

Language models

a.a. 2020-2021

Course of Information Retrieval

CdLM in Computer Science

University of Rome Tor Vergata

Prof. Giorgio Gambosi

Derived from slides produced by C. Manning and by H. Schütze



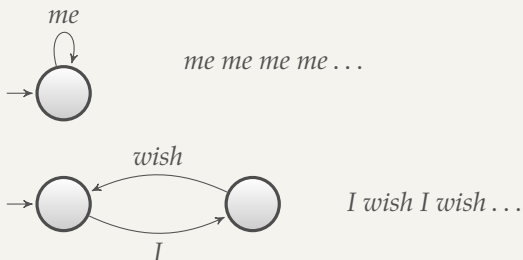
Using language models (LMs) for IR

- We view the document in terms of as a generative model that generates the query
- What we need to do:
 - Define the precise generative model we want to use
 - Estimate parameters (different parameters for each document's model)
 - Smooth to avoid zeros
 - Apply to query and find document most likely to have generated the query
 - Present most likely document(s) to user

What is a language model?

- Assume we are reading (or generating) a document d term by term
- We can view a language model M_d for d as a way to determine the next term which will be read (generated)

We can view the language model as a **finite state automaton**, where the transitions between states are associated to terms



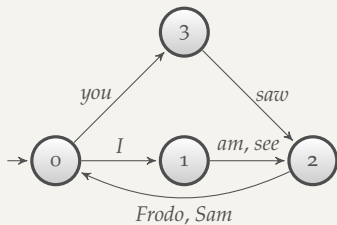
Cannot generate: "II", " wish wish wish" or "wish I wish": history counts

Probabilistic language models

Each document was generated by a different automaton like this, except that these automata are **probabilistic**.

- For each node, a probability distribution is defined on all transitions
- A document corresponds to (is generated as) a sequence of random sample on such distributions

A probabilistic language model

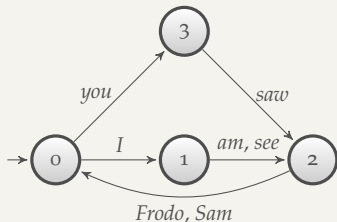


$P(\text{term}|\text{state})$

t	$p(t 0)$	$p(t 1)$	$p(t 2)$	$p(t 3)$
<i>I</i>	0.4			
<i>you</i>	0.4			
<i>am</i>		0.5		
<i>see</i>		0.5		
<i>saw</i>				1
<i>Frodo</i>			0.8	
<i>Sam</i>			0.2	
STOP	0.2			

- This is a probabilistic finite-state automaton and the transition distribution for its states 0, 1, 2, 3.
- **STOP** is not a word, but a special symbol indicating that the automaton stops.

A probabilistic language model



$P(\text{term}|\text{state})$

t	$p(t 0)$	$p(t 1)$	$p(t 2)$	$p(t 3)$
<i>I</i>	0.4			
<i>you</i>	0.4			
<i>am</i>		0.5		
<i>see</i>		0.5		
<i>saw</i>				1
<i>Frodo</i>			0.8	
<i>Sam</i>			0.2	
STOP	0.2			

- possible sequence generated:

w	<i>I</i>	<i>am</i>	<i>Frodo</i>	<i>you</i>	<i>saw</i>	<i>Sam</i>	STOP
$P(t s)$	0.4	0.5	0.8	0.4	1	0.2	0.2

Total sequence probability = 0.00256

A probabilistic unigram language model

A simple version of language models, that we will consider here, is provided by the case when there is a unique state, hence $p(t|s) = p(t)$ for each term



$P(\text{term})$	
t	$p(t)$
<i>I</i>	0.1
<i>you</i>	0.05
<i>am</i>	0.1
<i>see</i>	0.15
<i>saw</i>	0.1
<i>Frodo</i>	0.3
<i>Sam</i>	0.1
STOP	0.1

- possible sequence generated:

w	Frodo	I	Sam	Sam	saw	see	Frodo	STOP
$P(t s)$	0.3	0.1	0.1	0.1	0.15	0.2	0.3	0.1

Total sequence probability = $2.7 \cdot 10^{-7}$

A different language model for each document

M_{d_1} : language model of d_1

t	$p(t)$	t	$p(t)$
STOP	.1	I	.15
you	.2	am	.05
see	.05	saw	.05
Frodo	.2	Sam	.2

M_{d_2} : language model of d_2

t	$p(t)$	t	$p(t)$
STOP	.2	I	.1
you	.1	am	.05
see	.15	saw	.2
Frodo	.15	Sam	.05

query: Frodo saw Sam STOP

$$p(\text{query}|M_{d_1}) = 0.2 \cdot 0.05 \cdot 0.2 \cdot 0.1 = 2 \cdot 10^{-4}$$

$$p(\text{query}|M_{d_2}) = 0.15 \cdot 0.2 \cdot 0.05 \cdot 0.2 = 3 \cdot 10^{-4}$$

$p(\text{query}|M_{d_1}) < p(\text{query}|M_{d_2})$: thus, document d_2 is “more relevant” to the query “Frodo saw Sam STOP ” than d_1 is.

