Introduction to Information Retrieval

Language Models for IR

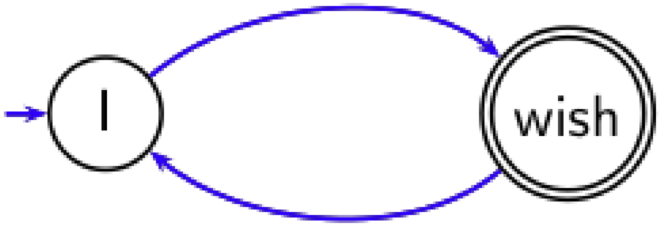
Overview

- Language models
- Language models for IR
- Discussion

Language Models

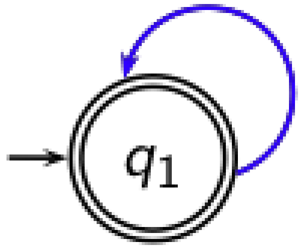
What is a language model?

- 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 (left) and the state emission distribution for its one state q_1 (right).
- STOP is not a word, but a special symbol indicating that the automaton stops.

String **: frog said that toad likes frog STOP**

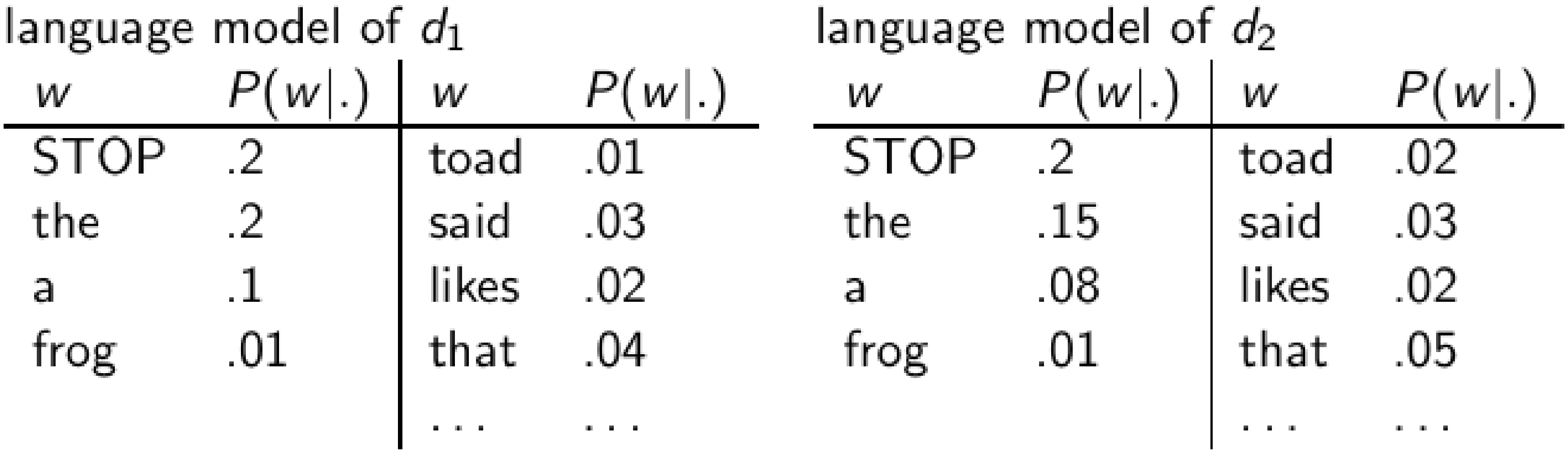
- $P(\text{String}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot$

To find the probability of a word sequence, we just multiply the probabilities which the model gives to each word in the sequence, together with the probability of continuing or stopping after producing each word. For example

$$\begin{aligned} P(\text{frog said that toad likes frog}) &= (0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01) \\ &\quad \times (0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.2) \\ &\approx 0.000000000001573 \end{aligned}$$

Most of the time, we will omit to include STOP and (1 − STOP) probabilities

A different language model for each document



language model of d_1

w	$P(w\|.)$	w	$P(w\|.)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
		...	...

language model of d_2

w	$P(w\|.)$	w	$P(w\|.)$
STOP	.2	toad	.02
the	.15	said	.03
a	.08	likes	.02
frog	.01	that	.05
		...	...

