Advanced Retrieval Models

[DAT640] Information Retrieval and Text Mining

Krisztian Balog
University of Stavanger

August 27, 2024

In this module

1. Fielded retrieval models
2. Query modeling
3. Web search
4. Learning to rank

Fielded retrieval models

Fielded retrieval models

- Quick recap of BM25 and LM
- Fielded extensions of BM25 and LM

BM25 scoring

BM25:

$$score(d,q) = \sum_{t \in q} \frac{c_{t,d} \times (1 + k_1)}{c_{t,d} + k_1(1 - b + b\frac{|d|}{avgdl})} \times idf_t$$

- Parameters
  - k_1: calibrating term frequency scaling
  - b: document length normalization
- Note: several slight variations of BM25 exist!

Language Models

Query likelihood scoring:

$$P(q|d) = \prod_{t \in q} P(t|\theta_d)^{c_{t,q}}$$

- θ_d is the document language model
  - Multinomial probability distribution over the vocabulary of terms
- $c_{t,q}$ is the raw frequency of term t in the query
- **Smoothing**: ensuring that $P(t|\theta_d)$ is > 0 for all terms

Smoothing

- Jelinek-Mercer smoothing:

$$P(t|\theta_d) = (1-\lambda)P(t|d) + \lambda P(t|C)$$

  - $\lambda \in [0, 1]$ is the smoothing parameter
  - Empirical document model (maximum likelihood estimate):

$$P(t|d) = \frac{c_{t,d}}{|d|}$$

  - Collection (background) language model (maximum likelihood estimate):

$$P(t|C) = \frac{\sum_{d'} c_{t,d'}}{\sum_{d'} |d'|}$$

- Dirichlet smoothing:

$$P(t|\theta_d) = \frac{c_{t,d} + \mu P(t|C)}{|d| + \mu}$$

  - μ is the smoothing parameter (typically ranges from 10 to 10000)

Fielded retrieval models

- Quick recap of BM25 and LM
- Fielded extensions of BM25 and LM

Motivation

- Documents are composed of multiple fields
  - E.g., title, body, anchors, etc.
- Modeling internal document structure may be beneficial for retrieval

Example

promise
PROMISE
Participative Research labOratory for Multimedia and Multilingual Information Systems Evaluation
Log-in
Overview
Achievements
Use cases
Publications
Events
CLEF
Media Center
Contacts
Search
Events
Winter School 2013
PROMISE Winter School 2013
Bressanone, Italy
Winter School 2013
Programme
Lecturers
Venue
Registration and Accomodation
Sponsor and Patronage
Flyer
Important Dates
Registration Deadline (extended): 28th
PROMISE Winter School 2013
Bridging between Information Retrieval and Databases
Bressanone, Italy 4 - 8 February 2013
The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured, semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as post-doctoral researchers form the fields of databases, information retrieval, and related fields.

Unstructured representation

```
PROMISE Winter School 2013
Bridging between Information Retrieval and Databases
Bressanone, Italy 4 - 8 February 2013
The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and
Databases" is to give participants a grounding in the core topics that constitute the
multidisciplinary area of information access and retrieval to unstructured, semistructured,
and structured information. The school is a week-long event consisting of guest lectures from
invited speakers who are recognized experts in the field. The school is intended for PhD
students, Masters students or senior researchers such as post-doctoral researchers form the
fields of databases, information retrieval, and related fields. [...]
```

Example

```html
<html>
<head>
   <title>Winter School 2013</title>
   <meta name="keywords" content="PROMISE, school, PhD, IR, DB, [...]" />
   <meta name="description" content="PROMISE Winter School 2013, [...]" />
</head>
<body>
   <h1>PROMISE Winter School 2013</h1>
   <h2>Bridging between Information Retrieval and Databases</h2>
   <h3>Bressanone, Italy 4 - 8 February 2013</h3>
   <p>The aim of the PROMISE Winter School 2013 on "Bridging between
   Information Retrieval and Databases" is to give participants a grounding
   in the core topics that constitute the multidisciplinary area of
   information access and retrieval to unstructured, semistructured, and
   structured information. The school is a week-long event consisting of
   guest lectures from invited speakers who are recognized experts in the
   field. The school is intended for  PhD students, Masters students or
   senior researchers such as post-doctoral researchers form the fields of
   databases, information retrieval, and related fields. </p>
   [...]
</body>
</html>
```

Fielded representation (based on HTML markup)

d_1: **title**	`Winter School 2013`
d_2: **meta**	`PROMISE, school, PhD, IR, DB, [...]` `PROMISE Winter School 2013, [...]`

d_3: **headings**	`PROMISE Winter School 2013`
	`Bridging between Information Retrieval and Databases`
	`Bressanone, Italy 4-8 February 2013`

d_4: **body**	`The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured,semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as postdoctoral researchers form the fields of databases, information retrieval, and related fields. [...]`

Fielded extension of retrieval models

- BM25 $\Rightarrow$ BM25F
- Language Models (LM) $\Rightarrow$ Mixture of Language Models (MLM)

BM25F

- Extension of BM25 incorporating multiple fields
- The soft normalization and term frequencies need to be adjusted
- Original BM25 retrieval function:

$$score(d, q) = \sum_{t \in q} \frac{c_{t,d} \times (1 + k_1)}{c_{t,d} + k_1 \times B} \times idf_t$$

