligned}$$

Unordered sequences

A Markov Random Field Model for Term Dependencies

$$\begin{aligned}\psi_U(c) &= \lambda_U \log P(\#\text{uwN}(q_i, \dots, q_j) | D) \\ &= \lambda_U \log \left[(1 - \alpha_D) \frac{tf_{\#\text{uwN}(q_i, \dots, q_j), D}}{|D|} + \alpha_D \frac{cf_{\#\text{uwN}(q_i, \dots, q_j)}}{|C|} \right]\end{aligned}$$

Using the feature functions

A Markov Random Field Model for Term Dependencies

$$\begin{aligned}P_\Lambda(D|Q) &\stackrel{\text{rank}}{=} \lambda_T \sum_{c \in T} f_T(c) + \lambda_O \sum_{c \in O} f_O(c) + \lambda_U \sum_{c \in O \cup U} f_U(c) \\ \lambda_T + \lambda_O + \lambda_U &= 1\end{aligned}$$

Evaluation

A Markov Random Field Model for Term Dependencies

	PI		SD		FD	
	AvgP	P9510	AvgP	P9510	AvgP	P9510
AP	0.1775	0.2912	0.1867* (+5.2%)	0.2980 (+2.3%)	0.1866* (+5.1%)	0.3068* (+5.4%)
WSJ	0.2592	0.4327	0.2776† (+7.1%)	0.4427 (+2.3%)	0.2738* (+5.6%)	0.4413 (+2.0%)
WT10g	0.2032	0.2866	0.2167* (+6.6%)	0.2948 (+2.9%)	0.2231** (+9.8%)	0.3031 (-5.8%)
GOV2	0.2502	0.4837	0.2832* (+13.2%)	0.5714* (+18.1%)	0.2844* (+13.7%)	0.5837* (+20.7%)
$(\lambda_T, \lambda_O, \lambda_U)$	(1.00, 0.00, 0.00)		(0.85, 0.10, 0.05)		(0.80, 0.10, 0.10)	

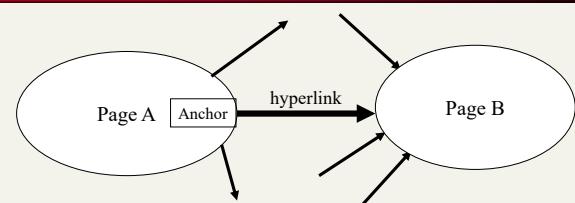
Web Retrieval

Some slides are adapted from the companion website to the "Introduction to IR" book by Manning, Raghavan and Schutze

Topics

- Anchor text
- Link analysis for ranking
 - Pagerank and variants
 - HITS

The Web as a Directed Graph

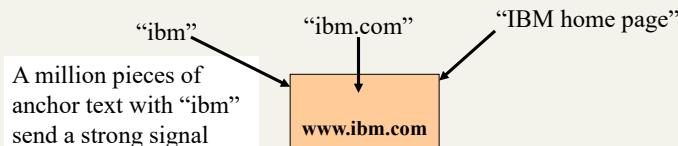


Assumption 1: A hyperlink between pages denotes author perceived relevance (quality signal)

Assumption 2: The anchor of the hyperlink describes the target page (textual context)

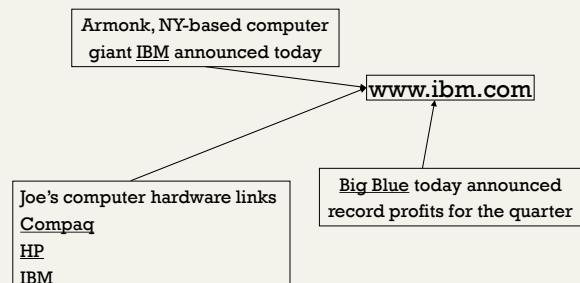
Anchor Text

McBryan '94



Indexing anchor text

- When indexing a document D , include anchor text from links pointing to D .



Indexing anchor text

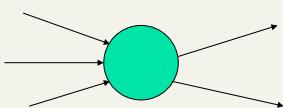
- Can sometimes have unexpected side effects - e.g., *evil empire*.
- Can index anchor text with less weight.

Citation Analysis

- Citation frequency
- Bibliographic coupling frequency
 - Articles that co-cite the same articles are related
- Citation indexing
 - Who is this author cited by? (Garfield [Garf72])
- Pagerank preview: Pinski&Narin '76

Query-independent ordering

- First generation: using link counts as simple measures of popularity.
- Two basic suggestions:
 - Undirected popularity:
 - Each page gets a score = the number of in-links plus the number of out-links ($3+2=5$).
 - Directed popularity:
 - Score of a page = number of its in-links (3).

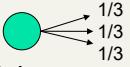


Query processing

- First retrieve all pages meeting the text query (say *venture capital*).
- Order these by their link popularity (either variant on the previous page).

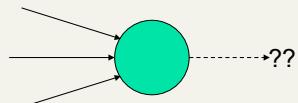
Pagerank scoring (Brin&Page '98)

- Imagine a browser doing a random walk on web pages:
 - Start at a random page
 - At each step, go out of the current page along one of the links on that page, equiprobably
- "In the steady state" each page has a long-term visit rate - use this as the page's score.



Not quite enough

- The web is full of dead-ends.
 - Random walk can get stuck in dead-ends.



Random surfer model

- At a dead end, jump to a random web page.
- At any non-dead end, with probability 10%, jump to a random web page.
 - With remaining probability (90%), go out on a random link.
- 10% - a parameter.

Result of random surfing

- Now cannot get stuck locally.
- There is a long-term rate at which any page is visited (not obvious, will show this).
- How do we compute this visit rate?

PageRank as a Markov chain

$$\text{InDegree}(v) = \sum_{u: u \rightarrow v} 1$$

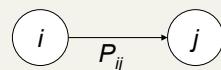
$$PR(v) \stackrel{\text{def}}{=} \sum_{u: u \rightarrow v} PR(u) \quad \text{Pinski and Narin '76}$$

$$PR(v) \stackrel{\text{def}}{=} \sum_u PR(u) \left(\frac{1-\lambda}{|X|} + \lambda \frac{\delta[u \rightarrow v]}{\sum_{v': u \rightarrow v'} 1} \right)$$

$\delta[u \rightarrow v] = 1$ iff there is an edge from u to v

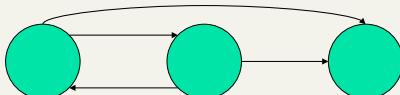
Markov chains

- A Markov chain consists of n states, plus an $n \times n$ transition probability matrix P .
- At each step, we are in exactly one of the states.
- For $1 \leq i, j \leq n$, the matrix entry P_{ij} tells us the probability of j being the next state, given we are currently in state i .



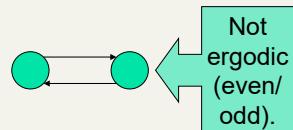
Markov chains

- Clearly, for all i , $\sum_{j=1}^n P_{ij} = 1$.
- Markov chains are abstractions of random walks.



Ergodic Markov chains

- A Markov chain is ergodic if
 - you have a path from any state to any other
 - For any start state, after a finite transient time T_0 , the probability of being in any state at a fixed time $T > T_0$ is nonzero.



Ergodic Markov chains

- For any ergodic Markov chain, there is a unique long-term visit rate for each state.
 - *Steady-state probability distribution.*
- Over a long time-period, we visit each state in proportion to this rate.
- It doesn't matter where we start.

Probability vectors

- A probability (row) vector $\mathbf{x} = (x_1, \dots, x_n)$ tells us where the walk is at any point.
- E.g., $(00\dots 1\dots 000)$ means we're in state i .

More generally, the vector $\mathbf{x} = (x_1, \dots, x_n)$ means the walk is in state i with probability x_i .

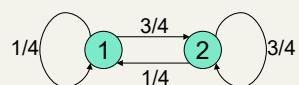
$$\sum_{i=1}^n x_i = 1.$$

Change in probability vector

- If the probability vector is $\mathbf{x} = (x_1, \dots, x_n)$ at this step, what is it at the next step?
- Recall that row i of the transition prob. Matrix \mathbf{P} tells us where we go next from state i .
- So from \mathbf{x} , our next state is distributed as $\mathbf{x}\mathbf{P}$.

Steady state example

- The steady state looks like a vector of probabilities $\mathbf{a} = (a_1, \dots, a_n)$:
 - a_i is the probability that we are in state i .



For this example, $a_1=1/4$ and $a_2=3/4$.

How do we compute this vector?

- Let $\mathbf{a} = (a_1, \dots, a_n)$ denote the row vector of steady-state probabilities.
- If our current position is described by \mathbf{a} , then the next step is distributed as $\mathbf{a}\mathbf{P}$.
- But \mathbf{a} is the steady state, so $\mathbf{a}=\mathbf{a}\mathbf{P}$.
- Solving this matrix equation gives us \mathbf{a} .
 - So \mathbf{a} is the (left) eigenvector for \mathbf{P} .
 - (Corresponds to the “principal” eigenvector of \mathbf{P} with the largest eigenvalue.)
 - Transition probability matrices always have largest eigenvalue 1.

One way of computing a

- Recall, regardless of where we start, we eventually reach the steady state \mathbf{a} .
- Start with any distribution (say $\mathbf{x}=(10\dots 0)$).
- After one step, we’re at $\mathbf{x}\mathbf{P}$;
- after two steps at $\mathbf{x}\mathbf{P}^2$, then $\mathbf{x}\mathbf{P}^3$ and so on.
- “Eventually” means for “large” k , $\mathbf{x}\mathbf{P}^k = \mathbf{a}$.
- Algorithm: multiply \mathbf{x} by increasing powers of \mathbf{P} until the product looks stable.

The Power method

PageRank as a Markov chain

$$PR(v) \stackrel{\text{def}}{=} \sum_u PR(u) \left(\frac{1-\lambda}{|X|} + \lambda \frac{\delta[u \rightarrow v]}{\sum_{v' : u \rightarrow v'} 1} \right)$$

$\delta[u \rightarrow v] = 1$ iff there is an edge from u to v

True if there are no dead ends!

PageRank as a Markov chain (dealing with dead ends)

$$PR(v) \stackrel{\text{def}}{=} \sum_{u:out(u)>0} PR(u) \left(\frac{1-\lambda}{|X|} + \lambda \frac{\delta[u \rightarrow v]}{\sum_{v' : u \rightarrow v'} 1} \right) + \sum_{u:out(u)=0} \frac{PR(U)}{|X|}$$

Pagerank summary

- Preprocessing:
 - Given graph of links, build matrix \mathbf{P} .
 - From it compute \mathbf{a} .
 - The entry a_i is a number between 0 and 1: the pagerank of page i .
- Query processing:
 - Retrieve pages meeting query.
 - Rank them by their pagerank.
 - Order is query-independent.

The reality

- Pagerank is used in google, but so are many other clever heuristics.

Pagerank: Issues and Variants

- How realistic is the random surfer model?
 - What if we modeled the back button? [Fagi00]
 - Surfer behavior sharply skewed towards short paths [Hube98]
 - Search engines, bookmarks & directories make jumps non-random.
- Biased Surfer Models
 - Weight edge traversal probabilities based on match with topic/query (non-uniform edge selection)
 - Bias jumps to pages on topic (e.g., based on personal bookmarks & categories of interest)

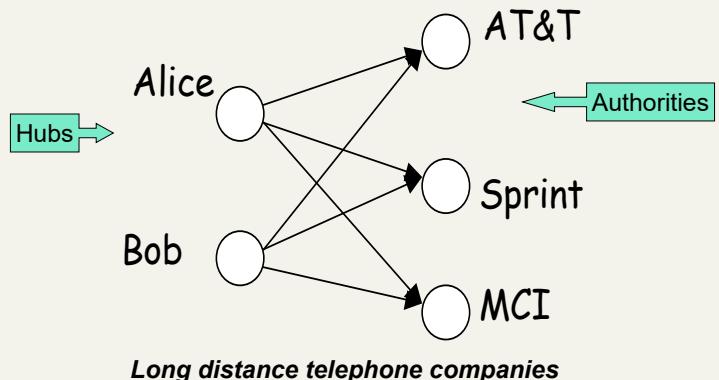
Hyperlink-Induced Topic Search (HITS) – Kleinberg '98

- In response to a query, instead of an ordered list of pages each meeting the query, find two sets of inter-related pages:
 - *Hub pages* are good lists of links on a subject.
 - e.g., "Bob's list of sports-related links."
 - *Authority pages* occur recurrently on good hubs for the subject.
- Best suited for "broad topic" queries rather than for page-finding queries.
- Gets at a broader slice of common *opinion*.

Hubs and Authorities

- Thus, a good hub page for a topic *points* to many authoritative pages for that topic.
- A good authority page for a topic is *pointed to* by many good hubs for that topic.
- Circular definition - will turn this into an iterative computation.

