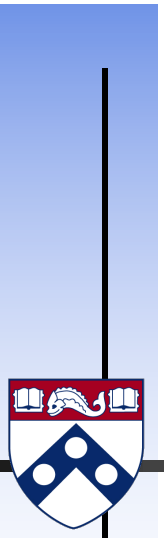


CIS 455/555: Internet and Web Systems

Information Retrieval

November 3, 2021

Plan for today

- Web services ✓
- Information retrieval ✓
  - Basics ✓
  - Precision and recall ✓
  - Taxonomy of IR models ✓
- Classic IR models
  - Boolean model ✓
  - Vector model NEXT
  - TF/IDF
- HITS and PageRank

Vector model

- Define:

  $w_{ij} > 0$ whenever $k_i \in d_j$

  $w_{iq} \geq 0$ associated with the pair (k_i, q)

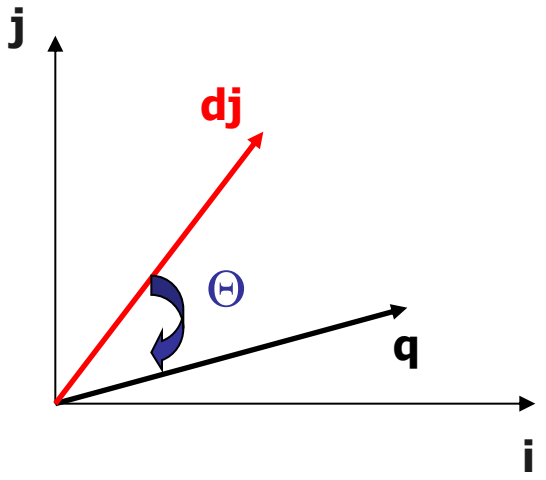
  $\vec{d_j} = (w_{1j}, w_{2j}, \ldots, w_{tj})$

  $\vec{q} = (w_{1q}, w_{2q}, \ldots, w_{tq})$

  - With each term k_i, associate a vector *vec(i)*
  - These vectors (e.g., *vec(i)* and *vec(j))* are assumed to be orthonormal (i.e., index terms are assumed to occur independently within the documents)
    - Does this assumption ("independence assumption") hold in practice?

- The *t* vectors *vec(i)* form an orthonormal basis for a t-dimensional space
- In this space, queries and documents are represented as weight vectors

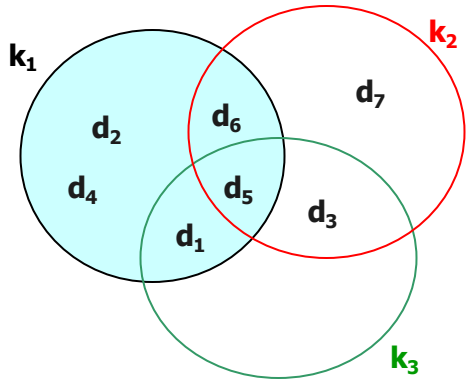
Cosine similarity



$$sim(d_j, q) = \cos\theta = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \cdot w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \cdot \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$$

- All weights are nonnegative; hence, $0 \le sim(q,d_j) \le 1$

Vector Model Example 1

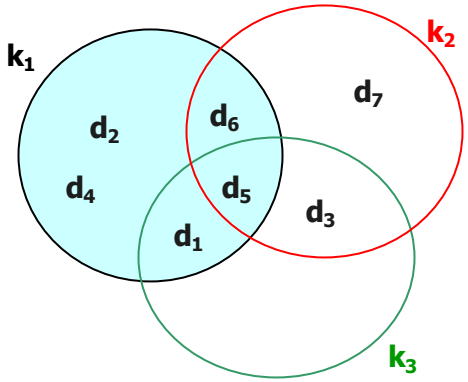


Uniform weights

Query: k_1 k_2 k_3

	k_1	k_2	k_3	$q \bullet d_j$
d_1	1	0	1	2
d_2	1	0	0	1
d_3	0	1	1	2
d_4	1	0	0	1
d_5	1	1	1	3
d_6	1	1	0	2
d_7	0	1	0	1
q	1	1	1	

Vector Model Example 2

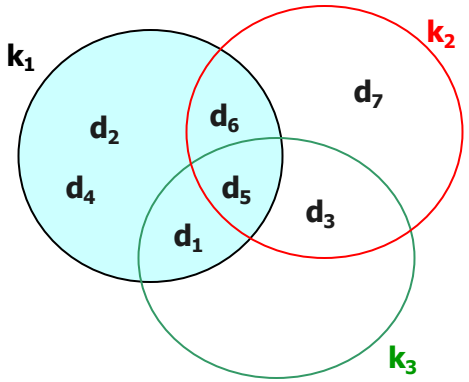


Query weights

Query: $k_3\ k_2\ k_3\ k_1\ k_2\ k_3$

	k_1	k_2	k_3	$q \bullet d_j$
d_1	1	0	1	4
d_2	1	0	0	1
d_3	0	1	1	5
d_4	1	0	0	1
d_5	1	1	1	6
d_6	1	1	0	3
d_7	0	1	0	2
q	1	2	3	

