Introduction to Information Retrieval

http://informationretrieval.org

IIR 12: Language Models for IR

Mihai Surdeanu

(Based on slides by Hinrich Schütze at informationretrieval.org)

Spring 2017

Overview

Take-away today

- Statistical language models: Introduction
- Statistical language models in IR: Unigram models
- Exercise: Bigram language models
- Discussion: Properties of different probabilistic models in use in IR

Outline

Using language models (LMs) for IR

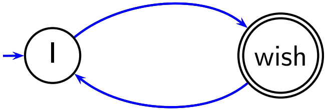
1. LM = language model
2. We view the document as a generative model that generates the query. That is, we compute $P(\text{query}|\text{document})$ for each document.
3. We rank documents in descending order of $P(\text{query}|\text{document})$.

Intro to LMs

Let's forget about IR for a minute.

What is a language model?

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

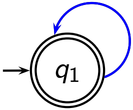


I wish I wish I wish I wish ...

Cannot generate: "wish I wish" or "I wish I"

In natural language: each text is generated by an automaton like this except that these automata are probabilistic.

A probabilistic language model



w	$P(w \mid q_1)$	w	$P(w \mid q_1)$
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

$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2 = 0.0000000000048$

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
		...	...

query: frog said that toad likes frog STOP

$P(\text{query}|M_{d1}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$
$= 0.0000000000048 = 4.8 \cdot 10^{-12}$

$P(\text{query}|M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.2$
$= 0.0000000000120 = 12 \cdot 10^{-12}$

$P(\text{query}|M_{d1}) < P(\text{query}|M_{d2})$
Thus, document d_2 is "more relevant" to the query "frog said that toad likes frog STOP" than d_1 is.

Outline

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 .
- For uniform prior: ranking documents according 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 in BIM and Naive Bayes (next lecture).
- $$P(q|M_d) = P(\langle t_1, \ldots, t_{|q|} \rangle | M_d) = \prod_{1 \le k \le |q|} P(t_k|M_d)$$

  ($|q|$: length of q; t_k: the token occurring at position k in q)
- This is equivalent to:

  $$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\mathrm{tf}_{t,q}}$$
- $\mathrm{tf}_{t,q}$: term frequency (# occurrences) of t in q

Parameter estimation

- Missing piece: Where do the parameters $P(t|M_d)$ come from?
- Start with maximum likelihood estimates (as we did for BIM and will do for Naive Bayes)
- $$\hat{P}(t|M_d) = \frac{\mathrm{tf}_{t,d}}{|d|}$$

  ($|d|$: length of d; $\mathrm{tf}_{t,d}$: # occurrences of t in d)
- As before, 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 $P(q|M_d) = 0$. – Thats's bad.
- We need to smooth the estimates to avoid zeros.

Smoothing

For BIM we saw the smoothing model where we would add a constant (0.5) to all counts. We will revisit that soon. But, now, let's look at a couple of different approaches.

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; $T = \sum_t \mathrm{cf}_t$: the total number of tokens in the collection.
- $$\hat{P}(t|M_c) = \frac{\mathrm{cf}_t}{T}$$
- We will use $\hat{P}(t|M_c)$ to "smooth" $P(t|d)$ away from zero.

Jelinek-Mercer smoothing

- Aka “linear interpolation” or “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.

Jelinek-Mercer smoothing: 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 $\lambda = 1/2$

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: $d_1 > d_2$

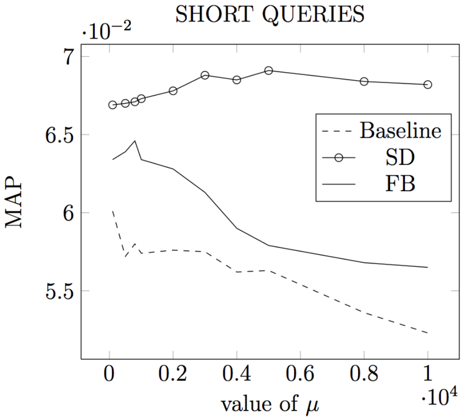
Dirichlet smoothing

- $$\hat{P}(t|d) = \frac{\mathrm{tf}_{t,d} + \alpha\hat{P}(t|M_c)}{L_d + \alpha}$$
- Intuition: Before having seen any part of the document we start with the background distribution as our estimate.
- As we read the document and count terms we update the background distribution.
- The weighting factor α determines how strong an effect the prior has.

Jelinek-Mercer or Dirichlet?

- Dirichlet performs better for keyword, i.e., short, queries, Jelinek-Mercer performs better for verbose queries.
- Both models are sensitive to the smoothing parameters – you shouldn't use these models without parameter tuning.

Sensitivity of Dirichlet to smoothing parameter



μ is the Dirichlet smoothing parameter (called α on the previous slides)

Outline

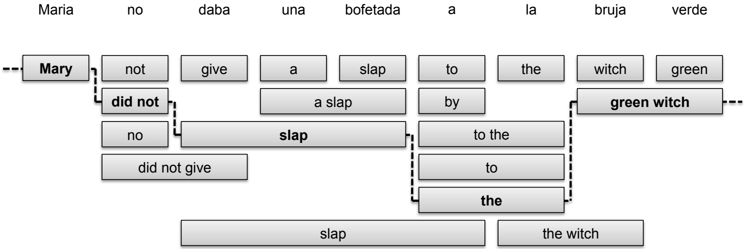
Bigram language models

- The word independence assumption is “unreasonably effective” for IR
- But for many other applications it is impossible to make. What applications are these?
- How does a LM look if the word independence assumption is not made?

Bigram language models

- The word independence assumption is “unreasonably effective” for IR
- But for many other applications it is impossible to make. What applications are these?
- How does a LM look if the word independence assumption is not made?
- Exercise: let’s design such a LM together!. Fundamentally, how do we compute $P(q|d)$, and how do we smooth the probabilities involved?

SMT: The role of the LM



(slide by Jordan Boyd-Graber)

Outline

1. Language models
2. Language Models for IR
3. Bigram Language Models
4. Discussion

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.

Vector space vs BM25 vs LM

- BM25/LM: based on probability theory
- Vector space: based on similarity, a geometric/linear algebra notion
- Term frequency is directly used in all three models.
  - LMs: raw term frequency, BM25/Vector space: more complex
- Length normalization
  - Vector space: Cosine normalization
  - LMs: probabilities are inherently length normalized
  - BM25: tuning parameters for optimizing length normalization
- idf: BM25/vector space use it directly.
- LMs: Mixing term 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.
- Collection frequency (LMs) vs. document frequency (BM25, vector space)

Language models for IR: Assumptions

- Simplifying assumption: Terms are conditionally independent.
  - Again, vector space model (and Naive Bayes) make the same assumption.
  - There are language models that do not make this assumption!
- Cleaner statement of assumptions than vector space
- Thus, better theoretical foundation than vector space
  - ... but "pure" LMs perform much worse than "tuned" LMs.

Take-away today

- Statistical language models: Introduction
- Statistical language models in IR: Unigram models
- Exercise: Bigram language models
- Discussion: Properties of different probabilistic models in use in IR