The hope



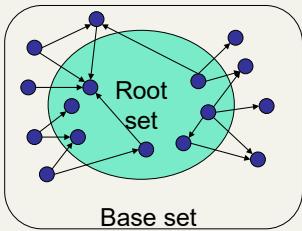
High-level scheme

- Extract from the web a base set of pages that *could* be good hubs or authorities.
- From these, identify a small set of top hub and authority pages;
→iterative algorithm.

Base set

- Given text query (say *browser*), use a text index to get all pages containing *browser*.
 - Call this the root set of pages.
- Add in any page that either
 - points to a page in the root set, or
 - is pointed to by a page in the root set.
- Call this the base set.

Visualization



Assembling the base set [Kleinberg '98]

- Root set typically 200-1000 nodes.
- Base set may have up to 5000 nodes.
- How do you find the base set nodes?
 - Follow out-links by parsing root set pages.
 - Get in-links (and out-links) from a *connectivity server*.
 - (Actually, suffices to text-index strings of the form `href="URL"` to get in-links to `URL`.)

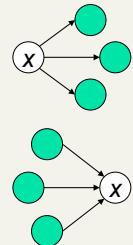
Distilling hubs and authorities

- Compute, for each page x in the base set, a hub score $h(x)$ and an authority score $a(x)$.
- Initialize: for all x , $h(x) \leftarrow 1$; $a(x) \leftarrow 1$;
- Iteratively update all $h(x)$, $a(x)$; 
- After iterations
 - output pages with highest $h()$ scores as top hubs
 - highest $a()$ scores as top authorities.

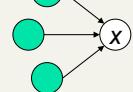
Iterative update

- Repeat the following updates, for all x :

$$h(x) \leftarrow \sum_{x \rightarrow y} a(y)$$



$$a(x) \leftarrow \sum_{y \leftarrow x} h(y)$$



Scaling

- To prevent the $h()$ and $a()$ values from getting too big, can scale down after each iteration.
- Scaling factor doesn't really matter:
 - we only care about the *relative* values of the scores.
- But, it's important to start with the values of hubs and authorities all set to one

How many iterations?

- Claim: relative values of scores will converge after a few iterations:
 - in fact, suitably scaled, $h()$ and $a()$ scores settle into a steady state!
 - proof of this comes later.
- We only require the relative orders of the $h()$ and $a()$ scores - not their absolute values.
- In practice, ~5 iterations get you close to stability.

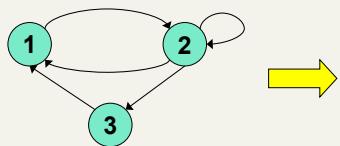
Japan Elementary Schools

Things to note

- Pulled together good pages regardless of language of page content.
 - Use *only* link analysis after base set assembled
 - iterative scoring is query-independent.
 - Iterative computation after text index retrieval - significant overhead.

Proof of convergence

- **$n \times n$ adjacency matrix A :**
 - each of the n pages in the base set has a row and column in the matrix.
 - Entry $A_{ij} = 1$ if page i links to page j , else = 0.



	1	2	3
1	0	1	0
2	1	1	1
3	1	0	0

Hub/authority vectors

- View the hub scores $h()$ and the authority scores $a()$ as vectors with n components.
 - Recall the iterative updates

$$h(x) \leftarrow \sum_{y \mapsto x} a(y)$$

$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$

Rewrite in matrix form

- $h = Aa$.
 - $a = A^t h$.

Recall A^t is the transpose of A .

Substituting, $\mathbf{h} = \mathbf{A}\mathbf{A}^t\mathbf{h}$ and $\mathbf{a} = \mathbf{A}^t\mathbf{A}\mathbf{a}$.

Thus, \mathbf{h} is an eigenvector of $\mathbf{A}\mathbf{A}^t$ and \mathbf{a} is an eigenvector of $\mathbf{A}^t\mathbf{A}$.

Further, our algorithm is a particular, known algorithm for computing eigenvectors: the *power iteration* method.

Guaranteed to converge.

Issues

- Topic Drift
 - Off-topic pages can cause off-topic “authorities” to be returned
 - E.g., the neighborhood graph can be about a “super topic”
 - Mutually Reinforcing Affiliates
 - Affiliated pages/sites can boost each others’ scores
 - Linkage between affiliated pages is not a useful signal

Learning to Rank (LTR) for Information Retrieval

Oren Kurland

Some slides are based on "Learning to Rank for Information Retrieval" by Tie-Yan Liu (2011)

Learning a Ranking Function

- Use train data to learn a ranker
 - Queries and documents marked as (non-)relevant for the queries
- Underlying principle: minimize a loss function

LTR Approaches

- Pointwise
- Pairwise
- Listwise

Pointwise Methods

- Regression
- Classification
- Ordinal regression

Regression

$\mathbf{x} = \{x_j\}_{j=1}^m$ is a set of documents
 $\mathbf{y} = \{y_j\}_{j=1}^m$ corresponding relevance categories
 f is a ranking function we want to learn

$$L(f; x_j, y_j) = (y_i - f(x_j))^2$$

Issues/extensions

- Downside of the squared loss function (can upper bound NDCG Cossock et al. '06)
- Weighted regression model (focus on errors for relevant documents; Cossock et al. '06)
- Regularization

Classification

Support Vector Machines (SVMs); Nallapati '04

- All relevant documents ($y_i = +1$) are positive examples and all non-relevant documents ($y_i = -1$) are negative examples

$$\begin{aligned} \min \frac{1}{2} \|w\|^2 + c \sum_{i=1}^n \sum_{j=1}^{m^{(i)}} \xi_j^{(i)} \\ \text{s.t.} \quad w^T x_j^{(i)} \leq -1 + \xi_j^{(i)} & \quad \text{if } y_j^{(i)} = -1 \\ w^T x_j^{(i)} \geq 1 - \xi_j^{(i)} & \quad \text{if } y_j^{(i)} = 1 \\ \xi_j^{(i)} \geq 0 & \quad j = 1, \dots, m^{(i)}; i = 1, \dots, n \end{aligned}$$

$m^{(i)}$ is the number of documents for query i

SVM Classification (Nallapati '04) — features

Feature	Feature
1 $\sum_{q_i \in Q \cap D} \log(c(q_i, D))$	4 $\sum_{q_i \in Q \cap D} (\log(\frac{ C }{c(q_i, C)}))$
2 $\sum_{i=1}^n \log(1 + \frac{c(q_i, D)}{ D })$	5 $\sum_{i=1}^n \log(1 + \frac{c(q_i, D)}{ D } \text{idf}(q_i))$
3 $\sum_{q_i \in Q \cap D} \log(\text{idf}(q_i))$	6 $\sum_{i=1}^n \log(1 + \frac{c(q_i, D)}{ D } \frac{ C }{c(q_i, C)})$

Figure 2: Features in the discriminative models: $c(w, D)$ represents the raw count of word w in document D , C represents the collection, n is the number of terms in the query, $|\cdot|$ is the size-of function and $\text{idf}(\cdot)$ is the inverse document frequency.

SVM Classification (Nallapati '04) — decision boundary

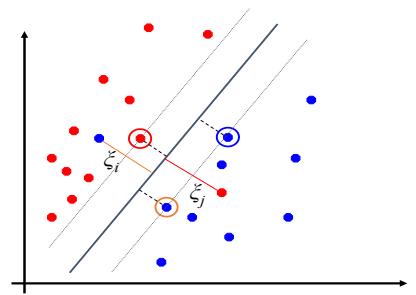


Figure taken from the Introduction to Information Retrieval book (Manning et al.)

Ordinal Regression

- The y_i 's are ordinal variables (K ordered categories)
- Finding a scoring function f such that one can easily use thresholds $b_1 \leq b_2 \leq \dots \leq b_{K-1}$ to distinguish the output of the scoring function into the different ordered categories

Examples

- Perceptron-based method: PRanking (Carmmer&Singer '02)
- Large-margin ordinal regression (Shshua&Levin '03)
 - Use SVM to learn w and the thresholds b_k for $k = 1, \dots, K$

Issues with Pointwise Methods

- The input is a single document; documents order cannot be naturally considered
- The fact that errors at high ranks count more than those at lower ranks is ignored

Pairwise Methods

- The focus is on the relative order between two documents
- In general, the goal is to minimize the number of mis-ranked document pairs

RankNet

Burges et al. '08

- x_u and x_v are documents for query q
- Ground truth: $\bar{P}_{u,v} = \frac{1}{2}(1 + S_{u,v})$; $S_{u,v} = 1$ if x_u is ranked before x_v ; $S_{u,v} = -1$ if x_v is ranked before x_u ; $S_{u,v} = 0$ for a tie
- Modeled probability: $P_{u,v}(f) = \frac{1}{1 + e^{-\sigma(f(x_u) - f(x_v))}}$

Loss function: the cross entropy between the target probability and modeled probability:

$$L(f; x_u, x_v, y_{u,v}) = -\bar{P}_{u,v}(f) \log P_{u,v}(f) - (1 - \bar{P}_{u,v}) \log(1 - P_{u,v}(f))$$

Model: neural network with one hidden layer

Optimization algorithm: gradient descent

RankSVM

Herbrich et al. '00, Joachims '02

$$f(x) = w^T x$$

$$\begin{aligned} & \min \frac{1}{2} \|w\|^2 + c \sum_{i=1}^n \sum_{u,v:y_{u,v}^{(i)}=1} \xi_{u,v}^{(i)} \\ \text{s.t. } & \quad w^T(x_u^{(i)} - x_v^{(j)}) \geq 1 - \xi_{u,v}^{(i)} \quad \text{if } y_{u,v}^{(i)} = 1 \\ & \quad \xi_{u,v}^{(i)} \geq 0 \end{aligned} \tag{1}$$

Implementation:

www.cs.cornell.edu/People/tj/svm_light/svm_rank.html

RankSVM-IR

Cao et al. '06

Issues with RankSVM

- Errors at high ranks should be more important than errors at low ranks
- Queries with many relevant documents shouldn't affect the learned ranker more than queries with few relevant documents

LambdaRank and LambdaMart

Burges et al.

- Lambda: The (position-based) evaluation measures are directly used to define the gradient with respect to each document pair in the training process
 - Goal: having the corresponding implicit loss function be more consistent with the way we evaluate the final ranking
- MART: multiple additive regression trees
- Bagging LambdaMART posts state-of-the-art performance (Ganjisaffar et al. '11)
- Implementations:
 - <https://sourceforge.net/p/lemur/wiki/RankLib/>
 - <https://github.com/yasserg/jforests>

Listwise Approaches

Take an entire set of documents associated with a query in the training data as the input and predict their ground truth labels (i.e., rankings)

- Loss function is explicitly related to the evaluation measures
- Loss function is not explicitly related to the evaluation measures

The LETOR dataset

<http://research.microsoft.com/en-us/um/beijing/projects/letor/>

Features

- Query-{document body, document title, anchor text} similarities
- Query-independent document measures (Bendersky et al. '11)
 - PageRank, CTR, dwell time, spam likelihood, readability level, entropy of term distribution, stopwords presence
- Document-independent query measures (Macdonald et al. '12)
 - query length, query clarity

Information Retrieval

Tutorial 2

Term Weighting

אביב 2023

Boolean Model

Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	Query:
Antony	1	1	0	0	0	Brutus AND Caesar
Brutus	1	1	0	1	0	but NOT Calpurnia
Caesar	1	1	0	1	1	
alpurnia	0	1	0	0	0	
leopatra	1	0	0	0	0	
mercy	1	0	1	1	1	
worser	1	0	1	1	1	

Disadvantages:

- Only exact matches are presented
 - Query formulation is hard to an average user
 - There is no information about which documents are more likely to be relevant

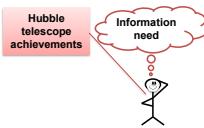
respect to the query

Solution:
Scoring documents with respect to the query

Scoring Documents

Problems

- The relevance is with respect to the information need, that we don't know
 - The query is usually short
 - The query might be poorly formulated
 - Vocabulary mismatch between the query terms and the document terms
 - Polysemy: a single term with multiple meanings
 - table, date..
 - Synonymy: a single meaning can be expressed by different terms
 - Car <> automobile, baby <> infant
 - The documents have different lengths and multiple topics
 - The query terms have different importance
 - Hubble telescope <> telescope achievements <> Hubble achievements



3

Weighting Terms

- **Purpose:** capture what query/document is about.
 - **Representation:** "bag-of-words"
 - The document "Mary is quicker than John" is, in this view, identical to the document "John is quicker than Mary".
 - **Idea:** frequent term is important.

4

Weighting Terms - First Try

Let's weight terms by the raw term frequency (tf)

- What is this document about (term size is proportional to tf)?

This is the course syllabus. Problem?

- The figure was generated using www.wordle.net



Removing Stop-words

- **Stop-words:** terms with VERY high frequency in collection
 - **Removing stop-words:**
 - ✓ has small impact on performance (at least for simple bag-of-words)
 - ✓ reduces index size
 - Removing stop-words \Leftrightarrow giving VERY common terms weight 0.

