Information Retrieval and Web Search

- Text Preprocessing and Representation: Vector Space Modeling
- Inverted Index, Boolean and Ranked Retrieval
- Web Search: Content Spamming, Trustworthy Webpages and Node Centrality
- PageRank: Dangling Nodes, Spider Traps, Random Teleporting
- Personalized and Topic Sensitive Pagerank
- Hyperlink-Induced Topic Search

Imdad ullah Khan

Information Retrieval (IR)

Information Retrieval (IR) is the science of searching

- for information in documents
- for documents themselves
- for metadata which describe data
- for information in images, sounds, videos etc.

Aim:

- To retrieve information that is relevant to the user's information need
- To organize and deliver the most relevant information efficiently.

Information Retrieval (IR)

Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Information Retrieval (IR) in the age of Big Data

- Data Overload: Sift through massive amounts of data effectively
- Decision Making: Facilitate informed decision making
- Accessibility: Make vast stores of data accessible and usable

- Web Search Engines: Web search, E-mail search, File search in your laptop (Google, Bing) ▷ Search (a fundamental computational problem) and communication are most popular uses of the computer
- Corporate Data Management: Helps in retrieving and managing internal documents and records
- Healthcare/Legal Practice: Used in medical and legal databases to help practitioners find relevant cases and studies
- Multimedia Retrieval: Used for finding images, videos, and audio based on content

Information Retrieval: Problem Formulation

The Information Retrieval Problem

- Given a collection of documents
  Assume it is a static collection
- Retrieve documents with information that is *"relevant"* to the user's need and helps the user complete a task

Main Goals of an IR System:

- Relevance: Retrieve documents that are relevant to the user's query
- Completeness: Ensure no relevant documents are omitted from retrieval
- Efficiency: Deliver results quickly and use resources efficiently
- Usability: Provide an interface that helps users in formulating queries and interpreting results

- Precision: The accuracy of the retrieval (correctness)
- Recall: The completeness of the retrieval (coverage)

Information Retrieval: Key Issues

- **Document Processing:** "effectively" describe information resources (documents containing information)
  - Feature extraction and Representation
  - Organization and Storage
  - Indexing
- **Query Processing:** find the "appropriate" information resources
  - Understand user's intent
  - Accessing
  - Filtering
  - Retrieving
- **Matching and Ranking:** present "relevant" information in ranked order
  - Ranking
  - Presenting

Information Retrieval: Static vs. Dynamic Collections

- Static Collections: Data sets that do not change frequently

  e.g., Archived articles, digital libraries

  ▷ require efficient indexing and retrieval strategies that do not need frequent updates

- Dynamic Collections: Continuously updating data sets

  e.g., Social media feeds, news websites

  ▷ pose challenges in terms of real-time indexing and the ability to quickly incorporate new information into the retrieval process

Information Retrieval: Keywords

- Automated IR systems designed in 60's to search news articles, scientific papers, patents, legal abstracts for keyword queries
- Users have a clear task in mind but express it vaguely

  ▷ Keywords are short and inexpressive
- Problem of **synonymy** (different words with same meaning)
  - "soccer" would not retrieve documents containing the word "football"
- Problem of **polysemy** (same words with different meanings)
  - "Lie" could mean to lay down and to tell something untruthful
  - "fair" could be an event or could refer to skin color, or as in "just"
- Early IR system had two distinct features
  1. Retrieval done by experts (reference librarians, patents attorneys)
  2. Search space (knowledge base) were collection of expert-written docs

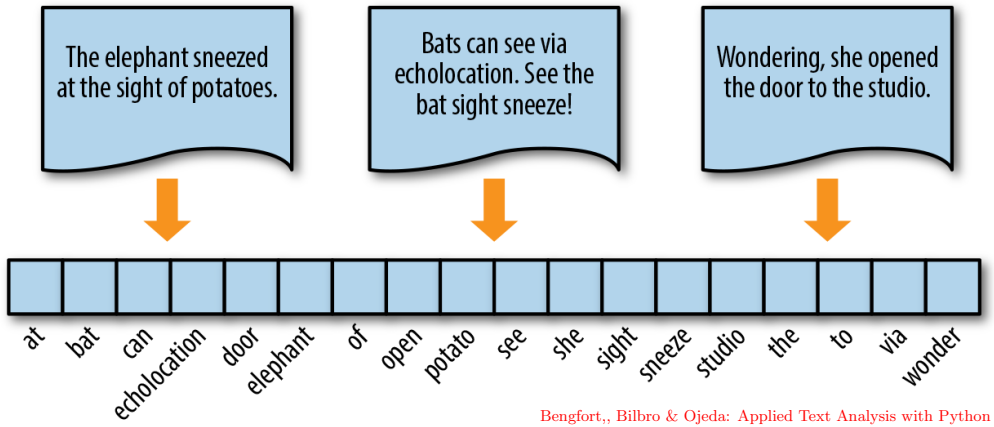
Now everyone is an author and everyone is a searcher

Text Representation

Vector Space Models

Vector Space Models

- Algorithms cannot work with raw texts directly
- Calculate similarity/difference between two documents?
- Convert texts to vectors. Vector Space Modeling

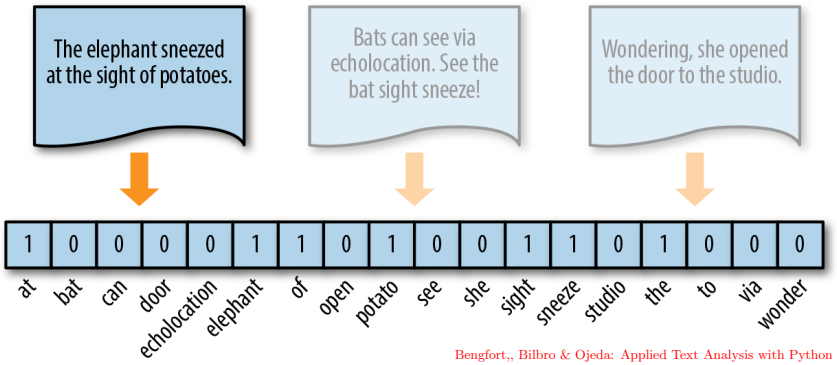


Bengfort,, Bilbro & Ojeda: Applied Text Analysis with Python

- Extract features from texts to reflect linguistic properties of the text
- Popular feature extraction methods (VSM variations) are
  - Set-of-Words: Binary word occurrences
  - Bag-of-Words: Word occurrences
  - TF-IDF: Term frequency-inverse document frequency
  - Word embedding

The Set-of-Words Model

- Text represented as a set of words it contains

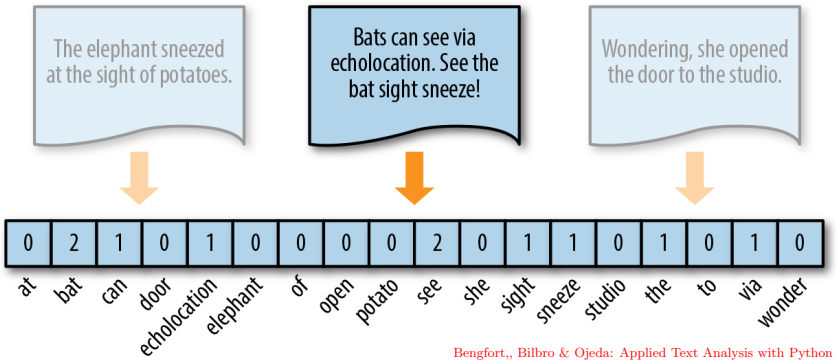


Set-of-Words: Documents represented by vectors $\in \{0,1\}^{|\Sigma|}$

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	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	
...							

The Bag of Words Model

- Text represented as a a bag (multiset) of words it contains



Bag-of-Words: Documents represented by term-frequency vectors $\in \mathbb{N}^{|\Sigma|}$

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	157	73	0	0	0	1	
BRUTUS	4	157	0	2	0	0	
CAESAR	232	227	0	2	1	0	
CALPURNIA	0	10	0	0	0	0	
CLEOPATRA	57	0	0	0	0	0	
MERCY	2	0	3	8	5	8	
WORSER	2	0	1	1	1	5	
...							

TF-IDF - Motivaiton

TF-IDF is a more refined model, to select features to represent texts

- Key idea is to find special words characterizing the document
- Reflect how significant a word is to a "document" in a "collection"
- **Frequency:** Most frequent words implies most significant in doc
- Actually exactly the opposite is true
- Most frequent words ("the", "are", "and") help English structure and build ideas but not significant in characterizing documents
- **Rarity:** Indicator of topics are rare words
- rare words overall but concentrated in a few docs "batsman", "prime-minister"
- ball, bat, pitch, catch, run $\Longrightarrow$ cricket related doc
- An indicator word is likely to be repeated if it appears once

TF-IDF

- TF-IDF value increases proportionally to the number of times a word appears in a document
- Offset by the number of documents in corpus containing that word
- Best known weighting scheme in IR. Value for a term increases with
  - Number of occurrences within a document
  - Rarity of the term in collection
- Helps to adjust for the fact that some words appear more frequently in general (frequent words are less meaningful than the rare ones)
- Involve two characteristics of words (terms: bigram, trigram)
  - Term frequency
  - Inverse document frequency

TF-IDF: Term Frequency

Documents: $D_1, \ldots D_N$. Terms (Σ): $t_1, \ldots, t_m$

- **Frequency, f_{ij}:** frequency of term t_i in document D_j
- Find a parameter to measure importance of t_i to D_j
- f_{ij} is not good, (very high for stop words in all documents)
- It is also possible that large docs D_j (books) have larger f_{ij}, than $f_{ij'}$ of short document $D_{j'}$ even if t_i is more important for $D_{j'}$ than D_j
- Recall normalization and scaling
- **Term Frequency:** $\text{TF}_{ij} := \frac{f_{ij}}{max_i f_{ij}}$
- Most frequent term t_i in D_j gets $\text{TF}_{ij} = 1$ others are < 1

TF-IDF: Inverse Document Frequency

Documents: $D_1, \ldots D_N$. Terms (Σ): $t_1, \ldots, t_m$

- Term frequency considers all t_i equally important
- Stop words appear frequently but have little importance
- Need to weigh down the frequent terms while scale up the rare ones
- Some terms are rare but appear in many documents a few times
- Weigh TF_{ij} (inversely) by the term's overall popularity in collection
- Suppose the term t_i appears in n_i out of N documents. Then
- **Inverse Document Frequency:** $\text{IDF}_i := \log\left(\frac{N}{n_i + 1}\right)$
- $+1$ in denominator avoids dividing by 0 if t_i doesn't appear in any doc

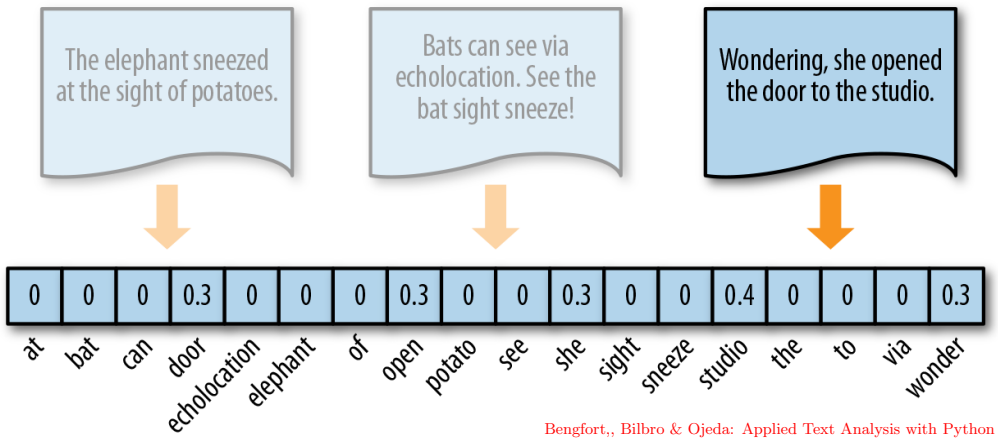
TF-IDF: Term frequency-inverse document frequency

Documents: $D_1, \ldots D_N$. Terms (Σ): $t_1, \ldots, t_m$

- Finally, weight or importance of a term t_i in document D_j is given as

$$\text{TF-IDF}(i,j) = \text{TF}_{ij} \times \text{IDF}_i$$

- Check the extreme cases
- If t_i appears in all the documents, then $\text{TF-IDF}(i,j) = 0$ in all D_j
- Many stop words would get score close to 0
- A term frequently appearing in some docs gets higher score there



Bengfort,, Bilbro & Ojeda: Applied Text Analysis with Python

TF-IDF: Example

- $D_1 = A$: "The car is driven on the road"
- $D_2 = B$: "The truck is driven on the highway"

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

- Common words score is zero (not significant)
- Score of "car", "truck", "road", and "highway" are non-zero (significant words)

The TF-IDF Model

Each document is represented by a real vector of TF-IDF weights $\in \mathbb{R}^{|\Sigma|}$

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35	
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0	
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0	
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0	
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0	
MERCY	1.51	0.0	1.90	0.12	5.25	0.88	
WORSER	1.37	0.0	0.11	4.15	0.25	1.95	
...							

Issues with Vector Space Models

- Dimensionality blows up, $|\Sigma|$ could be large $\implies$ high computational complexity
- (SoW) treats mere appearance of words as feature of document (Word appearing 1000 times versus one appearing once only)
- They do not preserve words' order that carries contextual information
- The disregard grammar and semantics of the text
- Following two documents produce identical vectors (in all 3 models), although the context and meaning is very different
  - *Mary is faster than John*
  - *John is faster than Mary*
- They ignore synonyms ("old bike" vs "used bike") and homonyms
- *n*-gram model of vocabulary takes care of context to some extent

Solution: Word embedding

Text Analytics: Text Normalization

- Initial Pre-procesing of text dataset
- The goal is to standardize sentence structure and vocabulary
- Helps reduce number of variables (dimensionality)

Exact preprocessing steps depends on application, they include

- Removing duplicate whitespaces, punctuations, accents, capital letters and, special characters
- Substituting word numerals by numbers (thirty $\rightarrow$ 30), values by type (\$100 $\rightarrow$ currency/money), contractions by phrases (I've $\rightarrow$ I have)
- Standardizing formats (e.g. dates), replacing abbreviations
- Stopwords removal
- Stemming
- Lemmatization

Text Analytics: Stop words

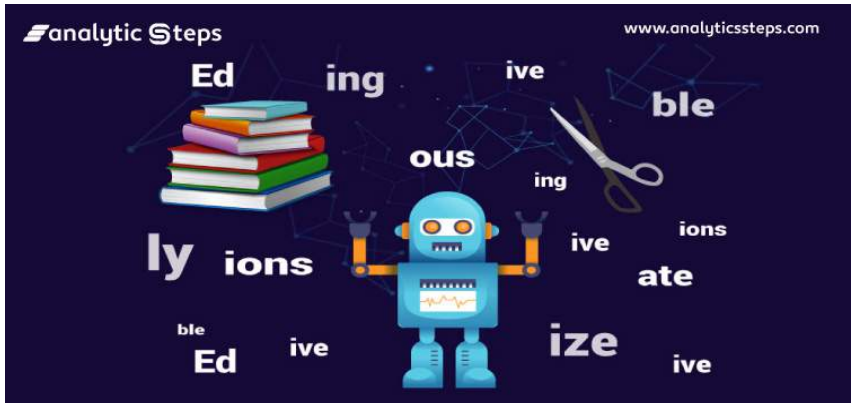
- Common words not providing useful information: the, it, is, are, an, a
- Often removed (filtered out) during preprocessing
- No universally good list of stop words
- Reduces time/space complexity, can improve analytics quality

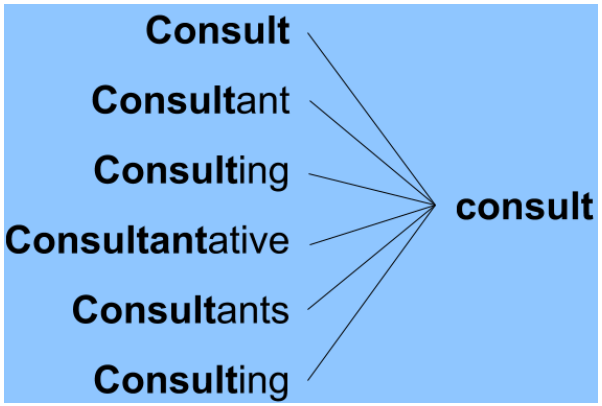
Sr.No	Tokenized sentences	SWR sentences
1	I have been on this medicine for 2 years	Medicine 2 years.
2	It has no side effects except for gaining of weight.	Side effects except gaining weight.
3	It also helps me sleep at night.	Also helps sleep night.
4	I was extremely suicidal and depressed	Extremely suicidal depressed.
5	completely did the opposite effect of what it is meant for	Completely opposite effect meant.

M Qasim (2018) Mining health reviews from online blogs and news

Text Analytics: Stemming and Lemmatization

Convert different variations of a word to a common root form





- **Stemming**: crude heuristic way of chopping off ends of words
- **Lemmatization**: grammatically sound words replacing
- am, are, is $\longrightarrow$ be
- car, cars, car's, cars' $\longrightarrow$ car
- "the boy's cars are different colors" $\longrightarrow$ "the boy car be differ color"

Term-Document Incidence Matrix

Documents are represented as set of words/terms

Term-document incidence matrix, M: A binary matrix indicating the presence (1) or absence (0) of terms in documents

A row for each term and a column for each document

$$M(t,d) = \begin{cases} 1 & \text{if doc } d \text{ contains term } t \\ 0 & \text{otherwise} \end{cases}$$

	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
...						

