

Introduction to Information Retrieval

<http://informationretrieval.org>

IIR 12: Language Models for IR

Hinrich Schütze

Institute for Natural Language Processing, Universität Stuttgart

2009.06.30

Overview

- 1 Recap
- 2 Language models
- 3 Language Models for IR
- 4 Discussion

Outline

- 1 Recap
- 2 Language models
- 3 Language Models for IR
- 4 Discussion

Indexing anchor text

- Anchor text is often a better description of a page's content than the page itself.
- Anchor text can be weighted more highly than the text on the page.
- A Google bomb is a search with “bad” results due to maliciously manipulated anchor text.
 - [dangerous cult] on Google, Bing, Yahoo

PageRank

- Model: a web surfer doing a random walk on the web
- Formalization: Markov chain
- PageRank is the **long-term visit rate** of the random surfer or the **steady-state distribution**.
- Need **teleportation** to ensure well-defined PageRank
- Power method to compute PageRank
 - PageRank is the principal left eigenvector of the transition probability matrix.

Computing PageRank: Power method

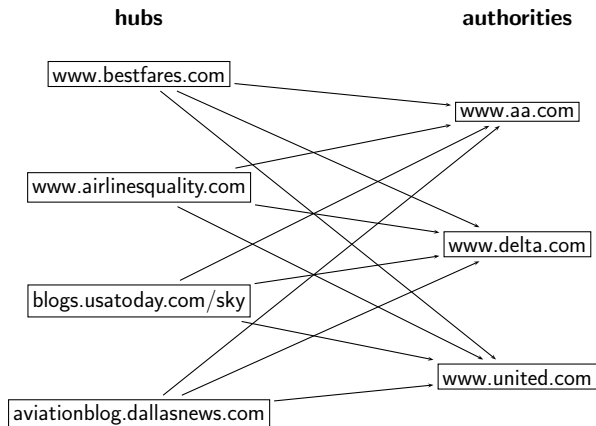
	x_1	x_2			
	$P_t(d_1)$	$P_t(d_2)$			
			$P_{11} = 0.1$	$P_{12} = 0.9$	
			$P_{21} = 0.3$	$P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
				...	
t_∞	0.25	0.75	0.25	0.75	$= \vec{x}P^\infty$

PageRank vector $= \vec{\pi} = (\pi_1, \pi_2) = (0.25, 0.75)$

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

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

HITS: Hubs and authorities



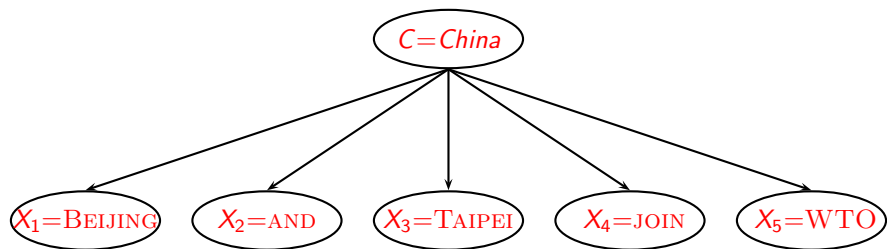
HITS update rules

- A : link matrix
- \vec{h} : vector of hub scores
- \vec{a} : vector of authority scores
- HITS algorithm:
 - Compute $\vec{h} = A\vec{a}$
 - Compute $\vec{a} = A^T\vec{h}$
 - Iterate until convergence
 - Output (i) list of hubs ranked according to hub score and (ii) list of authorities ranked according to authority score

Outline

- 1 Recap
- 2 Language models
- 3 Language Models for IR
- 4 Discussion

Recall: Naive Bayes generative model



Naive Bayes and LM generative models

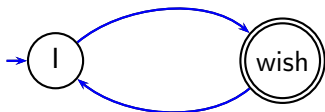
- We want to classify document d .
We want to classify a query q .
 - Classes: geographical regions like *China*, *UK*, *Kenya*.
Each document in the collection is a different class.
- Assume that d was generated by the generative model.
Assume that q was generated by a generative model
- Key question: Which of the classes is most likely to have generated the document? Which document (=class) is most likely to have generated the query q ?
 - Or: for which class do we have the most evidence? For which document (as the source of the query) do we have the most evidence?

