

CS 429

Information Retrieval

Final Exam Review

Jeremy E. Thompson

How to Prep

- Similar to mid term prep
- Lecture notes:
 - All the important topics have been dealt in the lecture notes
 - Go through class exercises
- Textbook:
 - Go through the relevant chapters in the textbook
 - Check out relevant textbook problems
- Additional materials
 - Go through quizzes and assignments
- No questions on python syntax

Relevance Feedback

Relevance Feedback

- *Problem*: Possible **disconnect** between information need and query leading to bad results
- *Idea*: Use user feedback on the utility/relevance of results to automatically **refine** and/or **reformulate** query
- User feedback on relevance of docs in initial set of results
 - User issues a query
 - User **marks** some results as relevant/non-relevant
 - System computes a **better** representation of the information need based on feedback
 - New results have (hopefully) **better** recall
 - Relevance feedback can go through **one or more** iterations

Relevance feedback

- We will use *ad hoc retrieval* to refer to regular retrieval **without relevance feedback**
- We now look at **two examples** of relevance feedback that highlight different aspects

Example: Initial query/results

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

Expanded query after relevance feedback

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

Initial query: *New space satellite applications*

Results for expanded query

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

Initial query: *New space satellite applications*

Key concept: Centroid

- “**Move**” the query towards the “**cluster**” of relevant documents
- The centroid is the center of mass of a set of points
- Recall that we represent documents as points in a high-dimensional space

- *Definition:* Centroid

$$\vec{\mu}(C) = \frac{1}{|C|} \sum_{d \in C} \vec{d}$$

- where **C** is a set of documents and \vec{d} is a document vector

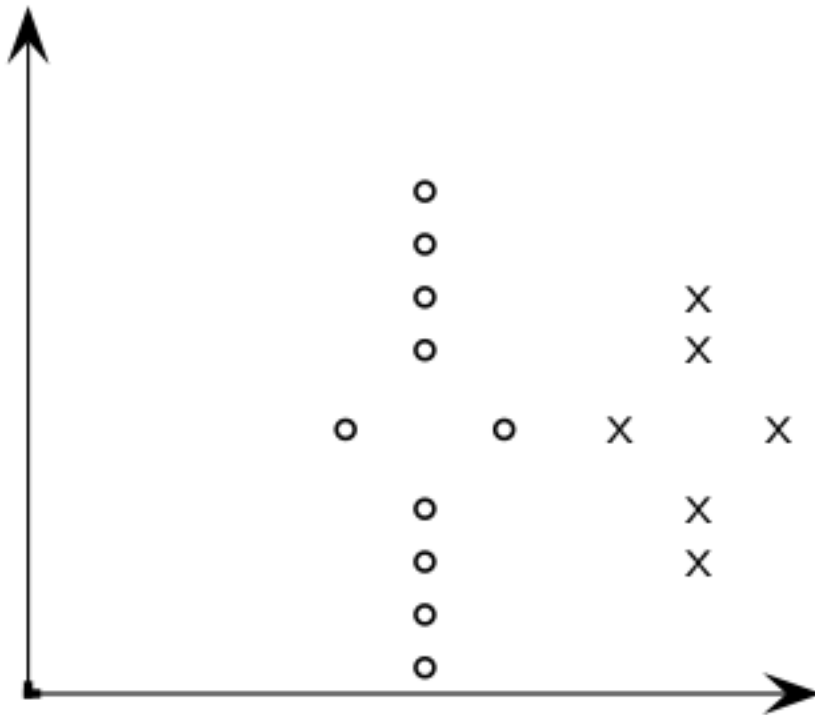
Rocchio' algorithm

- Rocchio' algorithm uses the vector space model
- Rocchio' chooses the query \vec{q}_{opt} that maximizes

$$\vec{q}_{opt} = \arg \max_{\vec{q}} [\text{sim}(\vec{q}, \mu(D_r)) - \text{sim}(\vec{q}, \mu(D_{nr}))]$$

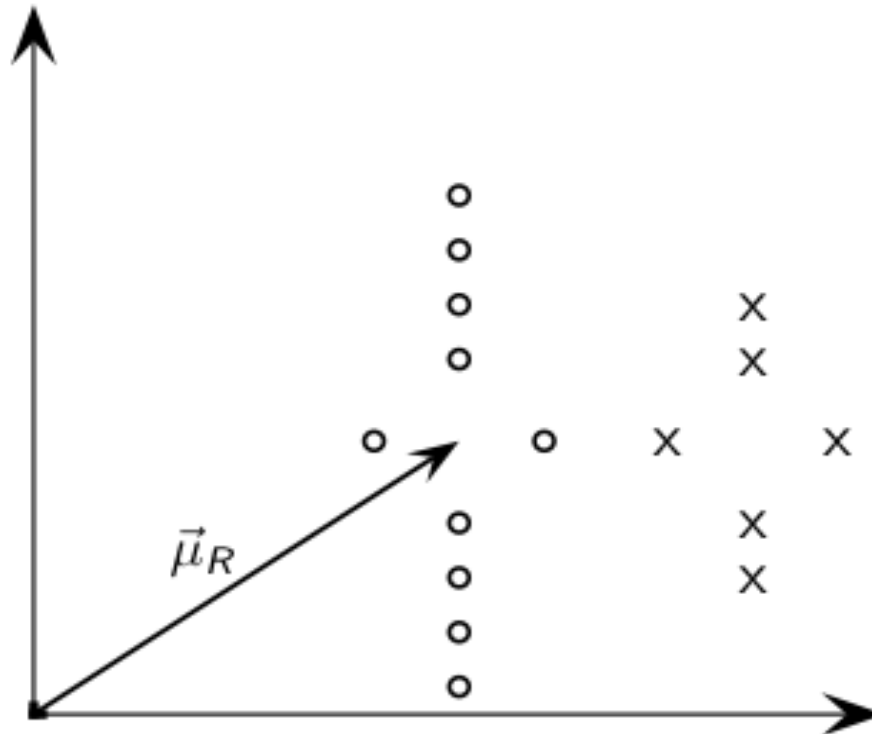
- D_r : set of **relevant** docs; D_{nr} : set of **nonrelevant** docs
- Intent: \vec{q}_{opt} is the vector that **separates** relevant and nonrelevant docs maximally

Illustration: Compute Rocchio' vector



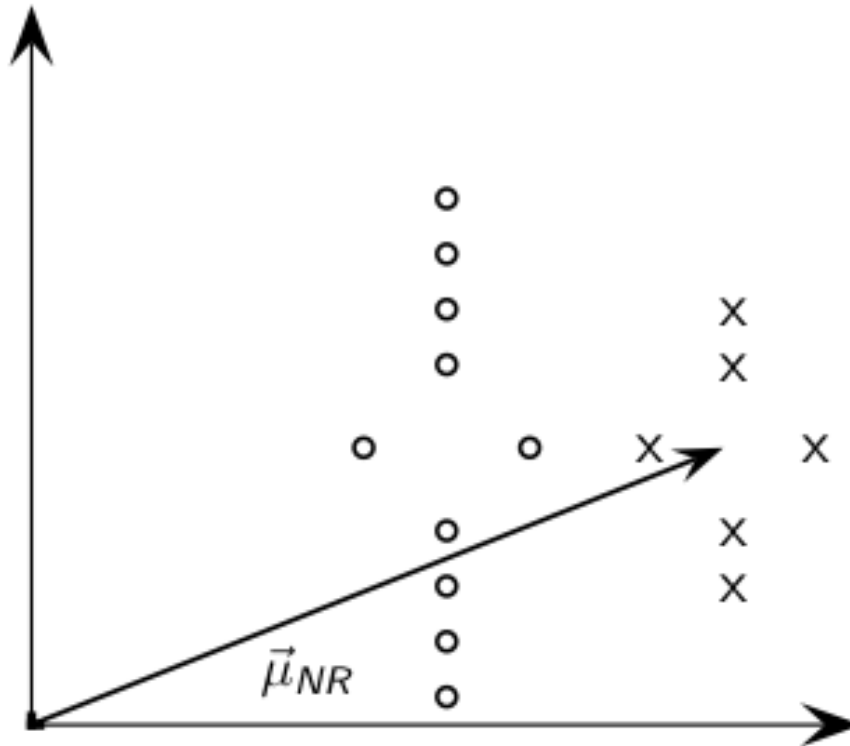
circles: relevant documents, Xs: nonrelevant documents

Rocchio' illustrated



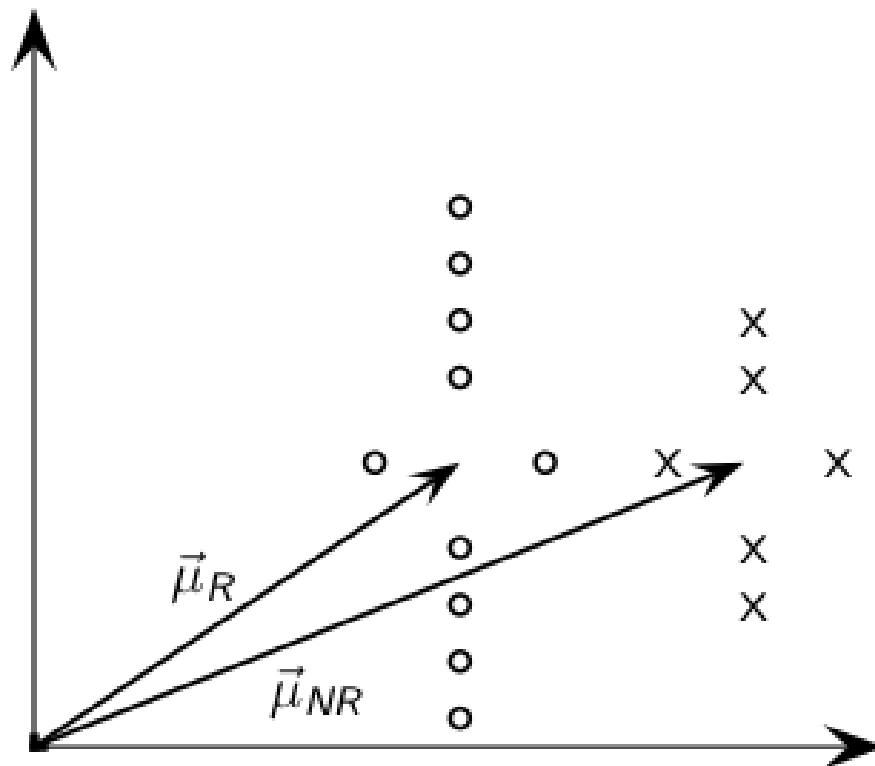
$\vec{\mu}_R$: centroid of relevant documents

Rocchio' illustrated

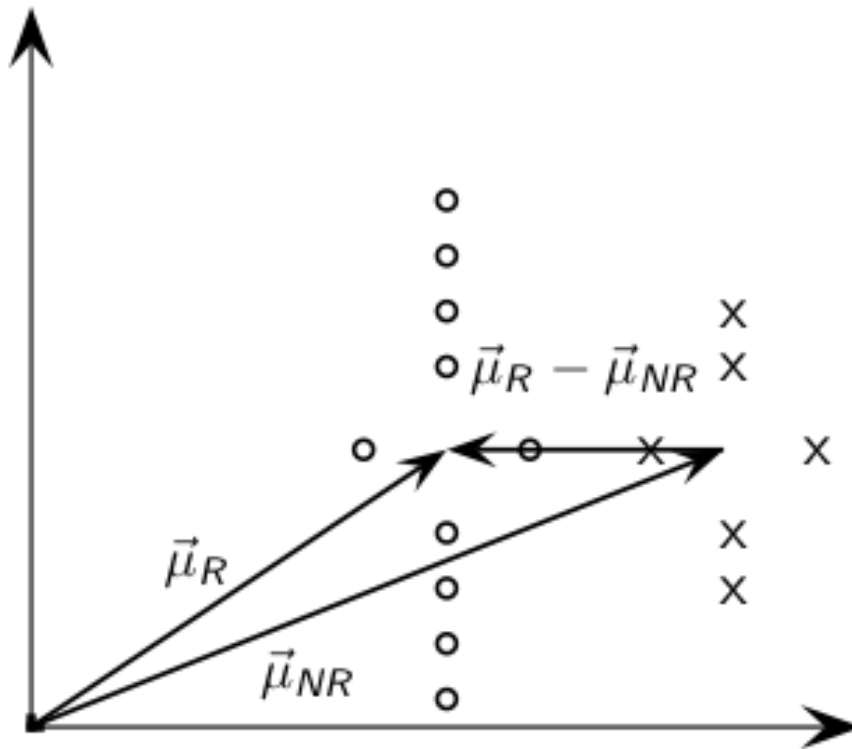


$\vec{\mu}_{NR}$: centroid of **nonrelevant** documents.