Term-Document Incidence Matrix

Retrieve Shakespeare's novel(s) that have words

Brutus AND Caesar AND NOT Calpurnia

- Bruteforce: linear scan all docs for each query term
- Bitwise AND of vectors (rows) for Brutus, Caesar and NOT Calpurnia

	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	1	0	1	1	1	1
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0
AND	1	0	0	1	0	0

- Returns two documents, "Antony and Cleopatra" and "Hamlet"

Term-Document Incidence Matrix: Limitations

- Consider $N = 10^6$ documents, each with about 1000 terms (tokens)
- 10^9 tokens at avg 6 bytes per token $\rightarrow$ $6GB$
- Assume there are $|\Sigma| = 500,000$ distinct terms in the collection
- Size of incidence matrix is then $500,000 \times 10^6$ (half a trillion bits)
- Generally, the term-document matrix is very sparse (contains no more than a billion 1's)

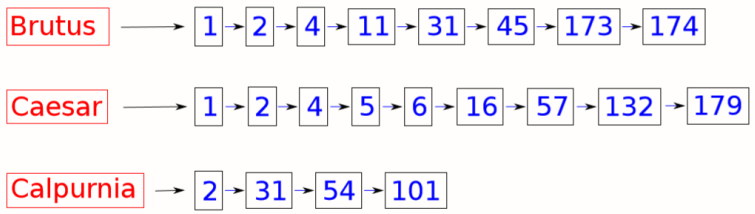
The Inverted Index

A key data structure for efficient document retrieval

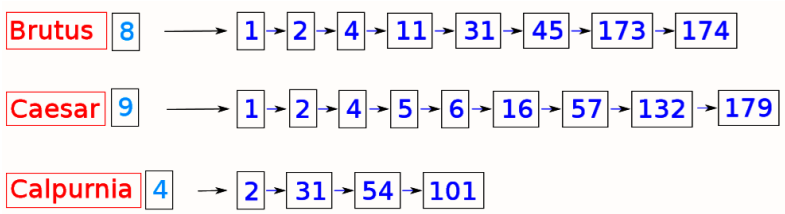
- Essential for efficient document retrieval, especially in large datasets

Σ : set of all terms/tokens (also called lexicon, vocabulary)

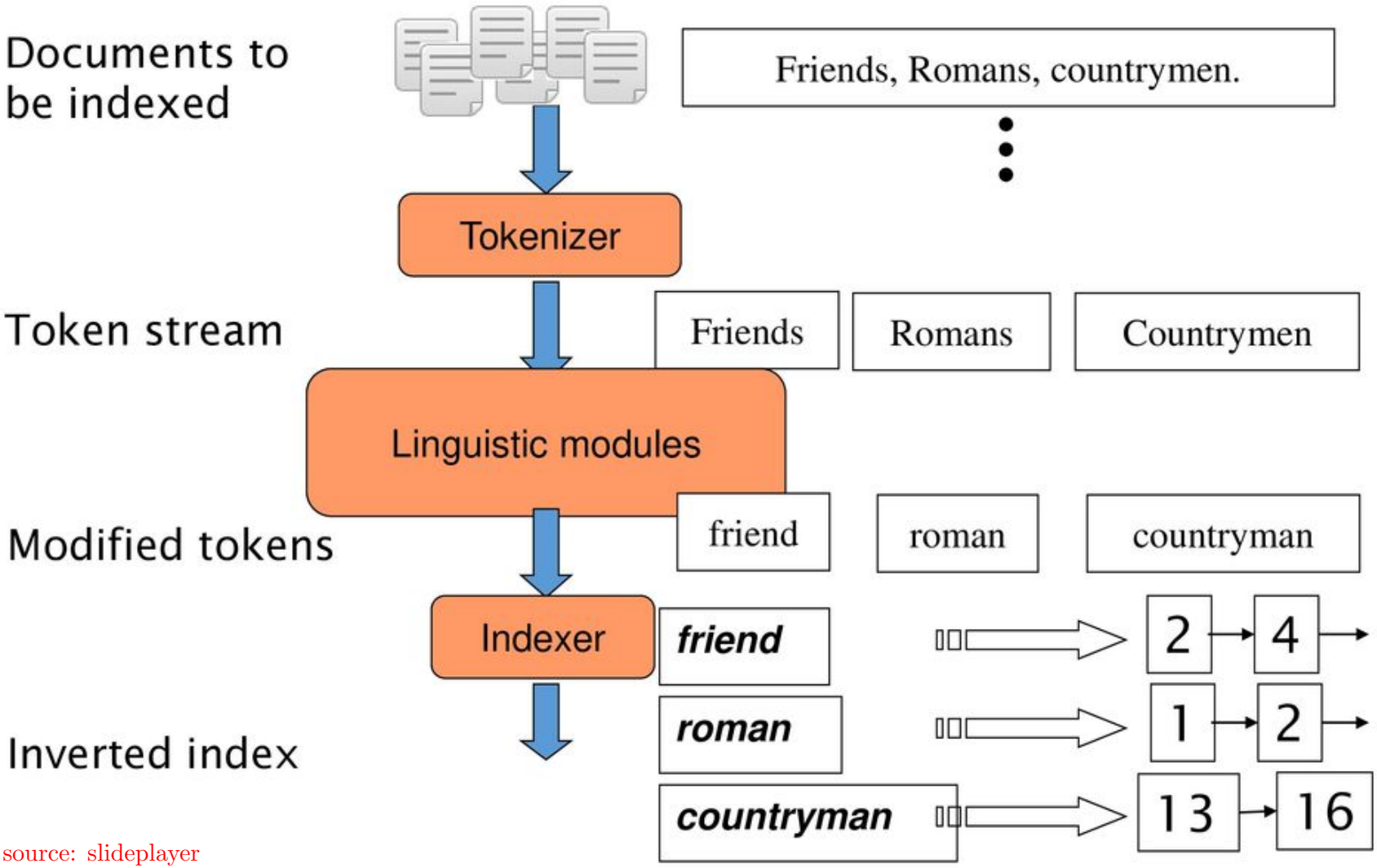
- A postings list for each term in Σ - list of docs where the term occurs



- Sort each posting and the list of posting for optimal processing
- Save length of lists at header



The Inverted Index - Steps



source: slideplayer

The Inverted Index - Token sequence

- Sequence of (Modified token, Document ID) pairs

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

The Inverted Index - Sort

- Sort by terms
- then by doc-ID

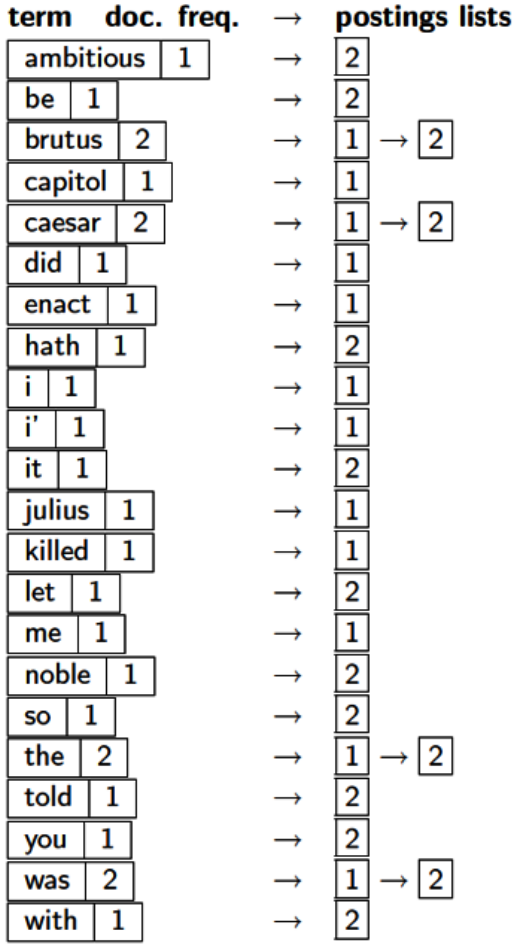
Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

The Inverted Index - Dictionary And Postings

- List document entries for each term
- Split into Dictionary and Postings
- Add number of docs information

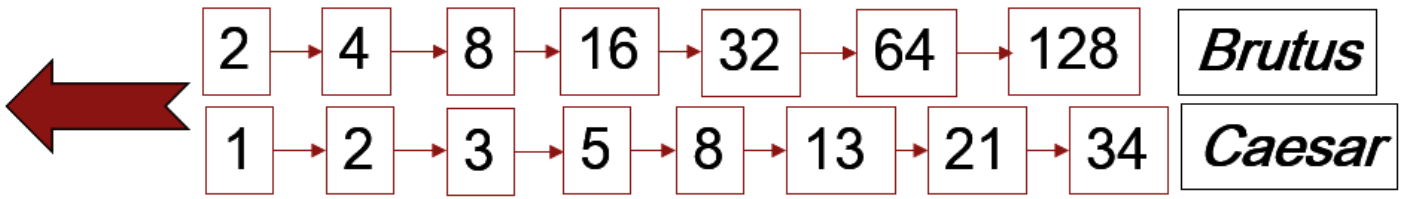
Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



The Inverted Index - Query processing

Query: Brutus AND Caesar

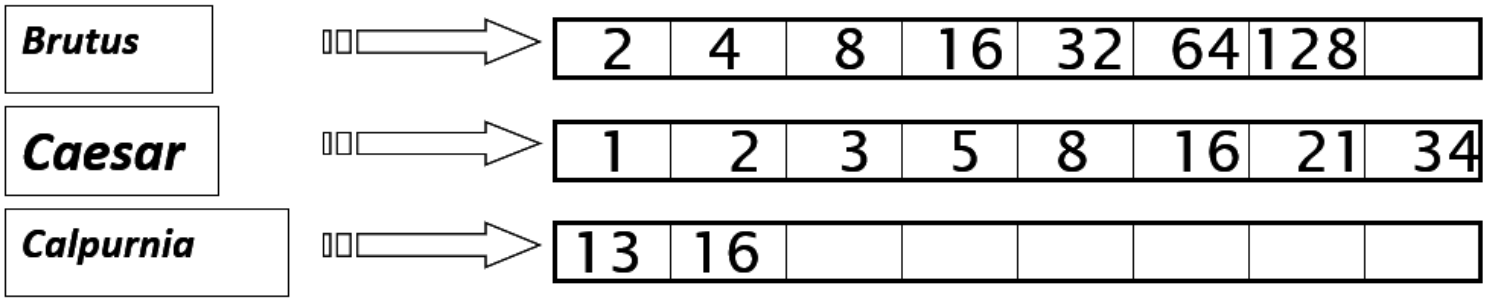
- Locate Brutus in the dictionary and retrieve its postings
- Locate Caesar in the dictionary and retrieve its postings
- "Merge" the two postings (to get intersection)
  - Walk through the two postings simultaneously, in time linear in the total number of postings entries
  - Let the list lengths be x and y, merge takes $O(x + y)$ operations (since postings are sorted by docID)



The Inverted Index - Query Optimization

Query: Brutus AND Calpurnia AND Caesar

- What is the best sequence of binary intersections to get $X \cap Y \cap Z$
- Process in order of increasing posting lengths
  - start with smallest pair of sets, then keep cutting further
  - this is why we kept document lengths in dictionary



Execute the query as (Calpurnia AND Brutus) AND Caesar

The Inverted Index - More general merges

Query: Brutus AND NOT Caesar

- Complement posting would be very long
- Can we still compute intersection in $O(x + y)$?

Query: (Brutus OR Caesar) AND NOT (Antony OR Cleopatra)

- Can we evaluate any set algebraic expression in "linear" time?

Inverted Index: Issues

Query: "Punjab University" as a phrase

- Documents containing "University of Punjab" are not a match
- Biword Indexes: Indexes bigrams in the text as phrases
  - Issues with Biword Indexes: Index blowup due to bigger dictionary
- Positional Indexes: also cater for context
  - In the postings, for each term store the position(s) where it appears
    <term, number of docs containing term;
    doc1: position1, position2 ... ;
    doc2: position1, position2 ... ;
    etc >
  - Issues with Positional Indexes:
    - A positional index is $2-4$ times as large as a non-positional index
    - Positional index size $35-50\%$ of volume of original text

Information Retrieval: Models

Retrieval models dictate how documents are indexed and retrieved

We briefly discuss two main information retrieval models

- Boolean Retrieval
- Ranked Retrieval

The goal is to set the stage for web information retrieval, where there are many other issues

Boolean Retrieval

A model for information retrieval in which documents are retrieved based on the presence or absence of terms specified in a Boolean query

Boolean Queries: Use logical operators such as AND, OR, and NOT to combine search terms

1. Documents are represented as set of words/terms
2. Queries are formulated as Boolean expressions of terms - combines sentences/documents with operators AND, OR, and NOT
3. A document matches a query if it satisfies the Boolean expression

Main techniques used in Boolean retrieval are

- Term Document Incidence Matrix
- Inverted Index

Boolean Retrieval: Advantages and Disadvantages

- Simplicity: Easy to understand and implement
- Precision: High precision in retrieving documents
  - ▷ document matches a condition or not
- It was the primary commercial retrieval tool for 3 decades
- Many search systems still in use are Boolean
  - Email, library catalog, Mac OS X Spotlight
- Complex Queries: Difficult to manage with large queries
  - Exact matching may retrieve too few or too many documents
    - ▷ AND gives too few, OR gives too many
  - The burden is on the user to formulate a good Boolean query
- Rigidity: Does not handle partial matches or relevance ranking
  - All terms are equally weighted
  - Output is not ordered in a useful fashion

Ranked Retrieval

Unlike Boolean model, ranked retrieval orders documents by relevance to the query, rather than a binary match

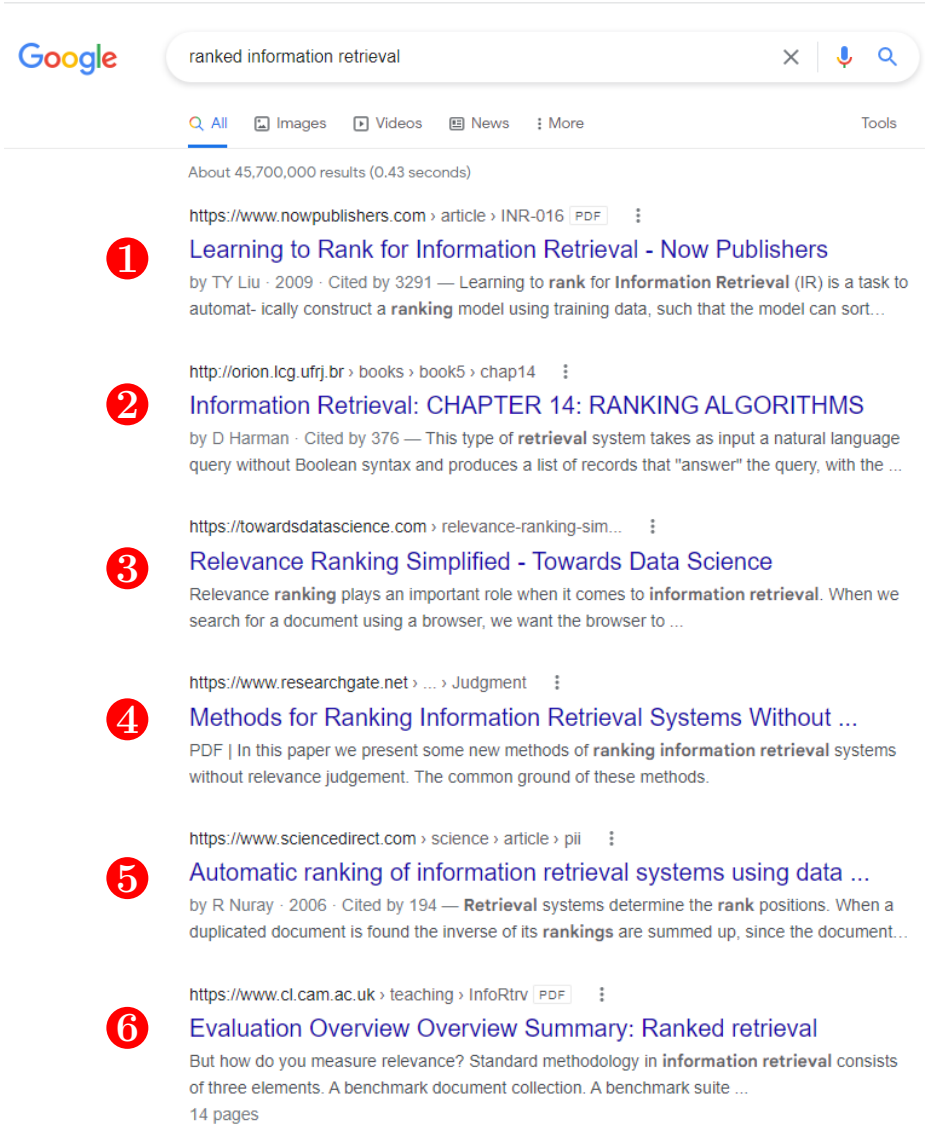
▷ Uses a scoring mechanism, typically based on the TF-IDF weighting scheme

- Returns the top k documents by relevance
- The size of the result is not an issue ($k \sim 10, 20, 100$)

Advantages:

- Allow for free text/natural language queries: Rather than considering query language of operators and expressions, it consider words of human language
- Allows for partial matching and results ranking, improved user experience

Ranked Retrieval



Scoring as Basis of Ranked Retrieval

How to rank-order documents in a collection with respect to a query?