Vector Model Example 3



Document + query weights

Query: $k_3\ k_2\ k_3\ k_1\ k_2\ k_3$

	k_1	k_2	k_3	$q \bullet d_j$
d_1	2	0	1	5
d_2	1	0	0	1
d_3	0	1	3	11
d_4	2	0	0	2
d_5	1	2	4	17
d_6	1	2	0	5
d_7	0	5	0	10
q	1	2	3	

Summary: Document Vector Model

- Document vector model captures both documents and queries as vectors, assuming each word is a different, orthogonal dimension
- We use the cosine of the angle between the vectors
  - More similar gets a higher score, closer to 1
  - Word frequencies in doc + query will matter!
- Issue: words and frequencies don't all have equal importance

Plan for today

- Web services ✓
- Information retrieval ✓
  - Basics ✓
  - Precision and recall ✓
  - Taxonomy of IR models ✓
- Classic IR models
  - Boolean model ✓
  - Vector model ✓
  - TF/IDF NEXT
- HITS and PageRank

Caveat: Words Aren't All Equal

- To this point we assumed each word was equally significant
- Let's consider the *value* of terms: TF*IDF

An example

The University of Pennsylvania

- What would be a good match for this query?

Weights in the vector model

$$sim(d_j, q) = \cos\theta = \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$$

- How do we compute the weights w_{ij} and w_{iq}?
- A good weight must take into account two effects:
  - quantification of intra-document contents (similarity)
    - tf factor, the term frequency within a document
  - quantification of inter-documents separation (dissimilarity)
    - idf factor, the inverse document frequency
- w_{ij} = tf(i,j) * idf(i)

Common TF and IDF preprocessing

- Let:

  N be the total number of docs in the collection

  n_i be the number of docs which contain k_i

  freq(i,j) raw frequency of k_i within d_j

- A normalized tf factor is given by

  $$f(i,j) = a + (1-a) * freq(i,j) / max(freq(l,j))$$

  where the maximum is computed over all terms which occur within the document d_j. (a is usually set to 0.4 or 0.5)

- The idf factor is computed as

  $$idf(i) = \log (N / n_i)$$

  the log is used to make the values of tf and idf comparable. It can also be interpreted as the amount of information associated with the term k_i

Putting it all together: Scoring

Specific document being scored

Computed across all documents

	Corpus		Docum.		Query		
Term	df	idf	tf	$w_{t,d}$	tf	$w_{t,q}$	product
auto	5000	2.3	1	0.41	0	0	0
best	50000	1.3	0	0	1	1.3	0
car	10000	2.0	1	0.41	1	2.0	0.82
insurance	1000	3.0	2	0.82	1	3.0	2.46

N=1000000

From: An Introduction to Information Retrieval, Cambridge UP

- Example: Query is for 'best car insurance'
  - Document: Use tf weighting without idf, but with Euclidean normalization
  - Query: Use idf
  - Net score for this document is sum of $w_{t,d}$*$w_{t,q}$: 0.41*0 + 0*1.3 + 0.41*2.0 + 0.82*3.0 = 3.28

Stop words

- What do we do about very common words ('the', 'of', 'is', 'may', 'a', ...)?
  - Do not appear to be very useful in general
  - ... though they may be in phrase searches
    - "President of the United States"
    - "To be or not to be"
- We can use a stop list to remove these entirely
  - Typically small (200-300 terms or less)
  - Ongoing trend is towards even smaller lists, or even no list at all (web search engines generally do not use them)

Stemming and lemmatization

- What if the document contains many similar word forms?
  - View, viewing, viewer, viewed, views, viewable, ...
  - Democracy, democratization, ...
- Can use stemming to 'normalize' words
  - A somewhat rough heuristic; chops off ends of words etc.
  - Most common algorithm: Porter stemmer
    - http://tartarus.org/~martin/PorterStemmer/
  - Far from perfect
    - Example: Operate, operating, operates, operation, operative, operatives, operational, ... are all stemmed to 'oper'
- Better: Use NLP tools (lemmatizer)
  - May use a vocabulary (e.g., 'are/is/were' -> 'be')

Example: Porter stemmer

> Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation

> such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret

- Above: Simple example of stemmer output
  - Example rules: SSES→SS, IES→I, …
  - Entire algorithm is fairly long and complex