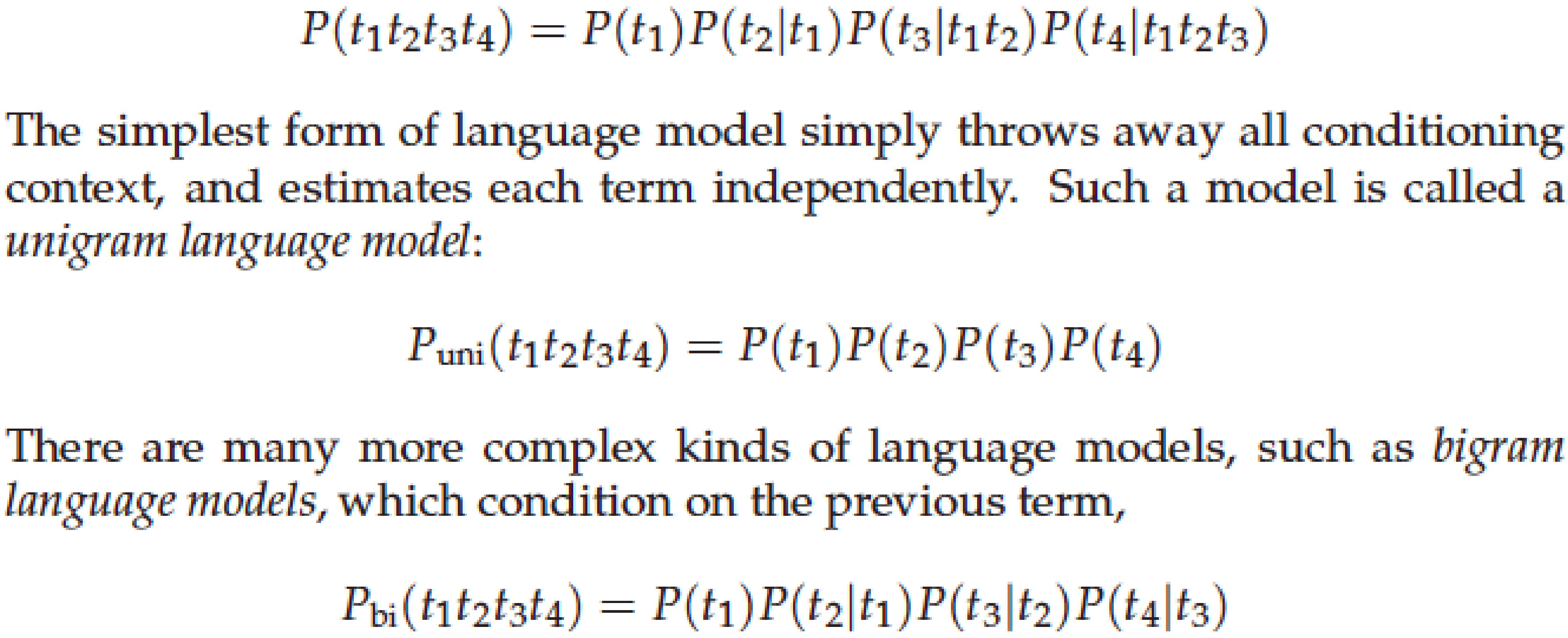
frog said that toad likes frog STOP

- $P(\text{string}|M_{d1}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 4.8 \cdot 10^{-12}$
- $P(\text{string}|M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 12 \cdot 10^{-12}$

$$P(\text{string}|M_{d1}) < P(\text{string}|M_{d2})$$

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

Types of Language Model



$$P(t_1t_2t_3t_4) = P(t_1)P(t_2|t_1)P(t_3|t_1t_2)P(t_4|t_1t_2t_3)$$

The simplest form of language model simply throws away all conditioning context, and estimates each term independently. Such a model is called a *unigram language model*:

$$P_{\text{uni}}(t_1t_2t_3t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$$

There are many more complex kinds of language models, such as *bigram language models*, which condition on the previous term,

$$P_{\text{bi}}(t_1t_2t_3t_4) = P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_4|t_3)$$

Language models for IR

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 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 if P(d) is same for all.

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 $P(q|d)$?

Note: From now onwards, $P(q|d)$ and $P(q|M_d)$ have been used interchangeably.

How to compute $P(q|d)$

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

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

M_d: language model for document d;

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

- This is equivalent to: $$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\mathrm{tf}_{t,q}}$$
- $\mathrm{tf}_{t,q}$: 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

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

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

- 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 $P(t|M_d) = 0$. – That’s bad.
- We need to smooth the estimates to avoid zeros.