Rocchio' illustrated

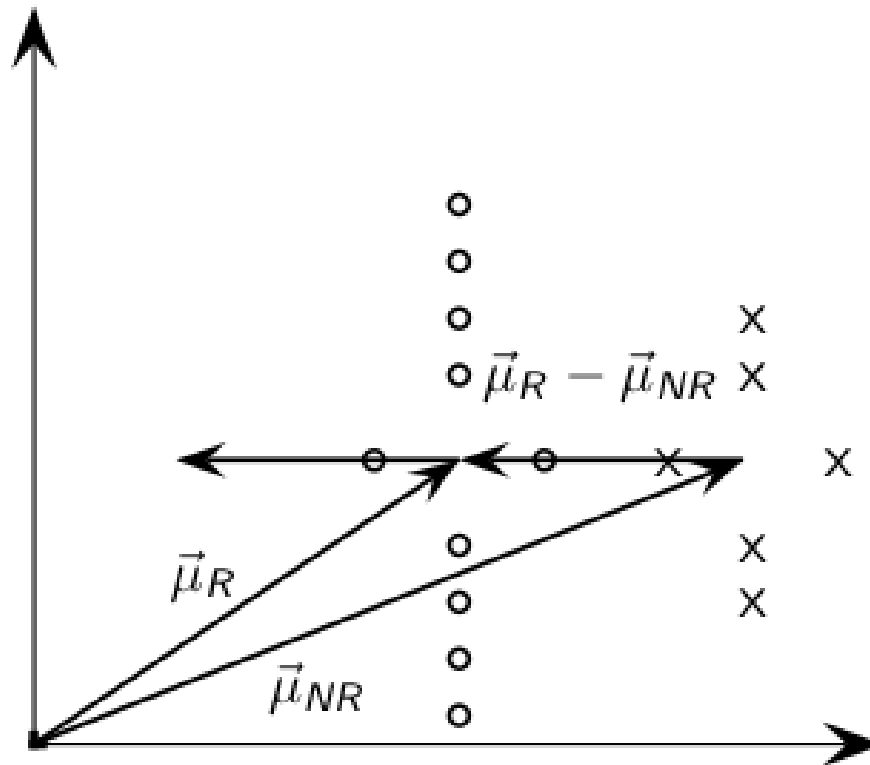


Rocchio' illustrated



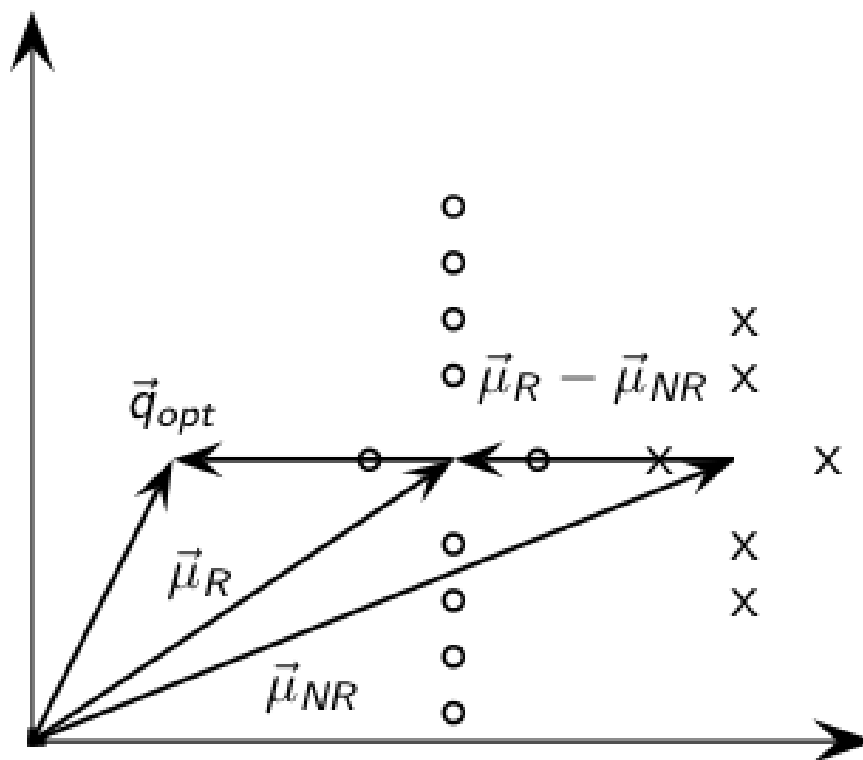
$\vec{\mu}_R - \vec{\mu}_{NR}$: difference vector

Rocchio' illustrated



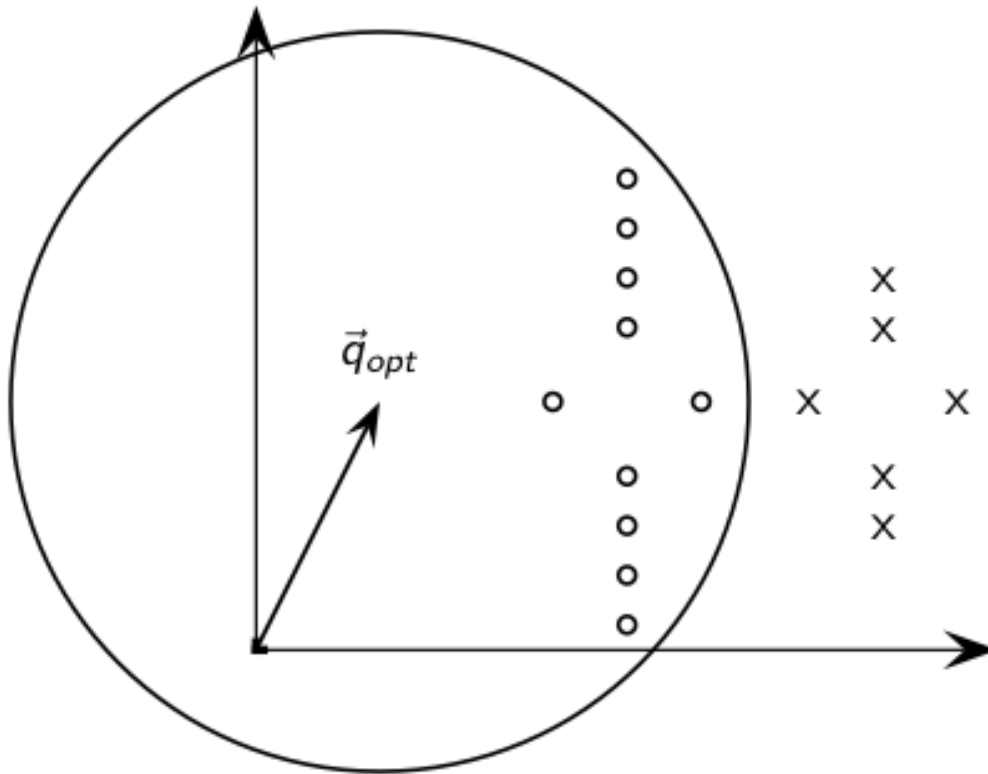
Add difference vector to $\vec{\mu}_R$...

Rocchio' illustrated



... to get \vec{q}_{opt}

Rocchio' illustrated



\vec{q}_{opt} separates relevant / nonrelevant perfectly

Rocchio' algorithm

- The **optimal query vector** is:

$$\begin{aligned}\vec{q}_{opt} &= \mu(D_r) + [\mu(D_r) - \mu(D_{nr})] \\ &= \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j + \left[\frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j \right]\end{aligned}$$

- We move the centroid of the relevant documents by the difference between the two centroids.

Rocchio 1971 Algorithm (SMART)

- Used in practice:

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

- D_r = set of known relevant doc vectors
- D_{nr} = set of known irrelevant doc vectors
- q_m = modified query vector; q_0 = original query vector; α, β, γ : weights (hand-chosen or set empirically)
- New query moves toward relevant documents and away from irrelevant documents

Exercise



- Construct the document vectors for

D1: good movie trailer

D2: shown trailer with good actor

D3: unseen movie

Given **dictionary**: {movie, trailer and good}, **query** Q1: *movie trailer*

Document Vectors: $D1=[1,1,1]$ $D2=[0,1,1]$ $D3=[1,0,0]$

$Q1=[1,1,0]$

Relevance feedback: $D_r=\{D1,D2\}$ $D_{nr}=\{D3\}$

Exercise cont.

$D1=[1,1,1]$ $D2=[0,1,1]$ $D3=[1,0,0]$ $Q1=[1,1,0]$ $D_r=\{D1,D2\}$
 $D_{nr}=\{D3\}$

Given $\alpha=1$, $\beta=0.75$, and $\gamma=0.15$

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

$$\vec{q}_1 = [1,1,0] + \frac{0.75}{2} [1,2,2] - \frac{0.15}{1} [1,0,0]$$

$$\vec{q}_1 = [1,1,0] + [0.375,0.75,0.75] - [0.15,0,0]$$

$$\vec{q}_1 = [1.390,1.75,0.75]$$

Exercise cont.

- **New query**
 - movie 1.39
 - Trailer 1.75
 - Good 0.75
- Some weights in query vector can go negative
 - **Negative** term weights are **ignored** (set to 0)

Relevance Feedback: Problems

- Relevance feedback is **expensive**
 - Relevance feedback creates **long modified queries**
 - Long queries are expensive to process
- Users are **reluctant** to provide explicit feedback
 - Most search engines do not have explicit feedback
- It's often **hard to understand** why a particular document was retrieved after applying relevance feedback

Pseudo relevance feedback

- Pseudo-relevance feedback **automates** the “manual” part of true relevance feedback
- Pseudo-relevance algorithm:
 - Retrieve a ranked list of hits for the user’s query
 - **Assume** that the top k documents are relevant
 - Do relevance feedback (e.g., Rocchio)
- Works very well on average
- But **can go horribly wrong** for some queries
- Several iterations can cause query drift

Relevance Feedback: Summary

- Relevance feedback has been shown to be very effective at **improving relevance** of results
 - Requires enough judged documents, otherwise it's unstable (≥ 5 recommended)
 - Requires queries for which the **set of relevant documents** is medium to large
- Full relevance feedback is **painful** for the user
- Full relevance feedback is **expensive**
- Other types of interactive retrieval may improve relevance by as much with less work

Ranked Retrieval

Vector Space model, Probabilistic IR and Language model

Ranked retrieval

- Thus far, our queries have all been Boolean
- Documents either match query terms or don't
- Good for expert users with precise understanding of their needs and the collection
- Also good for applications: Applications can easily consume 1000s of results.
- Not good for the majority of users
- Most users don't want to wade through 1000s of results.
 - Example: web search

Documents as vectors

- Each document is now represented as a real-valued vector of td-idf weights $\in \mathbb{R}^{|V|}$
 - So we have a $|V|$ -dimensional vector space
 - Terms are axes of the space
 - Documents are points or vectors in this space
 - Very high-dimensional: tens of millions of dimensions when you apply this to a web search engine
 - These are very sparse vectors - most entries are zero.

Term frequency tf

- Term frequency $tf_{t,d}$ of term t in document d is the number of times that t occurs in d
 - We want to use tf when computing query-document match scores. But how?
- Raw term frequency is not what we want:
- A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term
- But not 10 times more relevant
- Relevance does not increase proportionally with term frequency

idf weight

- df_t is the document frequency for term t
 - No. of documents that contain term t
- df_t is an inverse measure of the informativeness of t
- $df_t \leq N$
- idf_t : inverse document frequency of t

$$idf_t = \log_{10} \left(\frac{N}{df_t} \right)$$

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.
 - $tf-idf_{t,d} = \log(1 + tf_{t,d}) \times \log\left(\frac{N}{df_t}\right)$
- Best known weighting scheme in information retrieval
 - Increases with the number of occurrences within a document
 - Increases with the rarity of the term in the collection
 - Lowest when the term occurs in all documents
- Consider documents as a vector of terms

$$score(q, d) = \sum_{t \in q \cap d} tf_{t,d} \times idf_{t,d}$$

Cosine (query, document)

Dot product

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\|_2 \|\vec{d}\|_2} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i is the tf-idf weight of term i in the query
- d_i is the tf-idf weight of term i in the document
- V is the set of query terms
- $\cos(q, d)$ is the cosine similarity of q and d ... or, equivalently, the cosine of the angle between q and d .