- Assign a relevance score - say in [0,1] - to each document
- e.g query with one term:
  - If the term does not occur in the document, score is 0
  - The more frequent the query term in document; the higher the score

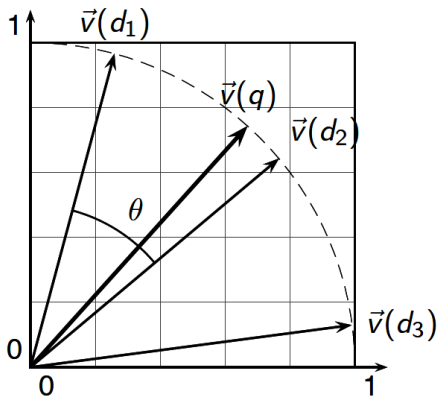
Different measures to calculate relevance/similarity in ranked retrieval

- Exact Match Models
  - Ranked Boolean retrieval model (based on set/bag of words model)
- Best Match Model (Vector Space Model)
  - $sim(q, d)$ (similarity/relevance between query and document)
  - VSM represent both docs and queries by vectors (e.g., TF-IDF)
  - Similarity is evaluated in the vector-space e.g. cosine similarity

Vector Space Model Ranking

- Represent the query by the TF-IDF vector
- Represent each document by the TF-IDF vector
- Compute cosine similarity between the query and document vectors
- Rank documents with respect to the query by cosine similarity
- Return top k to the user

$$cos(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q}.\mathbf{d}}{|\mathbf{q}||\mathbf{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2 \sum_{i=1}^{|V|} d_i^2}}$$



- q_i is the TF-IDF weight of term i in the the query
- d_i is the TF-IDF weight of the term i in the document

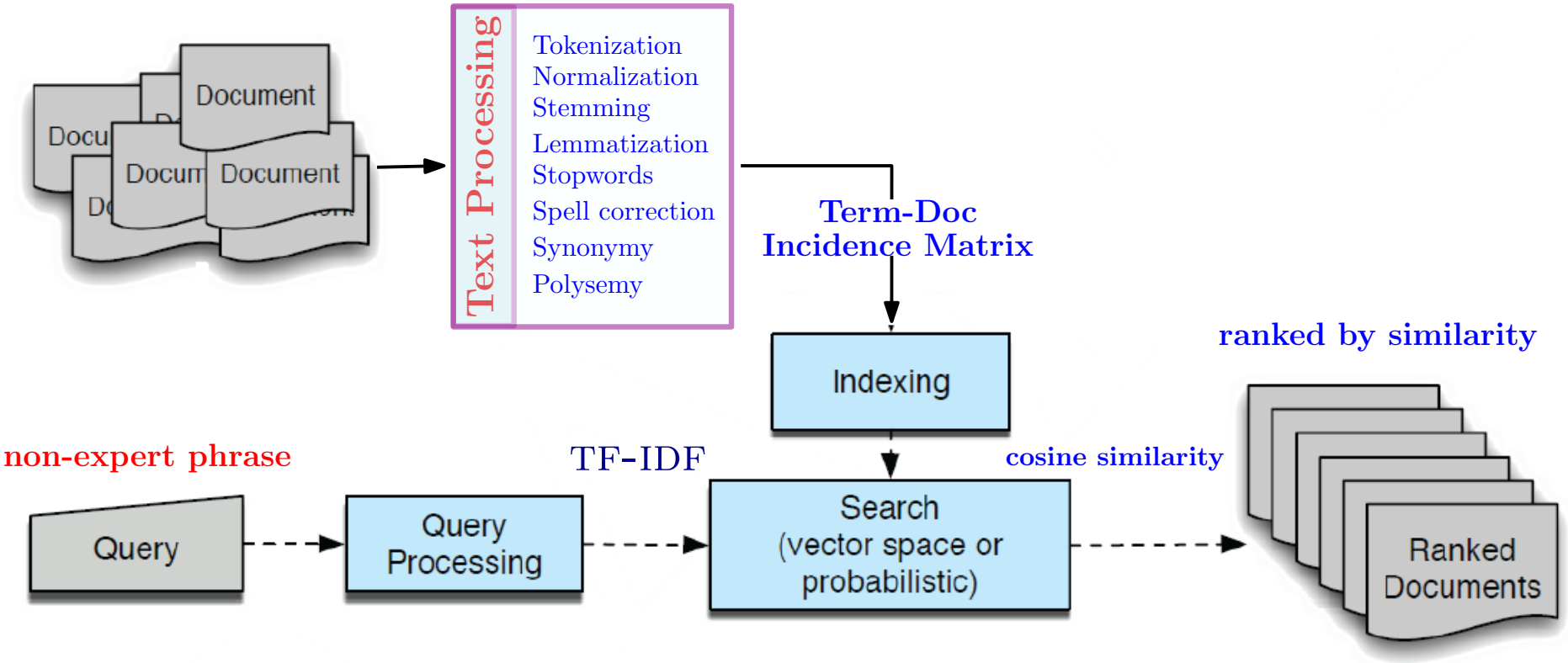
Vector Space Model -Example

- **Query**: "best car insurance"
- **Document**: "car insurance auto insurance"

word	query					document				product
	tf-raw	tf-wght	df	idf	weight	tf-raw	tf-wght	weight	n'lized	
auto	0	0	5000	2.3	0	1	1	1	0.52	0
best	1	1	50000	1.3	1.3	0	0	0	0	0
car	1	1	10000	2.0	2.0	1	1	1	0.52	1.04
insurance	1	1	1000	3.0	3.0	2	1.3	1.3	0.68	2.04

Key to columns: tf-raw: raw (unweighted) term frequency, tf-wght: logarithmically weighted term frequency, df: document frequency, idf: inverse document frequency, weight: the final weight of the term in the query or document, n'lized: document weights after cosine normalization, product: the product of final query weight and final document weight

Information Retrieval



Query Processing

- Query Parsing: Breaking down a query into its constituent parts
- Query Expansion: Enriching the query to include additional terms based on user intent and context
- Query Optimization: Refining the query process to enhance performance and relevance of results

- User Intent Analysis: Understanding what the user really wants from the query
- Contextual Query Expansion: Automatically adding terms to the query based on contextual understanding of the user's needs
- Feedback Mechanisms: Utilizing user feedback to continuously refine and improve the query process

Handling Complex Queries with Multiple Terms

- Logical Relationships: Understanding and interpreting the logical operators (AND, OR, NOT) between terms
- Phrase Queries: Techniques to ensure phrases are treated as single units rather than separate terms
- Proximity Queries: Implementing proximity operators to handle queries where the order and distance between terms matter

Enhancing Semantic Understanding: using thesauri and ontologies, to disambiguate query terms and improving the accuracy of retrieval

- Thesauri: provide lists of related terms to expand or refine queries
- Ontologies: to understand relationships between terms in a domain

Web Search and Information Retrieval

Web Search and Information Retrieval (IR)

Information Retrieval: Finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

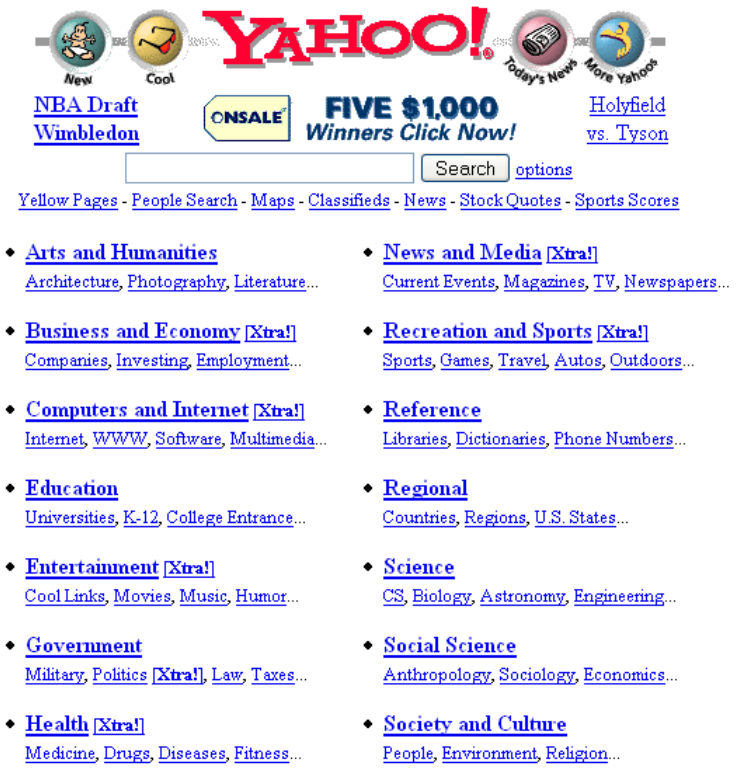
Web Search: Retrieve relevant documents from a vast collection of web pages using query keywords, and returns a ranked list of results

Applications

- Information Retrieval from the Web: Facilitates access to billions of web pages
- E-commerce: Product search and digital marketing
- Navigation: Services through geospatial search (e.g., "restaurants near me")
- Research: Academic and scientific literature
- Social Media: Content discovery and trends

Web Search: History

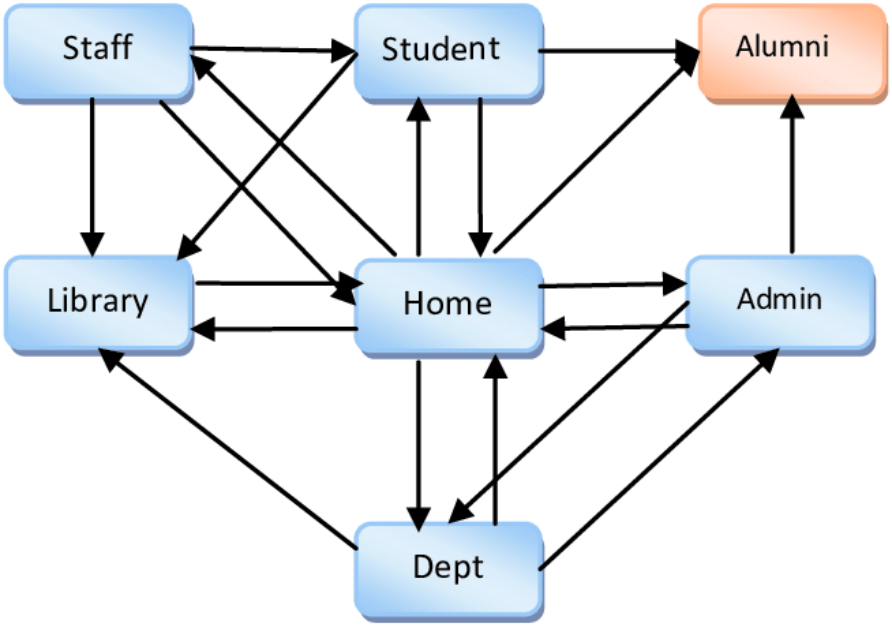
Initial Solution: Information organized in a directory structure for browsing



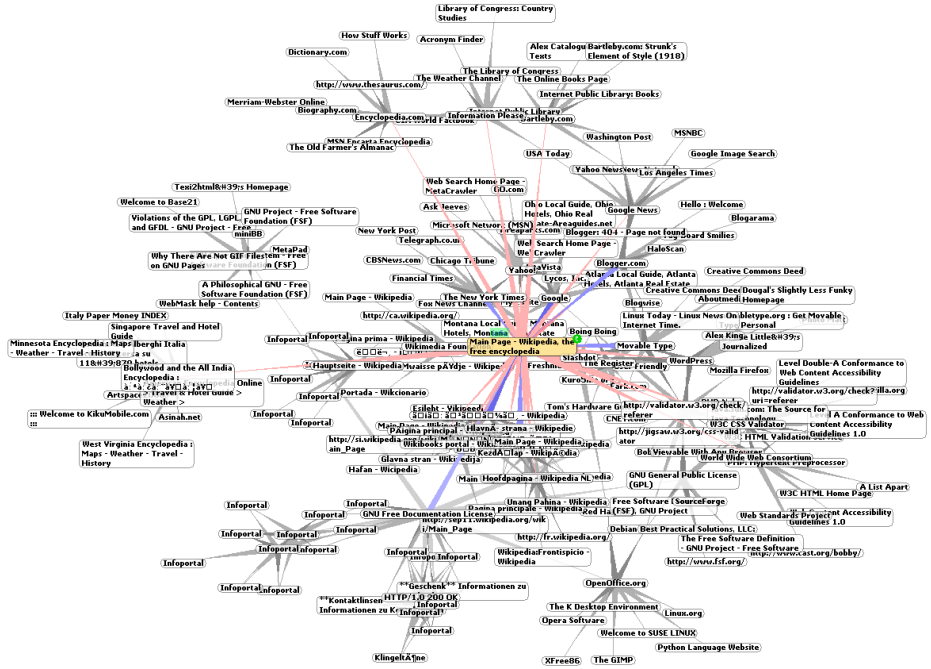
- Human edited - Yahoo listed webpages as standard and paid versions
  - Initially called Jerry and David's Guide to the World Wide Web
- DMOZ (directory.mozilla.org) maintained by volunteers (Open Directory Project)

The Web Graph

- A directed graph with webpages as vertices $v_1, v_2, \ldots$
- Page i has a hyperlink to page j, *implies* $(v_i, v_j) \in E$

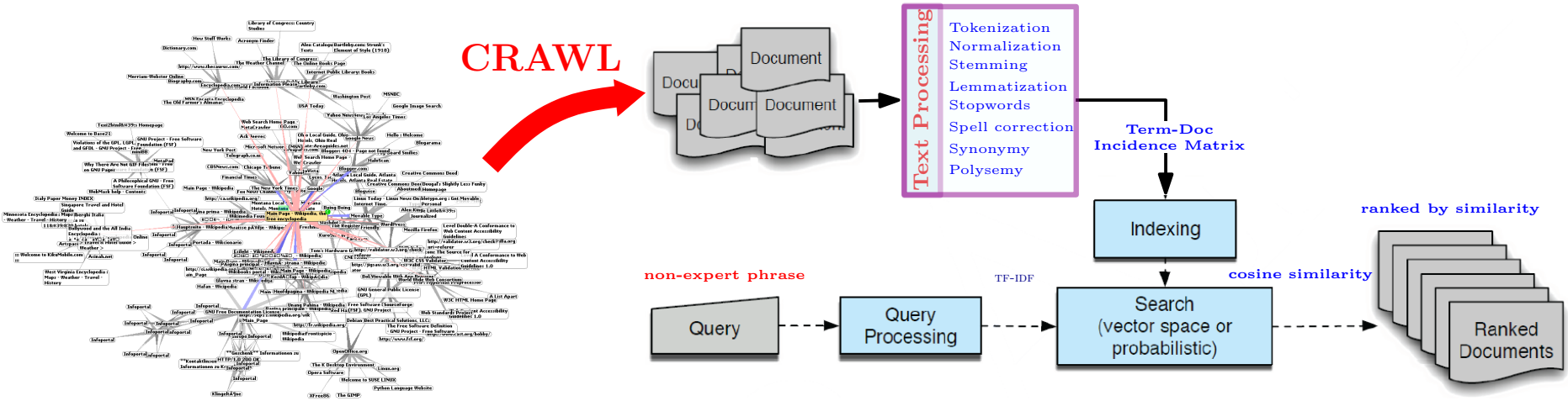


A sample Web Graph W of a University. source: A Singh (2013)



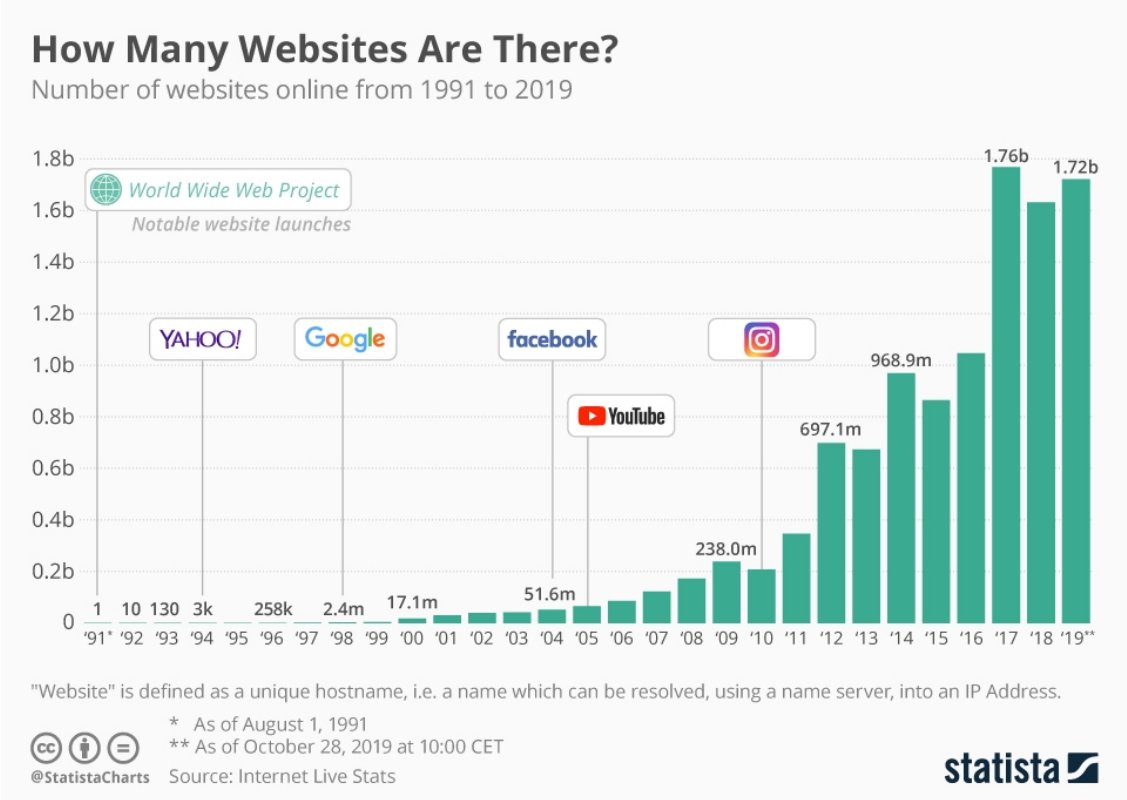
The webgraph is CRAWLED to get the collection of webpages