- where B is is the soft normalization:

$$B = (1 - b + b\frac{|d|}{avgdl})$$

BM25F

- Replace term frequencies $c_{t,d}$ with *pseudo term frequencies* $\tilde{c}_{t,d}$
- BM25F retrieval function:

$$score(d,q) = \sum_{t \in q} \frac{\tilde{c}_{t,d}}{k_1 + \tilde{c}_{t,d}} \times idf_t$$

- Pseudo term frequency calculation

$$\tilde{c}_{t,d} = \sum_{i} w_i \times \frac{c_{t,d_i}}{B_i}$$

- where
  - i corresponds to the field index
  - w_i is the field weight (such that $\sum_i w_i = 1$)
  - B_i is soft normalization for field i, where b_i becomes a field-specific parameter

$$B_i = (1 - b_i + b_i \frac{|d_i|}{avgdl_i})$$

Mixture of Language Models (MLM)

- Idea: Build a separate language model for each field, then take a linear combination of them

$$P(t|\theta_d) = \sum_i w_i P(t|\theta_{d_i})$$

- where
  - i corresponds to the field index
  - w_i is the field weight (such that $\sum_i w_i = 1$)
  - $P(t|\theta_{d_i})$ is the field language model

Field language model

- Smoothing goes analogously to document language models, but term statistics are restricted to the given field i
- Using Jelinek-Mercer smoothing:

$$P(t|\theta_{d_i}) = (1 - \lambda_i)P(t|d_i) + \lambda_i P(t|C_i)$$

- where both the empirical field model $(P(t|d_i))$ and the collection field model $(P(t|C_i))$ are maximum likelihood estimates:

$$P(t|d_i) = \frac{c_{t,d_i}}{|d_i|} \qquad P(t|C_i) = \frac{\sum_{d'} c_{t,d'_i}}{\sum_{d'} |d'_i|}$$

Setting parameter values

- Retrieval models often contain parameters that must be tuned to get the best performance for specific types of data and queries
- For experiments
    - Use training and test data sets
    - If less data available, use cross-validation by partitioning the data into k subsets
- Many techniques exist to find optimal parameter values given training data
    - Standard problem in machine learning
- For standard retrieval models, involving few parameters, *grid search* is feasible
    - Perform a sweep over the possible values of each parameter, e.g., from 0 to 1 in steps of 0.1

Query modeling

Query modeling based on feedback

- Take the results of a user's actions or previous search results to improve retrieval
- Often implemented as updates to a query, which then alters the list of documents
- Overall process is called **relevance feedback**, because we get feedback information about the relevance of documents
    - **Explicit feedback**: user provides relevance judgments on some documents
    - **Pseudo relevance feedback** (or *blind feedback*): we don't involve users but "blindly" assume that the top-k documents are relevant
    - **Implicit feedback**: infer relevance feedback from users' interactions with the search results (clickthroughs)

Feedback in an IR system

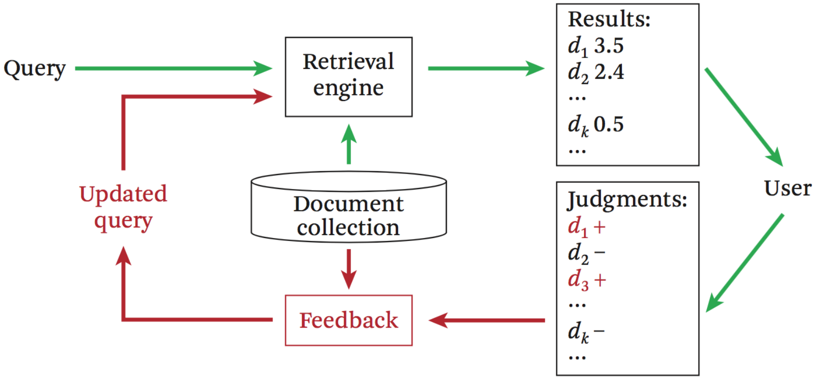


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 7.1]

Feedback in the Vector Space Model

- It is assumed that we have examples of relevant (D^+) and non-relevant (D^-) documents for a given query
- General idea: modify the query vector (adjust weight of existing terms and/or assign weight to new terms)
  - As a result, the query will usually have more terms, which is why this method is often called **query expansion**

Rocchio feedback

- Idea: adjust the weights in the query vector to move it closer to the cluster of relevant documents

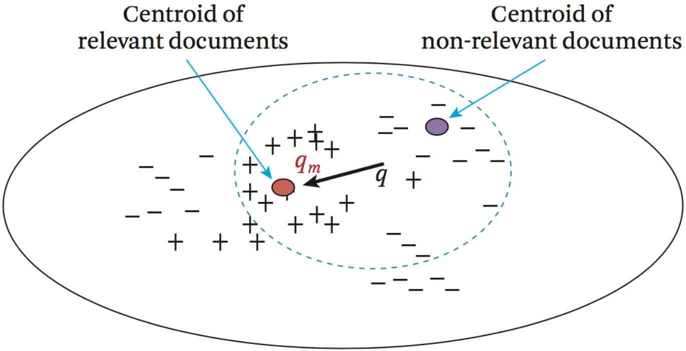


Figure: Illustration is taken from (Zhai&Massung, 2016)[Fig. 7.2]