Top K search

- Indexing
 - calculate weight (e.g. TF-IDF) vectors for all documents
- Query processing
 - calculate weight vector for query
- Calculate similarity (e.g. cosine) between query and all documents
- Sort by similarity and return top K

Precision and Recall

- **Precision (P)**: fraction of retrieved docs that are relevant

$$P = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = P(\text{relevant} | \text{retrieved})$$

- **Recall (R)**: fraction of relevant docs that are retrieved

$$R = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = P(\text{relevant} | \text{retrieved})$$

F measure

- F measure allows us to trade off precision against recall.
- $F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha) \frac{1}{R}}$ where $\alpha \in [0, 1]$
- $\Rightarrow \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$ where $\beta^2 = \frac{1-\alpha}{\alpha}$, $\beta \in [0, \infty]$
 - Weighted Harmonic mean: reciprocal of the weighted arithmetic mean of the reciprocals.
- Balanced F Measure with $\alpha = 0.5 \Rightarrow \beta = 1$
- What value range of β weights recall higher than precision?

Probabilistic IR Models at a Glance

- Use probability to model the relevance of documents
- Classical probabilistic retrieval model
 - Probability ranking principle
 - Binary Independence Model, BestMatch25 (Okapi)
- Probabilistic methods are one of the oldest topics in IR but still active area

Probability Ranking Principle (PRP)

- Let d be a document in the collection and q be a given query.
- $R_{d,q}$ is a random dichotomous (two states) variable, such that
 - $R_{d,q} = 1$ if document d is relevant w.r.t query q
 - $R_{d,q} = 0$ otherwise (if document d is non-relevant w.r.t query q)

$$P(R = 1|d, q) = \frac{P(d|R = 1, q)P(R = 1|q)}{P(d|q)}$$

$$P(R = 0|d, q) = \frac{P(d|R = 0, q)P(R = 0|q)}{P(d|q)}$$

Binary Independence Model (BIM)

- $P(R|d, q)$ is modelled using term incidence vectors as $P(R|\vec{x}, \vec{q})$

$$P(R = 1|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 1, \vec{q})P(R = 1|\vec{q})}{P(\vec{x}|\vec{q})}$$

$$P(R = 0|\vec{x}, \vec{q}) = \frac{P(\vec{x}|R = 0, \vec{q})P(R = 0|\vec{q})}{P(\vec{x}|\vec{q})}$$

- Where
 - $P(\vec{x}|R = 1, \vec{q})$: probability that if a relevant document is retrieved, then that document's representation is \vec{x}
 - $P(\vec{x}|R = 0, \vec{q})$: probability that if a non-relevant document is retrieved, then that document's representation is \vec{x}
 - $P(R = 1|\vec{q})$: prior probability of retrieving a relevant document for a query q
 - $P(R = 0|\vec{q})$: prior probability of retrieving a nonrelevant document for a query q

Deriving a Ranking Function for Query Terms

- Rank documents by their odds of relevance

$$\begin{aligned} O(R|\vec{x}, \vec{q}) &= \frac{P(R = 1|\vec{x}, \vec{q})}{P(R = 0|\vec{x}, \vec{q})} = \frac{\frac{P(R=1|\vec{q})P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q})P(\vec{x}|R=0,\vec{q})}{P(\vec{x}|\vec{q})}} \\ &= \frac{P(R = 1|\vec{q})}{P(R = 0|\vec{q})} \cdot \frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})} \end{aligned}$$

- $\frac{P(R=1|\vec{q})}{P(R=0|\vec{q})}$ is a constant for all document given a query

Deriving a Ranking Function for Query Terms

- Independence assumption: Presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):
- Therefore,

$$\frac{P(\vec{x}|R = 1, \vec{q})}{P(\vec{x}|R = 0, \vec{q})} = \prod_{t=1}^M \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}$$

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^M \frac{P(x_t|R = 1, \vec{q})}{P(x_t|R = 0, \vec{q})}$$

Deriving a Ranking Function for Query Terms

- Since each x_t is either 0 or 1, we can separate the terms to give:

$$O(R|\vec{x}, \vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t = 1|R = 1, \vec{q})}{P(x_t = 1|R = 0, \vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t = 0|R = 1, \vec{q})}{P(x_t = 0|R = 0, \vec{q})}$$

- $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in relevant document
 - $u_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a non-relevant document
- Visualise as contingency table:

document		relevant ($R = 1$)	nonrelevant ($R = 0$)
Term present	$x_t = 1$	p_t	u_t
Term absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

Probability Estimates in Practice

$$c_t = \log \frac{p_t(1 - u_t)}{u_t(1 - p_t)} = \log \frac{p_t}{(1 - p_t)} + \log \frac{1 - u_t}{u_t}$$

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for non-relevant documents by statistics from the whole collection
- Hence, u_t (the probability of term occurrence in non-relevant documents for a query) is df_t/N and
- $\log[(1 - u_t)/u_t] = \log[(N - df_t)/df_t] \approx \log N/df_t$
- What about $\log \frac{p_t}{(1 - p_t)}$?

Probability Estimates in Practice cont.

1. Use **relevance feedback weighting** if available

OR

2. Set p_t as constant

E.g., **assume** that p_t is same for all terms x_t in the query and that $p_t = 0.5$

- **Each term is equally likely** to occur in a relevant document, and so the p_t and $(1 - p_t)$ factors **cancel out** in the expression for RSV
- Weak estimate, but doesn't disagree with **expectation** that query terms appear in **many but not all** relevant documents
- Combined with earlier approximation for u_t document ranking determined by which query terms occur in documents **scaled by their idf weighting**
- For short documents (titles or abstracts) in one-pass retrieval situations, this estimate can be quite satisfactory

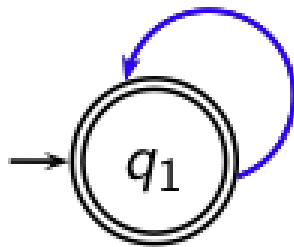
What is a language model?

- We can view a finite state automaton as a deterministic language model.



- What strings can it generate?
 - Generates - “I wish I wish I wish I wish . . . ”
- Can it generate the following?
 - “wish I wish”
 - “I wish I”
- Our basic model: each document was generated by a different automaton like this except that these automata are probabilistic.

Probabilistic Language Model



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

- This is a one-state probabilistic finite-state automaton – a unigram language model – and the state emission distribution for its one state q_1 . STOP is not a word, but a special symbol indicating that the automaton stops.
- What is the probability of generating the string “frog said that toad likes frog”?
- frog said that toad likes frog STOP

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02$$

$$= 0.00000000000048$$

STOP is not a word, but a special symbol indicating that the automaton stops.

A different language model for each document

language model of d_1				language model of d_2			
w	$P(w .)$	w	$P(w .)$	w	$P(w .)$	w	$P(w .)$
STOP	.2	toad	.01	STOP	.2	toad	.02
the	.2	said	.03	the	.15	said	.03
a	.1	likes	.02	a	.08	likes	.02
frog	.01	that	.04	frog	.01	that	.05
	

- Consider the string “frog said that toad likes frog”. What is the probability that it is generated by the models M_{d_1} and M_{d_2}
- $P(\text{string} | M_{d_1}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 0.00000000000048 = 4.8 \cdot 10^{-12}$
- $P(\text{string} | M_{d_2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 0.00000000000120 = 12 \cdot 10^{-12}$
- $P(\text{string} | M_{d_1}) < P(\text{string} | M_{d_2})$
- Thus, document d_2 is “more relevant” to the string “frog said that toad likes frog STOP” than d_1 is.

Query-Likelihood Model

- Each document is treated as (the basis for) a language model.
- Given a query q
- Our central IR problem: Rank documents based on their relevance (from Probabilistic IR), i.e. rank documents based on $P(d|q)$

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

- $P(q)$ is the same for all documents, so ignore
- $P(d)$ is the prior – often treated as the same for all d
 - We can give a prior to “high-quality” documents, e.g., those with high PageRank.
- $P(q|d)$ is the probability of q given d
- Rank documents according to relevance to q is equivalent to ranking according to $P(q|d)$

Query-Likelihood Model cont.

- What we need to do:
 - Step 1: Define the precise generative model we want to use
 - Step 2: Estimate parameters (different parameters for each document's model M_{d_i})
 - Step 3: Estimate the probability $P(q|M_{d_i})$ that document model M_{d_i} generated the query
 - Step 4: Present most likely document(s) to user
- Next: how do we compute $P(q|M_{d_i})$

How to compute $P(q|M_{d_i})$?


- Independence Assumption: We make the assumption that the events of the model generating the different terms are independent
- $P(q|M_d) = P(\langle t_1, \dots, t_k, \dots, t_{|q|} \rangle | M_d) = \prod_{1 \leq t \leq |q|} P(t_i | M_d)$
 - $|q|$: length of q ; t_k : the token occurring at position k in q)
- Missing piece: Where do the parameters $P(t|M_d)$. come from?
- Simplest estimation: Maximum Likelihood Estimates (MLEs)

How to compute $P(q|M_{d_i})$?

- $P(t|M_d) = \frac{tf_{t,d}}{|d|}$
($|d|$: length of d ; $tf_{t,d}$: # occurrences of t in d)
- $P(q|M_d) = \prod_{1 \leq t \leq |q|} P(t_i|M_d)$

Steps in LM Retrieval

- Steps:
 - Step 1: Define the precise generative model we want to use
 - Step 2: Estimate parameters (different parameters for each document's model)
 - Step 3: Apply smoothing to avoid zeros
 - Step 4: Apply to query and find document most likely to have generated the query
 - Step 5: Present most likely document(s) to user



Additional
Smoothing step

Smoothing: Mixture model

$$P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$$

- Mixes the probability from the document with the general collection frequency of the word
- Correctly setting λ is very important for good performance
- $P(t|M_c) = \frac{cf_t}{T}$
 - M_c : collection model
 - cf_t : the number of occurrences of term t in the collection
 - $T = \sum_t cf_t$: the total number of tokens in the collection

LMs vs. Vector Space Model

- LMs vs. vector space model: **commonalities**
 - Term frequency is used directly in LM
 - Probabilities are inherently “length-normalized”.
 - Mixing document and collection frequencies has an effect similar to idf.
- LMs vs. vector space model: **differences**
 - LMs: based on probability theory
 - Vector space: based on similarity, a geometric/ linear algebra notion
 - Collection frequency vs. document frequency
 - Details of term frequency, length normalization etc.