Weighting Terms - First Try

Example: removing stop-words from previous example:



- But terms with the same tf still have different importance
 - Low importance - high weight, compare: "e.g." <->"search"
 - High importance - low weight: where is "syllabus"?
 - Idea: let's weight terms according their prevalence in the collection.

Weighting Terms

Collection Statistics

- Rare terms in collection are more informative
 - But how to quantify “rare”?
 - Term distribution in the collection is a property of the language
 - Large collection resembles general language (English for instance)
 - Recall Zipf’s distribution from lecture ([short video](#))
 - Term distribution among documents is connected to topicality
 - A single document has a limited number of topics
 - Quantify informativeness by inverse document frequency (idf)

$$idf_t = \log\left(\frac{N}{df_t}\right)$$
 - N – the total number of documents in a collection.
 - df_t – the document frequency of term t : the number of documents in the collection that contain t .

Weighting Terms

Example

term t	df _t	idf _t =log(N/df _t)
Hubble	132	3.6
telescope	380	3.14
achievements	43063	1.09

where N is the total number of documents in collection (above N=528155)

Where is “information”?

Variant TF.IDF Functions

Logarithmic tf

- A document with 10 occurrences of a term is unlikely to be 10 times more relevant than a document with 1 occurrence
 - $Wf_{t,d} = \begin{cases} 1 + \log(tf_{t,d}), & tf_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$
 - $tf_{t,d} = \log(tf_{t,d}): 0 \rightarrow 0.1 \rightarrow 1.2 \rightarrow 1.3, 10 \rightarrow 2, 1000 \rightarrow 4, \dots$

Variant TF.IDF Functions

Maximum tf normalization

- Normalization of tf weights of all terms occurring in a document by the maximum tf if in that document.
 - $ntf_{t,d} = a + (1 - a) \frac{tf_{t,d}}{tf_{max}(d)}$
 - a is a value between 0 and 1 (set to 0.5 following prior work)
 - $tf_{max}(d) = max_{t' \in d} tf_{t',d}$
 - Used to mitigate the anomaly of longer documents
 - Longer documents tend to repeat the same words over and over again
 - Issues:
 - Stopword removal
 - Outlier term with large number of occurrences

Vector Space Model

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1.1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

- Each document d is represented by a vector of $W_{f,t,d} \cdot idf$
 - $W_{f,t,d} = 1 + \log(tf)$, iff $tf > 0$, otherwise 0
 - Terms are axes, $|V|$ is the vocabulary size, $d \in R^{|V|}$
 - We treat the query q similarly as a vector of $W_{t,f} \cdot idf$
 - Usually, cosine similarity is used to rank documents with respect to q .

Exercise

"Introduction to IR" book by Manning, Raghavan and Schütze (Changed)

query					document				
word	tf	wf	df	idf	$q_i = wf * idf$	tf	wf	$d_i = \text{normalized wf}$	$q_i * d_i$
digital			10,000						
video		100,000							
cameras		50,000							

- Queries: $q_1 = \text{"digital cameras"}$, $q_2 = \text{"digital"}$
- Document $d_1 = \text{"digital cameras and video cameras"}$, $d_2 = \text{"digital cameras"}$
- N = 100,000,000 (corpus size)

- Use logarithmic term weighting (wf columns) for query and document
- idf weighting for the query only and cosine normalization for document only
- "and" is a stopword

1. What is the similarity score between q_1 and d_1 ?

word	query - q_1					document - d_1				$q_1 * d_1$
	tf	wf	df	idf	$q_1 = wf * idf$	tf	wf	$d_1 = \text{normalized wf}$	$\frac{1}{2.2} = 0.45$	
digital	1	1	10,000	9.2	9.2	1	1	1	0.45	4.14
video	0	0	100,000	6.9	0	0	0	1	0.45	0
cameras	1	1	50,000	7.6	7.6	2	1.69	0.77	5.852	

- Cosine normalization = L_2 norm:
 - $|d_1| = \sqrt{1^2 + 1^2 + 1.69^2} = 2.2$
 - $\text{Sim}(q_1, d_1) = 4.14 + 0 + 5.852 = \underline{\underline{9.992}}$

2. What is the similarity score between q_2 and d_2 ?

query - q_2					document - d_2				
word	tf	wf	df	idf	$q_2 = wf * idf$	tf	wf	$d_2 = \text{normalized wf}$	$q_2 * d_2$
digital	1	1	10,000	9.2	9.2	1	1	$\frac{1}{1.414} = 0.707$	6.5
video	0	0	100,000	6.9	0	0	0	0	0
cameras	0	0	50,000	7.6	0	1	1	0.707	0

- Cosine normalization = L_2 norm:
 - $|d_2| = \sqrt{1^2 + 0 + 1^2} = 1.414$
 - $\text{Sim}(q_2, d_2) = 6.5 + 0 + 0 = \underline{\underline{6.5}}$

3. Rank d_1 and d_2 with respect to q_2 with and without idf weighting for q_2 , explain.

With idf:

- $\text{Sim}(q_2, d_1) = 9.2 * 0.45 = \underline{\underline{4.14}}$
- $\text{Sim}(q_2, d_2) = 9.2 * 0.707 = \underline{\underline{6.5}}$

Without idf:

- $\text{Sim}(q_2, d_1) = 1 * 0.45 = \underline{\underline{0.45}}$
- $\text{Sim}(q_2, d_2) = 1 * 0.707 = \underline{\underline{0.707}}$

When the query has only one term
idf weighting has no effect on document ranking

Information Retrieval

TUTORIAL 3

Evaluation

System Evaluation

- To measure retrieval effectiveness, we need:
 - A document collection
 - A set of information needs, expressible as queries
 - A set of relevance judgments
- Relevance is assessed **with respect to the** information need, **not the** query
- Example
 - Information need: The history of the Jaguar manufacturer
 - Query: Jaguar

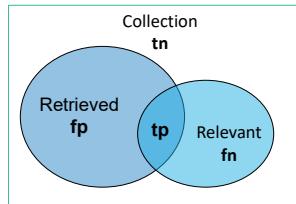
Precision and Recall

- After we run the query q , all the documents in our collection can be categorized as following:

	Relevant	Not Relevant
Retrieved	true positives (tp)	false positives (fp)
Not Retrieved	false negatives (fn)	true negatives (tn)

$$\text{Precision: } P = \frac{tp}{tp+fp} = \frac{\# \text{ Relevant Retrieved}}{\# \text{ Retrieved}}$$

$$\text{Recall: } R = \frac{tp}{tp+fn} = \frac{\# \text{ Relevant Retrieved}}{\# \text{ Relevant}}$$



3

Exercise

- Calculate recall and precision @5, @10, @15, for the following list, given that the total of relevant documents is 55

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
rel.	1	1	0	1	0	0	0	1	0	1	0	1	0	0	1

$$\text{Recall@5} = \frac{3}{55}, \text{Recall@10} = \frac{5}{55}, \text{Recall@15} = \frac{7}{55}$$

$$P@5 = \frac{3}{5}, P@10 = \frac{5}{10}, P@15 = \frac{7}{15}$$

- Recall is a non-decreasing measure
- There is a tradeoff between recall and precision
 - Usually, when recall is increased, precision is decreased

4

F Measure

- We want both recall and precision to be high
 - For typical Web search, precision at top ranks is more important
 - If you are a doctor, looking for all medical cases, recall is important
- We need a combined measure
 - The F measure [CJ van Rijsbergen '79], allows to control the importance of recall vs. precision

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \quad \beta^2 = \frac{1-\alpha}{\alpha}$$

- $\alpha \in [0,1], \beta^2 \in [0, \infty]$
- Balanced F measure: $F_{\alpha=0.5} = F_{\beta=1} = \frac{2PR}{P+R}$, also denoted: F1-measure

5

Average Precision (AP)

$$AP(q) = \frac{1}{|R|} \sum_{i:d_i \in R} P@i$$

- R is the set of relevant documents (judged for query q)
- $P@i$ is precision at rank i
- d_i is the relevant document at rank i
 - The more relevant documents we retrieve, the more elements are in the sum
- Example: calculate AP@15, $|R| = 55$

rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
rel.	1	1	0	1	0	0	0	1	0	1	0	1	0	0	1

$$AP(q) = \frac{1}{55} (1 + \frac{2}{2} + \frac{3}{4} + \frac{4}{8} + \frac{5}{10} + \frac{6}{12} + \frac{7}{15}) = 0.086$$

6

Mean Average Precision (MAP)

- Usually, we average the evaluation measures across queries to get a single estimate of system effectiveness.
- The average of AP is MAP (mean average precision)

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q)$$

7

Mean Reciprocal Rank (MRR)

- The user wants to see only one relevant document
- If the user looks down the ranking until a relevant document is found, and that document is at rank k, then $p@k = \frac{1}{k}$
- Reciprocal Rank: $RR = \frac{1}{k}$
 - k is the rank at which the first relevant document was retrieved
 - RR = 1 if a relevant document was retrieved at rank 1
 - RR = 0.5 if a relevant document was retrieved at rank 2
 - etc...
- MRR is the average of RR across queries

8

Non-Binary Relevance Judgments

- So far, all judgments were binary 0/1
- But some documents tend to be more relevant than others
- Use multiple levels of relevance (graded relevance judgments), e.g.,
 - 0 – not relevant
 - 1 – relevant
 - 2 – highly relevant
 - ...

9

Cumulative Gain (CG) & Discounted Cumulative Gain (DCG) [Jarvelin&Kekalainen '02]

$$CG@k = \sum_{i=1}^k rel_i$$

- rel_i is the graded relevance score of a document at rank i
- The order of documents doesn't affect CG

$$DCG@k = \sum_{i=1}^k \frac{rel_i}{\log_2(i+1)}$$

- Document are penalized as a function of rank i
- The base of the logarithm can be different

10

Normalized DCG (nDCG)

- DCG is not comparable across different queries (why?)
- Normalize by IDCG – the DCG of the ideal ranking

$$nDCG@k = \frac{DCG@k}{IDCG@k}$$

- Example:

rank	1	2	3	4	5
rel.	3	2	4	0	1

 what's the ideal ranking?

$$DCG@5 = \frac{3}{\log_2(2)} + \frac{2}{\log_2(3)} + \frac{4}{\log_2(4)} + \frac{0}{\log_2(5)} + \frac{1}{\log_2(6)} = 6.649$$

$$IDCG@5 = \frac{4}{\log_2(2)} + \frac{3}{\log_2(3)} + \frac{2}{\log_2(4)} + \frac{1}{\log_2(5)} + \frac{0}{\log_2(6)} = 7.323$$

$$nDCG@5 = \frac{6.649}{7.323} = 0.908$$

11

Normalized DCG (nDCG)

- To obtain a measure for a set of queries the nDCG values can be averaged.
- In a perfect ranking algorithm, DCG@k will be the same as IDCG@k, producing nDCG@k of 1
- $0 \leq nDCG \leq 1$
- An alternative formulation of DCG places stronger emphasis on retrieving relevant documents:

$$DCG@k = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

12

Normalized DCG (nDCG)

- The same example with the alternate formulation

$$DCG@k = \sum_{i=1}^k \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

- Example:

rank	1	2	3	4	5
rel.	3	2	4	0	1

 what's the ideal ranking?

$$DCG@5 = \frac{2^3 - 1}{\log_2(2)} + \frac{2^2 - 1}{\log_2(3)} + \frac{2^4 - 1}{\log_2(4)} + \frac{2^0 - 1}{\log_2(5)} + \frac{2 - 1}{\log_2(6)} = 16.779$$

$$IDCG@5 = \frac{2^4 - 1}{\log_2(2)} + \frac{2^3 - 1}{\log_2(3)} + \frac{2^2 - 1}{\log_2(4)} + \frac{2^1 - 1}{\log_2(5)} + \frac{2^0 - 1}{\log_2(6)} = 21.347$$

$$nDCG@5 = \frac{16.779}{21.347} = 0.786$$

13

Expected Reciprocal Rank (ERR) [Chapelle et al. '09]

- nDCG is a simple user browsing model
 - The usefulness of a document at rank i is independent of the usefulness of the documents at rank less than i
- ERR assumes a cascade browsing model that captures real user browsing behavior
 - The likelihood a user examines the document at rank i depends on how satisfied the user was with previously seen documents

$$ERR = \sum_{i=1}^k \frac{1}{i} P(\text{user stops at position } i) \\ = \sum_{i=1}^k \frac{1}{i} \prod_{j=1}^{i-1} (1 - R_j) R_i$$

The probability that the user is not satisfied with documents $d_1 \dots d_{i-1}$

The probability that the user is satisfied with document d_i

14

Expected Reciprocal Rank (ERR) 2

- The probability that a user finds a document d_i satisfying is a function of its graded relevance:

$$R_i = \frac{2^{rel_i} - 1}{2^{rel_{max}}}, rel_i \in \{0, \dots, rel_{max}\}$$

- If the graded relevance scores are {0, ..., 4}:

rel	R _i
0	0
1	1/16
2	3/16
3	7/16
4	15/16

15

ERR - Example

- The system returned documents d1,d2,d3. The relevance judgments are 3,2,4.