Rocchio feedback

- Modified query vector:

$$\vec{q}_m = \alpha \vec{q} + \frac{\beta}{|D^+|} \sum_{d \in D^+} \vec{d} - \frac{\gamma}{|D^-|} \sum_{d \in D^-} \vec{d}$$

  - $\vec{q}$: original query vector
  - D^+, D^-: set of relevant and non-relevant feedback documents
  - α, β, γ: parameters that control the movement of the original vector
- The second and third terms of the equation correspond to the centroid of relevant and non-relevant documents, respectively

Practical considerations

- Modifying all the weights in the query (and then using them all for scoring documents) is computationally heavy
  - Often, only terms with the highest weights are retained
- Non-relevant examples tend not to be very useful
  - Sometimes negative examples are not used at all, or γ is set to a small value

Exercise

E3-1 Rocchio feedback

Feedback in Language Models

- We generalize the query likelihood function to allow us to include feedback information more easily
- (Log) query likelihood

$$\log P(q|d) \propto \sum_{t \in q} c_{t,q} \times \log P(t|\theta_d)$$

- Generalize $c_{t,q}$ to a query model $P(t|\theta_q)$

$$\log P(q|d) \propto \sum_{t \in q} P(t|\theta_q) \times \log P(t|\theta_d)$$

  - Often referred to as **KL-divergence** retrieval, because it provides the same ranking as minimizing the Kullback-Leibler divergence between the query model θ_q and the document model θ_d
  - Using a maximum likelihood query model this is rank-equivalent to query likelihood scoring

Query models

- Maximum likelihood estimate (original query)

$$P_{ML}(t|\theta_q) = \frac{c_{t,q}}{|q|}$$

  - I.e., the relative frequency of the term in the query
- Linear interpolation with a feedback query model $\hat{\theta}_q$

$$P(t|\theta_q) = \alpha P_{ML}(t|\theta_q) + (1-\alpha)P(t|\hat{\theta}_q)$$

  - α has the same interpretation as in the Rocchio feedback model, i.e., how much we rely on the original query

Relevance models

- **Relevance models** are a theoretically sound and effective way of estimating feedback query models
- Main idea: consider other terms that co-occur with the original query terms in the set of feedback documents $\hat{D}$
  - Commonly taken to be the set of top-k documents (k=10 or 20) retrieved using the original query with query likelihood scoring
- Two variants with different independence assumptions
- Relevance model 1
  - Assume full independence between the original query terms and the expansion terms:

$$P_{RM1}(t|\hat{\theta}_q) \approx \sum_{d \in \hat{D}} P(d) P(t|\theta_d) \prod_{t' \in q} P(t'|\theta_d)$$

  - Often referred to as *RM3* when linearly combined with the original query

Relevance models

- Relevance model 2
  - The original query terms $t' \in q$ are still assumed to be independent of each other, but they are dependent on the expansion term t:

$$P_{RM2}(t|\hat{\theta}_q) \approx P(t) \prod_{t' \in q} \sum_{d \in \hat{D}} P(t'|\theta_d) P(d|t)$$

  - where $P(d|t)$ is computed as

$$P(d|t) = \frac{P(t|\theta_d)P(d)}{P(t)} = \frac{P(t|\theta_d)P(d)}{\sum_{d' \in \hat{D}} P(t|\theta_{d'})P(d')}$$

Illustration

t	$P_{ML}(t\|\theta_q)$	t	$P(t\|\theta_q)$
machine	0.5000	vision	0.2796
vision	0.5000	machine	0.2762
		image	0.0248
		vehicles	0.0224
		safe	0.0220
		cam	0.0214
		traffic	0.0178
		technology	0.0176
		camera	0.0173
		object	0.0147

Table: Baseline (left) and expanded (right) query models for the query *machine vision*; only the top 10 terms are shown.

Feedback summary

- Overall goal is to get a richer representation of the user's underlying information need by enriching/refining the initial query
- Interpolation with the original query is important
- Relevance feedback is computationally expensive! Number of feedback terms and expansion terms are typically limited (10..50) for efficiency considerations
- Queries may be hurt by relevance feedback ("query drift")

Web search

Web search

- Before the Web: search was small scale, usually focused on libraries
- Web search is a major application that everyone cares about
- Challenges
    - Scalability (users as well as content)
    - Ensure high-quality results (fighting SPAM)
    - Dynamic nature (constantly changing content)

Some specific techniques

- Crawling
  - Freshness
  - Focused crawling
  - Deep Web crawling
- Indexing
  - Distributed indexing
- **Retrieval** $\Leftarrow$
  - Link analysis

Deep (or hidden) Web

- Much larger than the "conventional" Web
- Three broad categories:
  - Private sites
    - No incoming links, or may require log in with a valid account
  - Form results
    - Sites that can be reached only after entering some data into a form
  - Scripted pages
    - Pages that use JavaScript, Flash, or another client-side language to generate links

Question

How to make content on the Deep Web searchable (indexable)?