Questions to ponder (not comprehensive)

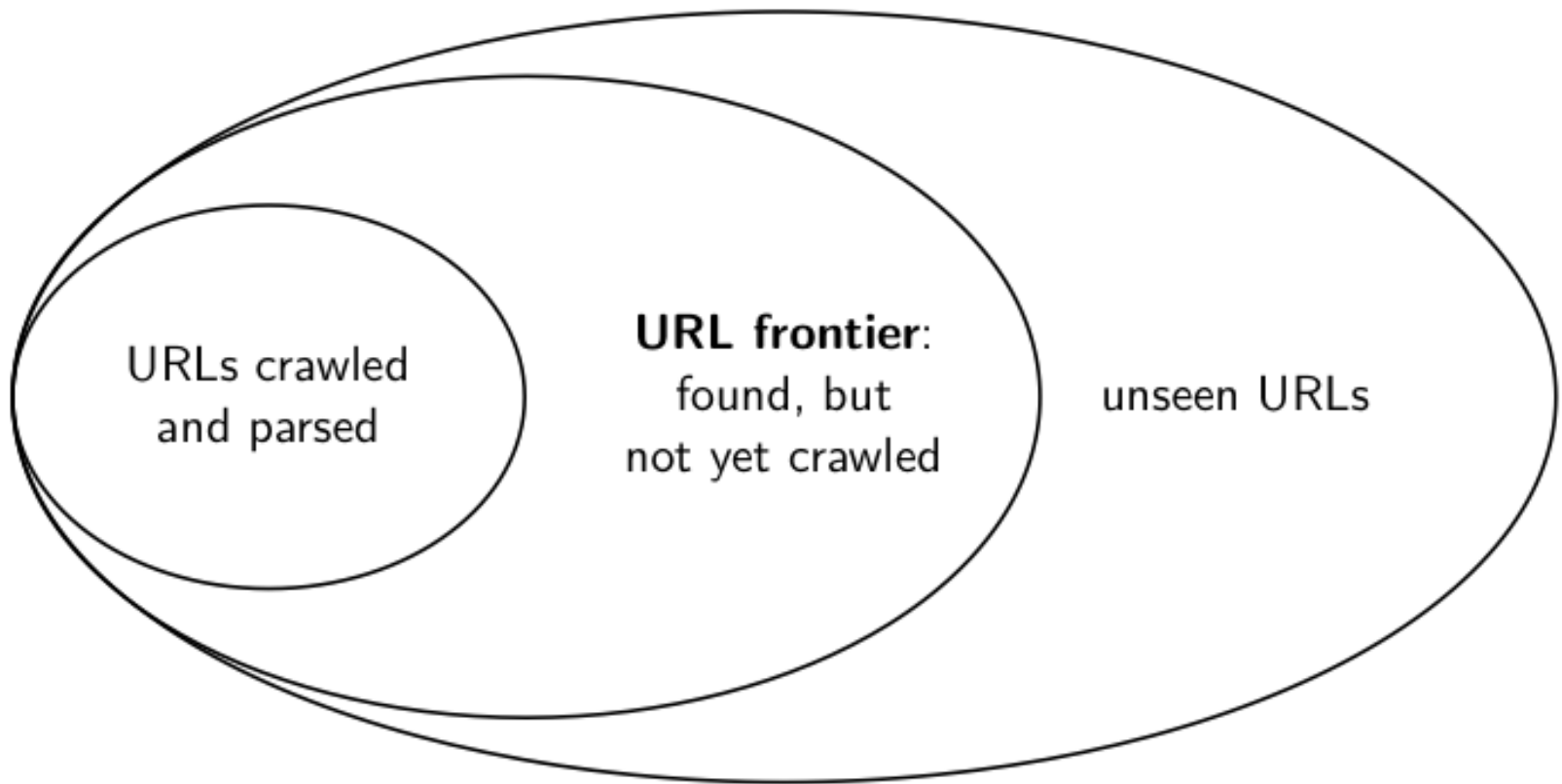
- What are the differences and similarities between the probabilistic IR, language and vector space models?
- What are the strengths and weaknesses for these models?
- How to calculate the parameters for the models?
- What is smoothing?

Link Analysis

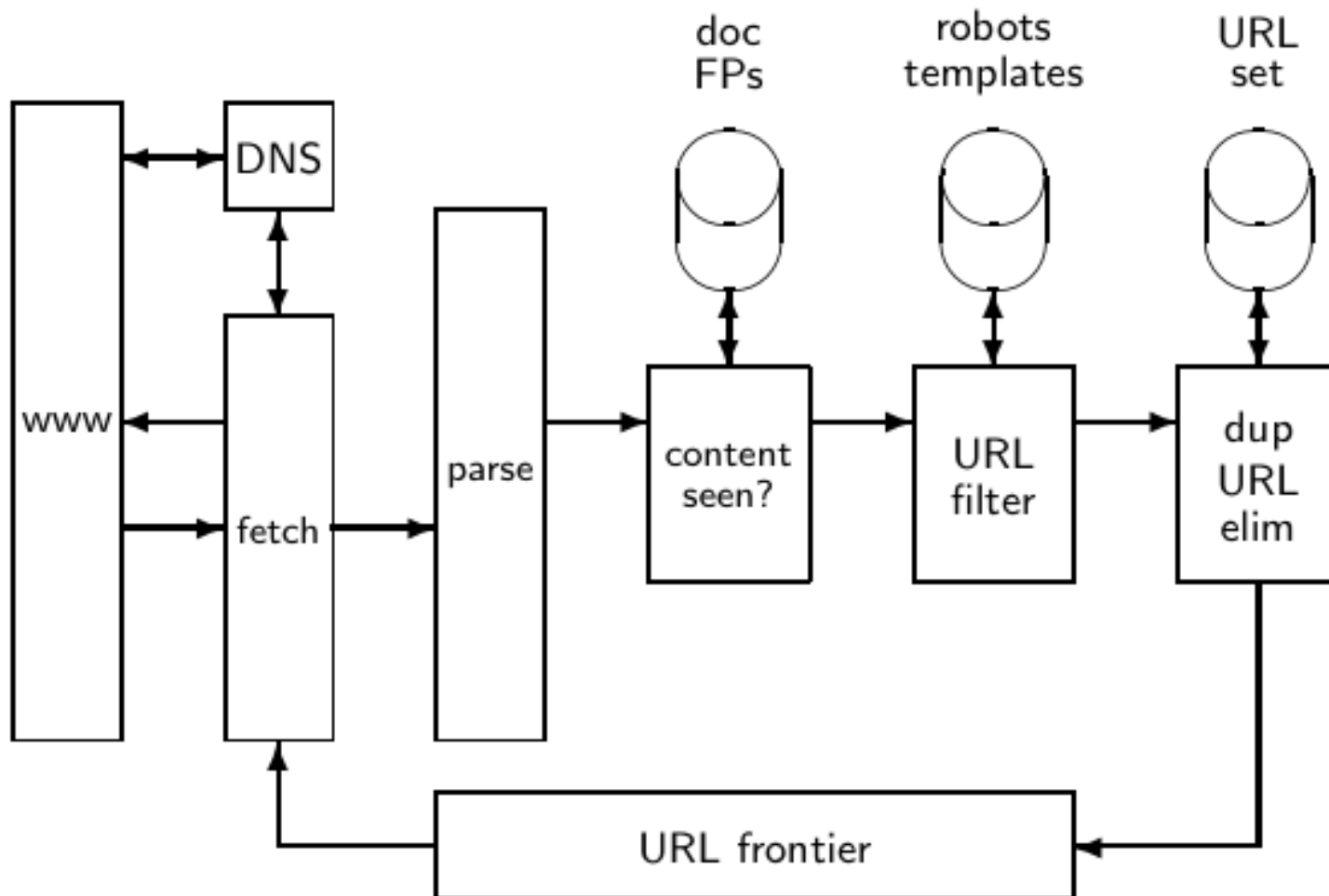
Web Crawler

- Finds and downloads web pages automatically
 - provides the collection for searching
- Challenges:
 - Web is huge and constantly growing
 - Difference with other IR
 - Web is not under the control of search engine providers
 - Getting the content of the documents is easier for many other IR systems.
 - E.g., indexing all files on your hard disk: just do a recursive descent on your file system
 - For web IR, getting the content of the documents takes longer . .
 - because of latency.
 - Web pages are constantly changing
 - Crawlers also used for other types of data