From: "An introduction to Information Retrieval", Cambridge University Press, page 34

Summary: Vector Model

- The best term-weighting schemes use tf-idf weights:

$$w_{i,j} = f(i,j) * \log(\frac{N}{n_i})$$

- For the query term weights, a suggestion is

$$w_{i,q} = 0.5 + \left[0.5 * \frac{freq(i,q)}{\max_l f\ req(l,q)}\right] * \log(\frac{N}{n_i})$$

- This model is very good in practice:
  - tf-idf works well with general collections
  - Simple and fast to compute
  - Vector model is usually as good as the known ranking alternatives

Pros & Cons of the vector model

- Advantages:
  - Term-weighting improves quality of the answer set
  - Partial matching allows retrieval of docs that approximate the query conditions
  - Cosine ranking formula sorts documents according to degree of similarity to the query

- Disadvantages:
  - Assumes independence of index terms; not clear if this is a good or bad assumption

Comparison of classic models

- Boolean model does not provide for partial matches and is considered to be the weakest classic model
- Some experiments indicate that the vector model outperforms the third alternative, the probabilistic model, in general
  - IR research has focused on improving probabilistic models for some time – but these haven't made their way to Web search
- Generally, we use a variation of the vector model in most text search systems

Further reading

- "An introduction to Information Retrieval"
  - Christopher D. Manning, Prabhakar Rhagavan, Hinrich Schuetze; Cambridge University Press, 2009
  - Available as a PDF from: http://nlp.stanford.edu/IR-book/
- Contains more details on many topics covered in this lecture
  - Examples: Scoring, tokenization, lemmatization, ...
- If you're the ranking expert in your final project team, you should have a look!
  - ... and possibly even if you're not (interesting!)

Web search before 1998

- Based on information retrieval
  - Boolean / vector model, etc.
  - Based purely on 'on-page' factors, i.e., the text of the page
- Results were not very good
  - Web doesn't have an editor to control quality
  - Web contains deliberately misleading information ($\rightarrow$SEO)
  - Great variety in types of information: Phone books, catalogs, technical reports, slide shows, ...
  - Many languages, partial descriptions, jargon, ...
- How to improve the results?

Plan for today

- Information retrieval ✓
- Classic IR models ✓
- HITS NEXT
  - Hubs and authorities
- PageRank
  - Iterative computation
  - Random-surfer model
  - Refinements: Sinks and Hogs
- Google

Goal: Find authoritative pages

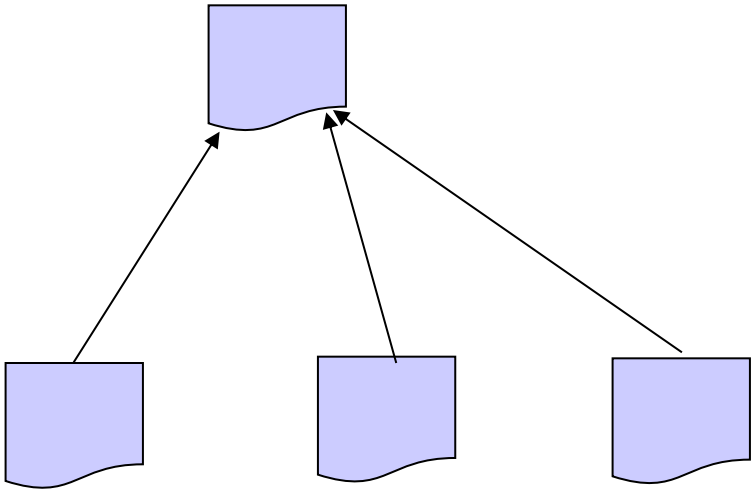
- Many queries are relatively broad
  - "cats", "harvard", "iphone", ...
- Consequence: Abundance of results
  - There may be thousands or even millions of pages that contain the search term, incl. personal homepages, rants, ...
  - IR-type ranking isn't enough; still way too much for a human user to digest
  - Need to further refine the ranking!
- Idea: Look for the most authoritative pages
  - But how do we tell which pages these are?
  - Problem: No endogenous measure of authoritativeness → Hard to tell just by looking at the page.
  - Need some 'off-page' factors

Idea: Use the link structure

- Hyperlinks encode a considerable amount of human judgment
- What does it mean when a web page links another web page?
  - Intra-domain links: Often created primarily for navigation
  - Inter-domain links: Confer some measure of authority
- So, can we simply boost the rank of pages with lots of inbound links?

Relevance ≠ Popularity!