Surfacing the Deep Web

- Pre-compute all interesting form submissions for each HTML form
- Each form submission corresponds to a distinct URL
- Add URLs for each form submission into search engine index

Link analysis

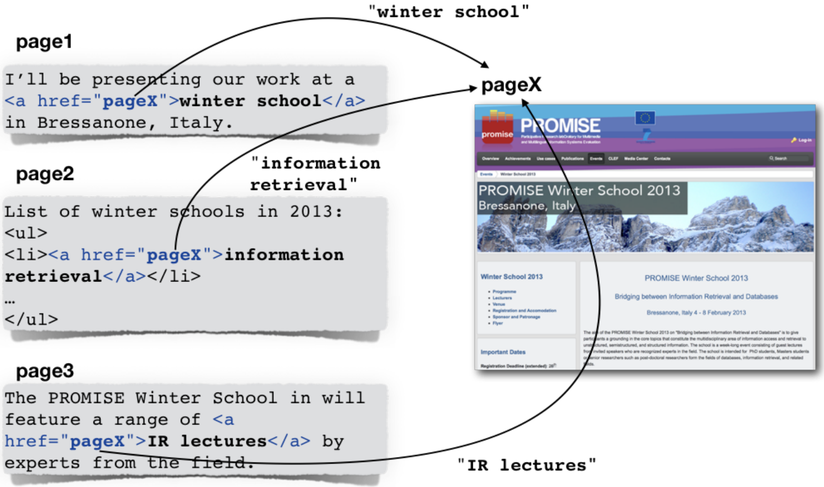
- Links are a key component of the Web
- Important for navigation, but also for search

- Both anchor text and links are used by search engines

Anchor text

- Aggregated from all incoming links and added as a separate document field
- Tends to be short, descriptive, and similar to query text
  - Can be thought of a description of the page “written by others”
- Has a significant impact on effectiveness for *some types of queries*

Example



page1
I’ll be presenting our work at a
winter school
in Bressanone, Italy.
page2
List of winter schools in 2013:

information
retrieval
…

page3
The PROMISE Winter School in will
feature a range of <a
href="pageX">IR lectures by
experts from the field.
"winter school"
"information
retrieval"
"IR lectures"
pageX
PROMISE
promise
PROMISE Winter School 2013
Bressanone, Italy
Winter School 2013
Programme
Lecturers
Venue
Registration and Accomodation
Sponsor and Patronage
Flyer
Important Dates
PROMISE Winter School 2013
Bridging between Information Retrieval and Databases
Bressanone, Italy 4 - 8 February 2013

Fielded document representation

title	Winter School 2013
meta	PROMISE, school, PhD, IR, DB, [...] PROMISE Winter School 2013, [...]
headings	PROMISE Winter School 2013 Bridging between Information Retrieval and Databases Bressanone, Italy 4-8 February 2013
body	The aim of the PROMISE Winter School 2013 on "Bridging between Information Retrieval and Databases" is to give participants a grounding in the core topics that constitute the multidisciplinary area of information access and retrieval to unstructured,semistructured, and structured information. The school is a week-long event consisting of guest lectures from invited speakers who are recognized experts in the field. The school is intended for PhD students, Masters students or senior researchers such as postdoctoral researchers form the fields of databases, information retrieval, and related fields. [...]
anchors	winter school information retrieval IR lectures

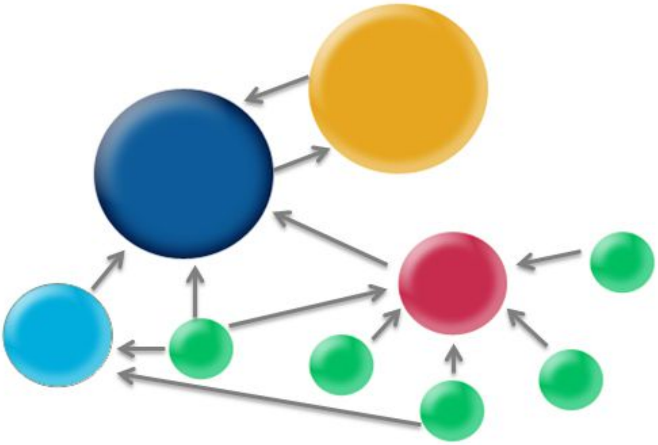
Document importance on the Web

- What are web pages that are popular and useful to *many* people?
- Use the links between web pages as a way to measure popularity
- The most obvious measure is to count the number of *inlinks*
    - Quite effective, but very susceptible to SPAM

PageRank

- Algorithm to rank web pages by popularity
- Proposed by Google founders Sergey Brin and Larry Page in 1998
- Main idea: **A web page is important if it is pointed to by other important web pages**
- PageRank is a numeric value that represents the importance of a web page
  - When one page links to another page, it is effectively casting a vote for the other page
  - More votes implies more importance
  - Importance of each vote is taken into account when a page's PageRank is calculated

Illustration



Source: `https://www.shoutmeloud.com/how-to-calculate-pagerank-google-seo.html`

Random Surfer Model

- PageRank simulates a user navigating on the Web randomly as follows
- The user is currently at page a
  - She moves to one of the pages linked from a with probability $1 - q$
  - She jumps to a random web page with probability q
    - This is to ensure that the user doesn't "get stuck" on any given page (i.e., on a page with no outlinks)
- Repeat the process for the page she moved to
- The PageRank score of a page is the average probability of the random surfer visiting that page