- Compute ERR at rank k=3

	i	1/i	rel _i	R _i	P(user stops at position i)
d1	1	1/1	3	7/16	7/16
d2	2	1/2	2	3/16	3/16 · (1 - 7/16)
d3	3	1/3	4	15/16	15/16 · (1 - 7/16) · (1 - 3/16)

$$ERR = \frac{1}{1} \cdot \frac{7}{16} + \frac{1}{2} \cdot \frac{3}{16} \cdot \left(1 - \frac{7}{16}\right) + \frac{1}{3} \cdot \frac{15}{16} \cdot \left(1 - \frac{7}{16}\right) \cdot \left(1 - \frac{3}{16}\right) = 0.633$$

16

Information Retrieval

TUTORIAL 4 Query Expansion, Relevance Feedback & Rocchio

Query Expansion Methods

- Global methods
 - Query expansion/reformulation
 - Thesauri (or Wordnet)
 - Automatic thesaurus generation
- Local methods
 - Relevance feedback
 - Pseudo relevance feedback



2

Vocabulary Mismatch

The two main reasons for vocabulary mismatch:

Polysemy:

- a single term with multiple meanings.
- Examples: table, count, tank...

Synonymy:

- a single meaning can be expressed by different terms.
- Examples: Car <-> automobile, baby <-> infant

Natural language is ambiguous, which leads to vocabulary mismatch between a document and a query.

3

Dealing with linguistic ambiguity

Goal: to pick the **correct** set of synonyms

Possible approaches:

- Thesaurus / WordNet
- Relevance feedback
- Pseudo-relevance feedback

4

Global Methods

5

Synonyms: MeSH thesaurus

- Used to index all medical research studies in the MEDLINE database.
- Addresses the problem of researchers using different terminology to refer to the same concept.
- + Provides excellent results in **exact** domains (medical, legal, etc.).
- Errors in unrestricted search. Example: searching in Google "Neck Pain" using "Cervicalgia".
- Development and maintenance of the Mesh thesaurus is time and resource intensive

MeSH Heading	Neck Pain
Tree Number	C10.597.617.576
Tree Number	C23.888.592.612.553
Tree Number	C23.888.646.501
Entry Term	Cervical Pain
Entry Term	Neckache
Entry Term	Anterior Cervical Pain
Entry Term	Anterior Neck Pain
Entry Term	Cervicalgia
Entry Term	Cervicodinia
Entry Term	Neck Ache
Entry Term	Posterior Cervical Pain
Entry Term	Posterior Neck Pain

6

(n) **squid, calamari, calamar** ((Italian cuisine) squid prepared as food)
 ◦ **domain category**
 ◦ **direct hypernym / inherited hypernym / sister term**

- S: (n) **seafood** (edible fish (broadly including freshwater fish) or shellfish or roe etc)

◦ **domain region**
 ◦ **part holonym**
 (n) **squid** (widely distributed fast-moving ten-armed cephalopod mollusk having a long tapered body with triangular tail fins)
 ◦ **direct hypernym / full hypernym**

- S: (n) **loligo** (somewhat flattened cylindrical squid)
- S: (n) **ommastrephes** (extremely active cylindrical squid with strong arms and large rhombic terminal fins)
- S: (n) **architeuthis, giant squid** (largest mollusk known about but never seen (to 60 feet long))

◦ **part meronym**
 ◦ **direct hypernym / inherited hypernym / sister term**

- S: (n) **decapod** (cephalopods having eight short tentacles plus two long ones)
 - **direct hypernym / full hypernym**
 - S: (n) **squid** (widely distributed fast-moving ten-armed cephalopod mollusk having a long tapered body with triangular tail fins)
 - S: (n) **cuttlefish, cuttle** (ten-armed oval-bodied cephalopod with narrow fins as long as the body and large calcareous internal shell)
 - S: (n) **spirula, Spirula peronii** (a small tropical cephalopod of the genus Spirula having prominent eyes and short arms and a many-chambered shell coiled in a flat sp

WordNet

- **Problems**
 - No notion of context-WordNet is designed of expanding single terms
 - Out-of-vocabulary meanings
 - Any guess of the first result retrieved by Google for "squid"?

7

Local Methods - Relevance Feedback & Rocchio

8

Review: Vector Space Model

- Represent everything as a vector
 - Documents and queries
 - Dimensions = words
- Term weighting: how important is a word in a document
 - tf
 - idf
- Vector similarity: is the document d "near" the query q ?
 - Cosine similarity

9

Relevance Feedback

Idea: Involve the user in the retrieval process in order to improve the final results.

1. The user issues a short query.
2. The system retrieves an initial ranked list of documents.
3. The user marks some returned documents as relevant or non-relevant.
4. The system captures a better representation of the information need.
5. The system displays a revised set of retrieved results.

10

Relevance Feedback



Relevance feedback:

- Allows the system to update the original query representation
- May be iterative
- May be implicit (e.g., from clicks)

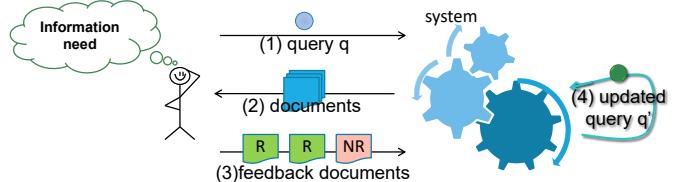


Pseudo-relevance feedback:

- No judgments given, but we assume relevance of the first documents retrieved

11

Relevance Feedback



12

Rocchio's Algorithm

[Rocchio '71]

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

Centroid of relevant documents

Centroid of non-relevant documents

- q_0, q_m – initial and modified query vectors
- D_r – set of relevant documents
- D_{nr} – set of non-relevant documents
- α, β, γ – tune the importance of relevant/non-relevant terms

13

Example

Given the following vectors and parameters, calculate the resulting feedback query.

- Relevant documents vectors:
 - R1: 2, 3, 0, 1, 5
 - R2: 6, 1, 4, 1, 1

- Non-Relevant document vector:
 - NR1: 2, 2, 3, 0, 4

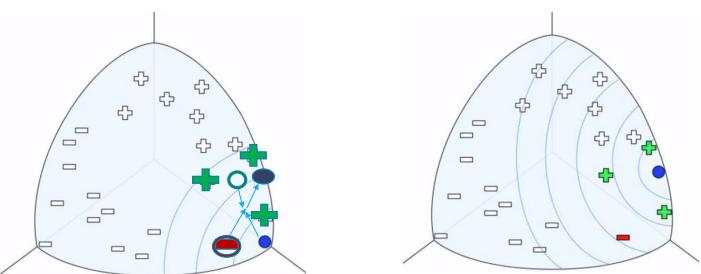
- Original query vector:
 - Q0: 0, 3, 1, 2, 2

- Constants $\alpha=1, \beta=0.75, \gamma=0.25$

$$q_m = 1 \cdot (2, 3, 0, 1, 5) + 0.75 * 0.5(2 + 6, 3 + 1, 0 + 4, 1 + 1, 5 + 1) - 0.25 \cdot (2, 2, 3, 0, 4) \\ = (4.5, 4, 0.75, 1.75, 6.25)$$

Example: Rocchio Feedback

- Run the original query, rank the documents
- User annotates 3 relevant, 1 non-relevant documents
- Construct a new query representation
- Compare the initial ranking to the one induced using the new query representation



15



Poison Pills

- Documents that are marked as relevant yet yield reduced retrieval effectiveness.

- Relevant documents may contain unrelated terms that do not represent the set of relevant documents and using these terms for retrieval will result in poor effectiveness.

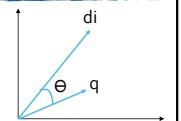
16

Information Retrieval

Tutorial 5
Probabilistic Model of IR
Some slides are adopted from Victor Lavrenko

Why Using Vector Space?

- Why does it work? no notion of relevance anywhere
- A well-formed algebraic space for retrieval
 - Key Idea: user's query can be viewed as a (very) short document
- Query becomes a vector in the same space as the documents
- Can measure each document's proximity to it



Why Do We Need A Probabilistic Model?

- Motivation
 - Speech recognition
 - Given that we see "John" and "feels", which word is more likely to appear next? "happy" or "habit"
 - Classification
 - Given that we observe "football" three times and "game" once in a news article, how likely is it about "sports"?
 - Ad hoc retrieval
 - Given that a user is interested in sports news, how likely would the user use "football" in a query?
- Mathematical formalism for relevant / non-relevant sets

3

Probability Ranking Principle (PRP)

- Robertson (1977)
 - "If a reference retrieval system's response to each request is a **ranking** of the documents in the collection in order of decreasing **probability of relevance** to the user who submitted the request,
Where the **probabilities** are **estimated as accurately** as possible on the basis of whatever data have been made available to the system for this purpose,
The overall **effectiveness** of the system to its user **will be the best** that is obtainable on the basis of those data."
- Basis for most probabilistic approaches to IR

4

What Does The PRP Mean?

- Rank documents by probability of relevance
 - $P(\text{relevant} | \text{document}) \equiv P(R|X)$
- Estimated as accurately as possible
 - $P_{\text{est}}(R|X) \rightarrow P_{\text{true}}(R|X)$ in some way
- Based on whatever data is available to system
 - $P_{\text{est}}(\text{relevant} | \text{document, session, context, collection, user profile, ...})$
- What is $P_{\text{true}}(R|X)$?
 - $P_{\text{true}}(R|Q, X)$
 - Proportion of all unseen users / contexts / tasks for which X would have been judged relevant to Q

5

How Do We Estimate $P(\text{relevant} | \text{query, document})$?

- Assumption A1:
 - Relevance of X doesn't depend on any other document
 - Made by almost every retrieval model (exception: cluster-based)
- Rank documents by $P(R|X)$
 - $\{R, NR\} = \{1, 0\}$ - Random Variables indicating relevance and non-relevance
 - X represents content of the document
- Rank-equivalent:
$$P(R | X) \cong \frac{P(R|X)}{P(NR|X)} \cong \frac{P(X|R)P(R)}{P(X|NR)P(NR)}$$
 - $x/(1-x)$: monotonic function
- Why Bayes? Want a generative model
 - $P(\text{observation} | \text{class})$ sometimes easier with limited data
 - What is the purpose of $P(R)$ and $P(NR)$?

6

Assumptions

- A2: $X = \{X_w\}$ one random variable for every word w
 - Bernoulli values (0,1)
 - A3: X_w are mutually independent given R
 - Allows us to write:

$$P(R|X) \approx \frac{P(X|R)}{P(X|NR)} \approx \frac{\prod_w P(X_w|R)}{\prod_w P(X_w|NR)}$$
 - Define $p_w = P(X_w = 1|R)$ and $r_w = P(X_w = 1|NR)$
- $$P(R|X) \approx \prod_{w \in X} \frac{p_w}{r_w} \cdot \prod_{w \in X} \frac{1-p_w}{1-r_w} = \prod_{w \in X} \frac{p_w(1-r_w)}{r_w(1-p_w)} \cdot \prod_{w \in X} \frac{1-p_w}{1-r_w} \approx \prod_{w \in X} \frac{p_w(1-r_w)}{r_w(1-p_w)}$$

Estimation With Relevant Examples

- Suppose we have (partial) relevance judgments
 - D_r ... relevant, D_{nr} ... non-relevant documents marked
 - Word w observed in $D_r(w)$, $D_{nr}(w)$ docs
 - $P(w)$ = % of docs that contain at least one occurrence of w
 - Includes smoothing avoids zeros, reduces variance
- $$P_w = \frac{D_r(w)+0.5}{D_r+1.0}, r_w = \frac{D_{nr}(w)+0.5}{D_{nr}+1.0}$$

- What if we don't have relevance information?