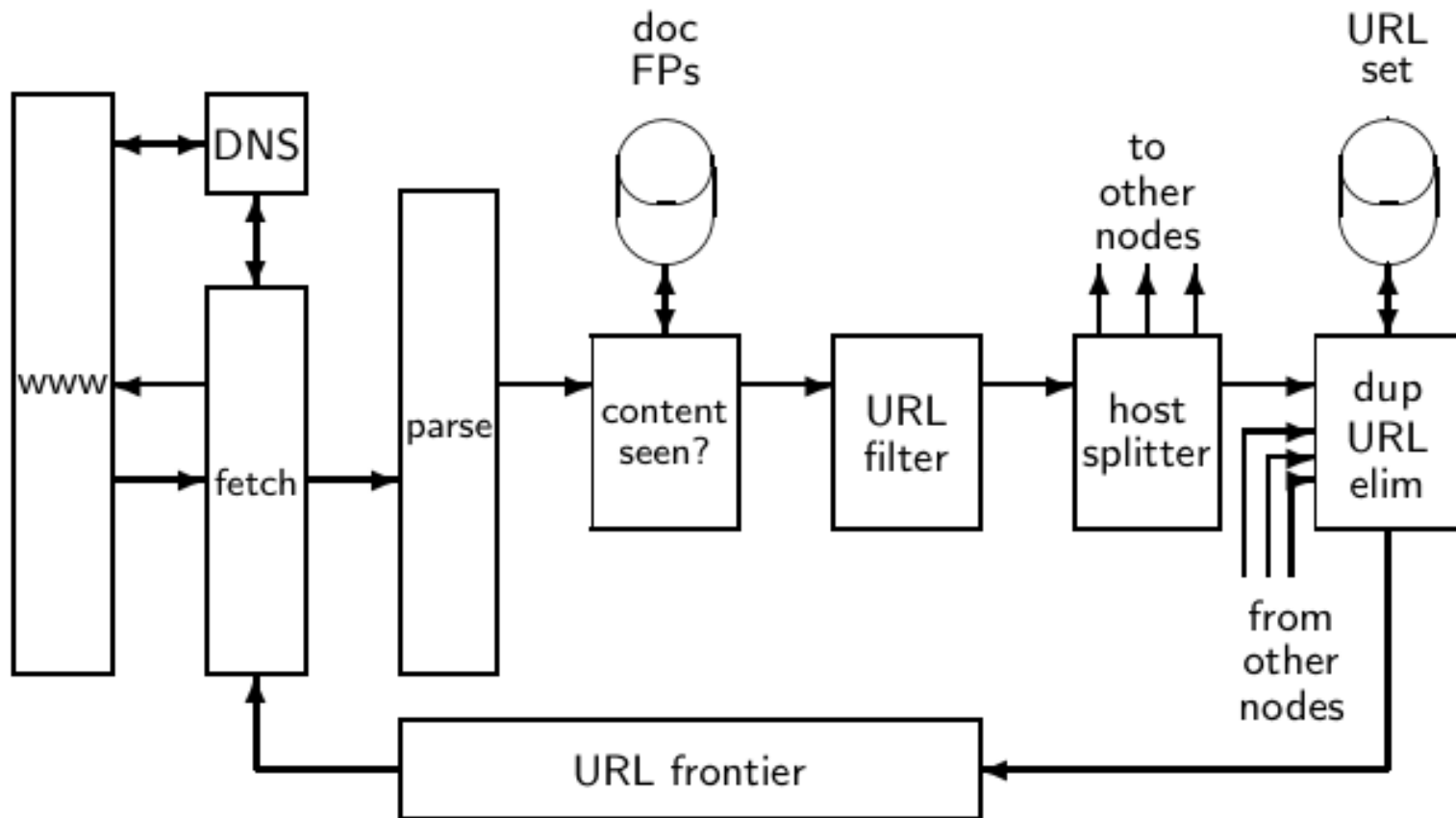
URL Frontier



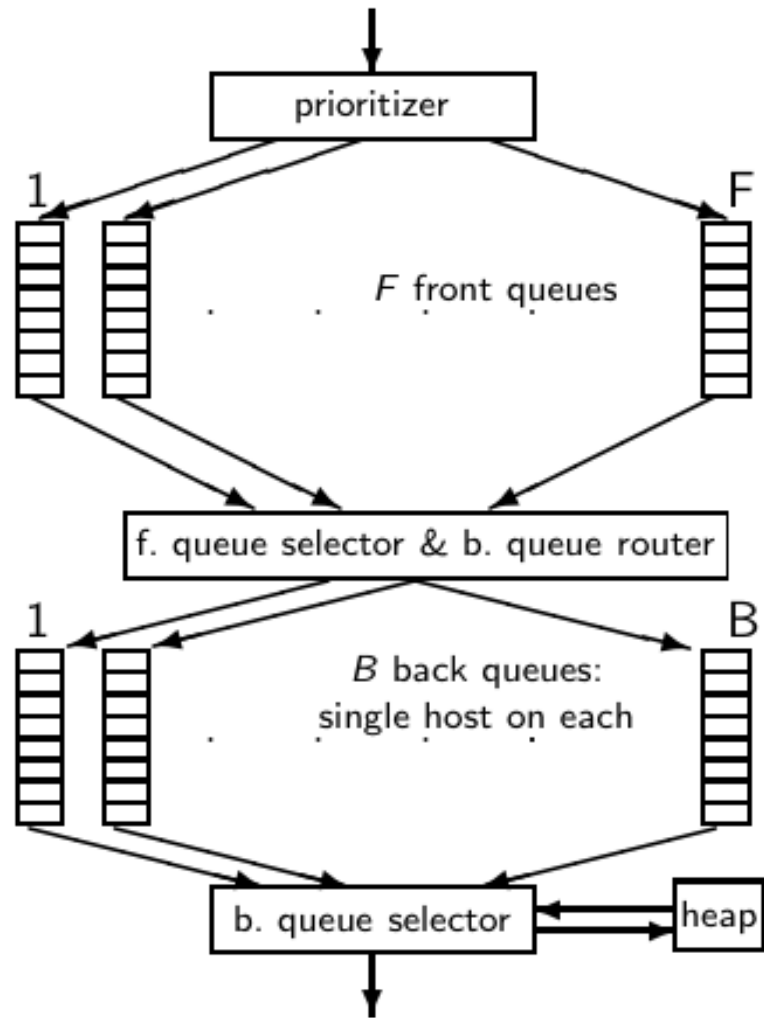
Basic crawl architecture



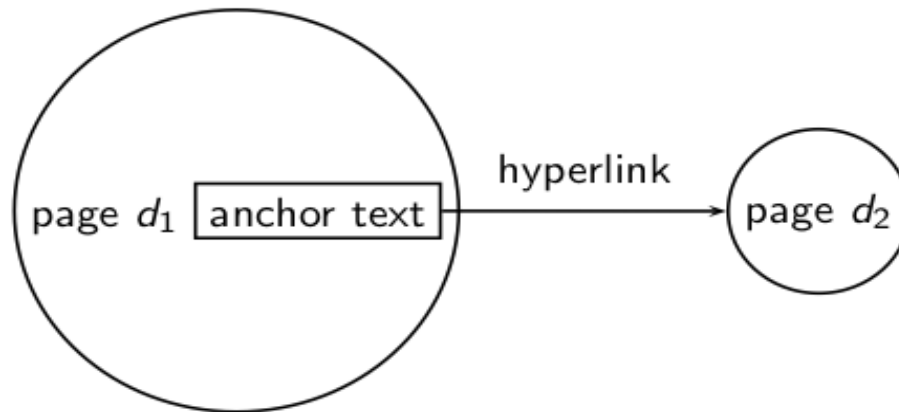
Distributed crawler



Mercator URL Frontier



Web as a Directed Graph



- Assumption 1: A hyperlink is a quality signal.
 - The hyperlink $d_1 \rightarrow d_2$ indicates that d_1 's author deems d_2
 - high-quality and relevant.
- Assumption 2: The anchor text describes the content of d_2 .
 - We use anchor text somewhat loosely here for: the text
 - surrounding the hyperlink .
 - Example: “You can find cheap cars here .”
 - Anchor text: “You can find cheap here”

Anchor text containing *IBM* pointing to www.ibm.com

www.nytimes.com: "IBM acquires Webify"

www.slashdot.org: "New IBM optical chip"

www.stanford.edu: "IBM faculty award recipients"



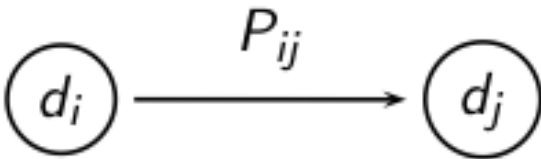
Diagram illustrating backlinks to www.ibm.com. Three sources are shown with dashed arrows pointing to the target URL:

- www.nytimes.com: "IBM acquires Webify"
- www.slashdot.org: "New IBM optical chip"
- www.stanford.edu: "IBM faculty award recipients"

The target URL, www.ibm.com, is enclosed in a rectangular box at the bottom.

Formalization of random walk: Markov chains

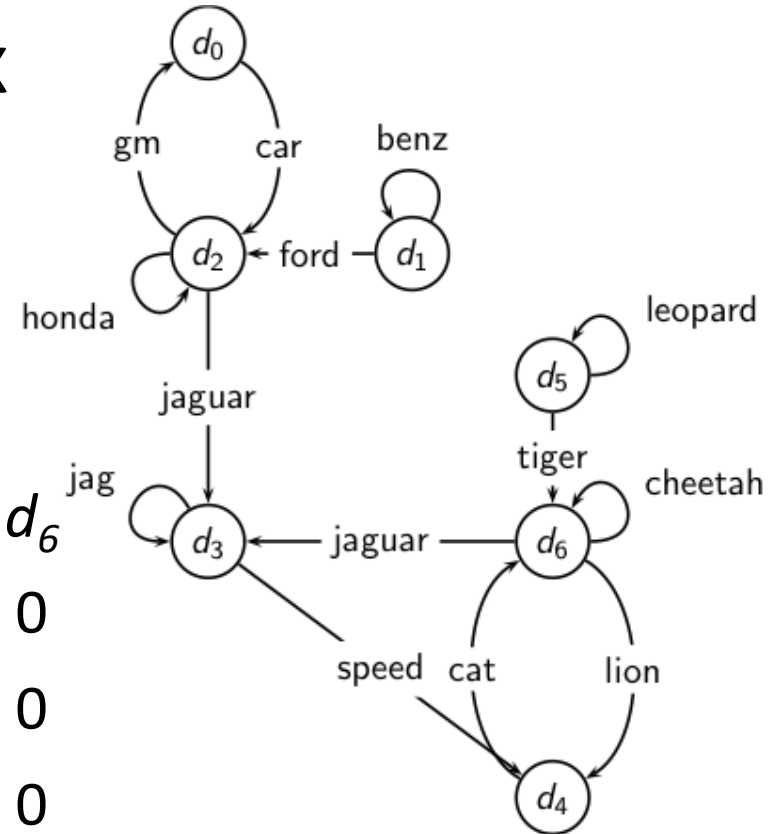
- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P .
 - state = page
- At each step, we are on exactly one of the pages.
- For $1 \leq i, j \leq N$, the matrix entry P_{ij} tells us the probability of j being the next page, given we are currently on page i .
- Clearly, for all i , $\sum_{j=1}^N P_{ij} = 1$



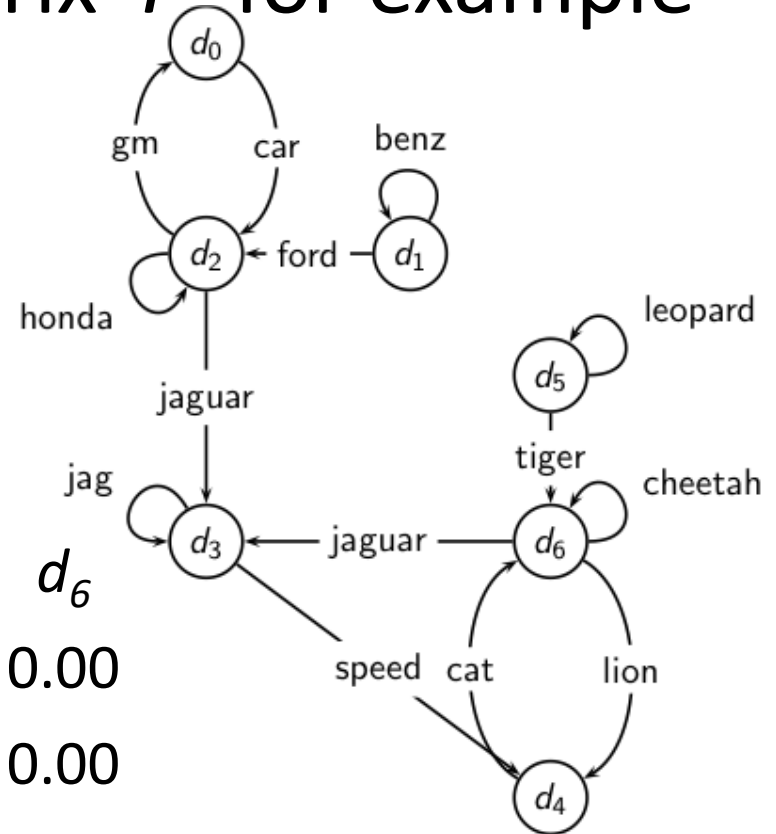
Example web graph

And adjacency matrix

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1



Transition probability matrix P for example



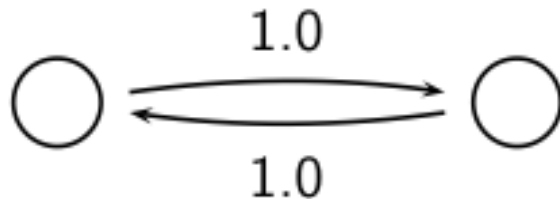
	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

Long-term visit rate

- Recall: PageRank = long-term visit rate
- Long-term visit rate of page d is the probability that a web surfer is at page d at a given point in time
- Next: what properties must hold of the web graph for the long-term visit rate to be well defined?
- The web graph must correspond to an **ergodic** Markov chain

Ergodic Markov chains

- A Markov chain is ergodic if it is irreducible and aperiodic.
- **Irreducibility.** Roughly: there is a path between a page to any other page.
- **Aperiodicity.** Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially.
- A non-ergodic Markov chain:



Ergodic Markov chains

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

Steady state

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \pi_2, \dots, \pi_N)$ of probabilities.
- (We use $\vec{\pi}$ to distinguish it from the notation for the probability vector \vec{x} .)
 - $\vec{\pi}$ is the long-term visit rate (or PageRank) of page i .
- So we can think of **PageRank** as a **very long vector** – one entry per page

Eigen Vectors

- For a square $N \times N$ matrix C and a vector \vec{x} that is not all zeros, the values of λ satisfying

$$C\vec{x} = \lambda\vec{x}$$

are called the **eigenvalues** of C

- The N -vector \vec{x} satisfying the above equation for an eigenvalue is the corresponding right eigenvector.
 - Eigenvector corresponding to the eigenvalue of largest magnitude is called the principal eigenvector.
- Similarly, the **left eigenvectors** of matrix C are the N -vectors \vec{y} such that

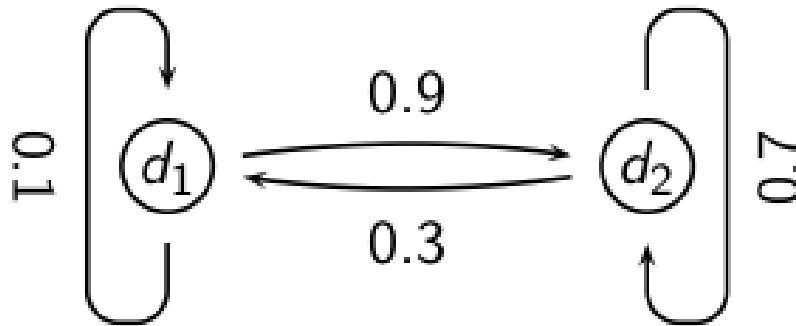
$$\vec{y}^T C = \lambda \vec{y}^T$$

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- $\vec{\pi} = (\pi_1, \pi_2, \dots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- ... and if the distribution in this step is x , then the distribution in the next step is xP .
- But $\vec{\pi}$ is the steady state!
- So: $\vec{\pi} = \vec{\pi}P$
- Solve this equation to get $\vec{\pi}$
 - Use **Power iteration** method
 - $\vec{\pi}$ is the principal left eigenvector for P ...
 - ... that is, π is the left eigenvector with the largest eigenvalue

Power Iteration Method: Example

- What is the PageRank / steady state in this example?



Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$	
			$P_{11} = 0.1$ $P_{12} = 0.9$ $P_{21} = 0.3$ $P_{22} = 0.7$
t_0	0	1	$= \vec{x}P$
t_1			$= \vec{x}P^2$
t_2			$= \vec{x}P^3$
t_3			$= \vec{x}P^4$
..			...
t_∞			$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$
t_0	0	1	0.3	0.7
t_1				
t_2				
t_3				
				\dots
t_∞				

$$= \vec{x}P$$

$$= \vec{x}P^2$$

$$= \vec{x}P^3$$

$$= \vec{x}P^4$$

$$\dots$$

$$= \vec{x}P^\infty$$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$
t_0	0	1	0.3	0.7
t_1	0.3	0.7		
t_2				
t_3				
t_∞				

$= \vec{x}P$
 $= \vec{x}P^2$
 $= \vec{x}P^3$
 $= \vec{x}P^4$
 \dots
 $= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2					$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					\dots
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76			$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					\dots
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					\dots
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748			$= \vec{x}P^4$
					\dots
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
					\dots
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
		
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
		
t_∞	0.25	0.75			$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$	PageRank vector = $\vec{\pi} = (\pi_1, \pi_2) = (0.25, 0.75)$		
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \mathbf{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{\mathbf{x}}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{\mathbf{x}}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{\mathbf{x}}P^4$
				...	\rightarrow ...
t_∞	<u>0.25</u>	<u>0.75</u>	<u>0.25</u>	<u>0.75</u>	$= \mathbf{x}P^\infty$ \rightarrow

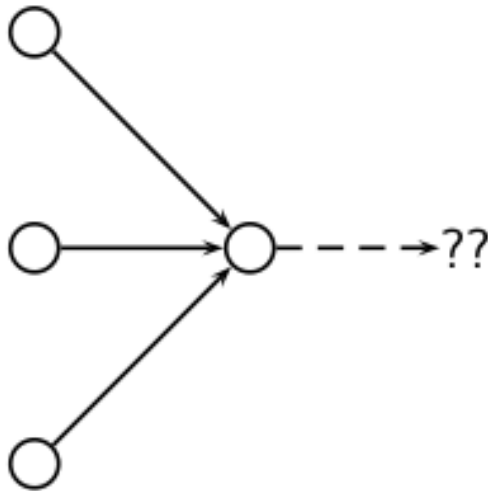
$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$

$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

PageRank issues

- Real surfers are **not random** surfers
 - Examples of nonrandom surfing: back button, short vs. long paths, bookmarks, directories – and search!
 - Markov model is not a good model of surfing
 - But it's good enough as a model for our purposes
- Simple PageRank ranking (as described on previous slide) produces **bad results** for many pages
 - Consider the query [video service]
 - The Yahoo home page (i) has a very high PageRank and (ii) contains both **video** and **service**
 - If we rank all Boolean hits according to PageRank, then the Yahoo home page would be top-ranked
 - Clearly not desirable for a video service query

Special Issue: Dead ends

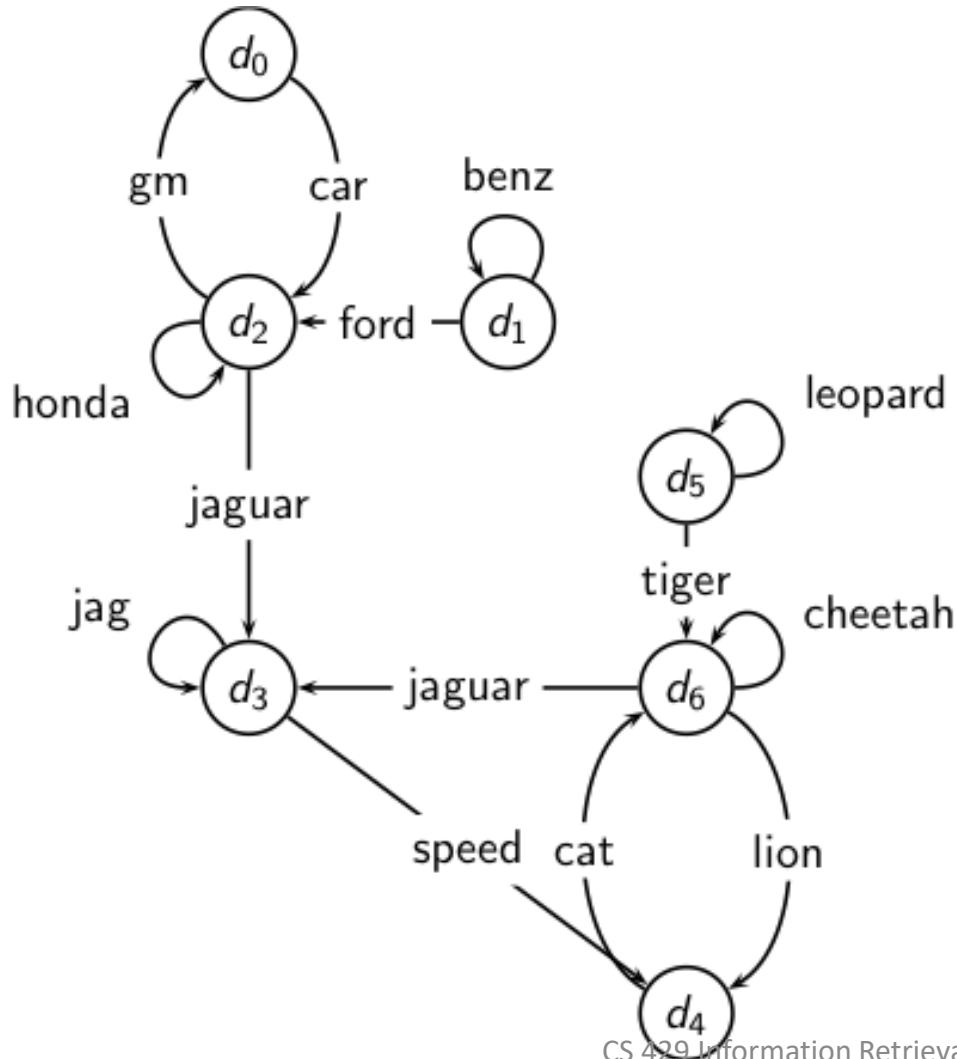


- The web is full of dead ends
- Random walk can get stuck in dead ends
- If there are dead ends, long-term visit rates are not well-defined (or non-sensical)

Teleporting – to get us of dead ends

- At a **dead end**, jump to a random web page with prob.
- $1/N$.
- At a **non-dead end**, with probability 10%, jump to a random web page (to each with a probability of $0.1/N$)
- With remaining probability (90%), go out on a random hyperlink
 - For example, if the page has 4 outgoing links: randomly choose one with probability $(1-0.10)/4=0.225$
- 10% is a parameter, the **teleportation rate (α parameter)**
- Note: “jumping” from dead end is independent of teleportation rate

Example web graph



Teleportation rate = $\alpha=0.14$

Adjacency Matrix

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0	0	1	0	0	0	0
d_1	0	1	1	0	0	0	0
d_2	1	0	1	1	0	0	0
d_3	0	0	0	1	1	0	0
d_4	0	0	0	0	0	0	1
d_5	0	0	0	0	0	1	1
d_6	0	0	0	1	1	0	1

Exercise

- Calculate the transition probability matrix with teleportation rate $\alpha = 0.14$

Calculating the Transition Matrix with Teleporting

- Given adjacency matrix A of the web graph.
- If a row of adjacency matrix A has no 1's, then replace each element by $1/N$.
- For all other rows proceed as follows.
 - Divide each 1 in A by the number of 1's in its row. Thus, if there is a row with three 1's, then each of them is replaced by $1/3$.
 - Multiply the resulting matrix by $(1 - \alpha)$.
 - Add $\frac{\alpha}{N}$ to every entry of the resulting matrix, to obtain transition matrix P .

Transition matrix **with teleporting**

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.02	0.02	0.88	0.02	0.02	0.02	0.02
d_1	0.02	0.45	0.45	0.02	0.02	0.02	0.02
d_2	0.31	0.02	0.31	0.31	0.02	0.02	0.02
d_3	0.02	0.02	0.02	0.45	0.45	0.02	0.02
d_4	0.02	0.02	0.02	0.02	0.02	0.02	0.88
d_5	0.02	0.02	0.02	0.02	0.02	0.45	0.45
d_6	0.02	0.02	0.02	0.31	0.31	0.02	0.31

Transition (probability) matrix **without teleportation**

	d_0	d_1	d_2	d_3	d_4	d_5	d_6
d_0	0.00	0.00	1.00	0.00	0.00	0.00	0.00
d_1	0.00	0.50	0.50	0.00	0.00	0.00	0.00
d_2	0.33	0.00	0.33	0.33	0.00	0.00	0.00
d_3	0.00	0.00	0.00	0.50	0.50	0.00	0.00
d_4	0.00	0.00	0.00	0.00	0.00	0.00	1.00
d_5	0.00	0.00	0.00	0.00	0.00	0.50	0.50
d_6	0.00	0.00	0.00	0.33	0.33	0.00	0.33

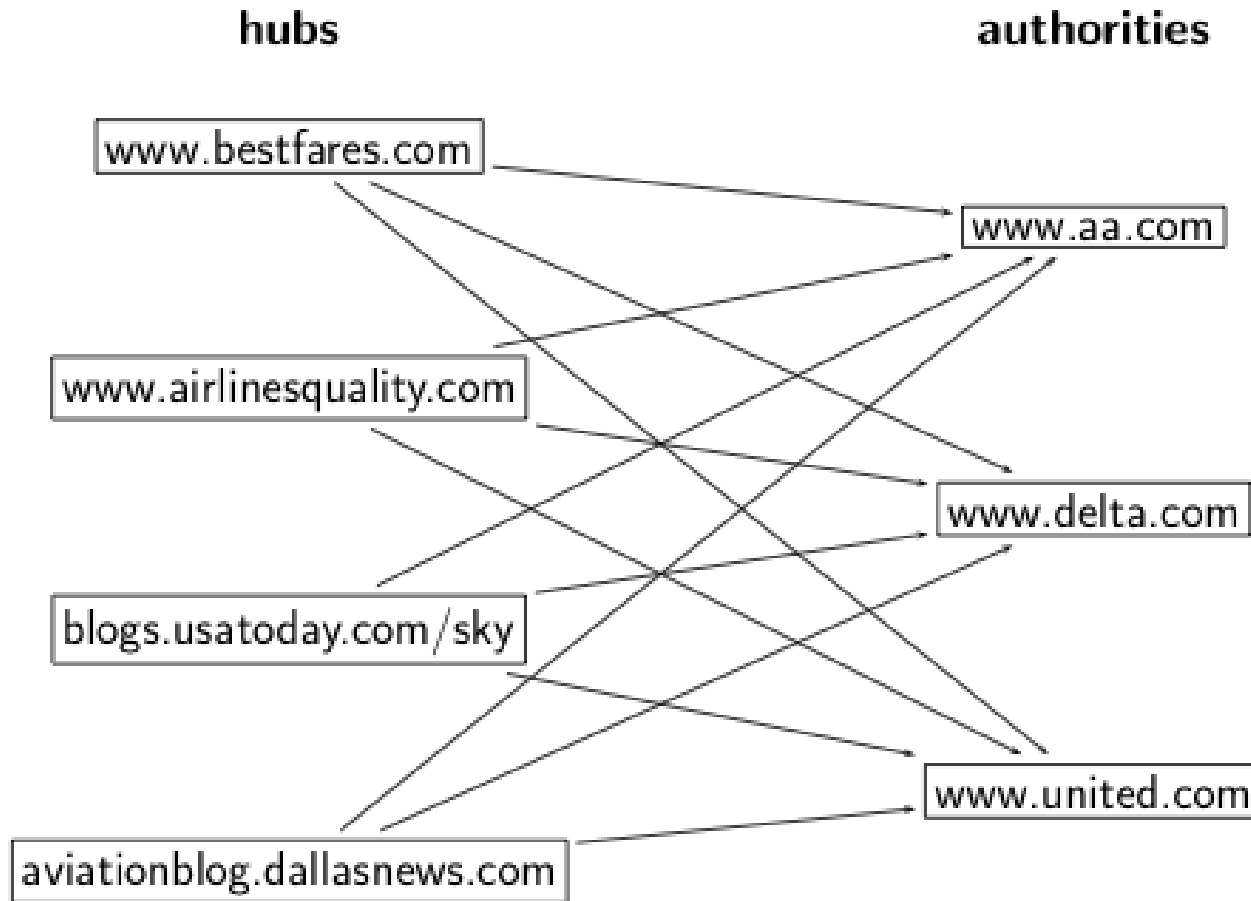
Power method vectors $\vec{x}P^k$

	\vec{x}	$\vec{x}P^1$	$\vec{x}P^2$	$\vec{x}P^3$	$\vec{x}P^4$	$\vec{x}P^5$	$\vec{x}P^6$	$\vec{x}P^7$	$\vec{x}P^8$	$\vec{x}P^9$	$\vec{x}P^{10}$	$\vec{x}P^{11}$	$\vec{x}P^{12}$	$\vec{x}P^{13}$
d_0	0.14	0.06	0.09	0.07	0.07	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.05
d_1	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_2	0.14	0.25	0.18	0.17	0.15	0.14	0.13	0.12	0.12	0.12	0.12	0.11	0.11	0.11
d_3	0.14	0.16	0.23	0.24	0.24	0.24	0.24	0.25	0.25	0.25	0.25	0.25	0.25	0.25
d_4	0.14	0.12	0.16	0.19	0.19	0.20	0.21	0.21	0.21	0.21	0.21	0.21	0.21	0.21
d_5	0.14	0.08	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
d_6	0.14	0.25	0.23	0.25	0.27	0.28	0.29	0.29	0.30	0.30	0.30	0.30	0.31	0.31