Using language models (LMs) for IR

- 1 LM = language model
- 2 We view the document as a generative model that generates the query.
- 3 What we need to do:
- 4 Define the precise generative model we want to use
- 5 Estimate parameters (different parameters for each document's model)
- 6 Smooth to avoid zeros
- 7 Apply to query and find document most likely to have generated the query
- 8 Present most likely document(s) to user
- 9 Note that $x \rightarrow y$ is pretty much what we did in Naive Bayes.

What is a language model?

We can view a **finite state automaton** as a **deterministic** language



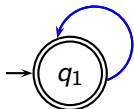
model.

I wish I wish I wish I wish ... Cannot generate: "wish I wish"

or "I wish I" Our basic model: each document was generated by a

different automaton like this except that these automata are **probabilistic**.

A 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. frog said that toad likes frog STOP

$$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$$
$$= 0.00000000000048$$

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
	

frog said that toad likes frog STOP $P(\text{string}|M_{d1}) = 0.01 \cdot 0.03$

$$\cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 = 0.0000000000048 = 4.8 \cdot 10^{-12}$$

$$P(\text{string}|M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.2$$

$$= 0.0000000000120 = 12 \cdot 10^{-12} \quad P(\text{string}|M_{d1}) < P(\text{string}|M_{d2})$$

Thus, document d_2 is “more relevant” to the string “frog said that toad likes frog STOP” than d_1 is.

Outline

- 1 Recap
- 2 Language models
- 3 Language Models for IR**
- 4 Discussion

Using language models in IR

- Each document is treated as (the basis for) a language model.
- Given a query q
- 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
 - But we can give a higher prior to “high-quality” documents, e.g., those with high PageRank.
- $P(q|d)$ is the probability of q given d .
- So to rank documents according to relevance to q , ranking according to $P(q|d)$ and $P(d|q)$ is equivalent.

Where we are

- In the LM approach to IR, we attempt to model the **query generation process**.
- Then we rank documents by **the probability that a query would be observed as a random sample from the respective document model**.
- That is, we rank according to $P(q|d)$.
- Next: how do we compute $P(q|d)$?

How to compute $P(q|d)$

- We will make the same conditional independence assumption as for Naive Bayes.
-

$$P(q|M_d) = P(\langle t_1, \dots, t_{|q|} \rangle | M_d) = \prod_{1 \leq k \leq |q|} P(t_k | M_d)$$

($|q|$: length of q ; t_k : the token occurring at position k in q)

- This is equivalent to:

$$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\text{tf}_{t,q}}$$

- $\text{tf}_{t,q}$: term frequency ($\#$ occurrences) of t in q
- **Multinomial model** (omitting constant factor)

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates (as we did for Naive Bayes)

-

$$\hat{P}(t|M_d) = \frac{\text{tf}_{t,d}}{|d|}$$

($|d|$: length of d ; $\text{tf}_{t,d}$: # occurrences of t in d)

- As in Naive Bayes, we have a problem with zeros.
- A single t with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod P(t|M_d)$ zero.
- We would give a single term “veto power”.
- For example, for query [Michael Jackson top hits] a document about “top songs” (but not using the word “hits”) would have $P(q|M_d) = 0$. – That’s bad.
- We need to smooth the estimates to avoid zeros.

Smoothing

- Key intuition: A nonoccurring term is possible (even though it didn't occur), ...
- ...but no more likely than would be expected by chance in the collection.
- Notation: M_c : the collection model; cf_t : the number of occurrences of t in the collection; $T = \sum_t cf_t$: the total number of tokens in the collection.



$$\hat{P}(t|M_c) = \frac{cf_t}{T}$$

- We will use $\hat{P}(t|M_c)$ to “smooth” $P(t|d)$ away from zero.