- Web crawler (aka spiderbot): Internet bot to systematically traverse the web graph (think BFS/DFS starting from a few seeds)
- Collects webspages for (web) indexing (aka web spidering)
- Crawling is also used for web archiving



Web Search Challenges: Size of Web

- 1.83 billion websites, $\sim$ 1.58 billion inactive ▷ internetlivestats, 2021
- Google indexed $\sim$ 55.2 billion webpages in Jan 2021 ▷ worldwidewebsize
- Google Search index is well over 100,000,000 gigabytes

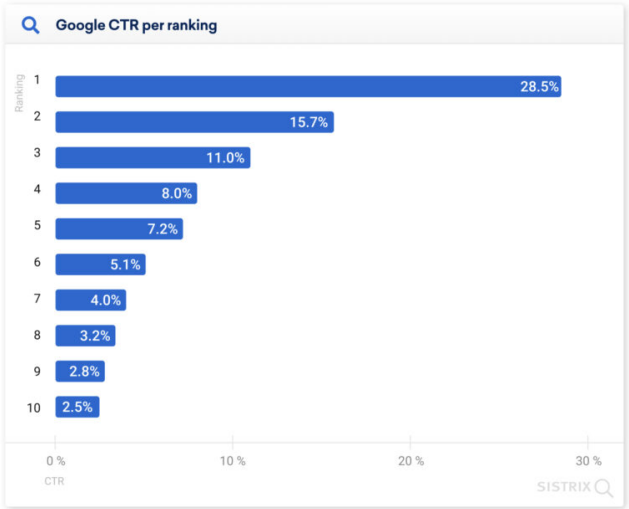


Web Search Challenges: Manipulation

- Retrieve web pages from inverted index
- Rank by cosine similarity b/w TF-IDF vectors of page and query phrase

Easy to manipulate to attract traffic to a web address with financial incentives

- Daily 3.5 billion Google searches – $\sim$ 10% growth p.a ▷ internetlivestats, 2019
- 35% of product searches start on Google ▷ eMarketer
- 46% of product searches begin with Google ▷ Jumpshot, 2018
- 90% of a survey respondents: likely to click on the first set of results ▷ searchengineland, 2018



Web Search Challenges: Dynamic and low quality content

Data Freshness: Constant updates to websites (news, blogs, etc.) require search engines to continuously re-index and keep results up to date

Duplicate Content: Many pages on the web are duplicates, which search engines must identify and filter to avoid redundant results

Evaluation of Search Engines

Evaluation of Search Engines

Assess how well it retrieves and ranks documents in response to user queries

1. Quantitative evaluation using formal metrics
2. Qualitative evaluation based on user feedback

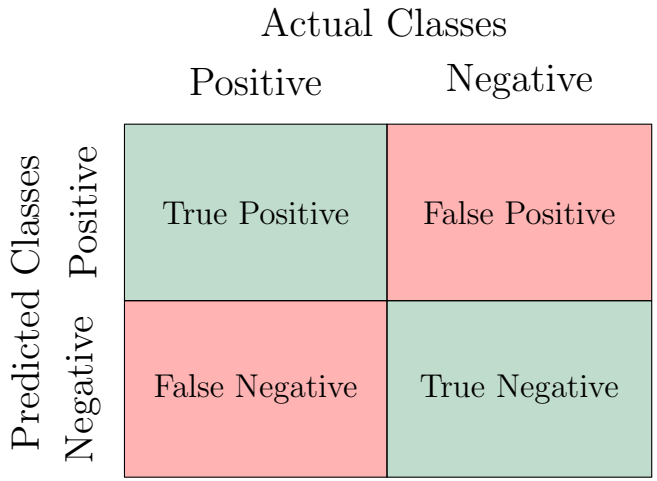
Effectiveness: How well the search engine meets the information needs of users

Efficiency: How quickly and resource-effectively the search engine can retrieve results

Evaluation typically involves comparing the results returned by the search engine against a set of predefined relevant documents (the ground truth)

Metrics for Quantitative Evaluation

Binary Decisions (e.g., classification into two classes) are evaluated by tabulating the classification results in a Confusion Matrix



Some summary statistics of the confusion matrix are

$$\text{ACCURACY} = \frac{\text{TP} + \text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \qquad \text{ERROR} = \frac{\text{FP} + \text{FN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$$

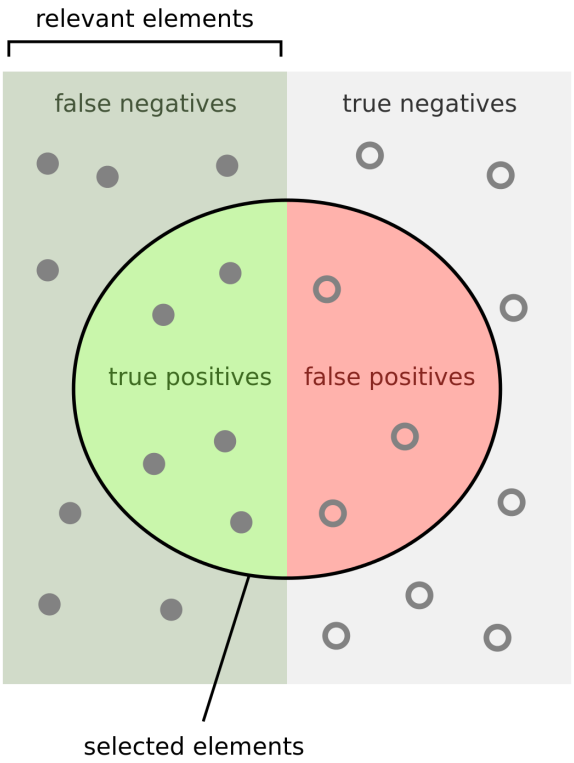
ACCURACY and ERROR are usually reported as percentages

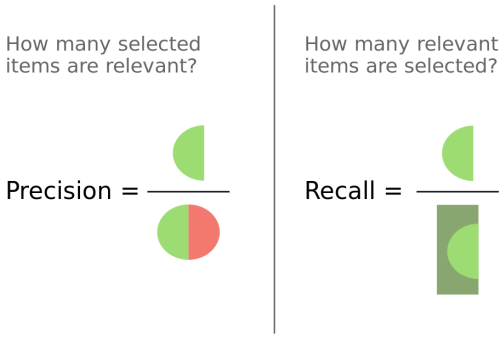
Metrics for Quantitative Evaluation

		Actual Classes: Positive	Actual Classes: Negative
Predicted Classes	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

- With big imbalance in classes, ACCURACY and ERROR are misleading
  - In a tumors dataset 99% samples are negative
  - (Blindly) predicting all as negatives gives 99% accuracy
  - But cancer is not detected
- Have to use cost matrix/loss function (essentially weighted accuracy)

Metrics for Quantitative Evaluation





$$\text{PRECISION} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

▷ sensitivity (measure of exactness)

$$\text{RECALL} = \frac{\text{TP}}{\text{TP}+\text{FN}}$$

▷ specificity (measure of completeness)

- F-measure: Maximizes both
- $F_1 = \dfrac{2}{\frac{1}{\text{PRECISION}} + \frac{1}{\text{RECALL}}}$
- F_k measure weighs PRECISION and RECALL differently

▷ Weighted harmonic mean

Qualitative Evaluation: A/B Testing

Focus on user satisfaction, ease of use, and relevance of results

A/B Testing: Comparing two versions of a search engine (A and B) to determine which one performs better

1. Randomly split users into two groups
   - Group A: Uses the current version of the system (version A)
   - Group B: Uses a modified version with changes to the ranking algorithm, layout, or new features
2. Show version A to Group A and version B to Group B
3. Measure performance metrics (click-through rate, user engagement, time spent on results)
4. Analyze results to determine the better version

Qualitative Evaluation: User Satisfaction Metrics

Collect users feedback on how well the search engine meets their needs

User Satisfaction is evaluated through both explicit and implicit feedback

Explicit Feedback: Users rate search results, provide comments, or answer surveys about the quality of results

Implicit Feedback: Behavioral metrics

- Click-Through Rate (CTR): Ratio of users who click on a specific link to the number of total users who view a page.

$$\text{CTR} = \frac{\text{Number of Clicks}}{\text{Number of Impressions}}$$

- Dwell Time: Time a user spends on a page after clicking a link
- Bounce Rate: Percentage of visitors who navigate away from the site after viewing only one page

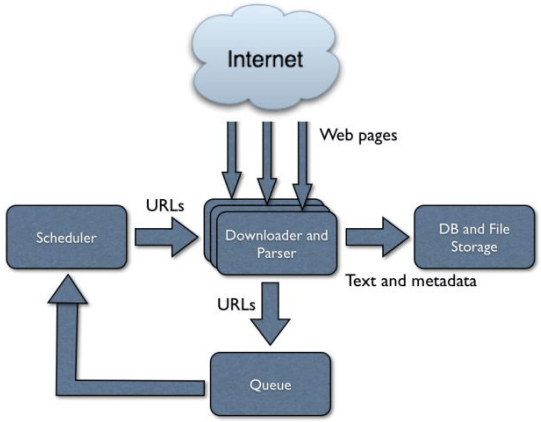
Web Crawling: Strategies

- Breadth-First Search (BFS): Explores all neighbors at the present depth before moving on to nodes at the next depth level
- Depth-First Search (DFS): Explores as far as possible along each branch before backtracking
- Focused Crawling: Targets specific topics or types of content
  - Content-based Filtering: Analyzing the content of web pages
  - Link-based Filtering: Using the structure of hyperlinks
- Incremental Crawling: Updates the index by re-crawling only the changed or new content
- Priority Crawling: Uses heuristics to prioritize URLs based on their importance (e.g., PageRank or site traffic)

Crawling must be efficient to cover a large portion of the web and manage frequent content updates

Web Crawling: Architecture

- Seed URLs: Initial set of URLs to start the crawl
- URL Frontier: List of URLs to be crawled
- Downloader: Fetches the web pages
- Parser: Extracts links and relevant data from fetched pages
- Indexer: Stores parsed data for efficient retrieval
- Crawl Depth: The level of recursion from the seed URL (i.e., how many links deep the crawler should go)



Web Crawling: Challenges

- Scalability: Handling the vast size of the web
- Politeness and Ethics
  - Politeness: Respecting the rules set by web servers (robots.txt)
  - Robots.txt: A file that specifies the rules for web crawlers
  - Rate Limiting: Controlling the speed of requests to avoid overloading servers
  - Ethical Considerations: Ensuring privacy and avoiding harmful activities
- Content Quality: Filtering out low-quality or irrelevant content
- Dynamic Content: Dealing with frequently changing web pages

Indexing

Indexing organizes the web crawler gathered content into a structure to enable fast search and retrieval

The most common index structure for search engines is the Inverted Index

Additional indexing techniques:

Forward Index: Maps documents to the terms they contain (reverse of inverted index)

Distributed Indexing: Spreads index data across multiple servers to handle large-scale datasets

Trustworthy Webpages

Web Search Challenges: Spamming

Financial incentive for traffic to a webpage $\Longrightarrow$ tricks to increase relevance of webpages to multiple keywords

- Termed as *search engine optimization* or *search engine spamming*
  - Depending on which side it is coming from
- Two broad categories of tricks (spamdexing) are
  - Content Spamming: (e.g., Keyword Stuffing, Invisible Text)
    - Content manipulation to attract traffic
  - Link Spamming: (e.g., Link Farms, Scraper Sites)
    - Graph Structure (hyperlinks) manipulation

Search engines have developed countermeasures, penalizing sites that engage in spamming through algorithm updates

Web Search Challenges: Content Spamming

The goal here is to increase TF-IDF scores of (many) keywords

- Keyword Stuffing: placing many keywords on webpage
  - Search engines truncate large pages, avoided with multiple pages
- Invisible Text: Background-colored text or hidden within html codes
- Doorway Pages: Pages with targeted keywords redirecting traffic to another page aka Cloaking
- Scrapper Sites: Pages using contents of other pages (e.g. top results against a keyword)
- Article spinning or translation: to avoid penalty for duplicate contents - contain rephrased or machine translated articles
- Deceiving Page Titles: Page titles irrelevant to content (title and header terms carry higher scores)

Trustworthy Webpages

- In addition to relevance/similarity of page with query
- Assign a trustworthiness (popularity/reliability) score to each page
- Page score is based on its "location" in the webgraph ▷ Link Analysis

Node centrality (called prestige in digraphs e.g. Twitter or Web)

- Degree centrality: how many neighbors a node has

$$C_d(v) := deg(v)$$

- Closeness centrality: how "close" a node is to other nodes

$$C_{close}(v) := \frac{1}{\sum_{u \neq v \in V} d_G(v,u)}$$

- Betweenness centrality: how often a node is on the shortest paths

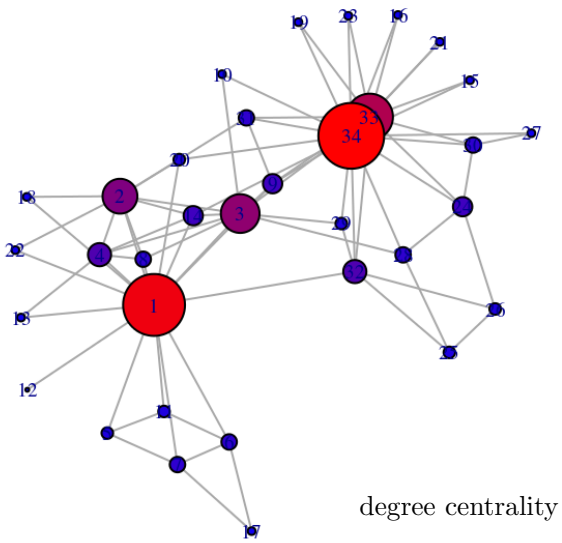
$$C_{bw}(v) := \sum_{s,t \neq v \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

$\sigma_{st}(v)$: number of shortest paths between s and t through v
σ_{st}: number of shortest paths between s and t

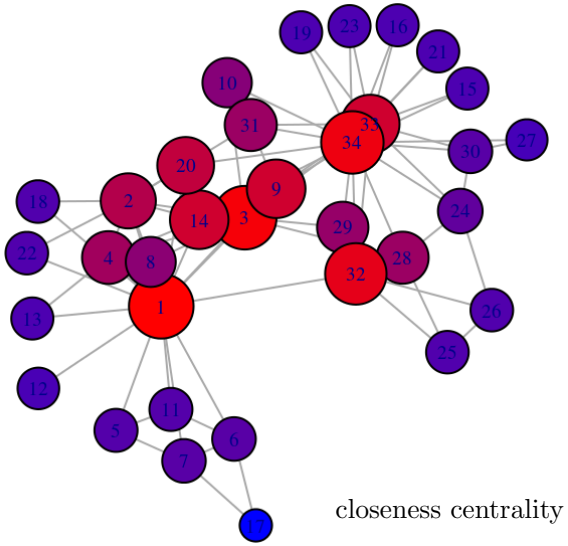
- Eigenvector centrality: Value of eigenvector at corresponding coordinate

$\mathbf{x}[i]$ $\mathbf{x}$: eigen vector corresponding to leading eigenvalue

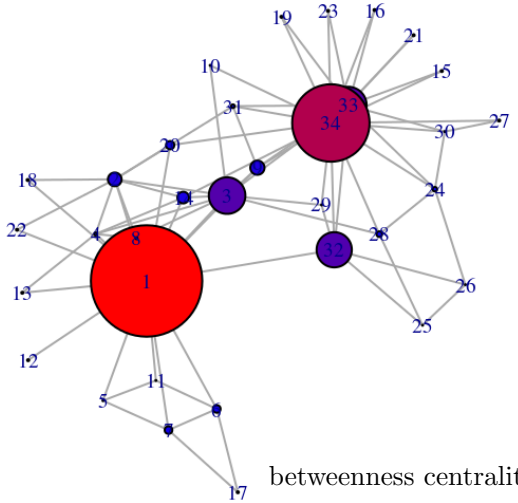
Node Centrality in Graphs



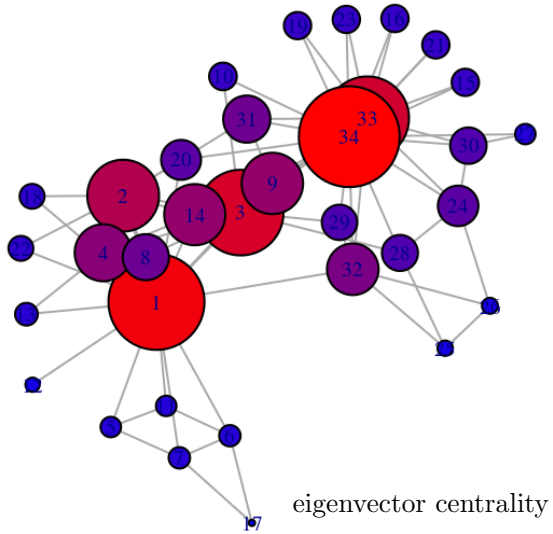
degree centrality



closeness centrality



betweenness centrality



eigenvector centrality

source: D. Petrov, Y. Dodonova, A. Shestakov (2015)