PageRank formula

Jump to a random page with this probability (q is typically set to 0.15)

Follow one of the hyperlinks in the current page with this probability

PageRank value of page p_i

$$PR(a) = \frac{q}{T} + \overbrace{(1-q)} \sum_{i=1}^{n} \frac{PR(p_i)}{L(p_i)}$$

PageRank of page a

Total number of pages in the Web graph

page a is pointed by pages $p_1 \ldots p_n$

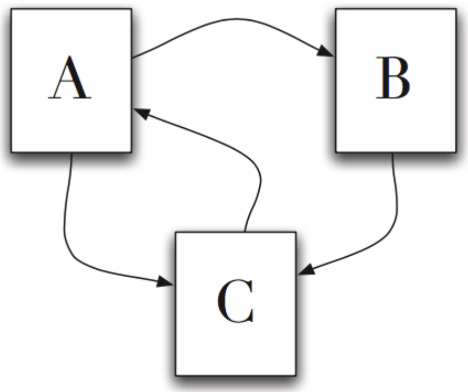
Number of outgoing links of page p_i

Technical issues

- This is a recursive formula. PageRank values need to be computed iteratively
  - We don't know the PageRank values at start. We can assume equal values $(1/T)$
- Number of iterations?
  - Good approximation already after a small number of iterations; stop when change in absolute values is below a given threshold

Example #1

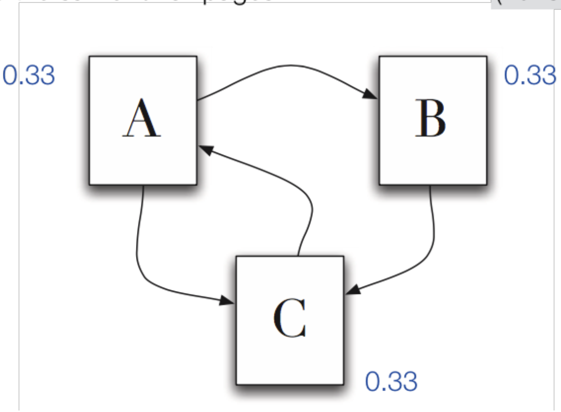
q=0
(no random jumps)



Example #1

Iteration 0: assume that the PageRank values are the same for all pages

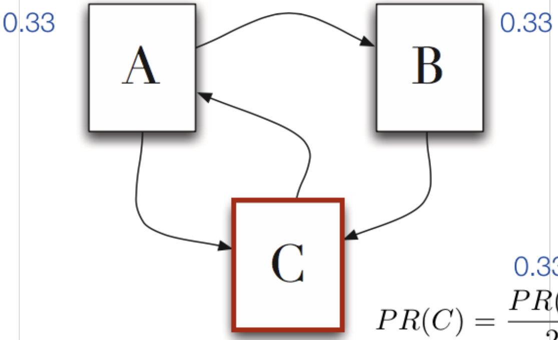
q=0
(no random jumps)



Example #1

Iteration 1

q=0
(no random jumps)



$$PR(C) = \frac{\overset{0.33}{PR(A)}}{2} + \frac{\overset{0.33}{PR(B)}}{1}$$

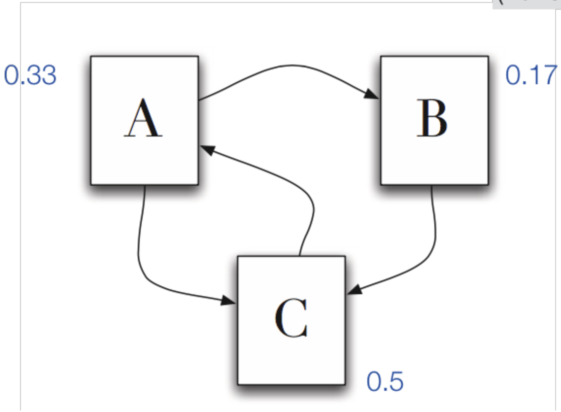
=0.5

PageRank of C depends on the PageRank values of A and B

Example #1

at the end of **Iteration 1**

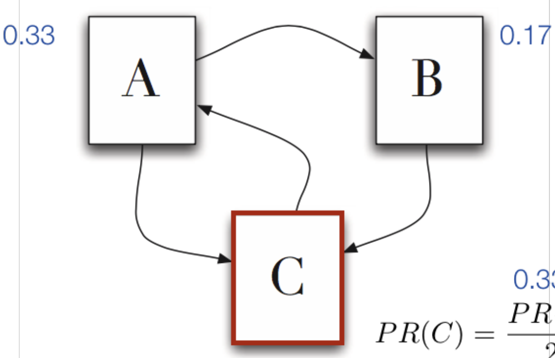
q=0
(no random jumps)



