Introduction to

Information Retrieval

Lecture 12: Language Models for IR

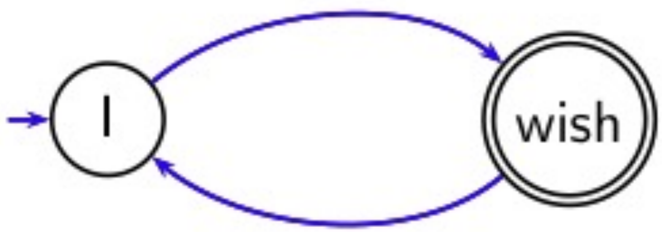
Using language models (LMs) for IR

- LM = language model
- We view the document as a generative model that generates the query.

- *What we need to do:*
- Define the precise generative model we want to use
- Apply to query and find the document(s) that are most likely to have generated the query
- Present most likely document(s) to user

What is a language model?

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

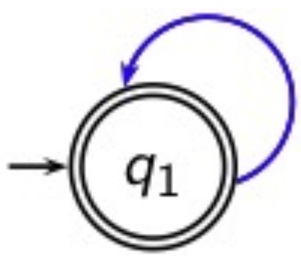


I wish I wish I wish I wish . . .

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

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

$= 0.0000000000048$

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
		...	...

frog said that toad likes frog STOP

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

$P(\text{string}|M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 0.0000000000120 = 12 \cdot 10^{-12}$ $\quad P(\text{string}|M_{d1}) < P(\text{string}|M_{d2})$

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

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 in Web search.
- $P(q|d)$ 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 $P(q|d)$?

- Notation: M_d : the document model

How to compute $P(q|d)$

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

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

($|q|$: 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
- Multinomial model (omitting constant factor)

Parameter estimation

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

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

($|d|$: 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 $P(q|M_d) = \prod P(t|M_d)$
  - We would give a single term "veto power".
  - E.g., 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 in the particular document), . . .
- . . . 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.

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

- We will use $\hat{P}(t|M_c) = \mathrm{cf}_t / T$

Mixture model

- We will use $\hat{P}(t|M_c)$ to "smooth" $P(t|d)$ away from zero.
- $P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$
- 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 1

- Collection of two docs: 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: $d_2 > d_1$

Example 2

- 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: $d_1 > d_2$

Language Models vs. Vector Space

LMs vs. vector space model

- 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: Assumptions

- Simplifying assumption: Queries and documents are objects of same type.
  - May not be true!
  - The vector space model makes the same assumption.
- Simplifying assumption: Terms are conditionally independent.
  - Not true in most cases
  - 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.

LMs vs. vector space model: differences

- 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.

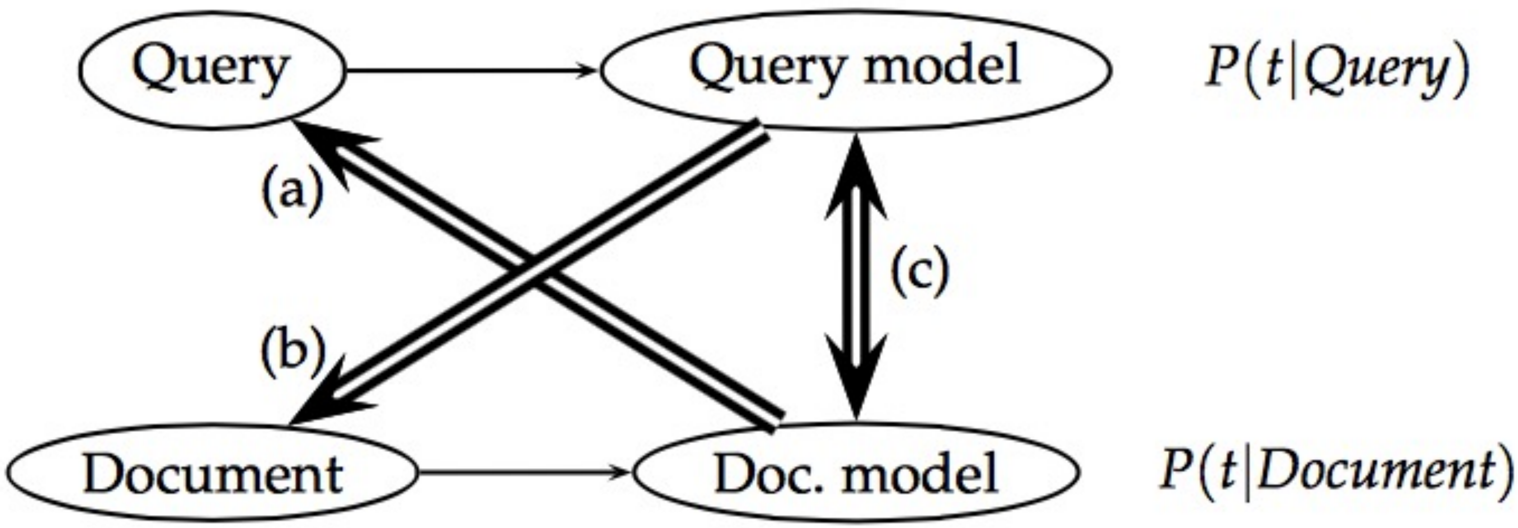
Alternative Language Modeling approaches: Brief discussion

Alternative LM approaches

- Rather than looking at the probability of a document LM generating the query, can look at the probability of a query LM generating the document
  - Challenge: much less text to estimate a LM based on query
  - Advantage: easier to incorporate relevance feedback

- Can make two LMs from the document (M_d) and the query (M_q), and then ask how different these two LMs are
  - Develop a risk minimization approach for retrieval: compute risk of retrieving a document d as relevant to query q
  - Risk can be estimated as the KL divergence of M_d from M_q

Three ways of developing language modeling approach



Kullback-Leibler Divergence

$$R(d;q) = KL(M_d \| M_q) = \sum_{t \in V} P(t|M_q) \log \frac{P(t|M_q)}{P(t|M_d)}$$

Translation Model

- Basic LMs do not consider any deviation in use of language between queries and documents (e.g., synonymous words)

- A translation model allows generation of query words not in a document, by translation to alternate terms with similar meaning
  - Forms the basis of cross-language IR
  - Assume translation model represented by a conditional probability distribution $T(x|y)$ between vocabulary terms
  - Usually built using separate resources such as a thesaurus or bilingual dictionary or a statistical machine translation dictionary

Example: Query Expansion in Language Modeling

Basic Idea: *We assume that the translation model can be represented by a conditional probability distribution T(·|·) between vocabulary terms.*

The form of the translation query generation model:

$$P(q|M_d) = \prod_{t \in q} \sum_{v \in V} P(v|M_d) T(t|v)$$

P(): basic document language model
T(): translation model
v is a term in the vocabulary, but not contained in the query
t is a term contained in the query