Using language models in IR

- Each document is treated as (the basis for) a language model.
- Given a query q :
 - We wish to rank documents by

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

- $p(q)$ is the same for all documents, so ignore
- $p(d)$ is the prior – often treated as the same for all d
 - But we could give a higher prior to documents which are relevant wrt some other measure, e.g., those with high PageRank.
- $p(q|d)$ is **the probability of q given d**
- For uniform prior: ranking documents according to $p(q|d)$ and $p(d|q)$ is equivalent.

We may see $p(q|d)$ as the probability that the document the user had in mind when she was formulating the query was in fact this one.

- 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 their respective document models (probability distributions)
- That is, we rank according to $P(q|d)$.
- In general, a document model structure (type of probability distribution) is assumed and its parameters values are derived, for each document, from its content

How to compute $P(q|d)$

- We make the **Naive Bayes** conditional independence assumption:

$$p(q|M_d) = p(\langle w_1, \dots, w_{|q|} \rangle | M_d) = \prod_{i=1}^{|q|} p(w_k | M_d)$$

$|q|$: length of q ; w_k : token t occurring at position k in q

How to compute $P(q|d)$

Where do the parameters $p(w_k|M_d)$ come from?

- **Likelihood**: this is the probability of data given a model, in this case $p(q|M_d)$
 - For fixed data, this provides a measure associated to each model instance (parameter values): the probability that such data are generated in the probabilistic framework defined by the model instance (for example, probability distribution)
 - This can be seen as “how much” a model instance explains the given data

How to compute $P(q|d)$

An hypothesis on the model structure (and the generation process) must be assumed.

- hypothesis: all terms have an associated probability to be the next word generated; this probability is independent from previous occurrences
- the probability of observing k occurrences of term t in the query q is given by the **binomial distribution**

$$p(\text{tf}_{t,q} = k) = \frac{|q|!}{k!(|q| - k)!} p^k (1 - p)^{|q| - k}$$

- the probability of observing k_1, k_2, \dots, k_m occurrences of all terms t_1, \dots, t_m in the query q is given by the **multinomial distribution**

$$p(\text{tf}_{t_i,q} = k_i, i = 1, \dots, m) = \frac{|q|!}{\prod_{i=1}^m k_i!} \prod_{i=1}^m p_i^{k_i}$$

How to compute $P(q|d)$

- That is, for each document d ,

$$p(\text{tf}_{t_i,q} = k_i, i = 1, \dots, m | M_d) \approx \prod_{t \in q} p(t | M_d)^{\text{tf}_{t,q}}$$

since the multiplying factor

$$\frac{|q|!}{\prod_{i=1}^m \text{tf}_{t_i,q}!}$$

is independent from the document

- here M_d is an m -dimensional array

$$M_d = [p_1, p_2, \dots, p_m]$$

with $\sum_{i=1}^m p_i = 1$ and $p(t_i | M_d) = p_i$

Parameter estimation

- The probability of the term in the document model, estimated by maximum likelihood is

$$\hat{p}_i = \hat{p}(t_i|M_d) = \frac{\text{tf}_{t_i,d}}{|d|}$$

- $|d|$: length of d
- $\text{tf}_{t_i,d}$: #occurrences of t_i in d

Different models

Different hypotheses on the distribution (generative process) provide different estimations.

- **Multiple Poisson:** we assume a dependency exists between occurrences of a term. This is formalized by a Poisson distribution

$$p(\text{tf}_{t,q} = k) = \frac{e^{-\lambda|q|}(\lambda|q|)^k}{k!}$$

where $\lambda|q|$ is the expected number of occurrences of t in q

- for the whole query

$$p(\text{tf}_{t_i,q} = k_i, i = 1, \dots, m | M_d) = \prod_{i=1}^m \frac{e^{-\lambda_i|q|}(\lambda_i|q|)^{k_i}}{k_i!}$$

- here M_d is an m -dimensional array

$$M_d = [\lambda_1, \lambda_2, \dots, \lambda_m]$$

Smoothing in the multinomial model

- We have a problem with zeros: a single t with $p(t|M_d) = 0$ will make $p(q|M_d) = \prod p(t|M_d) = 0$
- We would give a single term “veto power”.
- For example, for query [Frodo goes to mount Doom] a document about “Frodo Sam Doom” would have $p(q|M_d) = 0$
- We need to **smooth** the estimates to avoid zeros.