8

Estimation With Relevant Examples

- Relevant docs: $X_1 = "a b c b d"$, $X_2 = "a b e f b"$
- Non-relevant: $X_3 = "b g c d"$, $X_4 = "b d e"$, $X_5 = "a b e g"$
- Word: a b c d e f g h

9

Estimation With Relevant Examples

- Relevant docs: $X_1 = "a b c b d"$, $X_2 = "a b e f b"$
 - Non-relevant: $X_3 = "b g c d"$, $X_4 = "b d e"$, $X_5 = "a b e g"$
 - Word: a b c d e f g h
- | $D_r(w)$: | 2 | 2 | 1 | 1 | 1 | 0 | 0 | | $D_r = 2$ |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------|
| $D_{nr}(w)$: | 1 | 3 | 1 | 2 | 2 | 0 | 2 | 0 | $D_{nr} = 3$ |
| p_w : | 2.5/3 | 2.5/3 | 1.5/3 | 1.5/3 | 1.5/3 | 0.5/3 | 0.5/3 | 0.5/3 | |
| r_w : | 1.5/4 | 3.5/4 | 1.5/4 | 2.5/4 | 2.5/4 | 0.5/4 | 2.5/4 | 0.5/4 | |
- New document $X_6 = "b g h"$

10

Estimation With Relevant Examples

- Relevant docs: $X_1 = "a b c b d"$, $X_2 = "a b e f b"$
 - Non-relevant: $X_3 = "b g c d"$, $X_4 = "b d e"$, $X_5 = "a b e g"$
 - Word: a b c d e f g h
- | $D_r(w)$: | 2 | 2 | 1 | 1 | 1 | 0 | 0 | | $D_r = 2$ |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|--------------|
| $D_{nr}(w)$: | 1 | 3 | 1 | 2 | 2 | 0 | 2 | 0 | $D_{nr} = 3$ |
| p_w : | 2.5/3 | 2.5/3 | 1.5/3 | 1.5/3 | 1.5/3 | 0.5/3 | 0.5/3 | 0.5/3 | |
| r_w : | 1.5/4 | 3.5/4 | 1.5/4 | 2.5/4 | 2.5/4 | 0.5/4 | 2.5/4 | 0.5/4 | |
- New document $X_6 = "b g h"$
- $$P(R|X_6) \approx \prod_{w \in X} \frac{p_w(1-r_w)}{r_w(1-p_w)} = \frac{2.5}{4} \left(1 - \frac{3.5}{4}\right) \frac{0.5}{3} \left(1 - \frac{2.5}{4}\right) \frac{0.5}{2} \left(1 - \frac{0.5}{4}\right)$$

11

Estimation Without Relevant Examples

- Assumption A4:
 - If the word is not in the query, it is equally likely to occur in relevant and non-relevant populations
 - Practical reason: restrict product to query-document overlap
 - $p_w = r_w \dots w \notin Q$
- Assumption A5:
 - On average, a query word will occur in half the relevant documents
 - Practical reason: p_w and $(1-p_w)$ cancel out
 - $p_w = 0.5 \dots w \in Q$
- Assumption A6:
 - Non-relevant set approximated by collection as a whole
 - Most documents are non-relevant
 - $r_w \approx \frac{N_w}{N} = \frac{N_w+0.5}{N+1}$

$$P(R|X) = \prod_{w \in X} \frac{p_w(1-r_w)}{r_w(1-p_w)} = \prod_{w \in X \cap Q} \frac{p_w(1-r_w)}{r_w(1-p_w)} \cdot \prod_{w \in X \setminus Q} \frac{p_w(1-r_w)}{r_w(1-p_w)} = \prod_{w \in X \setminus Q} \frac{(1-r_w)}{r_w} = \prod_{w \in X \setminus Q} \frac{N_w-N_w+0.5}{N_w+0.5}$$

IDF

Estimation Without Relevant Examples

- Documents: $X_1 = "a\ b\ c\ b\ d"$, $X_2 = "b\ e\ f\ b"$, $X_3 = "b\ g\ c\ d"$, $X_4 = "b\ d\ e"$, $X_5 = "a\ b\ e\ g"$, $X_6 = "b\ g\ h"$

Word:	a	b	c	d	e	f	g	h	N=6
$N(W)$	2	6	2	3	3	1	3	1	
$\frac{N-N_w}{N_w}$	4.5	0.5	4.5	3.5	3.5	1.5	3.5	1.5	

- Query: $Q = "a\ c\ h"$

$$P(R | X_1) \cong \prod_{w \in X_1 \cap Q} \frac{N-N_w+0.5}{N_w+0.5} = \frac{4.5}{2.5} * \frac{4.5}{2.5}$$

$$P(R | X_2) \cong 1$$

$$P(R | X_3) \cong \frac{4.5}{2.5}$$

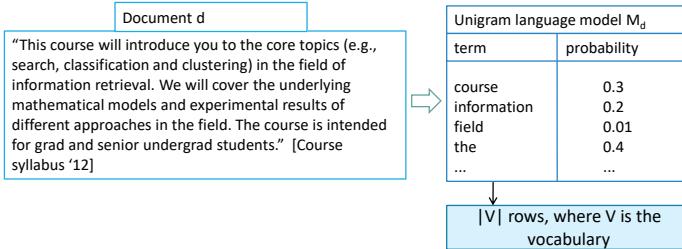
14

Information Retrieval

TUTORIAL 6 Language Models

Statistical Language Models

- Text: probability distribution of sequence of terms

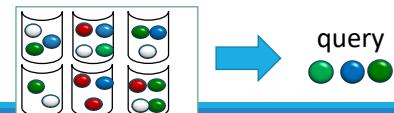


- Unigram language model - $P(w_1 w_2 \dots w_n) = P(w_1)P(w_2) \dots P(w_n)$
 - Probability of a term w is independent of preceding terms
 - Incorrect assumption**, but the derived model is efficient and relatively effective
- N-gram language model - $P(w_1 w_2 \dots w_n) = P(w_1)P(w_2|w_1) \dots P(w_n|w_1 \dots w_{n-1})$
 - Probability of a sequence depends on the probability of the n terms preceding it

2

Language Modeling (LM) Query Likelihood Approach

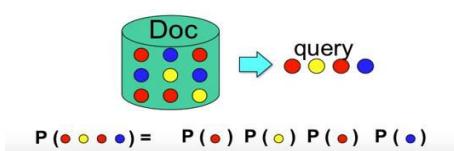
- A language model is a statistical approach for generating the probability of word sequences in a text
- Infer a language model M_{d_i} for each document d_i
- For a query q and each document d_i calculate $P(q | M_{d_i})$
- Query likelihood approach
 - Rank documents by $P(q | M_{d_i})$, the probability to generate a query q from M_{d_i}
- Why use $P(q | M_{d_i})$ to rank documents?
 - Assumption: the user "has a document in mind" when formulating a query



3

Unigram Language Models

- Words are "sampled" independently of each other
 - "bag of words" representation
 - Joint probability decomposes into a product of marginals



4

Estimation Of Language Models

- Document's d "bag of words" representation – a sample from a multinomial distribution
 - N independent trials (terms, N = length of document d)
 - $|V|$ possible outcomes for each trial (V = the vocabulary)
 - Each trial outcome has constant probability $P(t_i | M_d)$, $\sum_{t_i \in V} P(t_i | M_d) = 1$

$$P(d | M_d) = \frac{N!}{tf(t_1 \in d)!tf(t_2 \in d)! \dots tf(t_V \in d)!} P(t_1 | M_d)^{tf(t_1 \in d)} P(t_2 | M_d)^{tf(t_2 \in d)} \dots P(t_V | M_d)^{tf(t_V \in d)}$$

$tf(t_i \in d)$ is the number of occurrences of term t_i in d (raw term frequency)

- How to estimate $P(t_i | M_d)$?
 - Maximum likelihood estimate: $P(t_i | M_d)$ that maximizes the probability $P(d | M_d)$:

$$P_{MLE}(t_i | M_d) = \frac{tf(t_i \in d)}{N}$$

5

The Query Likelihood Model

- Assumption: query q is sampled from distribution M_d

$$P_{MLE}(q|M_d) = \prod_{q_i \in q} P_{MLE}(q_i|M_d) = \prod_{q_i \in q} \frac{tf(q_i|d)}{N}$$

- q_i are terms in q

6

Missing Terms Probabilities

- Documents are sparse
 - $tf_{qi} = 0 \Rightarrow P(Q|M_d) = 0$
- Idea: use background probabilities (the entire collection C)
 - A “large document” from multinomial distribution
 - $P_{MLE}(t_i|C) = \frac{tf(t_i|C)}{\sum_{t_i} tf(t_i|C)}$
- Smoothing with the collection
 - Major impact on performance
 - Smoothing parameters could be different per query/document
- Main roles
 - Eliminates 0 probabilities
 - Has an “idf effect” on terms’ probabilities

7

Smoothing Methods (There Are More)

1. Jelinek-Mercer (JM) smoothing

$$p_{JM}(w|M_d) = (1 - \lambda)p_{MLE}(w|M_d) + \lambda p_{MLE}(w|M_C)$$

2. Bayesian smoothing with Dirichlet priors

$$p_{Dir}(w|M_d) = \frac{tf(w \in d) + \mu p_{MLE}(w|M_C)}{dl_d + \mu}$$

- Equivalent to:

$$p_{Dir}(w|M_d) = (1 - \lambda)p_{MLE}(w|M_d) + \lambda p_{MLE}(w|M_C)$$

$$\text{where } \lambda = \frac{\mu}{dl_d + \mu}$$

- dl_d is the length of document d , μ is Dirichlet prior

8

Example

- Rank the following collection of documents, with respect to the query $q = \text{"onion soup onion"}$
- Use JM smoothing with $\lambda = 0.2$

D1: onion vegetable soup vegetable mushroom corn
 D2: corn onion soup onion mushroom corn
 D3: potato pumpkin tofu potato tofu potato

- First, we estimate the collection LM - M_C

- Total terms 18, unique terms 8

term	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
tf								
P_{MLE}								

9

Example

- Rank the following collection of documents, with respect to the query $q = \text{"onion soup onion"}$
- Use JM smoothing with $\lambda = 0.2$

D1: onion vegetable soup vegetable mushroom corn
 D2: corn onion soup onion mushroom corn
 D3: potato pumpkin tofu potato tofu potato

- First, we estimate the collection LM - M_C

- Total terms 18, unique terms 8

term	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
tf	3	2	2	2	3	3	1	2
P_{MLE}	3/18	2/18	2/18	2/18	3/18	3/18	1/18	2/18

10

Example (2)

- Unsmoothed document LMs:

term/ doc.	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
D1								
D2								
D3								

11

Example (2)

- Unsmoothed document LMs:

term/ doc.	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
D1	1/6	2/6	1/6	1/6	1/6	0	0	0
D2	2/6	0	1/6	1/6	2/6	0	0	0
D3	0	0	0	0	0	3/6	1/6	2/6

- Smoothed LM

- ($\lambda = 0.2$):
- $p(\text{onion}|D1) = 0.8*(1/6) + 0.2*(3/18) = 1/6$
- $P(\text{soup}|D1) = 0.8*(1/6) + 0.2*(2/18) = 0.156$

term/ doc.	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
D1	1/6	0.29	0.156	0.156	1/6	0.033	0.011	0.022
D2	0.3	0.022	0.156	0.156	0.3	0.033	0.011	0.022
D3	0.033	0.022	0.022	0.022	0.033	0.433	0.144	0.29

12

Example (3)

term/ doc.	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
D1	1/6	0.29	0.156	0.156	1/6	0.033	0.011	0.022
D2	0.3	0.022	0.156	0.156	0.3	0.033	0.011	0.022
D3	0.033	0.022	0.022	0.022	0.033	0.433	0.144	0.29

- Query likelihood

- $p(q|D1)=p(\text{onion}|D1)p(\text{soup}|D1)p(\text{onion}|D1)=1/6*0.156*1/6=0.0043$
- $p(q|D2)=p(\text{onion}|D2)p(\text{soup}|D2)p(\text{onion}|D2)=0.3*0.156*0.3=0.014$
- $p(q|D3)=p(\text{onion}|D3)p(\text{soup}|D3)p(\text{onion}|D3)=0.033*0.022*0.033=0.00002$
- Ranking: D2 -> D1 -> D3

Can we find a Dirichlet prior which will give the same scores?
yes, only because all documents have the same length

13

Alternative LM Approaches

- Document likelihood

- Rank documents by $P(d|M_q)$
- Simplest way to estimate M_q
- $P_{MLE}(t|M_q) = \frac{\text{tf}(teq)}{|q|}$
- More advanced: use relevance (or pseudo) feedback to estimate M_q [Zhai & Lafferty '01, Lavrenko & Croft '01]
- Model comparison
- Rank documents by “similarity” between M_q and M_d
- Possible “similarity” measure [Zhai & Lafferty '01]:

$$-KL(M_q | M_d) = -\sum_w p(w|M_q) \log \frac{p(w|M_q)}{p(w|M_d)}$$

14

Example – KL As Scoring Function

- Estimate M_q : $p(\text{onion}|M_q) = 2/3$, $p(\text{soup}|M_q) = 1/3$
- Note: query LM is usually not smoothed

$$KL(M_q | M_{D1}) = \frac{2}{3} \log \frac{2/3}{1/6} + \frac{1}{3} \log \frac{1/3}{0.156} = 0.511$$

$$KL(M_q | M_{D2}) = \frac{2}{3} \log \frac{2/3}{0.3} + \frac{1}{3} \log \frac{1/3}{0.156} = 0.341$$

$$KL(M_q | M_{D3}) = \frac{2}{3} \log \frac{2/3}{0.033} + \frac{1}{3} \log \frac{1/3}{0.022} = 1.263$$