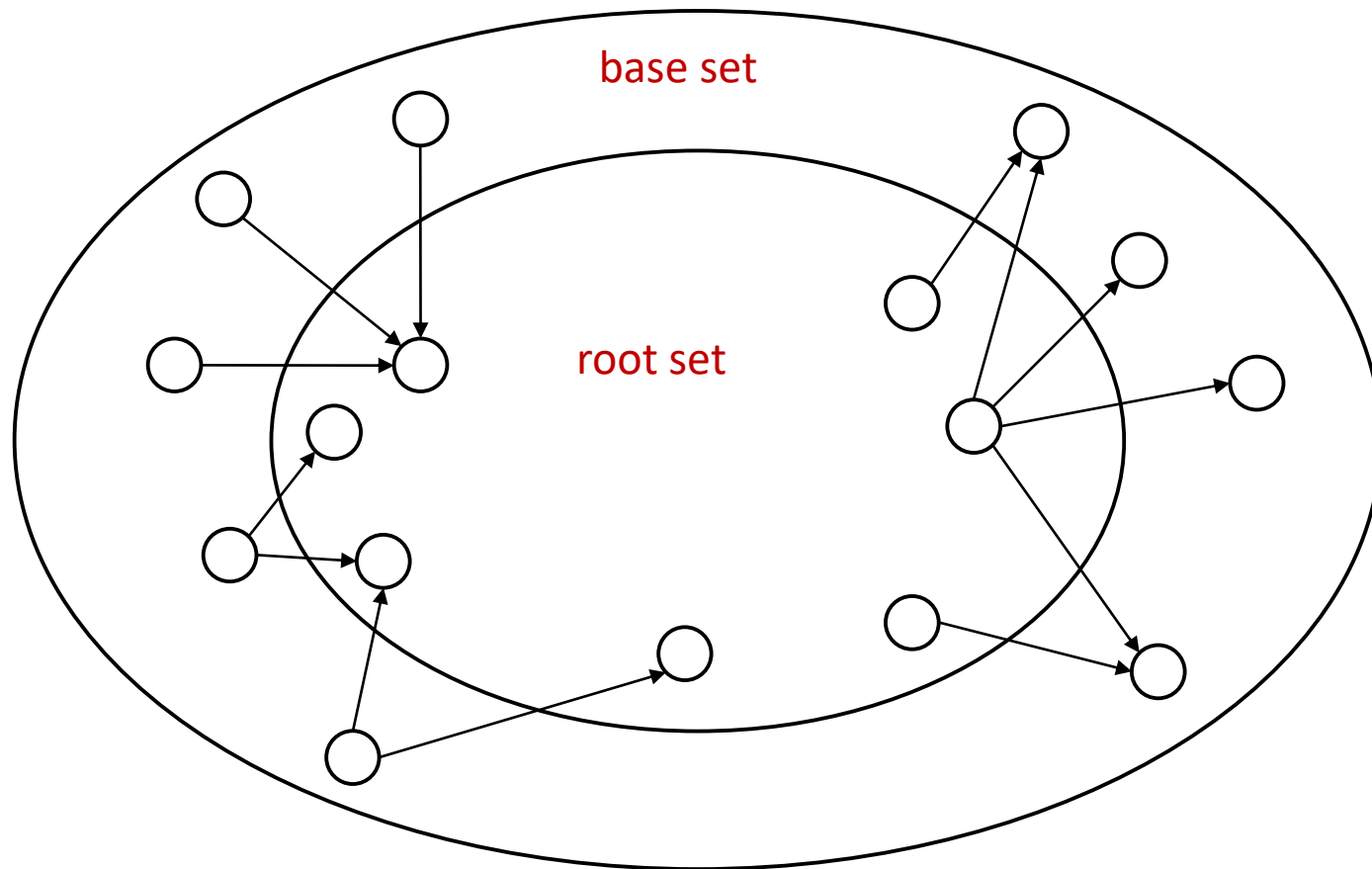
HITS – Hyperlink-Induced Topic Search

- Premise: there are two different types of relevance on the web.
- Relevance type 1: **Hubs**. A hub page is a good list of links to pages answering the information need.
 - E.g, for query [chicago bulls]: Bob's list of recommended resources on the Chicago Bulls sports team
- Relevance type 2: **Authorities**. An authority page is a direct answer to the information need.
 - The home page of the Chicago Bulls sports team
 - By definition: Links to authority pages occur repeatedly on hub pages.
- Most approaches to search (including PageRank ranking) don't make the distinction between these two very different types of relevance.

Example for hubs and authorities

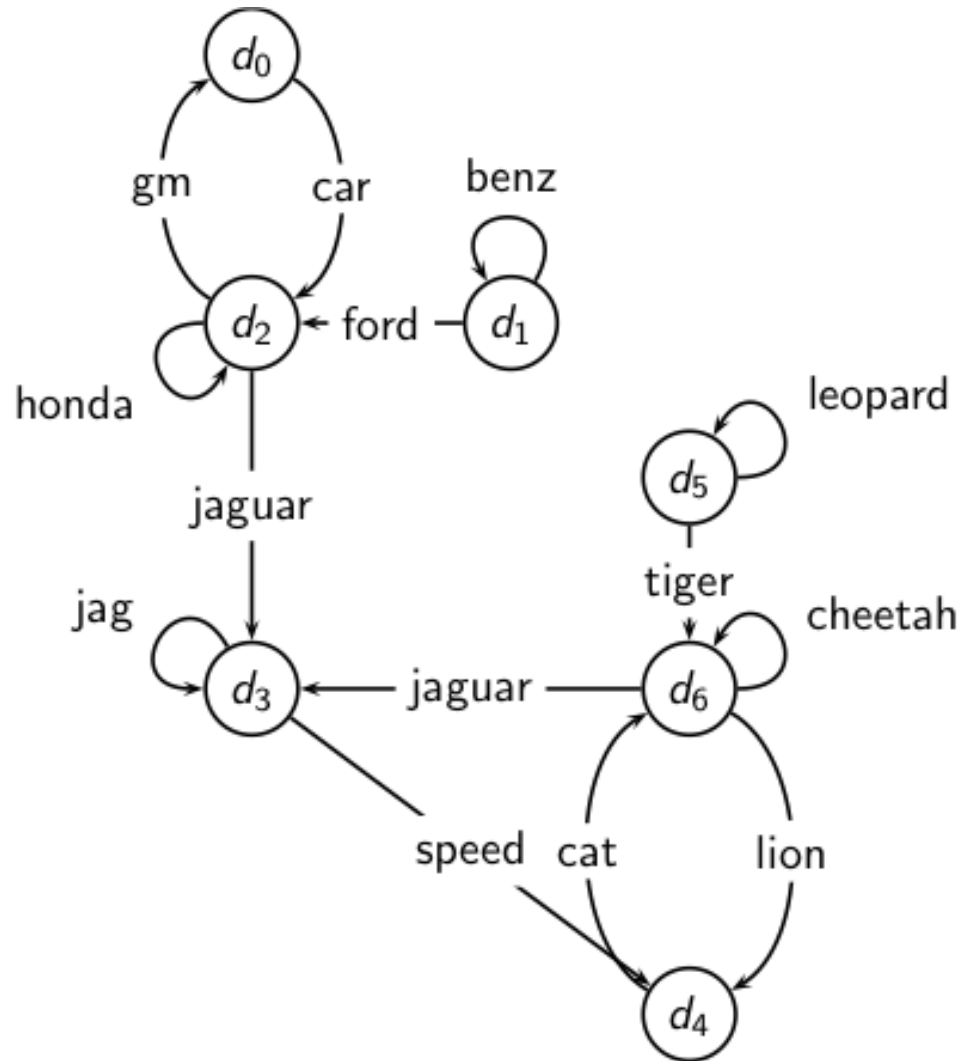


Root set and base set



The base set

Example web graph



PageRank vs. HITS: Discussion

- PageRank can be precomputed, HITS has to be computed at query time.
 - HITS is too expensive in most application scenarios
- PageRank and HITS make two different design choices concerning
 - i. the eigenproblem formalization
 - ii. the set of pages to apply the formalization to.
- These two are orthogonal
 - We could also apply HITS to the entire web and PageRank to a small base set
- Claim: On the web, a good hub almost always is also a good authority
 - The actual difference between PageRank ranking and HITS ranking is therefore not as large as one might expect

Questions to ponder (not comprehensive)

- What is the basic architecture of a web crawler? What is the reason it works?
- What is the difference between link analysis based and bag-of-words based approaches?
- What aspect of a document does link analysis capture?
- What are some key algorithms for link analysis?
- What are the key assumptions behind PageRank algorithm?
- What is the procedure to calculate the PageRank?
- What are the strengths and weaknesses of PageRank algorithm?

Questions to ponder (not comprehensive) cont.

- What are the basic intuitions behind HITS?
- How to calculate the hub and authority scores?
- What are the strengths and weakness of the HITS algorithm?
- Compare and contrast HITS and PageRank

Classification

Examples of how search engines use classification

- The automatic detection of **spam pages** (spam vs. nonspam)
- The automatic detection of sexually **explicit content** (sexually explicit vs. not)
- **Topic-specific** or *vertical* search – restrict search to a “vertical” like “related to health” (relevant to vertical vs. not)
- **Standing queries** (e.g., Google Alerts)
- **Sentiment detection**: is a movie or product review positive or negative (positive vs. negative)
- **Language identification** (classes: English vs. French etc.)

The Naive Bayes classifier

- The Naive Bayes classifier is a probabilistic classifier.
- We compute the probability of a document d being in a class c as follows:

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

- n_d is the length of the document. (number of tokens)
 - $P(t_k | c)$ is the conditional probability of term t_k occurring in a document of class c
 - $P(t_k | c)$ as a measure of **how much evidence** t_k contributes that c is the correct class.
 - $P(c)$ is the prior probability of c
- If a document's terms do not provide clear evidence for one class vs. another, we choose the c with highest $P(c)$

Maximum a posteriori class

- Our goal in Naive Bayes classification is to find the “best” class.
- The best class is the most likely **or maximum a posteriori (MAP) class** c_{map} :

$$c_{\text{map}} = \arg \max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg \max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$

Parameter estimation take 1: Maximum likelihood

- Estimate parameters $\hat{P}(c)$ and $\hat{P}(t_k|c)$ from training/test data:
How?

- Prior:
$$\hat{P}(c) = \frac{N_c}{N}$$

- N_c : number of docs in class c ; N : total number of docs
- Conditional probabilities:

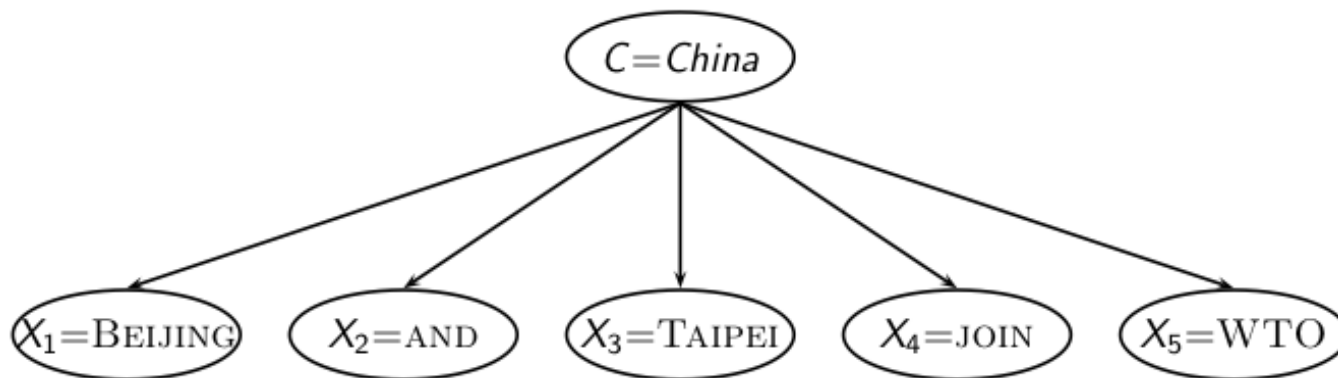
$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- T_{ct} is the number of tokens of t in training documents from class c (includes multiple occurrences)
- We've made a **Naive Bayes independence assumption** here:

$$\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$$

The problem with maximum likelihood estimates: **Zeros**

$$c_{\text{map}} = \arg \max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg \max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c)$$



$$P(\text{China}|d) \propto P(\text{China}) \cdot P(\text{BEIJING}|\text{China}) \cdot P(\text{AND}|\text{China}) \\ \cdot P(\text{TAIPEI}|\text{China}) \cdot P(\text{JOIN}|\text{China}) \cdot P(\text{WTO}|\text{China})$$

- If WTO never occurs in the training set for class China:

$$\hat{P}(\text{WTO}|\text{China}) = \frac{T_{\text{China}, \text{WTO}}}{\sum_{t' \in V} T_{\text{China}, t'}} = \frac{0}{\sum_{t' \in V} T_{\text{China}, t'}} = 0$$

The problem with maximum likelihood estimates: Zeros(cont)

- If there were no occurrences of WTO in documents in class China, we'd get a zero estimate:

$$\hat{P}(\text{WTO}|\text{China}) = \frac{T_{\text{China}, \text{WTO}}}{\sum_{t' \in V} T_{\text{China}, t'}} = 0$$

- We will get $P(\text{China}|\text{d}) = 0$ for any document that contains WTO!