Pagerank

Node Centrality in Graphs

$N^+(p)$ ($N^-(p)$): out(in)-neighbors $d^+(p)$ and $d^-(p)$: out(in)-degree

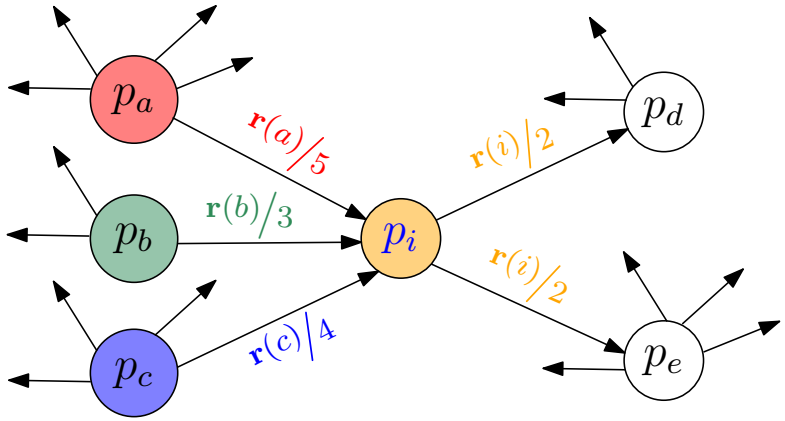
- Rating based on out-degree
  - Very easy to inflate
- Rating based on in-degree
  - Can be manipulated by spam farm many interlinked *'fake'* pages
- Rating based on weighted in-degree
  - Weights would be ratings of in-linking pages
  - Compare rating of p_1 and p_2 with in-link from p_x and p_y, resp.
  - Suppose p_x and p_y are equally rated
  - What if p_x links to 1000 other pages and p_y only links to p_2

Score page p_i based on 'weighted voting' of in-neighbors

For weights consider

- rank of in-neighbors
- importance of incoming links

$$\mathbf{r}(p_i) = \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}(p_j)}{d^+(p_j)}$$



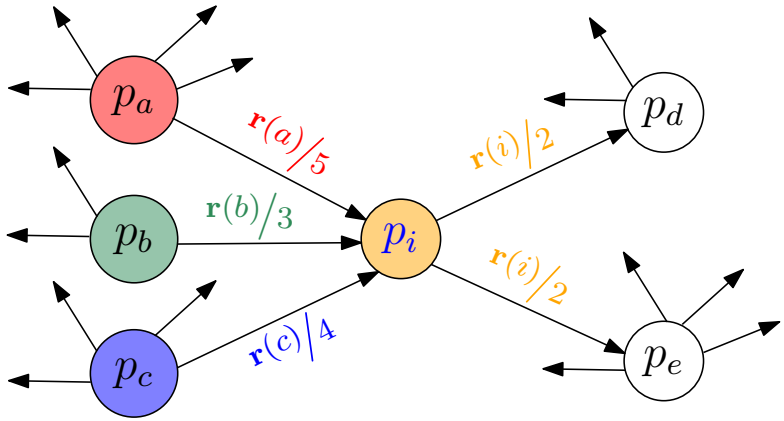
- Each page divides its score equally among its out-neighbors
- Rating of a page is directly proportional to rating of pages linking to it

Pagerank

For weights consider

- rank of in-neighbors
- importance of incoming links

$$\mathbf{r}(p_i) = \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}(p_j)}{d^+(p_j)}$$



- Each page divides its score equally among its out-neighbors
- Rating of a page is directly proportional to rating of pages linking to it
- Recursive Formulation!

$\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ At time $t = 0$ each page has equal rating

$$\mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$$ ▷ repeated improvement at time t

Pagerank Algorithm

Input: Web graph $G = (V, E)$, V is the set of pages and E: hyperlinks

Output: Pagerank score for each page

Algorithm Pagerank Algorithm

1: $\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ $n = |V(\text{WEB})|$

2: **for** $t = 0$ to k **do**

3: $$\mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$$

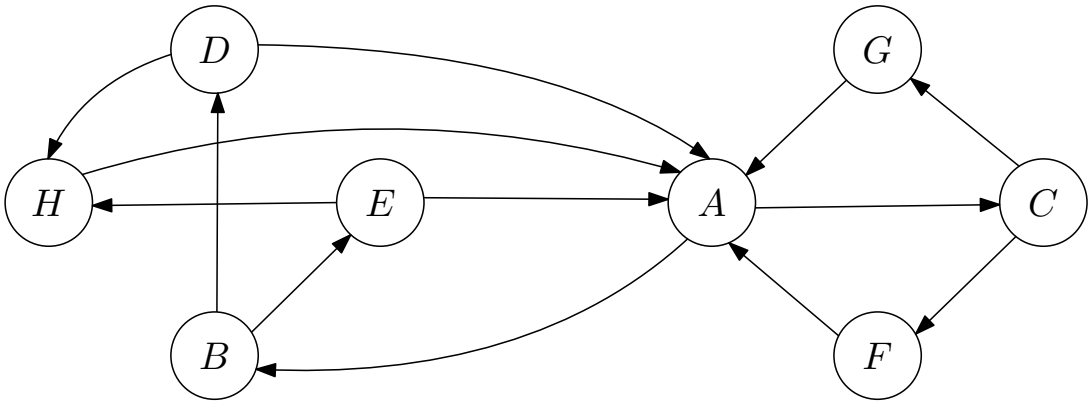
- Uses principle of repeated improvement
- Total pagerank (sum over all pages) remains constant = 1

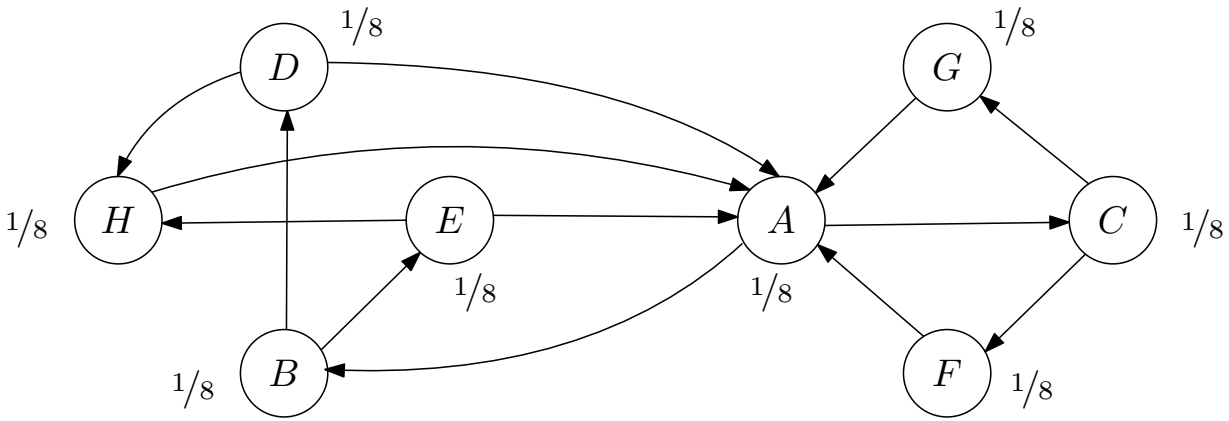
Pagerank Algorithm

$\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ $n = |V(\text{WEB})|$

for $t = 0$ to k **do**

$\quad \mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$





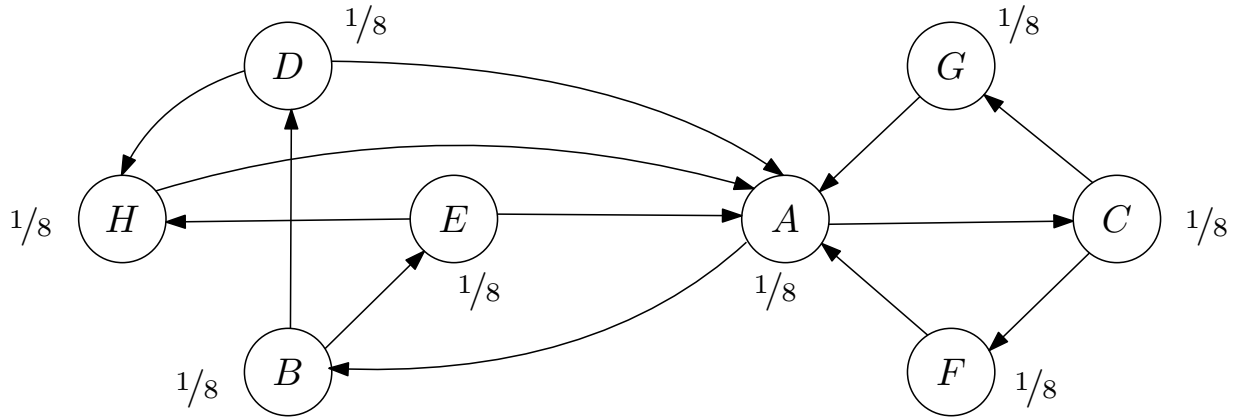
node	$\mathbf{r}(\cdot)$
A	1/8
B	1/8
C	1/8
D	1/8
E	1/8
F	1/8
G	1/8
H	1/8

Pagerank Algorithm

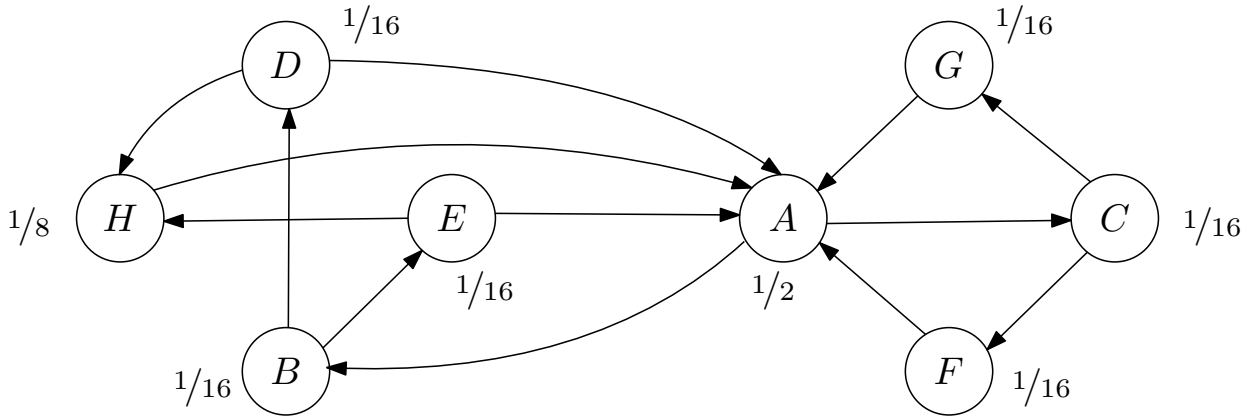
$\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ $n = |V(\text{WEB})|$

for $t = 0$ to k **do**

$\quad \mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$



node	$\mathbf{r}(\cdot)$
A	1/8
B	1/8
C	1/8
D	1/8
E	1/8
F	1/8
G	1/8
H	1/8



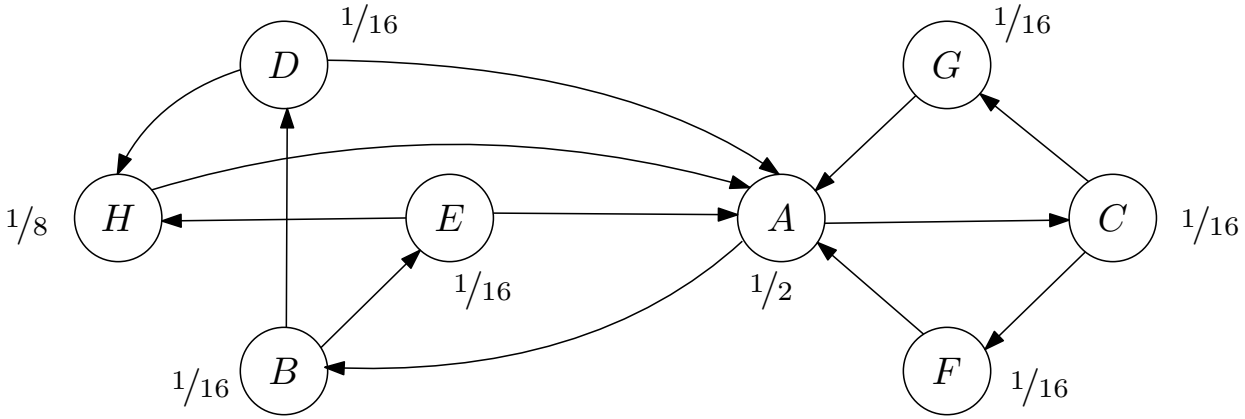
node	$\mathbf{r}(\cdot)$
A	1/2
B	1/16
C	1/16
D	1/16
E	1/16
F	1/16
G	1/16
H	1/8

Pagerank Algorithm

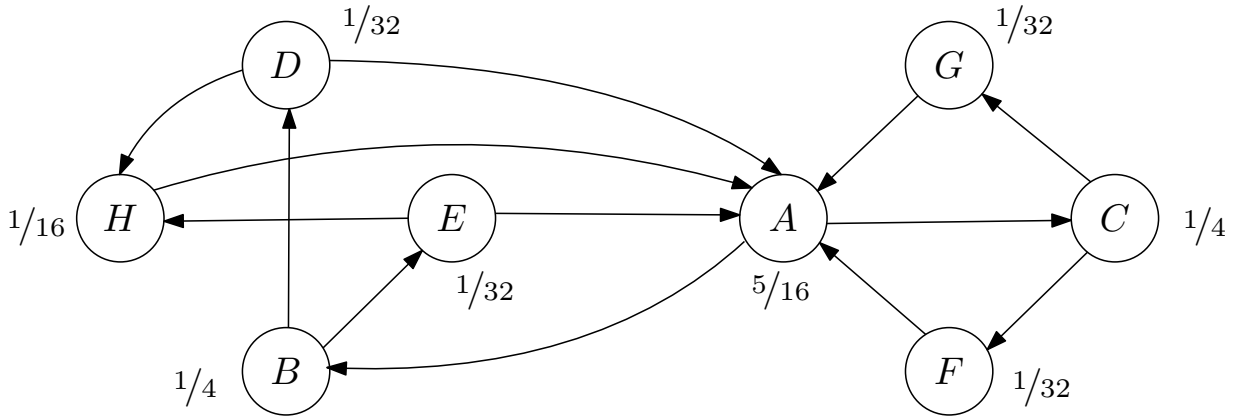
$\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ $n = |V(\text{WEB})|$

for $t = 0$ to k **do**

$\quad \mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$



node	$\mathbf{r}(\cdot)$
A	1/2
B	1/16
C	1/16
D	1/16
E	1/16
F	1/16
G	1/16
H	1/8



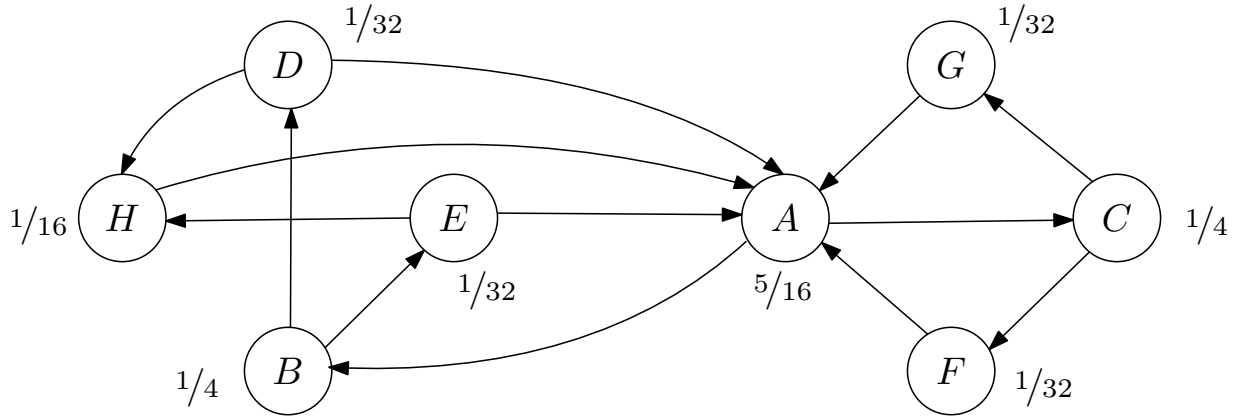
node	$\mathbf{r}(\cdot)$
A	5/16
B	1/4
C	1/4
D	1/32
E	1/32
F	1/32
G	1/32
H	1/16