Laplace smoothing

- Key intuition: A nonoccurring term is possible (even though it didn't occur), so we don't want to assign 0 probability to it
- We may avoid the zero probability case by adding a constant value, such as 1, to the count $\text{tf}_{t,d}$ in the maximum likelihood estimation: the numerator of the estimation ratio is now $\text{tf}_{t,d} + 1$
- This eliminates the zero probability case, but makes the normalization wrong, that is $\sum_t \text{tf}_{t,d} > |d|$. This can be avoided by summing M , the overall number of terms to $|d|$ at the denominator

$$\hat{p}(t|M_d) = \frac{\text{tf}_{t,d} + 1}{|d| + M}$$

Smoothing through collection model

- We may estimate its probability of a term in a document model by looking at the whole collection
- Let us consider the collection model M_c the collection model: we may estimate
- The maximum likelihood estimate of the probability of the term in the whole collection is given by

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

where cf_t is the number of occurrences of t in the collection and $T = \sum_t cf_t$ is the total number of tokens in the collection.

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

Jelinek-Mercer smoothing

The estimated probability of the term wrt the document is defined as a linear combination of the probability according to the document model and the probability according to the collection model

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

λ is a hyper-parameter which tunes the relevance of the document model wrt the collection model

- Mixtures of two distributions
- Correctly setting λ is very important for good performance

Jelinek-Mercer smoothing

Assuming the conditional independence of terms,

$$p(q|d) = \prod_{1 \leq k \leq |q|} \left(\lambda p(t_k|M_d) + (1 - \lambda)p(t_k|M_c) \right)$$

- Basic idea: we model the case that the user has a document in mind and generates the query from this document.
- - 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

Example

Collection:

- d_1 : “Frodo and Sam reached mount Doom with the help of Gollum”
- d_2 : “Gollum was attracted by the One Ring”

Query q : “Gollum Ring”

- Use mixture model with $\lambda = 1/2$
- $|d_1| = 11, |d_2| = 7, T = 18$
- $P(q|d_1) = [(1/11 + 2/18)/2] \cdot [(0/11 + 1/18)/2] \approx 0.0028$
- $P(q|d_2) = [(1/7 + 2/18)/2] \cdot [(1/7 + 1/18)/2] \approx 0.0125$
- Ranking: $d_2 > d_1$

Exercise: Compute ranking

- Collection: d_1 and d_2
- d_1 : Frodo had a small sword and a coat
- d_2 : The Shire was a small region in the west of Middle Earth
- Query q : west small
- Use mixture model with $\lambda = 1/2$
- $|d_1| = 8, |d_2| = 12, T = 20$

t	tf_{t,d_1}	tf_{t,d_2}	cf_t
west	0	1	1
small	1	1	2

- ...

Dirichlet smoothing

This is a bayesian approach: M_d is a random variable, with an associated distribution.

- a **prior distribution** $p(M_d)$ is defined from the collection model M_c : it is a distribution of probabilities of m -dimensional vectors (summing to 1) with expected value M_c and other parameters predefined
- by observing the document d content, a **posterior distribution** $p(M_d|d)$ is derived by applying Bayes rule

$$p(M_d|d) = \frac{p(d|M_d)p(M_d)}{p(d)}$$

- $p(d|M_d)$ is a multinomial distribution, by hypothesis
- $p(M_d)$ is chosen in such a way that $p(M_d)$ and $p(M_d|d)$ are of the same type (conjugate to the multinomial)
- typical choice, **Dirichlet distribution**

Dirichlet smoothing

- Dirichlet distribution

$$\text{Dir}(p_1, \dots, p_m | \alpha_1, \dots, \alpha_m) = \frac{\Gamma(\sum_{i=1}^m \alpha_i)}{\prod_{i=1}^m \Gamma(\alpha_i)} \prod_{i=1}^m p_i^{\alpha_i - 1}$$

- assume the prior distribution is defined as $\alpha_i = \mu \cdot p(t_i | M_c)$, where $p(t_i | M_c)$ is estimated as before
- under this hypothesis, it can be proved that the posterior distribution is then

$$\begin{aligned} p(M_d | d) &= p(p(t_1 | M_d), \dots, p(t_m | M_d) | d, M_c, \mu) \\ &\simeq \prod_{i=1}^m p(t_i | M_d)^{\text{tf}_{t_i, d} + \mu \cdot p(t_i | M_c) - 1} \end{aligned}$$

a Dirichlet with parameters $\alpha_i = \text{tf}_{t_i, d} + \mu \cdot p(t_i | M_c)$

Dirichlet smoothing

The resulting document model M_d is the expectation of the posterior distribution

- in general, in a Dirichlet distribution the expectation is the m -dimensional array with components

$$\frac{\alpha_i}{\sum_{k=1}^m \alpha_k}$$

- as a consequence, M_d is the m -dimensional array with components

$$\frac{\text{tf}_{t_i,d} + \mu \cdot p(t_i|M_c)}{\sum_{k=1}^m (\text{tf}_{t_k,d} + \mu \cdot p(t_k|M_c))} = \frac{\text{tf}_{t_i,d} + \mu \cdot p(t_i|M_c)}{|d| + \mu}$$

Dirichlet smoothing

- 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 μ determines how strong an effect the prior has.

Jelinek-Mercer or Dirichlet?

- Dirichlet performs better for keyword 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.

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: document vectors normalized
 - 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)