Modern Information Retrieval

Language Models for Information Retrieval[1]

Hamid Beigy

Sharif university of technology

April 4, 2025

[1] Some slides have been adapted from slides of Manning, Yannakoudakis, and Schütze.

Table of contents

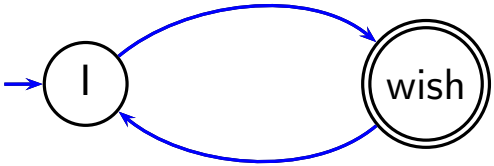
Introduction

Introduction

1. An language model is a model for how humans generate language.
2. We view the document as a generative model that generates the query.
3. What we need to do?
   - Define the precise generative model we want to use.
   - Estimate model parameters.
   - Smooth to avoid zeros.
   - Apply to query and find documents most likely to have generated the query.
   - Present most likely document(s) to user.

What is a language model?

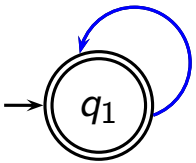
1. We can view a finite state automaton as a deterministic language model.



2. This automaton generates documents such as I wish I wish I wish I wish ....
3. But it can't generate documents such as I wish I or wish I wish.
4. Each document was generated by a different automaton like this except that these automata are probabilistic.

A probabilistic language model

1. Consider the following probabilistic automaton.



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
		...	...

2. This is a one-state probabilistic finite-state automaton (a unigram language model) and the state emission distribution for its one state q_1.
3. STOP is not a word, but a special symbol indicating that the automaton stops.
4. "frog said that toad likes frog STOP"

$$\begin{aligned} P(\textit{string}) &= 0.01 \times 0.03 \times 0.04 \times 0.01 \times 0.02 \times 0.01 \\ &\quad \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.2 \\ &\approx 0.0000000000048 \end{aligned}$$

Probabilistic Approach to IR

Language Model

1. A **language model** p is a distribution over sequences of tokens $x_{1:L}$

$$p(\mathit{The}, \mathit{mouse}, \mathit{ate}, \mathit{the}, \mathit{cheese}).$$

2. A language model can be used to score sequences.
3. It can also be used to perform conditional generation of a completion given a piece of text.

$$\mathit{the\ mouse\ ate} \rightsquigarrow \mathit{the\ cheese}.$$

4. Suppose we take a corpus of text $x_{1:L}$, for example:

$$\mathit{the\ mouse\ ate\ the\ cheese}$$

5. We can ask: **what is the probability the language model assigns to it?**

$$p(\mathit{the\ mouse\ ate\ the\ cheese})$$

6. We can break down the the joint probability into the product of the conditional probabilities for each token by the chain rule:

$$p(x_{1:L}) = \prod_{i=1}^{L} p(x_i \mid x_{1:i-1}).$$

Unigram Language Model

1. How do we build probabilities over sequences of terms?

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_1 t_2)P(t_4|t_1 t_2 t_3)$$

2. A unigram language model throws away all conditioning context, and estimates each term independently. As a result:

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$$

3. A bigram language model conditions on the previous term

$$P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_4|t_3)$$

What is a document language model?

1. A model for how an author generates a document on a particular topic.
2. The document itself is just one sample from the model (i.e., ask the author to write the document again and he/she will invariably write something similar, but not exactly the same).
3. A probabilistic generative model for documents.

Two Unigram Document Language Models

1. Consider two documents d_1 and d_2.

Language model for 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 for d_2

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

2. Consider query: $q =$ "frog said that toad likes frog STOP"
3. We have $p(q|M_{d_1}) = 0.0000000000048$
4. We have $p(q|M_{d_2}) = 0.0000000000120$
5. Since $p(q|M_{d_1}) < p(q|M_{d_2})$, hence document d_2 is more relevant to the query.