Pagerank Algorithm

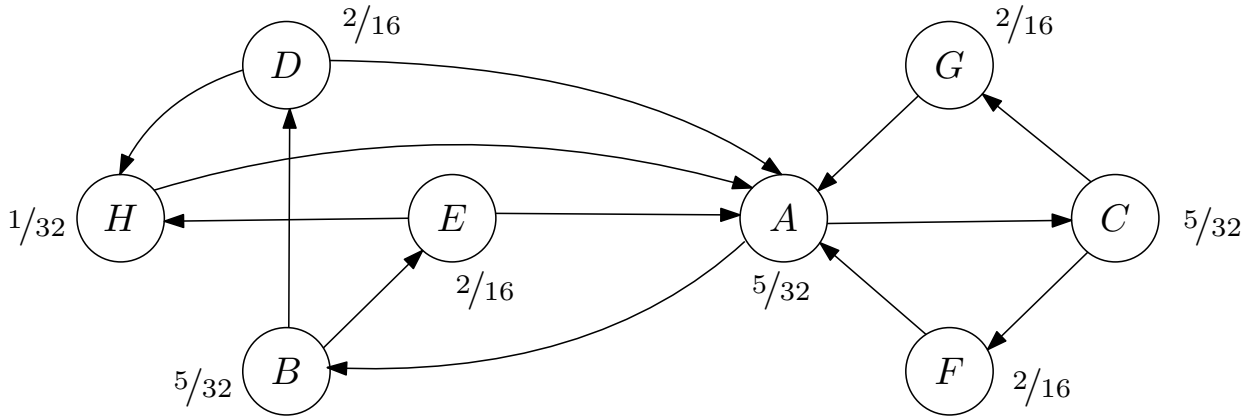
$\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ $n = |V(\text{WEB})|$

for $t = 0$ to k **do**

$\quad \mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$



node	$\mathbf{r}(\cdot)$
A	5/16
B	1/4
C	1/4
D	1/32
E	1/32
F	1/32
G	1/32
H	1/16



node	$\mathbf{r}(\cdot)$
A	5/32
B	5/32
C	5/32
D	2/16
E	2/16
F	2/16
G	2/16
H	1/32

Pagerank Algorithm

Algorithm Pagerank

$\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ $n = |V(\text{WEB})|$

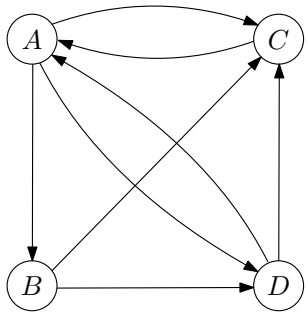
for $t = 0$ to **k** **do** ▷ What is k?

$$\mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$$

- Uses principle of repeated improvement
- No such k, stopping condition is $\|\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)}\| < \epsilon$ for $0 < \epsilon < 1$
- This approach is the power iteration method to compute eigenvector
- If the graph is not a degenerate case, the pagerank values converge to a limiting vector (equilibrium, stationary distribution)
- Total pagerank (sum over all pages) remains constant $= 1$

Random Walk Formulation of Pagerank

- Simulate a random walk (random surfer) across the web digraph
- The surfer chooses an outgoing link at random
- Score of a page is the (long-term) probability of visiting it



Link Matrix, L

Transpose of out-degree normalized adjacency matrix

$$\begin{bmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

- L encodes probabilities of visiting a page from another (transition)

$$\mathbf{r}(p_i) = \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}(p_j)}{d^+(p_j)} \qquad \mathbf{r}(p_i) = \sum_{p_j \in V} L(i,j)\, \mathbf{r}(p_j)$$

- $\mathbf{r} = L\mathbf{r}$ (the eigenvector of L with eigenvalue 1)
- $\mathbf{r} = L\mathbf{r}$ (also the stationary distribution of the Markov chain L)

Issues with Pagerank

Dead-ends, Spider Traps and Link Spamming

Pagerank Algorithm: Issues

Algorithm Pagerank

$\mathbf{r}^{(0)}(p_i) \leftarrow 1/n$ ▷ $n = |V(\text{WEB})|$

while $\|\mathbf{r}^{(t)} - \mathbf{r}^{(t-1)}\| > \epsilon$ **do**

$$\mathbf{r}^{(t+1)}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \frac{\mathbf{r}^{(t)}(p_j)}{d^+(p_j)}$$

Does it always converge to a unique and meaningful solution?

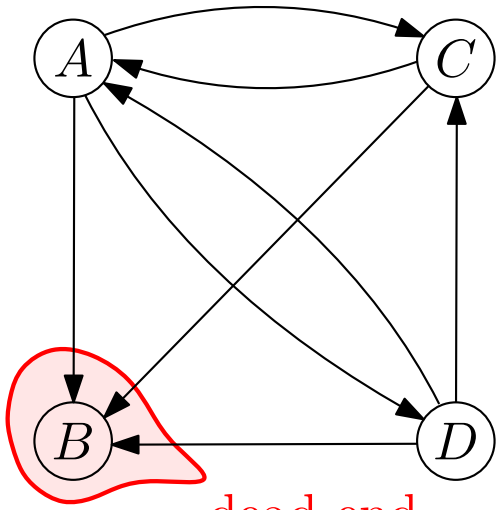
Fundamental problems

1. Dangling node or dead ends
   - Node(s) with out-degree 0 (sink nodes)
   - Since rating is not distributed, total rank leaks out
2. Spider Traps
   - All out-links within a component (sink components)
   - They eventually absorb all the rating

Both problems make the graph not strongly connected

Pagerank Algorithm: Dead Ends

Total rank leaks out to the dead end



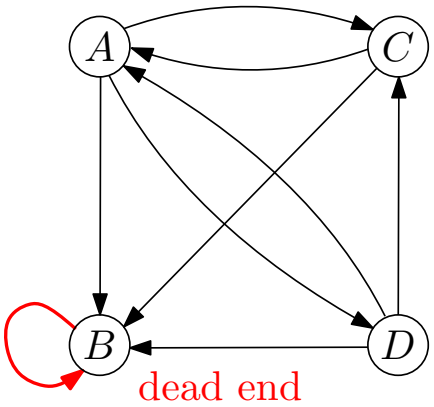
$$\begin{bmatrix} 0 & 0 & 1/2 & 1/3 \\ 1/3 & 0 & 1/2 & 1/3 \\ 1/3 & 0 & 0 & 1/3 \\ 1/3 & 0 & 0 & 0 \end{bmatrix}$$

Link Matrix, L

node	$\mathbf{r}^{(0)}(\cdot)$	$\mathbf{r}^{(1)}(\cdot)$	$\mathbf{r}^{(2)}(\cdot)$	$\mathbf{r}^{(3)}(\cdot)$
A	0.25	0.2083	0.1111	0.0718
B	0.25	0.2917	0.1806	0.1088
C	0.25	0.1667	0.0972	0.0602
D	0.25	0.0833	0.0694	0.0370
Total pagerank = 1.0		0.75	0.4583	0.2778

Pagerank Algorithm: Dead Ends

- Make a self-loop from a dangling node to itself
- This benefits such pages, become spider traps



$$\begin{bmatrix} 0 & 0 & 1/2 & 1/3 \\ 1/3 & 1 & 1/2 & 1/3 \\ 1/3 & 0 & 0 & 1/3 \\ 1/3 & 0 & 0 & 0 \end{bmatrix}$$

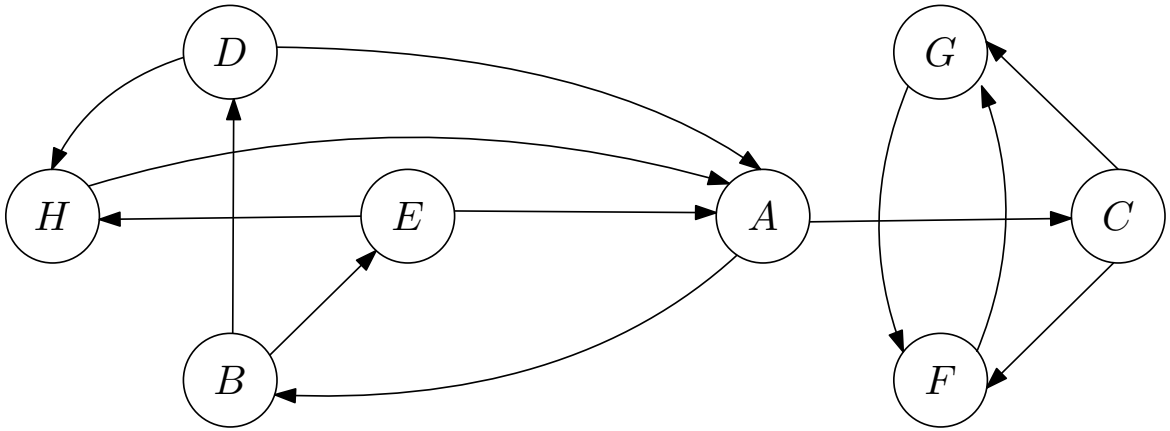
Link Matrix, L

node	$\mathbf{r}^{(0)}(\cdot)$	$\mathbf{r}^{(1)}(\cdot)$	$\mathbf{r}^{(2)}(\cdot)$	$\cdots$	$\mathbf{r}^{(*)}(\cdot)$
A	0.25	0.2083	0.1111		0.0
B	0.25	0.5417	0.7222		1.0
C	0.25	0.1667	0.0972		0.0
D	0.25	0.0833	0.0694		0.0

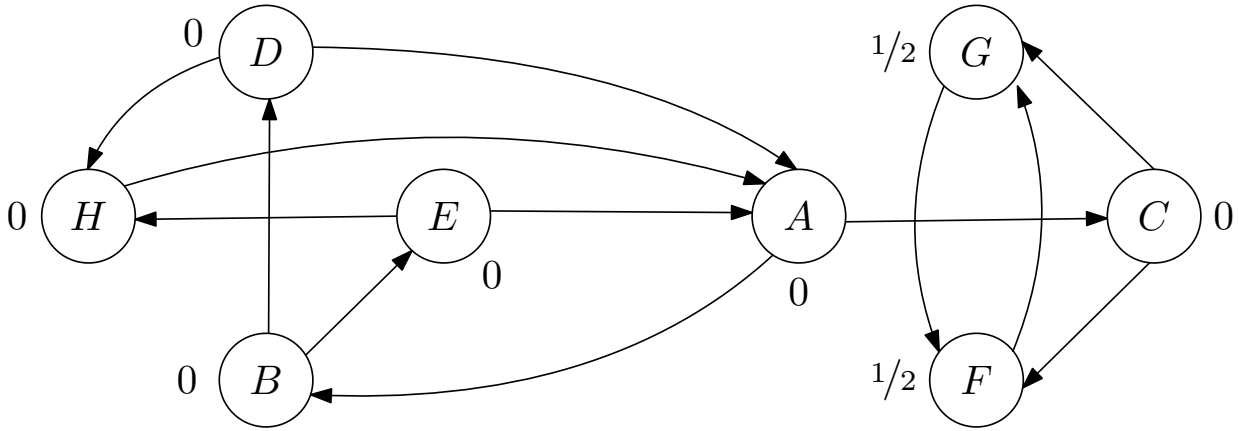
- A better solution is to recursively prune them out
- Compute pagerank for the remaining nodes
- Other methods to deal with them are also used (see below)

Pagerank Algorithm: Spider Traps

- Spider Traps: sink component(s)
- Eventually absorb all the rating



node	$\mathbf{r}(\cdot)$
A	$1/8$
B	$1/8$
C	$1/8$
D	$1/8$
E	$1/8$
F	$1/8$
G	$1/8$
H	$1/8$

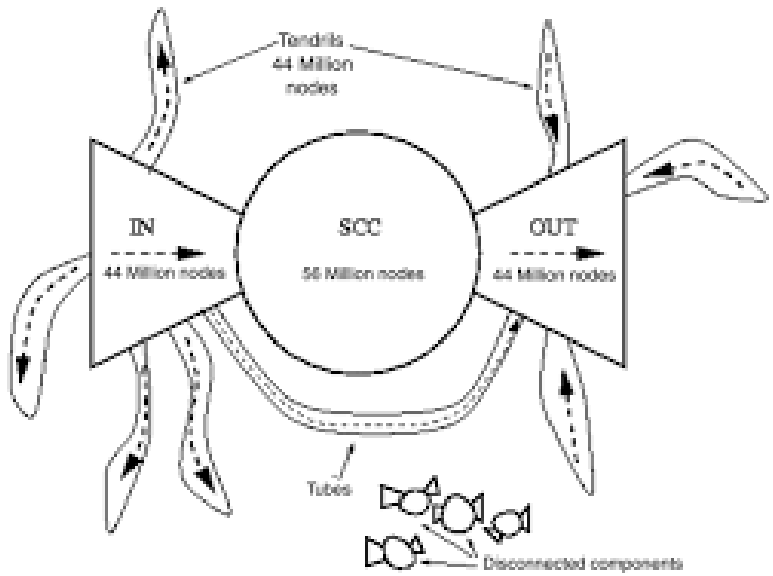


node	$\mathbf{r}(\cdot)$
A	0
B	0
C	0
D	0
E	0
F	$1/2$
G	$1/2$
H	0

Pagerank Algorithm: Web Structure

- Structural challenges to Web IR algorithms are not only hypothetical
- Broder et.al. (2000) SCC analysis of webgraph (AltaVista index)
- Study replicated for larger recent webgraphs reveal similar structure

- $\sim 200m$ pages, $\sim 1.5b$ links
- bow-tie structure (macroscopic)
- grouping of SCC's
- CORE: a giant SCC ($\sim 56m$) nodes
- IN: can reach CORE (unidirectional)
- OUT: can be reached from CORE
- TENDRILS:
  - reachable from IN cannot reach CORE
  - can reach OUT not reachable from CORE
- TUBES: both types of tendrils
- DISCONNECTED COMPONENTS



The bow-tie structure of the web (A. Broder et.al (2000))

Pagerank Algorithm: Random Teleports

- Spider Traps: sink component(s)
- Eventually absorb all the rating
- Google fix is "random restarts" (random teleport)
  - With probability $1 - \beta$ follow an out-link
  - With probability β jump to a random page
  - Generally, $\beta \in [0.1 - 0.15]$ ▷ β is called Damping Factor
- Random walker (surfer) will teleport out of a spider trap
- From dead-ends teleport with probability 1
  - Matrix becomes column-stochastic

Adjusted Formulation

$$\mathbf{r}(p_i) = \sum_{p_j \in N^-(p_i)} (1-\beta)\frac{\mathbf{r}(p_j)}{d^+(p_j)} + \beta\frac{1}{n}$$

$$L' = (1-\beta)L + \frac{\beta}{n}\mathbb{I}_n$$

Link Spamming

Search Engine Optimization (SEO)

Web Search: Link Spamming

SEO: techniques used to increase the visibility of a website in search engine results

Aim to improve a page's ranking through legitimate methods, such as keyword optimization and quality content

- On-page SEO
- Off-page SEO
- Technical SEO

Link Spamming attempt to manipulate search engines through unethical means, exploiting ranking algorithms like PageRank

Link spamming strategies include creating networks of low-quality or irrelevant links to artificially inflate the ranking of a page

Lead to low-quality pages on top in search results

Web Search: Link Spamming

- Link Farms: Densely connected subgraphs to increase ranks of pages
- Private Blog Networks: authoritative expired websites with influential in-links used to have out-links to targeted web pages
- Sybil Attack: spammer create multiple inter-linked websites at different domain names to inflate the number of links
- Spam Blogs: Blogs created for promoting a webpage
- Guest blog Spam: Posting on other blogs to add links back to the target website
- Referrer Log Spamming: Getting target website to appear in the logs of another site, tricking search engines to treat it as more relevant
- Forum Spam, Comment Spam, Wiki Spam Posting irrelevant links in forums, comments, and wikis to artificially increase backlinks

Link Spamming: Google's Countermeasures

Algorithm Updates: specifically target and penalize websites engaging in link spamming tactics, such as keyword stuffing and unnatural backlinks

- Penguin
- Panda

Manual Penalties: demoting or de-indexing sites that violate their guidelines

Google uses ML to detect and filter spam links from contributing to search rankings

Link Analysis: Search engines analyze the quality of backlinks, considering factors such as the relevance of the linking site, the anchor text, and the diversity of backlinks

Anchor Text

Google penalizes various content and link spamming tricks (Google Panda)