- Ranking by $-KL$: D2 -> D1 -> D3

- The same as by query-likelihood approach

- Note: under the specific assumption of using MLE for the query language model

15

Information Retrieval

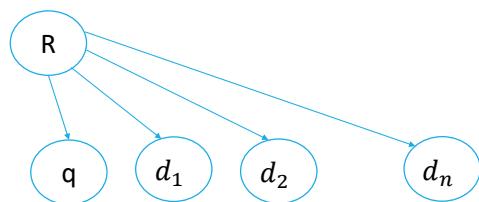
TUTORIAL 8

Relevance Models

2021 – תשלפ"א – חורף

Relevance Models (Lavrenko and Croft '01)

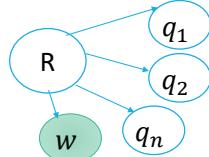
- Given query q and relevant documents d_1, d_2, \dots, d_n there is a relevance language model R that generates the terms both in q and in d_1, d_2, \dots, d_n



2

Estimating R

- The query and the documents are samples from an unknown relevant model R
- The sampling game
 - Relevant examples are missing
 - Let $Q = q_1 \dots q_n$ - the only sample that we have from the R distribution



- $P(w|R) \approx P(w|q_1, q_2, \dots, q_n)$
- $P(w|R) \approx \frac{P(w, q_1, q_2, \dots, q_n)}{P(q_1, q_2, \dots, q_n)}$
- $P(w, q_1, q_2, \dots, q_n) = ?$
- I.I.D. sampling

3

Relevance Models

(Lavrenko and Croft '01)

- Basic relevance model (RM1):

$$P(w|R) \approx \sum_{d \in D_{init}} P(w|M_d)P(M_d|q)$$

$$P(M_d|q) = \frac{P(q|M_d)P(M_d)}{\sum_{d \in D_{init}} P(M_d)P(q|M_d)}$$

- Interpolated relevance model (RM3, Abdul Jaleel et al. 04'):

$$P(W|IR) \approx \beta P_{MLE}(w|q) + (1 - \beta) \sum_{d \in D_{init}} P(w|M_d)P(M_d|q)$$

4

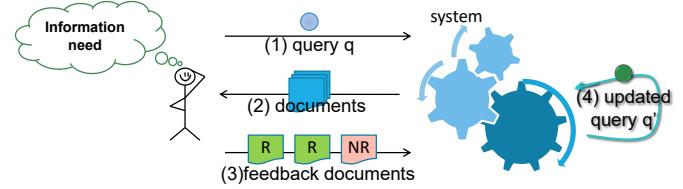
Model Comparison

- Query and documents are represented by language models:
 - M_q, M_d
- Documents are scored using the similarity between the document language model and that of the query language model

$$-KL(M_q | M_d) = -\sum_w p(w|M_q) \log \frac{p(w|M_q)}{p(w|M_d)}$$

5

Relevance Feedback



- Relevance feedback:
 - Allows the system to update the original query representation
 - May be iterative
 - May be implicit (e.g., from clicks)
- Pseudo-relevance feedback:
 - No judgments given, but we assume relevance of the first documents retrieved

6

Relevance Feedback

- Pseudo feedback
 - Retrieve an initial set of n documents
 - Assume that the top m ranked documents are relevant
 - Induce query model

$$P(M_d|q) = \frac{P(q|M_d)P(M_d)}{\sum_{d \in D_{init}} P(M_d)P(q|M_d)}$$
- True feedback
 - Retrieve an initial set of n documents
 - The user marks m relevant documents
 - Top-down process
 - Induce query model

$$P(M_d|q) = \frac{1}{m}$$

	Rank	Doc
"Relevant"	1	d1
	2	d2
	3	d3
	4	d4
	5	d5
	6	d6

	Rank	Doc
Relevant	1	d1
Relevant	2	d2
Relevant	3	d3
Relevant	4	d4
Relevant	5	d5
Relevant	6	d6

7

Example

- Rank the following 4 documents in response to the query "orange apple"
 - d1: orange orange lemon lemon clementine apple
 - d2: orange and red are beautiful colors
 - d3: orange and celcom offer a special deal
 - d4: the fruits I like most are orange and apple
- Construct a relevance model using the top 2 documents

8

Relevance Model Example

- Rank the following 4 documents in response to the query "orange apple"

We assume: Dirichlet smoothing with $\mu=1000$

$$p_{Dir}(w|M_d) = \frac{f(w \in d) + \mu p_{MLE}(w|M_C)}{dl_d + \mu}$$

$$\begin{aligned} p(q|d1) &= p(\text{orange}|d1) * p(\text{apple}|d1) = \left(\frac{2+\frac{5}{28}1000}{6+1000} \right) * \left(\frac{1+\frac{2}{28}1000}{6+1000} \right) = 0.0129 \\ p(q|d2) &= p(\text{orange}|d2) * p(\text{apple}|d2) = \left(\frac{1+\frac{5}{28}1000}{6+1000} \right) * \left(\frac{0+\frac{2}{28}1000}{6+1000} \right) = 0.01267 \\ p(q|d3) &= p(\text{orange}|d3) * p(\text{apple}|d3) = \left(\frac{1+\frac{5}{28}1000}{7+1000} \right) * \left(\frac{0+\frac{2}{28}1000}{7+1000} \right) = 0.01264 \\ p(q|d4) &= p(\text{orange}|d4) * p(\text{apple}|d4) = \left(\frac{1+\frac{5}{28}1000}{9+1000} \right) * \left(\frac{1+\frac{2}{28}1000}{9+1000} \right) = 0.01277 \end{aligned}$$

Relevance Model Example

- Construct a relevance model using the top 2 documents

$$P(w|R) \approx \sum_{d \in D_{init}} P(w|M_d)P(M_d|q)$$

$$P(M_d|q) = \frac{P(q|M_d)P(M_d)}{\sum_{d \in D_{init}} P(M_d)P(q|M_d)}$$

$$P(\text{clementine}|R) = \frac{1}{6} * \frac{0.0129}{0.0129+0.01277} = 0.0838$$

$$P(\text{fruit}|R) = \frac{1}{9} * \frac{0.01277}{0.0129+0.01277} = 0.055$$

$$P(\text{apple}|R) = \frac{1}{6} * \frac{0.0129}{0.0129+0.01277} + \frac{1}{9} * \frac{0.01277}{0.0129+0.01277} = 0.139$$

...

- Documents are ranked according to the similarity of their model to that of the query

$$-KL(M_q | M_d) = -\sum_w p(w|M_q) \log \frac{p(w|M_q)}{p(w|M_d)}$$

Clipping The Query Model

- The expanded query contains many terms
- The amount of unique terms in documents provided for feedback
- Problems:
 - Query drift – why?
 - Computationally demanding
- Clipping the query model
 - Use the top V terms assigned the highest probability
 - Underlying assumption: these terms are most likely to represent the information need

11

Clipping Example

- $q_0 = \text{"nobel prize"}$
- INEX collection
 - # of documents = 2,666,190
 - # of relevant documents to q_0 = 22
- Construct RM3 model
 - 3 relevant documents serve as true feedback
 - $\beta = 0.2$
 - Clipping: $v = 100$
- statistics:
 - 79 out of 100 terms are stopwords!
 - The top 12 terms with the highest probability are: nobel, prize, the, of, in, and, for, on, 2007, or, medicine, physiology

12

Additional Thoughts

- Given examples of relevant documents provided by the user
- Will the expanded query always be closer to the centroid of the relevant documents?
- Suggest an approach for ameliorating this drawback
- How would you estimate $P(M_d|q)$?
- Using KL for ranking results in an IDF effect while using the query likelihood model (MLE)
 - Does the IDF effect hold for any query model?
Raiber and Kurland 2017



13

Does the IDF effect hold for any query model?

- M_d : Smoothed unigram document language model
- $M_d \equiv (1 - \lambda)M_d^{MLE} + \lambda M_C^{MLE}$
- For $w \in d$:
 - $p_s(w|M_d) = (1 - \lambda)p(w|M_d^{MLE}) + \lambda p(w|M_C^{MLE})$
- For $w \notin d$:
 - $p_u(w|M_d) = \lambda p(w|M_C^{MLE})$
 - $-CE(M_q, M_d) = \sum_w p(w|M_q) \log p(w|M_d) = \sum_{w \in d} p(w|M_q) \log p_s(w|M_d) + \sum_{w \notin d} p(w|M_q) \log p_u(w|M_d) = \sum_{w \in d} p(w|M_q) \log \frac{p_s(w|M_d)}{p_u(w|M_d)} + \sum_w p(w|M_q) \log p_u(w|M_d) = \sum_{w \in d} p(w|M_q) \log \frac{p_s(w|M_d)}{\lambda p(w|M_C^{MLE})} + \log \lambda - CE(M_q, M_C^{MLE}) \propto \sum_{w \in d} p(w|M_q) \log \left(1 + \frac{1 - \lambda}{\lambda} \frac{p(w|M_d^{MLE})}{p(w|M_C^{MLE})} \right) + \log \lambda$

14

Information Retrieval

TUTORIAL 8 Divergence Minimization

Divergence Minimization Model (DMM) Zhai and Lafferty 01'

- A feedback-based approach in the language modeling framework
- Given a query q and a set of relevance-feedback documents F
 - The query language model M_q should capture what the user is interested in
 - Estimated using the MLE - $p_{MLE}(w|M_q) = \frac{tf(w|q)}{|q|}$
 - $tf(w|q)$ is the count of term w in query q
 - $|q|$ is the total number of terms in the query
 - Estimate a feedback language model M_F
 - Relevance feedback
 - Pseudo feedback
 - Interpolate the feedback model with the original query model

$$p(w|M'_q) = (1 - \alpha)p_{MLE}(w|M_q) + \alpha p(w|M_F)$$
 - α controls the influence of the feedback model

2

M_F Estimation

- DMM uses an idea similar to the Rocchio algorithm
- Assumptions:
 1. The feedback language model M_F is very close to the language model of every feedback document M_d
 2. The feedback language model M_F is far from the collection language model M_C
- $\hat{M}_F = \arg \min_{M_F} \left(\sum_{d \in F} \alpha_d KL(M_F || M_d) - \lambda KL(M_F || M_C) \right)$
- Smaller $KL(M_F || M_d)$ and higher $KL(M_F || M_C)$
- M_d is the smoothed language model of feedback document d
- $\alpha_d = \frac{1}{|F|}$,
 - uniform contribution of each document to the feedback model
- $\lambda \in [0, 1]$ controls the influence of the collection language model

3

M_F Estimation

- DMM uses an idea similar to the Rocchio algorithm
- $\hat{M}_F = \arg \min_{M_F} \left(\sum_{d \in F} \alpha_d KL(M_F || M_d) - \lambda KL(M_F || M_C) \right)$
 - Use Lagrange multiplier to solve the minimization problem

$$P(w|M_F) \propto \exp\left(\frac{1}{1-\lambda} \cdot \frac{1}{|F|} \sum_{d \in F} \log(p(w|M_d)) - \frac{\lambda}{1-\lambda} \log(p(w|M_C))\right)$$
 - use term clipping to estimate a distribution M_F

4

Maximum Entropy Divergence Minimization Model (MEDMM) Lv and Zhai 14'

- DMM is relatively less effective than RM3
 - Theoretically well-justified
 - The problem: the entropy of the feedback language model
- Diagnosing the problem of DMM

$$\begin{aligned} \hat{M}_F &= \arg \min_{M_F} \sum_{d \in F} \alpha_d KL(M_F || M_d) - \lambda KL(M_F || M_C) \\ &= \arg \min_{M_F} \sum_{d \in F} \alpha_d \cdot (CE(M_F, M_d) - H(M_F)) - \lambda \cdot (CE(M_F, M_C) - H(M_F)) \\ &= \arg \min_{M_F} \sum_{d \in F} \alpha_d \cdot CE(M_F, M_d) - \sum_{d \in F} \alpha_d \cdot H(M_F) - \lambda \cdot (CE(M_F, M_C) - H(M_F)) \\ &\stackrel{\sum \alpha_d = 1}{=} \arg \min_{M_F} \sum_{d \in F} \alpha_d CE(M_F, M_d) - \lambda CE(M_F, M_C) - (1 - \lambda)H(M_F) \end{aligned}$$

5

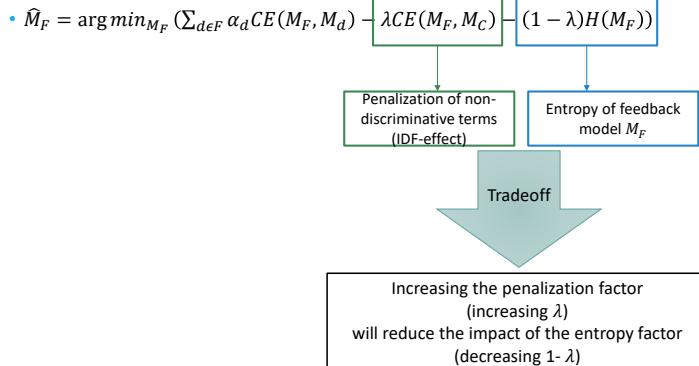
Maximum Entropy Divergence Minimization Model (MEDMM) Lv and Zhai 14'