Example #1

Iteration 2

q=0
(no random jumps)



$$PR(C) = \frac{\overset{0.33}{PR(A)}}{2} + \frac{\overset{0.17}{PR(B)}}{1}$$

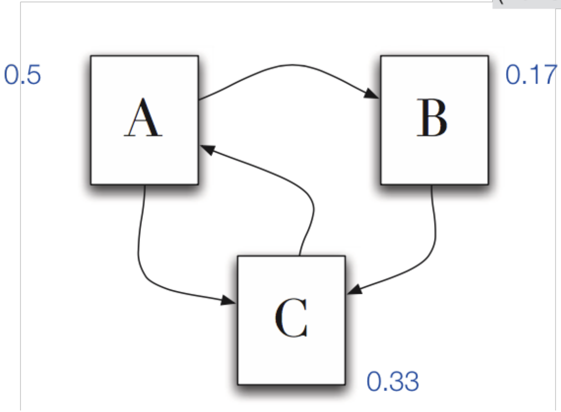
=0.33

PageRank of C depends on the PageRank values of A and B

Example #1

at the end of **Iteration 2**

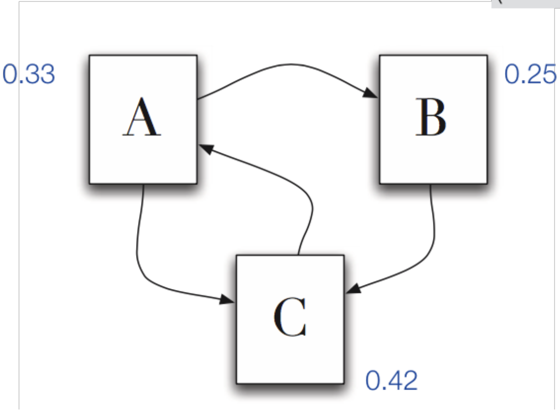
q=0
(no random jumps)



Example #1

at the end of **Iteration 3**

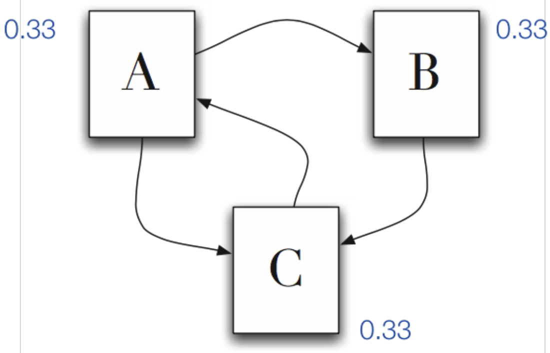
q=0
(no random jumps)



Example #2

Iteration 0: assume that the PageRank values are the same for all pages

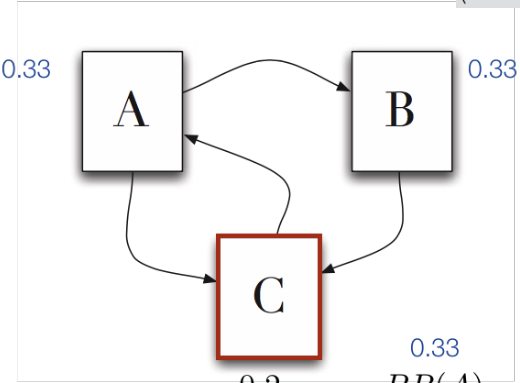
q=0.2
(with random jumps)



Example #2

Iteration 1

q=0.2
(with random jumps)



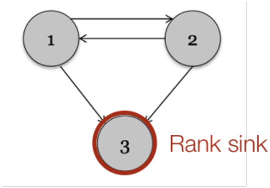
$$PR(C) = \frac{0.2}{3} + 0.8\left(\frac{\overset{0.33}{PR(A)}}{2} + \frac{\overset{0.33}{PR(B)}}{1}\right)$$ **=0.47**

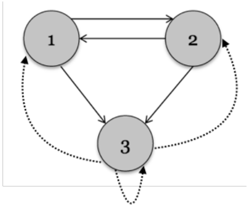
Question

How are PageRank scores affected by pages that do not have any outgoing links?

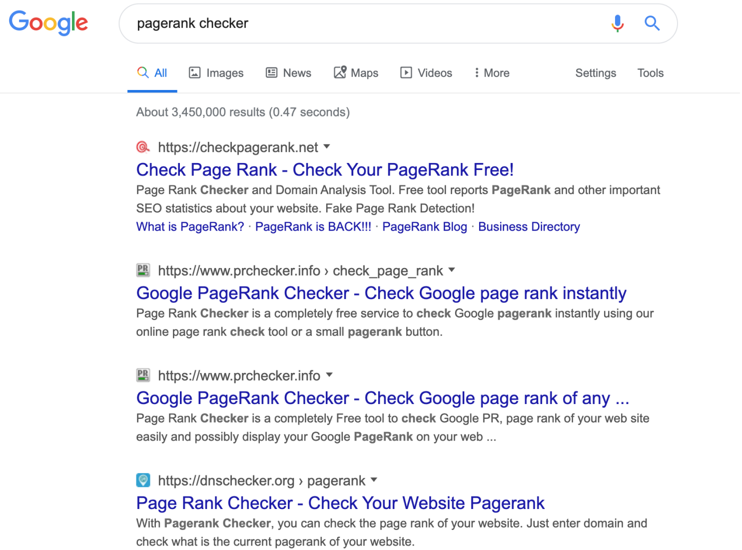
Dealing with "rank sinks"