Query Likelihood Method

1. Users often pose queries by thinking of words that are likely to be in relevant documents.
2. The query likelihood approach uses this idea as a principle for ranking documents.
3. We construct from each document d in the collection a language model M_d .
4. Given a query q, we rank documents by the likelihood of their document models M_d generating q: $P(q|M_d)$

Query Likelihood Method

1. Each document is treated as (the basis for) a language model.
2. Given a query q
3. Rank documents based on $P(d|q)$

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

4. $P(q)$ is the same for all documents, so we ignore it
5. $P(d)$ is the prior – often treated as the same for all d

   But we can give a higher prior to high-quality documents
6. $P(q|d)$ is the probability of q given d.
7. For uniform prior: ranking documents according according to $P(q|d)$ and $P(d|q)$ is equivalent.

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

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

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

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

2. This is equivalent to:

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

$\mathrm{tf}_{t,q}$: term frequency (#occurrences) of t in q

3. Multinomial model (omitting constant factor)

1. Missing piece: Where do the parameters $P(t|M_d)$ come from?
2. Start with maximum likelihood estimates

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

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

3. We have a problem with zeros, a single t with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod_t P(t|M_d)$ zero.
4. We need to smooth the estimates to avoid zeros.

Smoothing

1. Let
   - M_c be the collection model;
   - cf_t be the number of occurrences of t in the collection;
   - $T = \sum_t \mathrm{cf}_t$ be the total number of tokens in the collection.
2. We can use

$$\hat{P}(t|M_c) = \frac{\mathrm{cf}_t}{T}$$

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

Jelinek-Mercer smoothing

1. We can use a mix of the probability from the document with the general collection frequency of the word.

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

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

Jelinek-Mercer smoothing: Summary

1. Let

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

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

Example

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

Dirichlet smoothing (Bayesian smoothing)

1. In Dirichlet smoothing, we use

$$\hat{P}(t|d) = \frac{\mathrm{tf}_{t,d} + \alpha \hat{P}(t|M_c)}{L_d + \alpha}$$

2. The background distribution $\hat{P}(t|M_c)$ is the prior for $\hat{P}(t|d)$.
3. Intuition: Before having seen any part of the document we start with the background distribution as our estimate.
4. As we read the document and count terms we update the background distribution.
5. The weighting factor α determines how strong an effect the prior has.

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

1. BM25/LM: based on probability theory
2. Vector space: based on similarity, a geometric/linear algebra notion
3. Term frequency is directly used in all three models.
   - LMs: raw term frequency, BM25/Vector space: more complex
4. Length normalization
   - Vector space: Cosine or pivot normalization
   - LMs: probabilities are inherently length normalized
   - BM25: tuning parameters for optimizing length normalization
5. idf: BM25/vector space use it directly.
6. 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.
7. Collection frequency (LMs) vs. document frequency (BM25, vector space)

Language models for IR: Assumptions

1. 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.
2. Simplifying assumption: Terms are conditionally independent.
   - Again, vector space model (and Naive Bayes) make the same assumption.
3. Cleaner statement of assumptions than vector space
4. Thus, better theoretical foundation than vector space
   - But "pure" LMs perform much worse than "tuned" LMs.

There are three obvious ways to perform retrieval using language models:

1. **Query likelihood retrieval** trains a model on the document and estimates the query's likelihood.
2. **Document likelihood retrieval** trains a model on the query and estimates the document's likelihood. **Queries are very short, so these seem less promising.**
3. **Model divergence retrieval** trains models on both the document and the query, and compares them.

1. The most common way to compare probability distributions is with Kullback-Liebler (KL) divergence.

$$D_{KL}(p||q) = \sum_{e} p(e) \log \frac{p(e)}{q(e)}$$

2. Model divergence retrieval works as follows:
   - Choose a language model for the query, $p(w|q)$.
   - Choose a language model for the document, $p(w|d)$.
   - Rank documents by $-D_{KL}(p(w|d) \;||\; p(w|q))$.

     More divergence means a worse match.

References

Reading

1. Chapter 12 of Information Retrieval Book[2].
2. Section 7.2 of Search Engines - Information Retrieval in Practice Book[3].

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

[3] W. Bruce Croft, Donald Metzler, and Trevor Strohman (2009). *Search Engines - Information Retrieval in Practice*. Pearson Education.

References

- Croft, W. Bruce, Donald Metzler, and Trevor Strohman (2009). *Search Engines - Information Retrieval in Practice*. Pearson Education.
- Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Questions?