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.
- High value of λ : “conjunctive-like” search – tends to retrieve documents containing all query words.
- Low value of λ : more disjunctive, suitable for long queries
- Correctly setting λ is very important for good performance.

Mixture model: Summary



$$P(q|d) \propto \prod_{1 \leq k \leq |q|} (\lambda P(t_k|M_d) + (1 - \lambda)P(t_k|M_c))$$

- What we model: The user has a document in mind and generates the query from this document.
- The equation represents the probability that the document that the user had in mind was in fact this one.

Example

- Collection: d_1 and d_2
- d_1 : Jackson was one of the most talented entertainers of all time
- d_2 : Michael Jackson anointed himself King of Pop
- Query q : Michael Jackson
- Use mixture model with $\lambda = 1/2$
- $P(q|d_1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2] \approx 0.003$
- $P(q|d_2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2] \approx 0.013$
- Ranking: $d_2 > d_1$

Exercise: Compute ranking

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2 : Lucene narrows quarter loss but revenue decreases further
- Query q : revenue down
- Use mixture model with $\lambda = 1/2$
- $P(q|d_1) = [(1/8 + 2/16)/2] \cdot [(1/8 + 1/16)/2] = 1/8 \cdot 3/32 = 3/256$
- $P(q|d_2) = [(1/8 + 2/16)/2] \cdot [(0/8 + 1/16)/2] = 1/8 \cdot 1/32 = 1/256$
- Ranking: $d_1 > d_2$

Outline

- 1 Recap
- 2 Language models
- 3 Language Models for IR
- 4 Discussion

LMs vs. Naive Bayes

- Different **smoothing methods**: mixture model vs. add-one
- We classify the **query** in LMs; we classify **documents** in text classification.
- Each document is a class in LMs vs. classes are human-defined in text classification
- The formal model is the same: multinomial model.
 - Actually: The way we presented Naive Bayes, it's not a true multinomial model, but it's equivalent.

Vector space (tf-idf) vs. LM

Rec.	precision		%chg	significant?
	tf-idf	LM		
0.0	0.7439	0.7590	+2.0	
0.1	0.4521	0.4910	+8.6	
0.2	0.3514	0.4045	+15.1	*
0.4	0.2093	0.2572	+22.9	*
0.6	0.1024	0.1405	+37.1	*
0.8	0.0160	0.0432	+169.6	*
1.0	0.0028	0.0050	+76.9	
11-point average	0.1868	0.2233	+19.6	*

The

language modeling approach always does better in these experimentsbut note that where the approach shows significant gains is at higher levels of recall.

LMs vs. vector space model (1)

- LMs have some things in common with vector space models.
- Term frequency is directly in the model.
 - But it is not scaled in LMs.
- Probabilities are inherently “length-normalized”.
 - Cosine normalization does something similar for vector space.
- Mixing document and collection frequencies has an effect similar to idf.
 - Terms rare in the general collection, but common in some documents will have a greater influence on the ranking.

LMs vs. vector space model (2)

- LMs vs. vector space model: commonalities
 - Term frequency is directly in the model.
 - 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.

Language models for IR: Assumptions

- Simplifying assumption: **Queries and documents are objects of the same type.** Not true!
 - There are other LMs for IR that do not make this assumption.
 - The vector space model makes the same assumption.
- Simplifying assumption: **Terms are conditionally independent.**
 - Again, vector space model (and Naive Bayes) make the same assumption.
- Cleaner statement of assumptions than vector space
- Thus, better theoretical foundation than vector space
 - ...but “pure” LMs perform much worse than “tuned” LMs.

Resources

- Chapter 12 of IIR
- Resources at <http://ifnlp.org/ir>
 - Ponte and Croft's 1998 SIGIR paper (one of the first on LMs in IR)
 - Lemur toolkit (good support for LMs in IR)