- How to handle *rank sinks* ("dead ends"), i.e., pages that have no outlinks?
- Assume that it links to all other pages in the collection (including itself) when computing PageRank scores





Online PageRank checkers



PageRank summary

- Important example of query-independent document ranking
  - Web pages with high PageRank are preferred
- It is, however, not as important as conventional wisdom holds
  - Just one of the many features a modern web search engine uses
  - It tends to have the most impact on popular queries

Exercise

E3-2 PageRank

Incorporating document importance (e.g., PageRank)

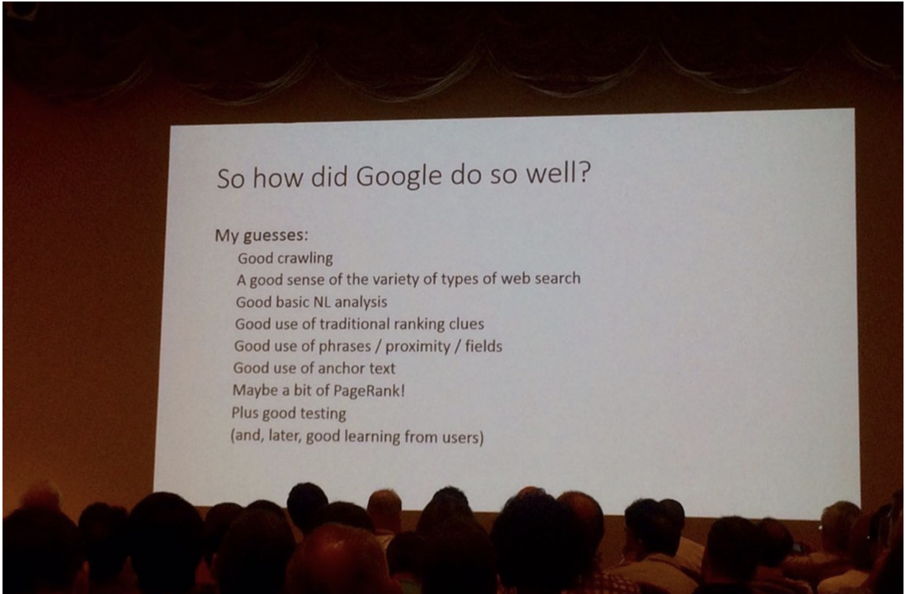
- How to incorporate document importance into the ranking?
- As a query-independent ("static") score component

$$score'(d,q) = score(d,q) \times score(d)$$

- In case of Language Models, document importance is encoded as the document prior $P(d)$

$$P(d|q) \propto P(q|d)P(d)$$

Stephen Robertson, SIGIR'17 keynote



Question

What is search engine optimization (SEO)?

Learning to rank

Learning to rank

- Motivation
- Learning to rank methods
- Practical considerations

Recap

- Classical retrieval models
  - Vector space model, BM25, LM
- Three main components
  - Term frequency
    - How many times query terms appear in the document
  - Document length
    - Any term is expected to occur more frequently in long document; account for differences in document length
  - Document frequency
    - How often the term appears in the entire collection

Additional factors

- So far: content-based matching
- Many additional signals, e.g.,
  - Document quality
    - PageRank
    - SPAM score
    - ...
  - Implicit (click-based) feedback
    - How many times users clicked on a document given a query?
    - How many times this particular user clicked on a document given the query?
    - ...
  - ...

Question

How to combine all these clues for ranking?

Learning to rank

- Motivation
- Learning to rank methods
- Practical considerations

Machine learning for IR

- We hypothesize that the probability of relevance is related to some combination of *features*
  - Each feature is a clue or signal that can help determine relevance
- We employ machine learning to learn an "optimal" combination of features, based on training data
  - There may be several hundred features; impossible to tune by hand
  - Training data is (item, query, relevance) triples
- Modern systems (especially on the Web) use a great number of features
  - In 2008, Google was using over 200 features[1]

[1] The New York Times (2008-06-03)

Some example features

- Log frequency of query word in anchor text
- Query word in color on page?
- #images on page
- #outlinks on page
- PageRank
- URL length
- URL contains “$\sim$”?
- Page length
- ...

Simple example

- We assume that the relevance of a document is related to a linear combination of all the features:

$$\log \frac{P(R=1|q,d)}{1-P(R=1|q,d)} = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

  - x_i is the value of the i^{th} feature
  - β_i is the weight of the i^{th} feature

- This leads to the following probability of relevance:

$$P(R=1|q,d) = \frac{1}{1+e^{(-\beta_0 - \sum_{i=1}^{n} \beta_i x_i)}}$$

- This *logistic regression* method gives us an estimate in $[0,1]$

Learning to Rank (LTR)

- Learn a function automatically to rank items (documents) effectively
  - Training data: (item, query, relevance) triples
  - Output: ranking function $h(q, d)$
- Three main groups of approaches
  - Pointwise
  - Pairwise
  - Listwise

Pointwise LTR

- Specifying whether a document is relevant (binary) or specifying a degree of relevance
  - Classification: Predict a categorical (unordered) output value (relevant or not)
  - Regression: Predict an ordered or continuous output value (degree of relevance) $\Leftarrow$
- All the standard classification/regression algorithms can be directly used
- Note: classical retrieval models are also point-wise: $score(q, d)$