Smoothing

- Key intuition: A non-occurring 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 collect $T = \sum_t \mathrm{cf}_t$: : the total number of tokens in the collection.

$$P(t/Mc) = cf_t/T$$

- We will use $\hat{P}(t|M_c)$ to “smooth” $P(t|M_d)$ 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

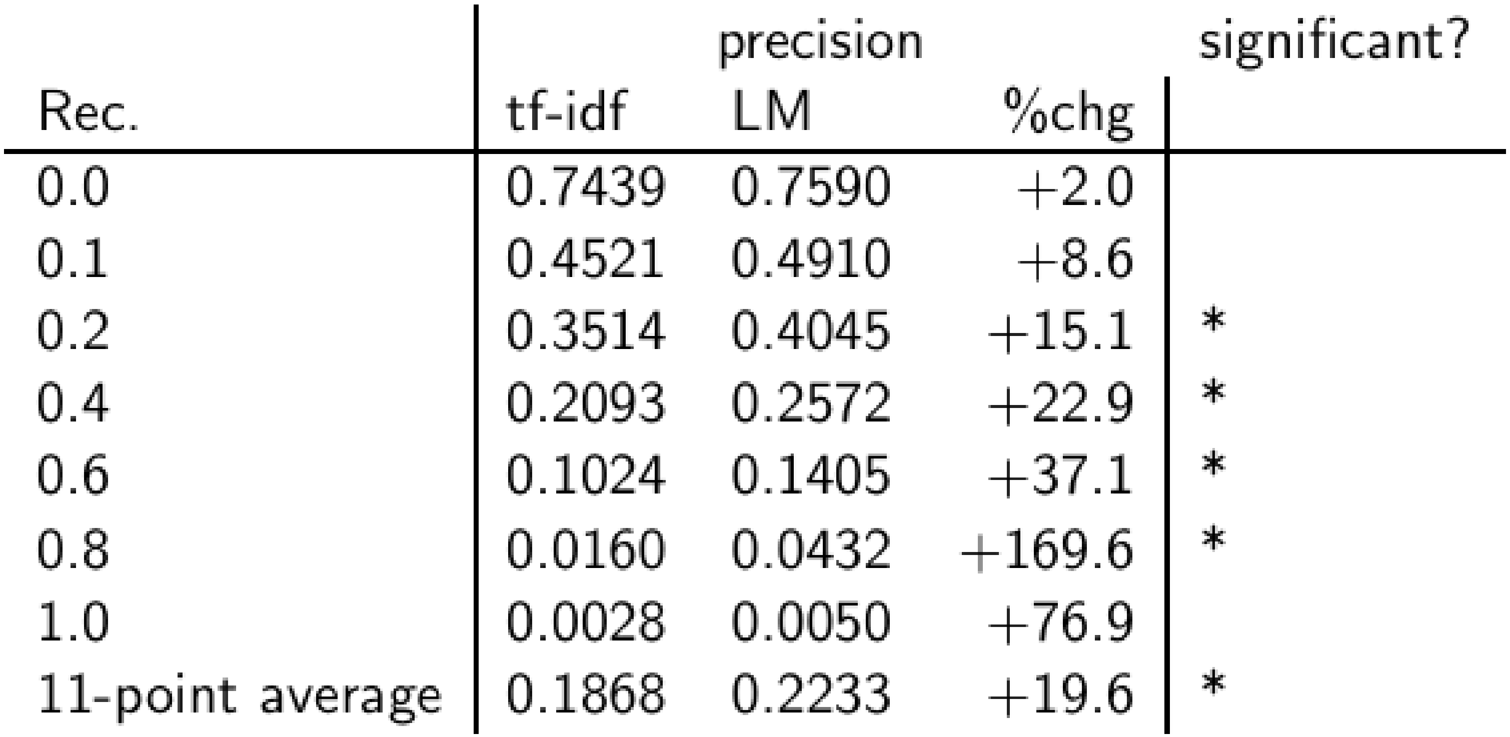
- Collection: d_1 and d_2
- d_1 : *Xerox reports a profit but revenue is down*
- d_2: *Lucene narrows quarter loss but decreases further*
- Query *q*: **revenue down**

- 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: $d_2 > d_1$

Discussions

Vector space (tf-idf) vs. LM



	precision			significant?
Rec.	tf-idf	LM	%chg	
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 experiments . . . . . . but 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 directed 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 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) makes 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.

Thank you

Questions?