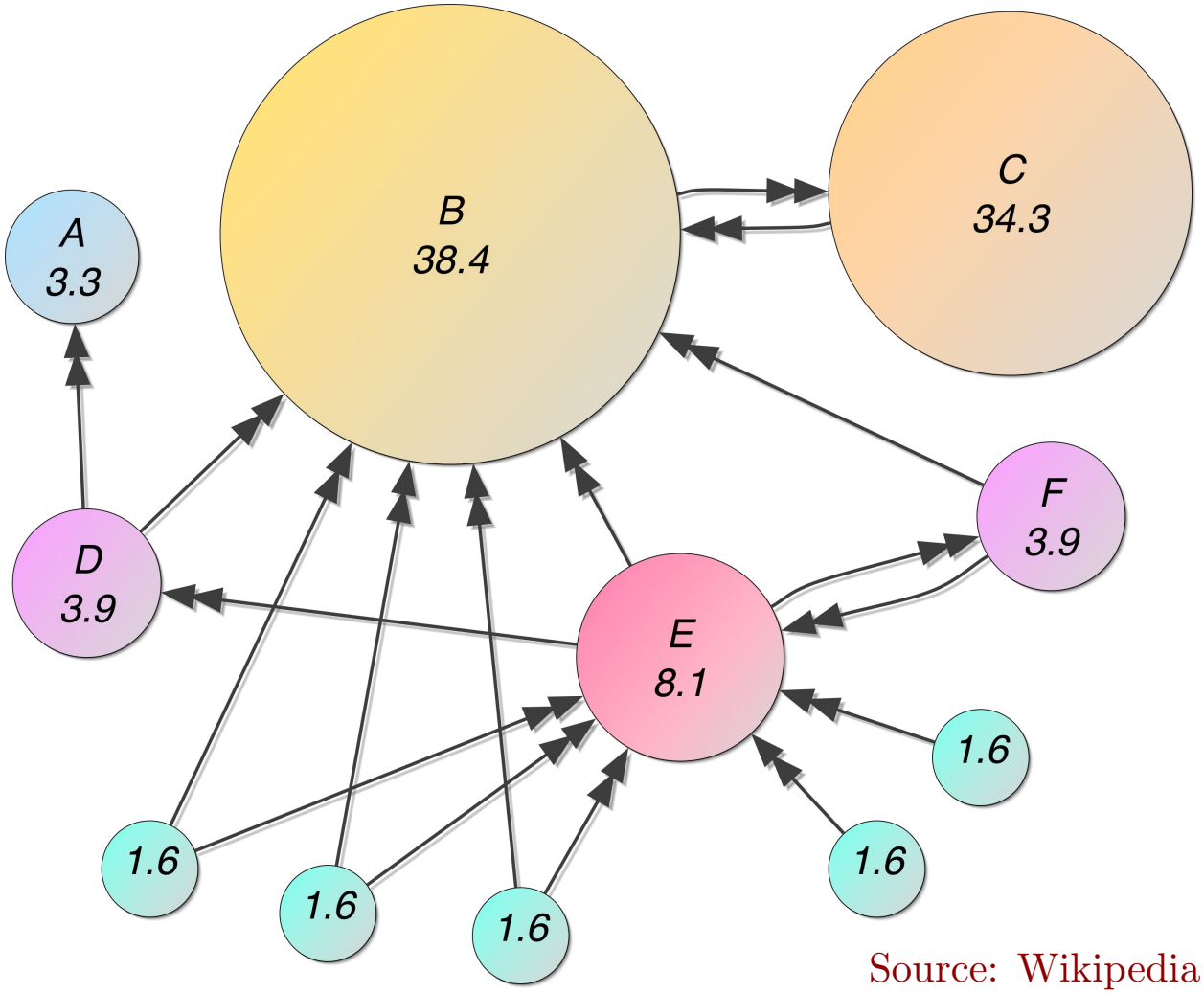
Anchor text, link label or link text is the visible, clickable text in hyperlinks

Links types: No anchor text (click here), Naked URL (www.abc.com) or (LUMS)

- (Legitimate) Anchor text describes the landing page better than the content of the page itself
- Anchor texts from links pointing to a page is included when the page is indexed
- Terms from anchor text are weighted highly in the vector representation of the page
- Help index non-html or non-text pages too (images/videos)

A related concept is that of Google Bomb

Pagerank Visualization



Source: Wikipedia

Visualization of pageranks (percentages) for a small graph
damping factor, $\beta = .85$

Topic Sensitive Pagerank

Personalized and Topic Sensitive Pagerank

- So far we discussed query independent ranking
  - Compute an apriori rating $\mathbf{r}$ of all web pages in the index
  - On query q, find the subset C of pages relevant to q
  - Present pages in C in decreasing order of $\mathbf{r}$
- Specialized Ranking w.r.t query
  - On query q by user u
  - Personalized Pagerank [Brin & Page, 1998] adjust ordering according to the user u
  - Topic Sensitive PageRank [Taher Haveliwala, 2002] adjust rating according to topic of the query q

Topic Sensitive Pagerank

Goal: Not just PageRank - rate pages also by relevance to topic of query q

In conventional PageRank we teleported to any of the pages equally likely

$$\mathbf{r}(p_i) = \sum_{p_j \in N^-(p_i)} (1-\beta)\frac{\mathbf{r}(p_j)}{d^+(p_j)} + \beta \underbrace{\frac{1}{n}}_{T}$$

T is a uniform distribution over all pages

Topic sensitive PageRank

- identify a subset of pages S (how?) ▷ teleport set
- related to the topic of q (what is topic of q?)

T_S: non-uniform probability distribution (high values at coordinates in S)

Random walker is biased towards S more likely to jump onto pages in S (hence spend more times), their ratings will be higher

Can we compute PageRank at query time?

Topic Sensitive Pagerank

Preprocessing:

- Fix a set of k topics
- Find pages about each topics (k teleport sets $S_1, S_2, \ldots, S_k$)
- For each topic find PageRank of all pages using T_{S_i} for teleporting
  - Each page has k PageRank scores, $\mathbf{r}_1(\cdot), \ldots, \mathbf{r}_k(\cdot)$ - one for each topic

Query-time processing:

- Compute distribution of likelihoods of q belonging to each of k topics
  - i.e. for $1 \leq i \leq k$ find $Pr[C_i|q]$ (probability that q's topic is C_i)
- TSPR of page u is weighted (by $Pr[C_i|q]$) sum of $\mathbf{r}_i(u)$

$$\text{TSPR}(u|q) = \sum_{i=1}^{k} Pr[C_i|q]\, \mathbf{r}_i(u)$$

Topic Sensitive Pagerank: Preprocessing and Query-time processing

Preprocessing:

- Find pages about the fixed k topics (k teleport sets $S_1, S_2, \ldots, S_k$)
- DMOZ top level, (sports, business, health, education)
- Use topic modeling, cluster TF-IDF vectors of pages, use document embedding and deep learning

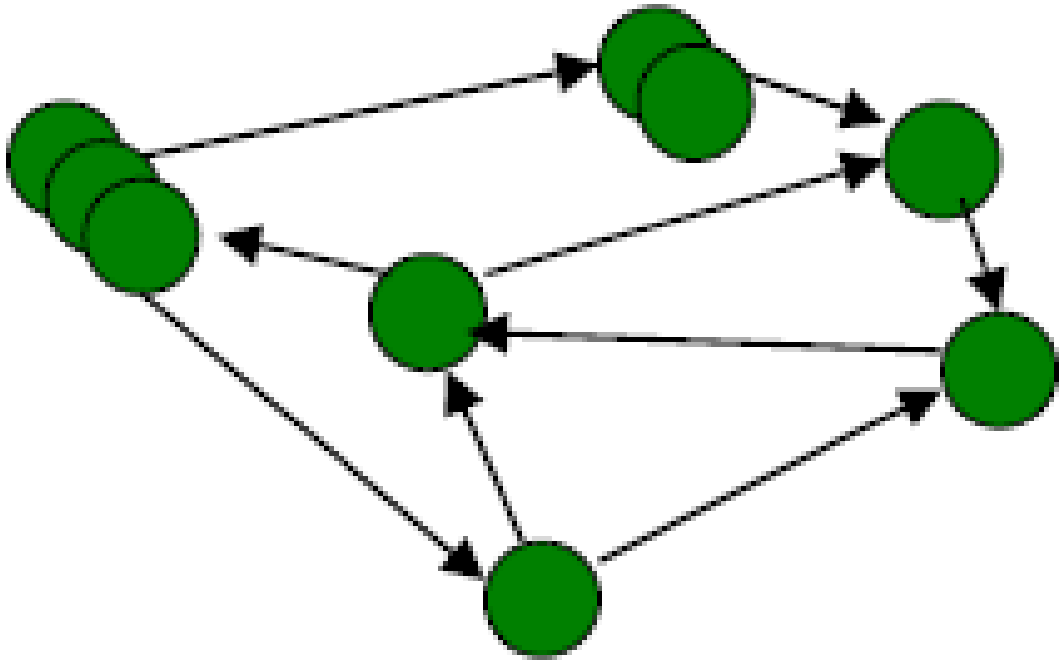
Query Classification: Classify the query into one or more topics.

Compute distribution of likelihoods of q belonging to each of k topics

- User can pick topic from a combo-box
- Use classification to classify query into a topic
- Use query launching context
  - e.g. query launched from a webpage (local search bar) about topic i
  - History of queries e.g. "basketball" followed by "Jordan"
- Use user context e.g. Users social media profile, attributes etc.

Topic Sensitive Pagerank

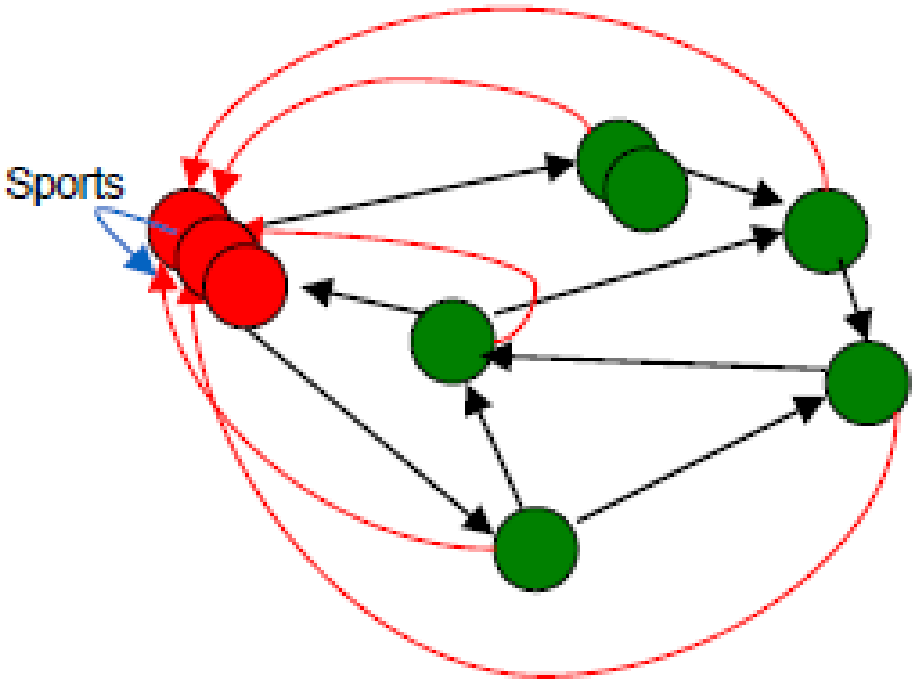
J. Magalhaes @Universidade NOVA de Lisboa



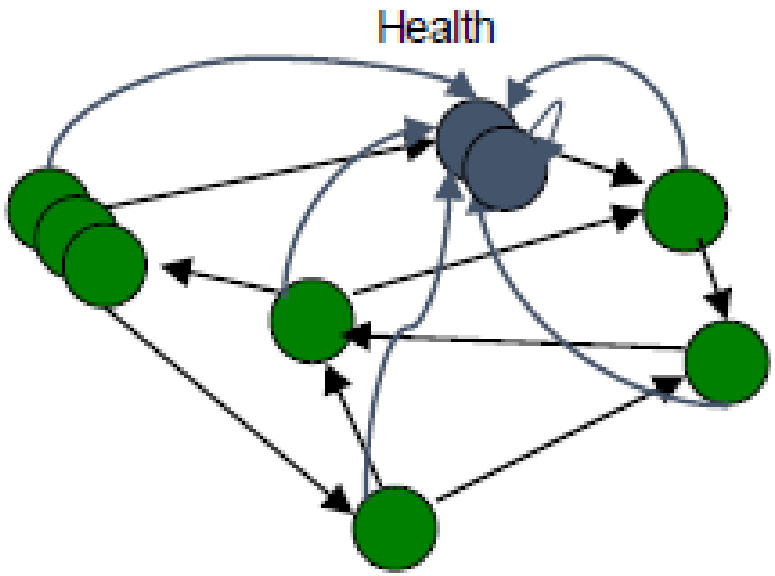
The query has 90% chance of being about **Sports**.

The query has 10% chance of being about **Health**.

Topic Sensitive Pagerank

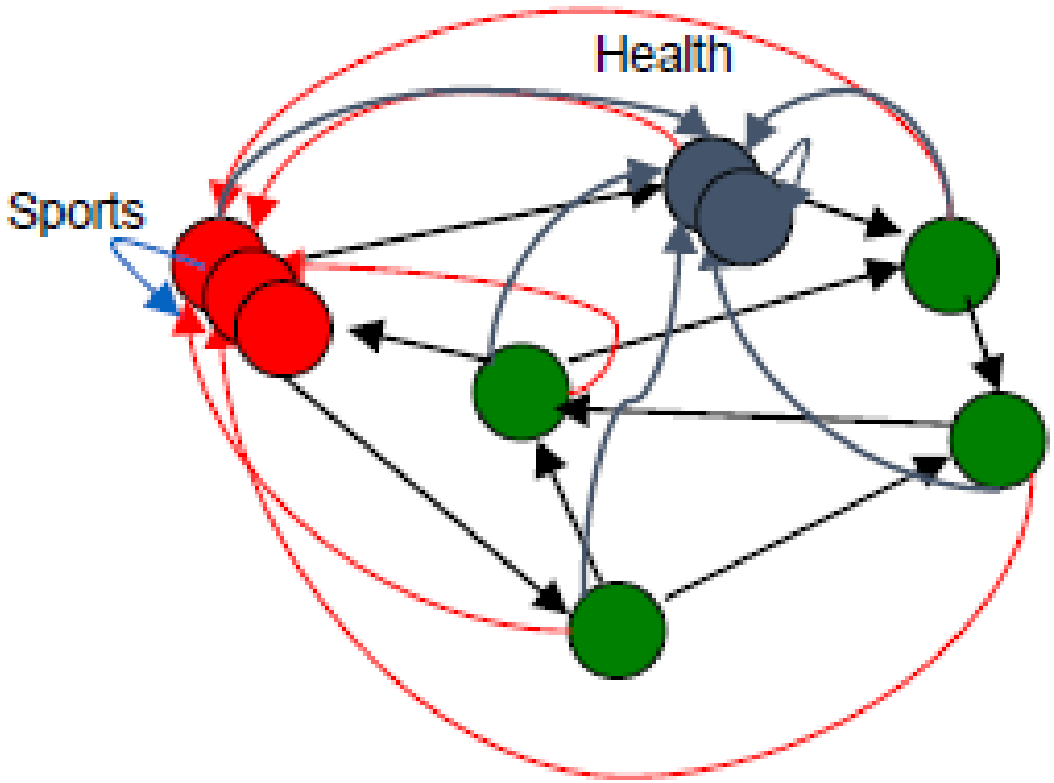


Sports teleportation



Health teleportation

Topic Sensitive Pagerank



pr = (0.9 PR_{sports} + 0.1 PR_{health}) gives you:
9% sports teleportation, 1% health teleportation

Hyperlink-Induced Topic Search (HITS)

Hyperlink-Induced Topic Search (HITS)

HITS developed by Jon Kleinberg (1998)

- PageRank: best suited for most reliable pages to specific queries
- HITS: best suited for "broad topic" queries
- It returns a broader common opinion
- Not only find pages that reliably has relevant content but also finds "experts" on the topic, pages linking to many relevant pages
- In response to a query HITS finds two sets of inter-related pages

HITS Computes two scores for each page: hub score and authority score

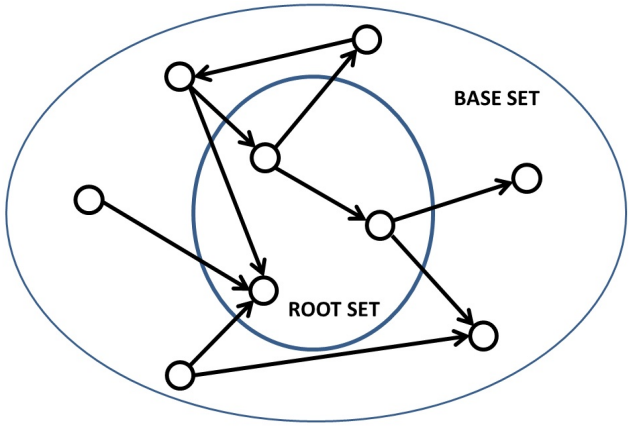
Hyperlink-Induced Topic Search (HITS)

Underlying Assumption: Page A links to Page $B \implies A$ recommends B

- Hubs (high quality experts): pages that link to many "good pages"
- Pages that link to many other pages
  - List of top data science conferences
  - Course bulletin
- Authorities (high quality content): pages listed on many "good hubs"
  - Conference webpages
  - Course home pages
- Hubs and authorities are mutually reinforcing.