To avoid zeros: Add-one smoothing

- Before:

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}}$$

- Now: Add one to each count to avoid zeros:

$$\hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B}$$

- B is the number of different words (in this case the size of the vocabulary: $|V| = M$)

Exercise

	docID	words in document	in $c = \textit{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- Estimate parameters of Naive Bayes classifier
- Classify test document

Exercise

	docID	words in document	in $c = \textit{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

How many classes are there?

Exercise

	docID	words in document	in $c = \textit{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- How many classes are there?
 - $\textit{China}(c)$ & all classes that are not china (\bar{c})

Exercise

	docID	words in document	in $c = \textit{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- How many classes are there?
 - $\textit{China}(c)$ & all classes that are not china (\bar{c})
- What are the values of the priors?
 - $\hat{P}(c) = ?$
 - $\hat{P}(\bar{c}) = ?$

Exercise

	docID	words in document	in $c = \textit{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- How many classes are there?
 - $c = \textit{China}$ & all classes that are not china (\bar{c})
- What are the values of the priors?
 - $\hat{P}(c) = \frac{3}{4}$
 - $\hat{P}(\bar{c}) = \frac{1}{4}$

Exercise

	docID	words in document	in $c = \textit{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- What are the values of the priors?
 - $\hat{P}(c) = \frac{3}{4}$
 - $\hat{P}(\bar{c}) = \frac{1}{4}$
- What are the conditional probabilities for each term?

Exercise

	docID	words in document	in $c = \text{China}$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

- What are the values of the priors?

- $\hat{P}(c) = \frac{3}{4}$ $\hat{P}(\bar{c}) = \frac{1}{4}$

- What are the conditional probabilities for each term?

- $\hat{P}(\text{CHINESE}|c) = \frac{T_{\text{CHINESE},c} + 1}{\sum_{t' \in V} T_{t',c} + 1}$

$$= \frac{T_{\text{CHINESE},c} + 1}{(T_{\text{CHINESE},c} + 1) + (T_{\text{BEIJING},c} + 1) + (T_{\text{SHANGHAI},c} + 1) + (T_{\text{MACAO},c} + 1) + (T_{\text{TOKYO},c} + 1) + (T_{\text{JAPAN},c} + 1)}$$

$$= \frac{6}{(5 + 1) + (1 + 1) + (1 + 1) + (1 + 1) + (0 + 1) + (0 + 1)} = \frac{6}{14}$$

Example: Parameter estimates

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\bar{c}) = 1/4$ Conditional probabilities:

$$\begin{aligned}\hat{P}(\text{CHINESE}|c) &= (5 + 1)/(8 + 6) = 6/14 = 3/7 \\ \hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) &= (0 + 1)/(8 + 6) = 1/14 \\ \hat{P}(\text{CHINESE}|\bar{c}) &= (1 + 1)/(3 + 6) = 2/9 \\ \hat{P}(\text{TOKYO}|\bar{c}) = \hat{P}(\text{JAPAN}|\bar{c}) &= (1 + 1)/(3 + 6) = 2/9\end{aligned}$$

The denominators are $(8 + 6)$ and $(3 + 6)$ because the lengths of text_c and $\text{text}_{\bar{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: Classification

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$

$$\hat{P}(\bar{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Thus, the classifier assigns the test document to $c = \textit{China}$. The reason for this classification decision is that the three occurrences of the positive indicator CHINESE in d_5 outweigh the occurrences of the two negative indicators JAPAN and TOKYO.

Taking the log

- Multiplying lots of small probabilities can result in floating point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, we can sum log probabilities instead of multiplying probabilities.
- Since log is a monotonic function, the class with the highest score does not change.
- So what we usually compute in practice is:

$$c_{\text{map}} = \arg \max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k | c)]$$

Naive Bayes classifier

- Classification rule:

$$c_{\text{map}} = \arg \max_{c \in \mathbb{C}} [\log \hat{P}(c) + \sum_{1 \leq k \leq n_d} \log \hat{P}(t_k|c)]$$

- Simple interpretation:
 - Each conditional parameter $\log \hat{P}(t_k|c)$ is a weight that indicates how good an indicator t_k is for c
 - The prior $\log \hat{P}(c)$ is a weight that indicates the relative frequency of c
 - The sum of log prior and term weights is then a measure of how much evidence there is for the document being in the class.
 - We select the class with the most evidence

Second (positional) independence assumption

$$\hat{P}(t_{k_1}|c) = \hat{P}(t_{k_2}|c)$$

- For example, for a document in the class *UK*, the probability of generating QUEEN in the first position of the document is the same as generating it in the last position.
- The two independence assumptions amount to **the bag of words** model.

Naive Bayes is Effective

- Naive Bayes has won competitions(e.g., KDD-CUP 97)
- More robust to non-relevant features than some more complex learning methods
- More robust to concept drift (changing of definition of class over time) than some more complex learning methods
- A good dependable baseline for text classification (but not the best)
- Optimal if independence assumptions hold (never true for text, but true for some domains)
- Very fast
- Low storage requirements

Evaluating classification

- Evaluation must be done on test data that are independent of the training data (usually a disjoint set of instances).
- It's easy to get good performance on a test set that was available to the learner during training (e.g., just memorize the test set).
- Measures: Precision, recall, F_1 , classification accuracy

Precision P and recall R

	in the class	not in the class
predicted to be in the class	true positives (TP)	false positives (FP)
predicted to not be in the class	false negatives (FN)	true negatives (TN)

$$P = TP / (TP + FP)$$

$$R = TP / (TP + FN)$$

Averaging: Micro vs. Macro

- We now have an evaluation measure (F_1) for **one class**
- But we also want a single number that measures the **aggregate performance** over all classes in the collection.
- **Macroaveraging**
 - Compute F_1 for each of the C classes
 - Average these C numbers
- **Microaveraging**
 - Compute TP, FP, FN for each of the C classes
 - Sum these C numbers (e.g., all TP to get aggregate TP)
 - Compute F_1 for aggregate TP, FP, FN

Feature Selection

- In text classification, we usually represent documents in a **high-dimensional** space, with each dimension corresponding to a term
- In this lecture: axis = dimension = word = term = feature
- Many dimensions correspond to rare words
 - Rare words can mislead the classifier
 - Rare misleading features are called **noise features**
- **Eliminating noise features** from the representation **increases efficiency and effectiveness** of text classification
- Eliminating features is called **feature selection**

Mutual information

- Compute the feature utility $A(t, c)$ as the **expected mutual information** (MI) of term t and class c
- MI tells us “how much information” the term contains about the class and vice versa.
- For example, if a term’s occurrence is independent of the class (same proportion of docs within/without class contain the term), then MI is 0.
- Definition:

$$I(U; C) = \sum_{e_t \in \{1,0\}} \sum_{e_c \in \{1,0\}} P(U=e_t, C=e_c) \log_2 \frac{P(U=e_t, C=e_c)}{P(U=e_t)P(C=e_c)}$$

How to compute MI values

- Based on maximum likelihood estimates, the formula we actually use is:

$$I(U; C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_{1.}N_{.1}} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_{0.}N_{.1}} \\ + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_{1.}N_{.0}} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_{0.}N_{.0}}$$

- N_{10} : number of documents that contain t ($e_t = 1$) and are not in c ($e_c = 0$); N_{11} : number of documents that contain t ($e_t = 1$) and are in c ($e_c = 1$); N_{01} : number of documents that do not contain t ($e_t = 0$) and are in c ($e_c = 1$); N_{00} : number of documents that do not contain t ($e_t = 0$) and are not in c ($e_c = 0$); $N = N_{00} + N_{01} + N_{10} + N_{11}$.

MI example for *poultry*/EXPORT in Reuters

	$e_c = e_{poultry} = 1$	$e_c = e_{poultry} = 0$	
$e_t = e_{EXPORT} = 1$	$N_{11} = 49$	$N_{10} = 27,652$	Plug
$e_t = e_{EXPORT} = 0$	$N_{01} = 141$	$N_{00} = 774,106$	

these values into formula:

MI example for *poultry*/EXPORT in Reuters

$$\begin{array}{l}
 e_t = e_{\text{EXPORT}} = 1 \\
 e_t = e_{\text{EXPORT}} = 0
 \end{array}
 \begin{array}{cc}
 e_c = e_{\text{poultry}} = 1 & e_c = e_{\text{poultry}} = 0 \\
 \hline
 N_{11} = 49 & N_{10} = 27,652 \\
 \hline
 N_{01} = 141 & N_{00} = 774,106
 \end{array}
 \text{ Plug}$$

these values into formula:

$$\begin{aligned}
 I(U; C) &= \frac{49}{801,948} \log_2 \frac{801,948 \cdot 49}{(49 + 27,652)(49 + 141)} \\
 &+ \frac{141}{801,948} \log_2 \frac{801,948 \cdot 141}{(141 + 774,106)(49 + 141)} \\
 &+ \frac{27,652}{801,948} \log_2 \frac{801,948 \cdot 27,652}{(49 + 27,652)(27,652 + 774,106)} \\
 &+ \frac{774,106}{801,948} \log_2 \frac{801,948 \cdot 774,106}{(141 + 774,106)(27,652 + 774,106)} \\
 &\approx 0.000105
 \end{aligned}$$

MI feature selection on Reuters

Class: <i>coffee</i>		Class: <i>sports</i>	
term	MI	term	MI
COFFEE	0.0111	SOCCER	0.0681
BAGS	0.0042	CUP	0.0515
GROWERS	0.0025	MATCH	0.0441
KG	0.0019	MATCHES	0.0408
COLOMBIA	0.0018	PLAYED	0.0388
BRAZIL	0.0016	LEAGUE	0.0386
EXPORT	0.0014	BEAT	0.0301
EXPORTERS	0.0013	GAME	0.0299
EXPORTS	0.0013	GAMES	0.0284
CROP	0.0012	TEAM	0.0264

Vector space classification

- As before, the training set is a set of documents, each labeled with its class
- In vector space classification, this set corresponds to a labeled set of points or vectors in the vector space
- **Premise 1:** Documents in the same class form a **contiguous region**
- **Premise 2:** Documents from different classes **don't overlap**
- We define lines, surfaces, hypersurfaces to divide regions

Rocchio vs. Naive Bayes

- In many cases, Rocchio performs worse than Naïve Bayes.
- One reason: Rocchio does not handle nonconvex, multimodal classes correctly.

Questions to ponder (not comprehensive)

- What is the main idea behind Naïve bayes classification?
- What are the assumption of Naïve bayes?
- How do we compute the prior and conditional probabilities?
- Compare the run time of the training and testing phases of Naïve bayes
- What are the strengths and weakness of Naïve bayes?
- How do we evaluate classifier performance? What is the distinction between microaveraging and macroaveraging?

Questions to ponder (not comprehensive)

- Why is feature selection useful?
- How to calculate important feature using mutual information?
- What are the main premises behind vector space model?
- What are the similarities between using Rocchio for relevance feedback and for classification?
- Compare the two vector space methods: rocchio and kNN.

kNN: Discussion

- No training necessary
 - But linear preprocessing of documents is as expensive as training Naive Bayes
 - We always preprocess the training set, so in reality training time of kNN is linear
- kNN is very accurate if training set is large
- But kNN can be very inaccurate if training set is small