- Diagnosing the problem of DMM

- $\hat{M}_F = \arg \min_{M_F} \left(\sum_{d \in F} \alpha_d KL(M_F || M_d) - \lambda KL(M_F || M_C) \right) = \arg \min_{M_F} \left(\sum_{d \in F} \alpha_d CE(M_F, M_d) - \lambda CE(M_F, M_C) - (1 - \lambda)H(M_F) \right)$
 - $H(M_F) = -\sum_w p(w|M_F) \log p(w|M_F)$ - the entropy of M_F
 - $CE(M_F, M_d)$ is the cross entropy between M_F and M_d
 - $CE(M_F, M_C)$ is the cross entropy between M_F and M_C

6

DMM Tradeoff



7

MEDMM

- $\hat{M}_F = \arg \min_{M_F} (\sum_{d \in F} \alpha_d CE(M_F, M_d) - \lambda CE(M_F, M_C) - (1 - \lambda)H(M_F))$
- Idea: reduce λ without harming the IDF effect
- Solutions:
 - Changing the smoothing method of M_d
 - $p(w|M_d) = \frac{tf(w|d) + \delta}{|d| + \delta|V_F|}$
 - $tf(w|d)$ is the count of term w in d
 - $|d|$ is the length of d
 - $|V_F|$ the total number of distinct feedback terms in F
 - δ is a free parameter (set to 0.1)
 - Additional entropy component $(1 - \lambda - \beta)H(M_F)$
 - Use the posterior of document language models for α_d
 - $p(M_d|q) = \frac{P(q|M_d)P(M_d)}{\sum_{d \in D_{init}} P(M_d)P(q|M_d)}$
- $\hat{M}_F = \arg \min_{M_F} (\sum_{d \in F} \alpha_d CE(M_F, M_d) - \lambda CE(M_F, M_C) - \beta H(M_F))$
 - λ controls the IDF effect
 - β controls the entropy of the feedback language model

8

MEDMM

- $\hat{M}_F = \arg \min_{M_F} (\sum_{d \in F} \alpha_d CE(M_F, M_d) - \lambda CE(M_F, M_C) - \beta H(M_F))$
 - λ controls the IDF effect
 - β controls the entropy of the feedback language model
- Use Lagrange multiplier to solve the minimization problem
- $P(w|M_F) \propto \exp(\frac{1}{\beta} \cdot \sum_{d \in F} \alpha_d \cdot \log(p(w|M_d) - \frac{\lambda}{\beta} \cdot \log(p(w|M_c)))$
- Use the posterior of document language models for α_d
 - $p(M_d|q) = \frac{P(q|M_d)P(M_d)}{\sum_{d \in D_{init}} P(M_d)P(q|M_d)}$
- use term clipping to estimate a distribution M_F

9

Information Retrieval

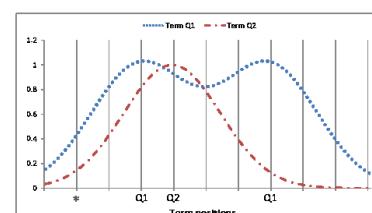
TUTORIAL 9 Positional Language Models

Positional Language Models Lv and Zhai 09'

- Motivation:** the language modeling approach does not consider the following
 - Term proximity heuristic
 - Passage retrieval
- Idea:** Positional Language Models (PLM)
 - Define a language model for each position of a document
 - Use the PLMs as the basis of the document score
 - The PLM at position i is estimated based on the propagated counts of words within a document
- Open questions:**
 - How to define the propagation function?
 - How to estimate the PLM and how to score the document?

2

Term Propagation

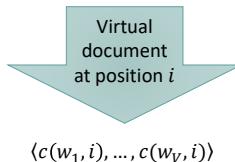


- PLM at each position can be estimated based on all propagated counts of all words to the position
- Treat a position as a document with all words appearing at that position with discounted counts

3

Estimation Of Positional Language Models

- How to induce a language model at document position i ?
 - $I(w, i) = 1$ iff term w occurs at position i
 - $k(i, j)$ – the propagated count to position i from a term at position j
 - Any non-increasing function of $|i - j|$
 - $c(w, i) = \sum_{j=1}^N I(w, j) * k(i, j)$, N is the document length
 - The total propagated count of term w at position i from the occurrences of w in all the positions
 - Even if $I(w, i) = 0$ the count may be greater than 0.



4

PLM & Smoothing

- PLM at position i (unsmoothed)

$$p_{MLE}(w|M_{d,i}) = \frac{c(w,i)}{\sum_{w' \in V} c(w',i)}$$

- Jelinek-Mercer (JM) smoothing

$$p_{JM}(w|M_{d,i}) = (1 - \lambda)p_{MLE}(w|M_{d,i}) + \lambda p_{MLE}(w|C)$$

- Bayesian smoothing with Dirichlet priors

$$p_{Dir}(w|M_{d,i}) = \frac{c(w,i) + \mu p_{MLE}(w|C)}{Z_i + \mu}$$

- $Z_i = \sum_{w \in V} c(w, i)$ - the length of the virtual document at position i

5

Similarity Measure

- To score a position given a query we can use the KL-divergence

$$S(q, d, i) = -KL(M_q | M_{d,i}) = -\sum_{w \in V} p(w|M_q) \log \frac{p(w|M_q)}{p(w|M_{d,i})}$$

where,

- V is the vocabulary
- $p(w|M_q)$ can be estimated using:
 - MLE
 - Pseudo relevance feedback algorithms – relevance model, MEDMM, mixture model

6

Propagation Functions - $k(i, j)$

1. Gaussian kernel

$$k(i, j) = \exp\left[-\frac{(i-j)^2}{2\sigma^2}\right]$$

2. Triangle kernel

$$k(i, j) = \begin{cases} 1 - \frac{|i - j|}{\sigma} & \text{if } |i - j| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

3. Passage kernel (non-proximity-based kernel)

$$k(i, j) = \begin{cases} 1 & \text{if } |i - j| \leq \sigma \\ 0 & \text{otherwise} \end{cases}$$

- σ – the steepness parameter

- “soft” passage representation

7

Analysis of the steepness parameter

- For $\sigma \rightarrow \infty$: $k(i, j) = 1$ for all i and j
- $c(w, i) = \sum_{j=1}^{|d|} I(w, j)k(i, j) = c(w, d)$
- Standard whole document language model ($p(w|M_d)$) is a private case of PLM
- Small σ emphasizes more local term proximity

8

PLM-Based Document Ranking

- How to score the entire document?

- We compute a position specific score using -KL
- We will use these scores for computing the entire document score

1. Best Position Strategy

- Score a document based on the score of its best matching position

$$S_{max}(q, d) \equiv \max_{i \in [1, N]} \{S(q, d, i)\}$$

2. Multi Position Strategy

- Score a document best on the top k positions

$$S_{multi}(q, d) \equiv \frac{1}{k} \sum_{i \in Top\ k} S(q, d, i)$$

3. Multi- σ Strategy

- Interpolation of the PLM and the regular document language model

$$S_{multi-\sigma}(q, d) \equiv \gamma * \max_{i \in [1, N]} \{S(q, d, i)\} + (1 - \gamma) * (-KL(M_q | M_d))$$

9

PLM – Results

method \ data	WT2G	TREC8	FR	AP88-89
KL	0.2931	0.2509	0.2697	0.2196
$\sigma = 25$	0.3247 ⁺	0.2562 ⁺	0.2936	0.2237⁺
$\sigma = 75$	0.3336⁺	0.2553 ⁺	0.2896 ⁺	0.2227
$\sigma = 125$	0.3330 ⁺	0.2559 ⁺	0.2885	0.2201
$\sigma = 175$	0.3324 ⁺	0.2574⁺	0.2858	0.2196
$\sigma = 275$	0.3255 ⁺	0.2561 ⁺	0.2852	0.2193

Table 5: The best performance of multi- σ strategy for different σ . '+' means that improvements over the baseline KL method are statistically significant.

10

Class Exercise

- Compute the similarity between the PLM induced at position 1 in document D2 with respect to the query $q = \text{"onion soup onion"}$
- Use DIR smoothing with $\mu = 100$ and MLE for the query model
- For the propagation function use the Gaussian kernel with $\sigma=5$

D1: onion vegetable soup vegetable mushroom

D2: corn onion soup

D3: potato pumpkin tofu potato tofu potato

term	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
tf	2	2	2	1	1	3	1	2
P_{MLE}	2/14	2/14	2/14	1/14	1/14	3/14	1/14	2/14

11

Class Exercise

D2: corn onion soup

- Estimate $M_{D_{2,1}}$
- For the propagation function use the Gaussian kernel with $\sigma=5$
- $c(\text{corn}, 1) = k(1,1) = 1$
- $c(\text{onion}, 1) = k(1,2) = \exp\left(-\frac{1}{2\sigma^2}\right) = \exp\left(-\frac{1}{50}\right) = 0.9801$
- $c(\text{soup}, 1) = k(1,3) = \exp\left(-\frac{4}{2\sigma^2}\right) = \exp\left(-\frac{4}{50}\right) = 0.9231$
- $Z_1 = c(\text{corn}, 1) + c(\text{onion}, 1) + c(\text{soup}, 1) = 1 + 0.9801 + 0.9231 = 2.9032$

term	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
tf	2	2	2	1	1	3	1	2
P_{MLE}	2/14	2/14	2/14	1/14	1/14	3/14	1/14	2/14

12

Class Exercise

D2: corn onion soup

- Estimate $M_{D_{2,1}}$
- Use DIR smoothing with $\mu = 100$
- , $c(\text{onion}, 1)=0.9801, c(\text{soup}, 1)=0.9231, Z_1=2.9032$
- $p(\text{onion}|M_{D_{2,1}}) = \frac{c(\text{onion}, 1)+\mu \cdot p(\text{onion}|C)}{Z_1+\mu} = \frac{0.9801+14.2857}{102.9032} = 0.1483$
- $p(\text{soup}|M_{D_{2,1}}) = \frac{0.9231+14.2857}{102.9032} = 0.1477$

term	onion	vegetable	soup	mushroom	corn	potato	pumpkin	tofu
tf	2	2	2	1	1	3	1	2
P_{MLE}	2/14	2/14	2/14	1/14	1/14	3/14	1/14	2/14

13

Class Exercise

D2: corn onion soup

- Compute the similarity between the PLM induced at position 1 in document D2 with respect to the query $q = \text{"onion soup onion"}$
- $p(\text{onion}|M_{D_{2,1}}) = \frac{c(\text{onion}, 1)+\mu \cdot p(\text{onion}|C)}{Z_1+\mu} = \frac{0.9801+14.2857}{102.9032} = 0.1483$
- $p(\text{soup}|M_{D_{2,1}}) = \frac{0.9231+14.2857}{102.9032} = 0.1477$
- $-KL(M_q || M_{D_{2,1}}) = -\frac{2}{3} \cdot \log\left(\frac{\frac{2}{3}}{0.1483}\right) - \frac{1}{3} \cdot \log\left(\frac{\frac{1}{3}}{0.1477}\right) = -1.2733$

14

Information Retrieval

TUTORIAL 10

Web Retrieval

Web Retrieval

- Web pages have links
 - Anchor text (hyperlink's text)
 - Pointer to a different page
- The WWW is directed graph
 - Links are edges
 - Pages are vertices
- Basic idea:
 - The graph structure is used to estimate $p(d)$ – the prior probability of a document to be relevant
 - Query independent

Page Rank

Brin&Page 98'

- Google's description of PageRank:
 - Relies on the "uniquely democratic" nature of the web
 - Interprets a link from page A to page B as "a vote"
- $A \rightarrow B$ means A thinks B is worth something
 - "wisdom of the crowds": many links mean B must be good
 - Content-independent measure of quality of B
- Use as a ranking feature, combined with content
 - But not all pages that link to B are of equal importance
 - A single link from BBC or CNN
- Google PageRank:
 - How many "good" pages link to B

Page Rank

Brin&Page 98'

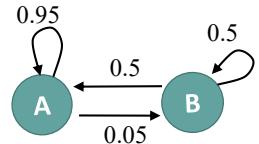
- Surfer's random walk on the web graph that contains N pages:
 1. Start at some page (current page)
 2. If the current page has out-links
 1. With probability λ select at random one out-link on current page and follow it
 2. With probability $(1-\lambda)$ "teleport" somewhere (out of N)
 3. Else (at "dead end") "teleport" somewhere (out of N)
 4. Go to step 2
- PageRank(d) = page d's visit rate during random walk

$$PR(v) = \sum_{u:out(u)>0} PR(u) \left(\frac{1-\lambda}{|N|} + \lambda \frac{\delta[u \rightarrow v]}{\sum_{v':u \rightarrow v'} 1} \right) + \sum_{u:out(u)=0} \frac{PR(u)}{|N|}$$

$\delta[u \rightarrow v] = 1$ iff there is an edge from u to v

Markov Chains – Abstractions Of Random Walks

- Markov chain:
 - Discrete-time stochastic process
 - N states + NxN transition probability matrix
 - Markov property: transition probability ($P_{i,j}$ from i to j) depends only on the current state (i)
 - $1 \leq i, j \leq N; 0 \leq P_{i,j} \leq 1$ and $\forall i, \sum_j P_{i,j} = 1$

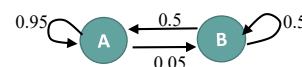


	A	B
A	0.95	0.05
B	0.5	0.5

• Example: A 2 states Markov chain

Markov Chains – Properties

- Probability vector $x \in R^N, 0 \leq x_i \leq 1$, describes the probability distribution over the states, $\sum_i x_i = 1$.
 - Examples:
 - $x=(1,0) \rightarrow$ at state A
 - $x=(0.5, 0.5) \rightarrow$ at A or B with equal probabilities.
- If at time t our probability distribution vector is x_t , then at time $t+1$: $x_{t+1} = x_t P$
- A Markov chain is ergodic if:
 - there exists a positive integer T_0 such that for **all pairs of states** i, j in the Markov chain, if it is started at time 0 in state i then for all $t > T_0$, the probability of being in state j at time t is **greater than 0**.