"A-Team" page

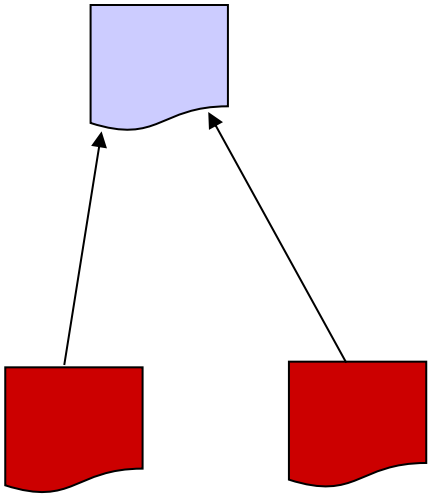


Failed Hollywood remakes of TV shows page

Mr. T's fan page

1980s TV Shows page

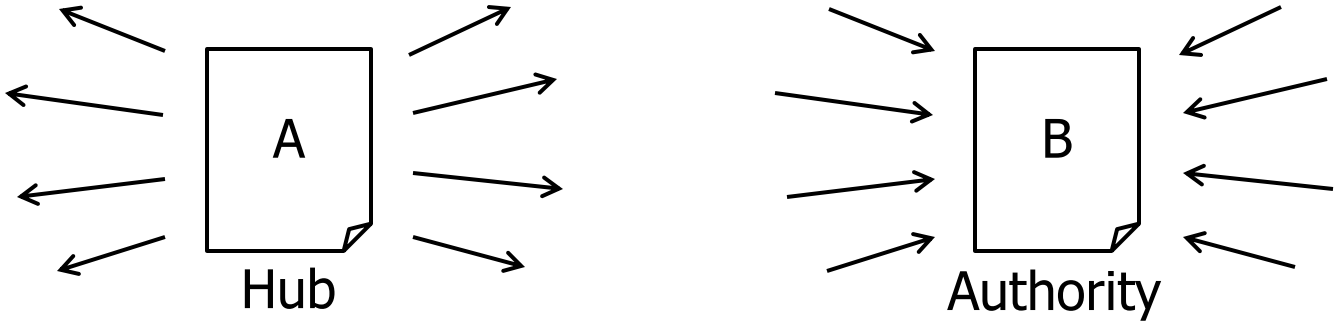
Team Sports - Wikipedia



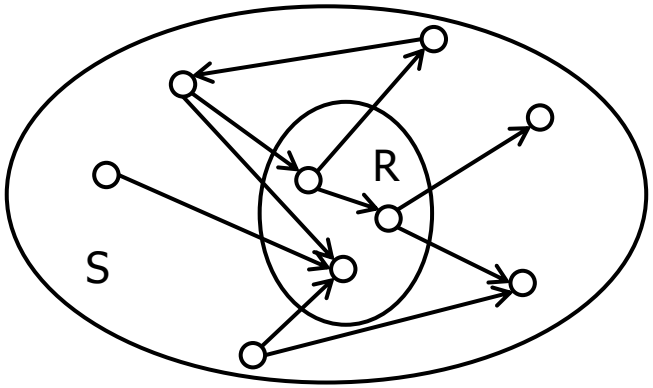
Yahoo Directory

Dictionary

Hubs and authorities



- Idea: Give more weight to links from hub pages that point to lots of other authorities
- Mutually reinforcing relationship:
  - A good hub is one that points to many good authorities
  - A good authority is one that is pointed to by many good hubs



HITS

- Algorithm for a query Q:
  1. Start with a root set R, e.g., the t highest-ranked pages from the IR-style ranking for Q
  2. For each $r \in R$, add all the pages r points to, and up to d pages that point to r. Call the resulting set S.
  3. Assign each page $p \in S$ an authority weight x^p and a hub weight y^p; initially, set all weights to be equal and sum to 1
  4. For each $p \in S$, compute new weights x^p and y^p as follows:
     - New x^p := Sum of all y^q such that $q \rightarrow p$ is an interdomain link
     - New y^p := Sum of all x^q such that $p \rightarrow q$ is an interdomain link
  5. Normalize the new weights such that both the sum of all the x^p and the sum of all the y^p are 1
  6. Repeat from step 4 until a fixpoint is reached
     - If A is adjacency matrix, fixpoints are principal eigenvectors of A^TA and AA^T, respectively

Recap: HITS

- Improves the ranking based on link structure
  - Intuition: Links confer some measure of authority
  - Overall ranking is a combination of IR ranking and this
- Based on concept of hubs and authorities
  - Hub: Points to many good authorities
  - Authority: Is pointed to by many good hubs
  - Iterative algorithm to assign hub/authority scores
- Query-specific
  - No notion of 'absolute quality' of a page; ranking needs to be computed for each new query
- Not very stable: small changes to graph result in large changes to scores
- Also somewhat inefficient