Hyperlink-Induced Topic Search (HITS)

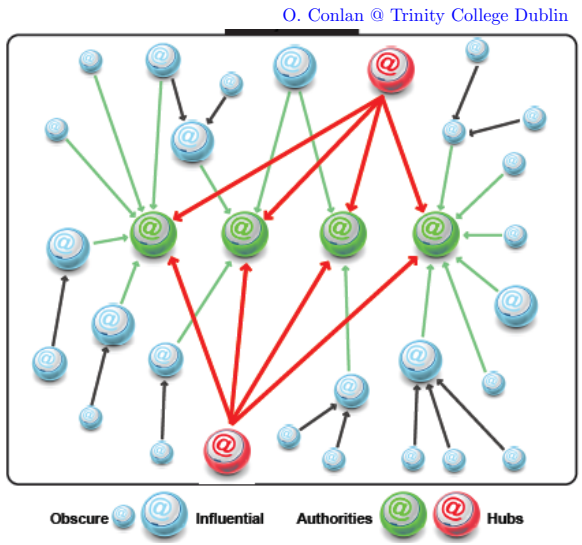
- Extract from the index a subset of pages that could be potentially good hubs or authorities (base set, B) as follows
  - On query q, get root set: of pages most "relevant" to q
  - Add pages that are either
    - linking to a page in the root set, or
    - linked to by a page in the root set



- For each page in B, compute its hub rating and authority rating
- Return the top k hubs and authorities from the base set

Hyperlink-Induced Topic Search (HITS)

- Each page x (in the base set) has two scores
  - $\mathbf{h}(x)$: hub score of x : measure quality of x as an "expert"
  - $\mathbf{a}(x)$: authority score of x : measure quality of x as "content"
- Initialize $\mathbf{h}(x)$ and $\mathbf{a}(x)$ to 1 for all x
- Principle of repeated improvement
- **A page is a good authority if it is linked to by good hubs**
- **A page is a good hub if it links to good authorities**



Hyperlink-Induced Topic Search (HITS)

- Each page x (in the base set) has two scores
  - $\mathbf{h}(x)$: hub score of x : measure quality of x as an "expert"
  - $\mathbf{a}(x)$: authority score of x : measure quality of x as "content"
- Initialize $\mathbf{h}(x)$ and $\mathbf{a}(x)$ to 1 for all x
- Principle of repeated improvement
- **A page is a good authority if it is linked to by good hubs**
- **A page is a good hub if it links to good authorities**
- $\mathbf{a}(x)$ is sum of hub scores of pages pointing to x

$$\mathbf{a}(x) \leftarrow \sum_{y \in N^-(x)} \mathbf{h}(y)$$

- $\mathbf{h}(x)$ is sum of authority scores of pages x is pointing to

$$\mathbf{h}(x) \leftarrow \sum_{y \in N^+(x)} \mathbf{a}(y)$$

HITS Algorithm

Hyperlink-Induced Topic Search (HITS)

Algorithm HITS

$\mathbf{h} \leftarrow \text{ONES}(|B|)$ ▷ B : base set

$\mathbf{a} \leftarrow \text{ONES}(|B|)$ ▷ Initialize $\mathbf{h}(\cdot)$ and $\mathbf{a}(\cdot)$ to 1

while stopping condition is not met **do**

$\text{NORMALIZE}(\mathbf{a}, \mathbf{h})$ ▷ $\mathbf{h} \leftarrow \mathbf{h}/\|\mathbf{h}\|$

For each $p_i \in B$

$$\mathbf{h}(p_i) \leftarrow \sum_{p_j \in N^+(p_i)} \mathbf{a}(p_j)$$

$$\mathbf{a}(p_i) \leftarrow \sum_{p_j \in N^-(p_i)} \mathbf{h}(p_j)$$

HITS Algorithm

- Each page i (in the base set) has two scores
  - $\mathbf{h}(i)$: hub score of i: measure quality of i as an "expert"
  - $\mathbf{a}(i)$: authority score of i: measure quality of i as "content"
- Initialize $\mathbf{h}(i)$ and $\mathbf{a}(i)$ to 1 for all i

$$\mathbf{a}(i) = \sum_{j \in N^-(i)} \mathbf{h}(j) \qquad \mathbf{h}(i) = \sum_{j \in N^+(i)} \mathbf{a}(j)$$

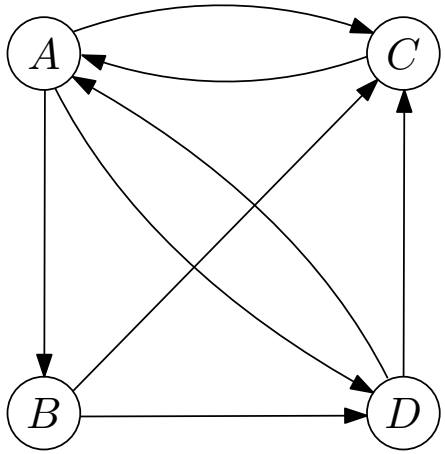
Let A be the adjacency matrix of the subgraph induced by B, $|B| = n$

A is $n \times n$ matrix with $A(i,j) = 1$ if $p_i \rightarrow p_j$

$$\mathbf{h}(i) = \sum_{j \in N^+(i)} \mathbf{a}(j) \quad \Leftrightarrow \quad \mathbf{h}(i) = \sum_j A(i,j)\, \mathbf{a}(j) \quad \Leftrightarrow \quad \mathbf{h} = A\mathbf{a}$$

$$\mathbf{a}(i) = \sum_{j \in N^-(i)} \mathbf{h}(j) \quad \Leftrightarrow \quad \mathbf{a}(i) = \sum_j A(j,i)\, \mathbf{h}(j) \quad \Leftrightarrow \quad \mathbf{a} = A^T\mathbf{h}$$

HITS Algorithm



Adjacency Matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{h} = A\,\mathbf{a}$$

$$\mathbf{a} = A^T\,\mathbf{h}$$

node	$\mathbf{h}^{(0)}(\cdot)$	$\mathbf{h}^{(1)}(\cdot)$	$\mathbf{h}^{(2)}(\cdot)$	$\mathbf{h}^{(3)}(\cdot)$	$\mathbf{h}^{(4)}(\cdot)$	$\mathbf{h}^{(5)}(\cdot)$
A	1	3	6	15	33	79
B	1	2	5	12	27	64
C	1	1	2	3	7	13
D	1	2	5	10	23	50

node	$\mathbf{a}^{(0)}(\cdot)$	$\mathbf{a}^{(1)}(\cdot)$	$\mathbf{a}^{(2)}(\cdot)$	$\mathbf{a}^{(3)}(\cdot)$	$\mathbf{a}^{(4)}(\cdot)$	$\mathbf{a}^{(5)}(\cdot)$
A	1	2	3	7	13	30
B	1	1	3	6	15	33
C	1	3	7	16	37	83
D	1	2	5	11	27	60

HITS Algorithm

Algorithm HITS

$\mathbf{h} \leftarrow \text{ONES}(|B|)$

$\mathbf{a} \leftarrow \text{ONES}(|B|)$

while stopping condition is not met **do**

$\text{NORMALIZE}(\mathbf{a}, \mathbf{h})$

$\mathbf{h} \leftarrow A\,\mathbf{a}$

$\mathbf{a} \leftarrow A^T\mathbf{h}$

$$\mathbf{h} = A\,\mathbf{a} = A\,\underbrace{A^T\mathbf{h}} = \underbrace{A\,A^T}\mathbf{h} \qquad \mathbf{a} = A^T\mathbf{h} = A^T\,\underbrace{A\mathbf{a}} = \underbrace{A^T\,A}\mathbf{a}$$

In $2k$ steps $\quad \mathbf{h} = (A\,A^T)^k\mathbf{h} \qquad \mathbf{a} = (A^T\,A)^k\mathbf{a}$

- Under some reasonable conditions on A, HITS converges to $\mathbf{h}^*$ and $\mathbf{a}^*$
- $\mathbf{h}^*$ is the principal eigenvector of AA^T
- $\mathbf{a}^*$ is the principal eigenvector of A^TA

PageRank vs. HITS

- PageRank: Measures importance of pages based on the link structure
- HITS: Differentiates between hubs and authorities
- Both use link analysis but have different applications and limitations
- HITS is better for Topic-specific search
- HITS is more Computationally intensive
- HITS is more sensitive to noise in the link structure
- Google uses a very complicated algorithm, incorporating more than a 100 factors
- The current Google algorithm is very close to HITS
- HITS is implemented in Ask.com and teoma

Advanced Topics and Current Trends

TrustRank

TrustRank conducts link analysis to separate useful webpages from spam and helps search engine rank pages in Search Engine Results Pages

Avoids the impractical manual review of the Internet

Introduced by researchers from Stanford and Yahoo! in 2004

Turstrank is a part of major web search engines like Yahoo! and Google

TrustRank

- TrustRank uses known-quality websites (seed websites) as a base for evaluating quality of other webpages
- The algorithm checks the outbound links from the trust seed websites and assesses the quality of the websites these links lead to
- It looks at the quality of backlinks to determine how reliable (trustworthy) each website is. More reliability = higher rankings

TrustRank is a semi-automated – it needs some initial human assistance

SALSA

Ranking is a fundamental problem in information retrieval

Link analysis views a hyperlink as an endorsement by a web page's author of another web page

Link- based ranking algorithms can be broadly grouped into two classes:

1. Query independent algorithms that estimate the quality of a web page
   ▷ e.g., Pagerank
2. Query-dependent ones that estimate pages' relevance to a query
   ▷ substantially more effective

Stochastic Approach for Link Structure Analysis (SALSA), a variation of HITS is a query-dependent link-based ranking algorithm

SALSA Algorithm

- It takes a result set R as input, and constructs a neighborhood graph from R in the same way as HITS
- Similarly, it computes an authority and a hub score for each vertex in the neighborhood graph, and these scores can be viewed as the principal eigenvectors of two matrices
- However, instead of using the straight adjacency matrix that HITS uses, SALSA weighs the entries according to their in and out-degrees
- The approach is based upon the theory of Markov chains, and relies on the stochastic properties of random walks performed on the collection of pages
- The input to the scheme consists of a collection of pages C which is built around a topic t
- Intuition suggests that authoritative pages on topic t should be visible from many pages in the subgraph induced by C. Thus, a random walk on this subgraph will visit t-authorities with high probability

Okapi BM25

The BM25 is a ranking algorithm used in search engines to score and rank documents that are relevant to a given query

- TF-IDF assigns weights to terms based on their frequency in a document, but does not take into account the length of the document or the average length of documents in the corpus
- BM25, an improvement over the TF-IDF, uses a similar approach, it also incorporates the inverse document frequency of each term, as well as a set of adjustable parameters that can be tuned to improve performance
- The result is a more accurate ranking of documents that are relevant to a given query

BM25 is flexible and effective. It can be adapted to different types of search tasks, from ad-hoc search to recommendation systems, and can be tuned to perform well on specific domains or languages

It is computationally efficient and easy to implement, making it a practical choice for large-scale search systems

Learning to Rank (LTR)

Machine learning approach to ranking

- Uses labeled training data (known relevance of documents to queries) to learn a ranking model
- LTR models can combine many features, term frequency, document length, user behavior data (click-through rates, dwell time), etc.
- Use models like decision tree, neural network, gradient-boosted tree
- Common algorithms: RankNet, RankBoost, LambdaMART

Three main approaches:

- Pointwise: Predict relevance score of individual documents, treating ranking as a regression problem
- Pairwise: Learn to rank by comparing pairs of documents and determining which one should rank higher
- Listwise: Directly optimize for the entire ranking of documents in response to a query

Personalized Search

Gather and analyze user information to model their behavior/ preferences/ intent and tailors search results to provide targeted/useful information

- Collaborative Filtering: Using data from similar users
- Content-Based Filtering: Using the content of items/webpages
- Hybrid Methods: Combining multiple techniques

- **Explicit User Profiling**: Based on user-provided information (e.g., demographics, location, social connections)
- **Implicit User Profiling**: Based on user behavior
  - Search History: Past search queries to infer preferences and needs
  - Click Behavior: Links clicked after search give insights of user interests
  - Location and Time: Geolocation and temporal data for localized results
  - Device Information: The device (mobile, desktop) provides context on user behavior (e.g., mobile searches often imply a need for quick, concise information)

Contextual Search

Tailoring search results based on the context of the query

Factors:

- Geolocation: Providing location-specific results
- Time: Adjusting results based on time of day or recency
  - ▷ e.g.,showing news results based on the latest stories
- Device Type: (e.g., mobile or desktop version of websites)
- User Intent: based on previous searches and session data Real-time Context: Contextual search adapts dynamically, taking into account the user's immediate environment

Multimedia Search

Searching for non-textual content (images, videos, audio)

Key challenges:

- Scalability: Requires handling large amounts of data and developing sophisticated methods to extract and match features
- Feature Extraction: Multimedia represented in a way that search engines can process
- Content Understanding: Difficulty in understanding and indexing multimedia content
- Semantic Gap: Bridging the gap between low-level features and high-level concepts
- Content Matching: Comparing multimedia content, (image similarity, video sequences, or audio patterns)
- Diversity: Providing diverse results to cover different aspects of query
- Relevance Ranking: Rank content based on relevance to user queries (text-based or multimedia-based (e.g., reverse image search))
- Diverse Formats: Managing different formats and standards

Image Search

Image Search allows users to search for images based on a query, which can be text (keywords) or an example image (reverse image search)

Text-Based Image Search (TBIR): Images are associated with metadata such as titles, tags, or descriptions. The search engine matches the query with this textual information

Content-Based Image Retrieval (CBIR): Uses visual features like color, texture, and shapes to find visually similar images to a given query image

- Color Histograms: Distribution of colors in the image
- Texture Analysis:
- Shape Descriptors: SIFT/SURF Descriptors, Detect and describe local features within the image
- Image Similarity Measure

Reverse Image Search: Allows users to provide an image as input, and the search engine retrieves visually similar images from the web

Video and Audio Search

Search and retrieve video and audio content for applications: in media monitoring, entertainment, security

- Text-Based Search: Metadata like titles, descriptions, and tags are used to find relevant videos
- Content-Based Video Retrieval: Extracts visual and motion features from video frames for content-based searches
- Shot Boundary Detection: Detects scene changes to segment videos and analyze keyframes
- Temporal Matching: Matches frames or visual events to a query video

Audio Search

- Speech Recognition: Converts spoken words in audio into text, allowing users to search based on spoken content
- Audio Fingerprinting: Extracts unique audio features (e.g., Mel-frequency cepstral coefficients) to identify or match audio clips
- Music Search: Matches a user's hummed or whistled melody to existing music tracks

Trends in Web Search: Neural Search Models

Use deep learning to represent queries and documents as vectors in a high-dimensional space, making it easier to measure similarity

Enable better semantic understanding by capturing more complex relationships between terms

Contextual Understanding: Models like BERT and GPT-3 can consider the context of words in a query, leading to more accurate results

- DSSM (Deep Structured Semantic Model): Learns semantic representations of queries and documents to improve ranking
- BERT-based Ranking: Pre-trained models like BERT capture deep linguistic features, improving search relevance for complex, natural language queries

Trends in Web Search: NLP in Search

Natural Language Processing (NLP) improves the way search engines understand and respond to user queries

NLP allows search engines to:

- Understand User Intent: Even when queries are vague or conversational, NLP helps determine the user's underlying intent
- Named Entity Recognition: Identify specific entities like people, places, and things mentioned in queries
- Query Rewriting: Automatically reformulate queries to improve retrieval accuracy (e.g., correcting typos or using synonyms)

For the query "movies by Christopher Nolan," NLP techniques help recognize "Christopher Nolan" as a director and return relevant movie titles, even though the query does not explicitly include the word "director"

Trends in Web Search: Retrieval Augmented Generation (RAG)

- Retrieval-Augmented Generation (RAG) grants generative artificial intelligence models information retrieval capabilities
- It modifies interactions with a large language model (LLM) so that the model responds to user queries with reference to a specified set of documents, using this information to augment information drawn from its own vast, static training data
- This allows LLMs to use domain-specific and/or updated information. Use cases include providing chatbot access to internal company data or giving factual information only from an authoritative source
- RAG can be used on unstructured (usually text), semi-structured, or structured data (for example knowledge graphs)

Trends in Web Search: Semantic Search

Search that understands the meaning behind queries

Techniques:

- Knowledge Graphs
- Ontologies

Trends in Web Search: Voice and Conversational Search

Growing trend with the rise of smart devices and virtual assistants, making search more accessible and conversational.

Driven by Voice Assistants (Google Assistant, Amazon Alexa, Apple's Siri)

- Speech Recognition: Converts spoken queries into text
- Personalized Responses: Voice assistants leverage user data and preferences to tailor responses (e.g., personalized news updates)
- Contextual Understanding: Understand intent and context from voice queries and conversations
- Conversational/Interactive Search: The search engine may respond with multi-turn dialogue, refining the results based on user feedback
- Integration with Services: Assistants can perform actions beyond search, (send message, play music, control smart home devices)

"What's the weather like today?" followed by, "What about tomorrow?"

Search engine must understand that second query refers to weather

Challenges and Opportunities in Voice Search

- Ambiguity: Voice queries are often more ambiguous and less precise than text queries, hard to determine user intent
- Accents and Dialects: Speech recognition systems may struggle with varied accents, dialects, or background noise
- Multi-turn Dialogues: Maintaining context across multiple interactions requires sophisticated conversational models

- Accessibility: Voice search opens up new possibilities for users with disabilities or those in hands-free environments
- Personalization: Deeper integration with user data allows for highly personalized and context-aware search experiences
- Growth in Smart Devices: Proliferation of smart home devices creates more opportunities for voice search and task automation

Web Search Consideration: Bias and Fairness

Bias: Systematic favoring or marginalization of certain groups, viewpoints, or content in search results ranking

- Algorithmic Bias: Bias embedded in the design of ranking algorithms, often unintentionally, due to the training data or model assumptions
- Data Bias: If the training data used to develop search models reflects existing societal biases (e.g., gender or racial bias), these biases can be perpetuated in search results
- Representation Bias: Over-representation or under-representation of specific topics, regions, or demographic groups in search results

Search engines must strive for fairness, ensuring that results do not unfairly favor or penalize specific individuals, groups, or viewpoints

Approaches to mitigate bias include diverse datasets, algorithm audits, and fairness-aware models

Challenges and Solutions in Protecting Privacy

- Anonymization: Removing identifying information is not always sufficient, as data can be re-identified by combining various sources
- Consent and Control: Ensuring users understand data collection and giving them control over its usage is challenging, especially with complex data policies
- Security: Safeguarding stored user data from breaches

- Differential Privacy: A technique that adds noise to data to protect individual privacy while still allowing for useful aggregate analysis
- Data Minimization: Collecting only the data necessary for the task at hand and deleting it after a reasonable time period
- User Transparency: Clear communication with users about data collection, use, and retention policies, and offering them control through privacy settings

DuckDuckGo emphasizes privacy by not tracking users or storing search histories

Additional Concepts

- Federated Learning
  - Training models across decentralized devices while keeping data local.
  - Enhances privacy and reduces data transfer.
- Explainable AI (XAI)
  - Making AI decisions transparent and understandable.
  - Importance in building trust and accountability.
- Multimodal Search
  - Combining text, image, and video search.
  - Examples: Google Lens, Bing Visual Search.