Markov Chains – Properties

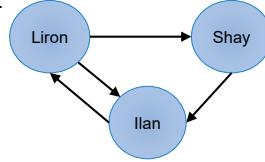
- Any transition probability matrix P has principal left eigenvector, i.e., there is v such as:

$$1v = vP$$
 (1 is eigenvalue)
- Ergodic Markov chain always has a "steady-state", i.e., there is a time T' such that for $t > T'$ a probability distribution x over states is fixed:

$$x_{t+1} = x_t P, x_{t+1} = x_t$$
- For ergodic Markov chain the "steady-state" probability vector x (that is the principal left eigenvector) equals to the long-term visit rate at each state.

Back To PageRank

- PageRank(d) is page d's long-time visit rate
- Surfer's random walk is ergodic Markov chain
- PageRank score is the steady state probability to be found at a page (state)
- Exercise:
 - Given the following network graph and $\lambda=0.9$:
 - Construct transition matrix
 - Present PR equations
 - Find PR scores



Solution

1. Transition matrix:

$$P_{Liron,Shay} = 0.5\lambda + \frac{(1-\lambda)}{3} = 0.45 + 0.033 = 0.483$$

2. Equations:

$$PR(v) = \sum_{u:out(u)>0} PR(u) \left(\frac{1-\lambda}{|N|} + \lambda \frac{\delta[u \rightarrow v]}{\sum_{v':u \rightarrow v'} 1} \right) + \sum_{u:out(u)=0} \frac{PR(u)}{|N|}$$

$$PR(Liron) = PR(Liron) \left(\frac{0.1}{3} \right) + PR(Shay) \left(\frac{0.1}{3} \right) + PR(Ilan) \left(\frac{0.1}{3} + 0.9 \right)$$

$$PR(Shay) = PR(Liron) \left(\frac{0.1}{3} + \frac{0.9}{2} \right) + PR(Shay) \left(\frac{0.1}{3} \right) + PR(Ilan) \left(\frac{0.1}{3} \right)$$

$$PR(Ilan) = PR(Liron) \left(\frac{0.1}{3} + \frac{0.9}{2} \right) + PR(Shay) \left(\frac{0.1}{3} + 0.9 \right) + PR(Ilan) \left(\frac{0.1}{3} \right)$$

Solution

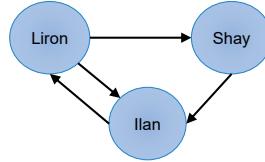
1. PR scores:

- Directly solving the equations
- Using the power method until convergence

$$PR(Liron) = 0.3919$$

$$PR(Shay) = 0.2097$$

$$PR(Ilan) = 0.3984$$



Hubs & Authorities(HITS)

Kleinberg '99

• Two types of web pages:

1. Hub pages contain good lists of links on a topic
2. Authority pages contain good information on a topic

• A circular definition

1. A good hub page points to many good authorities
2. A good authority page is pointed to by many good hubs

• Iteratively compute the hub and authority scores

Hubs & Authorities

Kleinberg '99

- Unlike page rank , HITS is not a query independent model
- We calculate the hub and authority score for each web page in the *base set*:

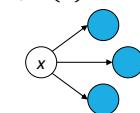
 1. Given a query, we use an initial ranking algorithm (for example, bm25 or LM) to retrieve a *root set*.
 2. The *base set* of pages include the root set as well as any page that either links to a page in the root set, or is linked to by a page in the root set

- We create two lists of the best hub and authority pages.

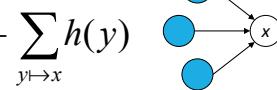
Hubs & Authorities

- Initialize: for all x , $h(x) \leftarrow 1$; $a(x) \leftarrow 1$

$$h(x) \leftarrow \sum_{x \mapsto y} a(y)$$



$$a(x) \leftarrow \sum_{y \mapsto x} h(y)$$



- To prevent the h and a values from getting too big, normalize after each iteration

- We only care about the relative values of the scores
- Use L2 normalization

Hubs & Authorities

Method 1:

if A is an adjacency matrix of the graph
for all x, $h(x) \leftarrow 1$; $a(x) \leftarrow 1$;

1. $h' = Aa$
2. $a' = A^T h$
3. $a=a'$, $h=h'$ [optional, L2 normalization of h' and a']
4. go to 1 or break if converged

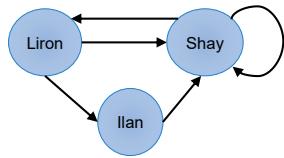
Method 2:

$$h' = Aa \quad \xrightarrow{\hspace{1cm}} \quad h' = AA^T h$$

$$a' = A^T h \quad a' = A^T A a$$

Example

- Calculate hub and authority scores for the following graph



Example

- Method 1:

A =	Liron	Shay	Ilan
Liron	0	1	1
Shay	1	1	0
Ilan	0	1	0

$A^T =$	Liron	Shay	Ilan
Liron	0	1	0
Shay	1	1	1
Ilan	1	0	0

iter. 0		iter. 1		iter. 2		iter. 3		iter. 4	
h	a	h	a	h	a	h	a	h	a
1	1	2	2	7	7	26	26	97	97
1	1	2	5	7	19	26	71	97	256
1	1	1	2	5	7	19	26	71	97

- Normalized h vector after iteration 3: (0.628, 0.628, 0.459)
- Normalized h vector after iteration 4: (0.628, 0.628, 0.459)
- Converged!**

Example

- Method 2:

$$a' = A^T A a$$

$$A^T A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$h' = AA^T h$$

$$AA^T = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

iteration	1	2	3
$a(Liron)$	2	7	26
$a(Shay)$	5	19	71
$a(Ilan)$	2	7	26

iteration	1	2	3
$h(Liron)$	4	15	56
$h(Shay)$	4	15	56
$h(Ilan)$	3	11	41

Converged after 3 iterations

Example

- What do the AA^T and $A^T A$ matrices represent?

$(AA^T)_{ij}$ is the number of pages both i and j point to

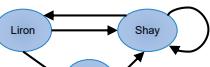
$(A^T A)_{ij}$ is the number of pages both i and j are pointed by

The number of out-links (hub)

The number of in-links (authority)

$$AA^T = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$



Information Retrieval

Tutorial 12
Learning to Rank

Recap

- A query: $q = (\dots \dots \dots)$
- A document: $d =$ 

2

Recap

- A query: $q = (\dots \dots \dots)$
- A document: $d =$ 

$f(\quad)$



Best

Worst

3

General Problem

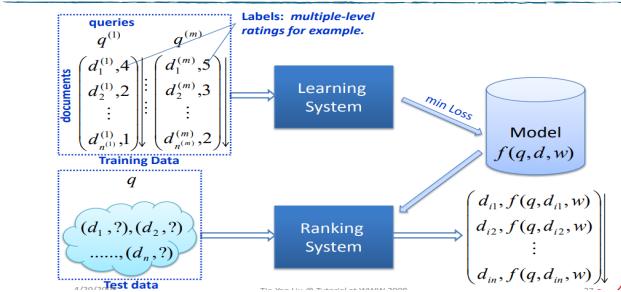
- Learn a ranker to increase the effectiveness of retrieval.
- Documents will be sorted by the score the ranker assigned.
- The data:
 - A set of queries and documents associated with each query.
 - Relevance judgements for the documents

General Problem

- Learning a scoring function f^* to sort a list of examples
 - Input: List of examples
 - Output: Scoring function f^* that produces the most optimal example ordering
 - RankSVM, LambdaMART, Neural Networks, etc...
- $q = (\dots \dots \dots)$  $V_{(d,q)} = (x_1, x_2, \dots, x_n)$
- $d =$ 

5

General Framework



*Taken from Liu '08

Cascading ranker

- In practice, Ranking all the documents with a model requires too much computation.
- 2 stage ranker:
 - Use an initial ranker (like bm25) to retrieve the top K documents for the query.
 - Rerank the top K results with the model!
- We can design a system with N stages and increase the complexity of the model used in each stage.

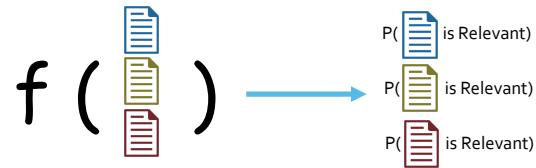
Question

- What are the 3 types of features that can be used in LTR model?
- Why using the tf-idf of the document as our feature vector is problematic?
- What happens when a document in our training set is retrieved by two different queries?

8

Pointwise Ranking

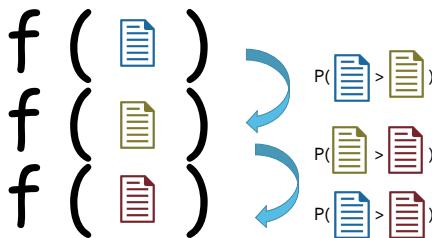
- Documents are considered independently of each other
 - Ordinal regression, classification
- $f(w, V_{(d,q)}) \rightarrow y$
 - w is a vector of model weights, y is the predicted value (relevance judgement)



9

Pairwise Ranking

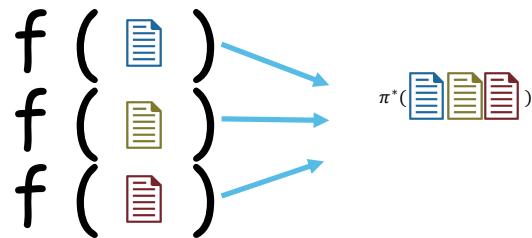
- Document pairs are considered
 - $f(w, V_{(d_i, q_j)}) \geq f(w, V_{(d_k, q_l)}) + \varepsilon$
 - RankNet, RankSVM



10

Listwise Ranking

- Consider the ordering of the entire list
 - ListNet, AdaRank



11

LTR approaches - summary

- Pointwise
 - The loss is calculated independently for each document
- Pairwise
 - The loss is calculated for each pair of documents with different relevance associated with the same query.
- Listwise
 - The ordering of the entire list is considered for the loss function

Features from Letor Dataset

examples of features used for the Letor

Dataset.

Full list available [here](#)

Table 2: Learning Features for the "Gov" Corpus

ID	Feature Description	Category
1	$\sum_{q_i \in q \cap d} c(q_i, d)$ in body	Q-D
2	$\sum_{q_i \in q \cap d} c(q_i, d)$ in anchor	Q-D
3	$\sum_{q_i \in q \cap d} c(q_i, d)$ in title	Q-D
4	$\sum_{q_i \in q \cap d} c(q_i, d)$ in URL	Q-D
5	$\sum_{q_i \in q} c(q_i, d)$ in whole document	Q-D
6	$\sum_{q_i \in q} idf(q_i)$ in body	Q
7	$\sum_{q_i \in q} idf(q_i)$ in anchor	Q
8	$\sum_{q_i \in q} idf(q_i)$ in title	Q
9	$\sum_{q_i \in q} idf(q_i)$ in URL	Q
10	$\sum_{q_i \in q} idf(q_i)$ in whole document	Q
11	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in body	Q-D
12	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in anchor	Q-D
13	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in title	Q-D
14	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in URL	Q-D
15	$\sum_{q_i \in q \cap d} c(q_i, d) \cdot idf(q_i)$ in whole document	Q-D
16	$ d $ of body	D
17	$ d $ of anchor	D
18	$ d $ of title	D
19	$ d $ of URL	D
20	$ d $ of whole document	D

Useful Libraries

- RankSVM: http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
- RankLib: <https://sourceforge.net/p/lemur/wiki/RankLib/>
- TFRanking: <https://github.com/tensorflow/ranking>
- LightGBM: <https://lightgbm.readthedocs.io/en/latest/>