Pointwise LTR

- **Input:** A set of instances, with queries, documents, and relevance labels
- **Objective:** Learn a model that can predict the relevance score for each document independently, without considering the order or relationship between documents in a ranked list
- **Loss function:** Typically, regression loss (e.g., mean squared error or mean absolute error) that measures the difference between predicted and actual relevance scores

Pairwise LTR

- The learning function is based on a pair of items
  - Given two documents, classify which of the two should be ranked at a higher position
  - I.e., learning relative preference
- E.g., Ranking SVM, LambdaMART, RankNet

Pairwise LTR

- **Input:** A set of instances, with queries, documents, and relevance labels
- **Objective:** Learn a model that can compare pairs of documents and determine which one is more relevant within each pair
- **Loss function:** Pairwise loss function (e.g., hinge loss or logistic loss) that encourages the correct ranking of pairs

Listwise LTR

- The ranking function is based on a ranked list of items
    - Given two ranked list of the same items, which is better?
- Directly optimizes a retrieval metric
    - Need a loss function on a list of documents
    - Can get fairly complex compared to pointwise or pairwise approaches
- Challenge is scale: huge number of potential lists
- E.g., AdaRank, ListNet, LamdaRank

Listwise LTR

- **Input:** A set of instances with queries and lists of documents, where each list is a ranked collection of documents with relevance scores
- **Objective:** Learn a model that can optimize the ranking order of entire lists of documents, taking into account the interactions and dependencies between documents in each list
- **Loss function:** Listwise loss function (e.g., Discounted Cumulative Gain, Normalized Discounted Cumulative Gain, or ListNet loss) that quantifies the quality of the entire ranked list

How to?

- Develop a feature set
  - The most important step!
  - Usually problem dependent
- Choose a good ranking algorithm
  - E.g., Random Forests work well for pairwise LTR
- Training, validation, and testing
  - Similar to standard machine learning applications

Features for document retrieval

- Query features
  - Depend only on the query
- Document features
  - Depend only on the document
- Query-document features
  - Express the degree of matching between the query and the document

Query features

- Query length (number of terms)
- Sum of IDF scores of query terms in a given field (title, content, anchors, etc.)
- Total number of matching documents
- Number of named entities in the query
- ...

Document features

- Length of each document field (title, content, anchors, etc.)
- PageRank score
- Number of inlinks
- Number of outlinks
- Number of slash in URL
- Length of URL
- ...

Query-document features

- Retrieval score of a given document field (e.g., BM25, LM, TF-IDF)
- Sum of TF scores of query terms in a given document field (title, content, anchors, URL, etc)
- Retrieval score of the entire document (e.g., BM25F, MLM)
- ...

Group work

Imagine you are working on building a job recommendation system for an online job portal. The goal of the system is to provide personalized job recommendations to users based on their profiles and preferences.
Design a feature set that could be used to build a machine learning model for this job recommendation system. Consider features related to user profiles, job descriptions, interactions, and any other relevant information that could enhance the quality of job recommendations.

Learning to rank

- Motivation
- Learning to rank methods
- Practical considerations

Feature normalization

- Feature values are often normalized to be in the $[0, 1]$ range *for a given query*
  - Esp. matching features that may be on different scales across queries because of query length
- Min-max normalization:

$$\tilde{x}_i = \frac{x_i - \text{min}(x)}{\text{max}(x) - \text{min}(x)}$$

  - $x_1, \ldots, x_n$: original values for a given feature
  - $\tilde{x}_i$: normalized value for the i^{th} instance

Class imbalance and computation cost

- Many more non-relevant than relevant instances
- Classifiers usually do not handle huge imbalance well
- Also, it is not feasible to extract features for all documents in the corpus
- Sampling is needed!

Two-stage ranking pipeline

- Implemented as a re-ranking mechanism (two-step retrieval)
  - Step 1 (initial ranking): Retrieve top-N (N=100 or 1000) candidate documents using a baseline approach (e.g., BM25). (This, essentially, is *document sampling*.)
  - Step 2 (re-ranking): Create feature vectors and re-rank these top-N candidates to arrive at the final ranking
- Often, candidate documents from first-pass retrieval and labeled (judged) documents are combined together for learning a model
  - Retrieved but not judged documents are assumed to be non-relevant
- Feature computation
  - Document features may be computed offline
  - Query and query-document features are computed online (at query time)
  - Avoid using too many expensive features!

Summary

- Fielded extensions of retrieval models (BM25F, MLM)
- Types of relevance feedback (explicit, implicit, pseudo/blind)
- Query expansion in the vector space model (Rocchio feedback)
- Query expansion in a language modeling setting (Relevance Models)
- Utilizing links on the Web: anchor text and link structure
- PageRank algorithm
- Learning ranking functions (learning to rank)
- Main classes of learning-to-rank approaches (pointwise, pairwise, listwise)
- Types of features (query, document, query-document)
- Practical considerations in learning-to-rank

Reading

- Text Data Management and Analysis (Zhai&Massung)
  - Chapter 7
  - Chapter 10: Sections 10.3, 10.4
- Relevance-Based Language Models (Lavrenko&Croft, 2001)
  - `https://sigir.org/wp-content/uploads/2017/06/p260.pdf`