Information Retrieval

Language Modeling for IR

Exploring a different IR modeling paradigm

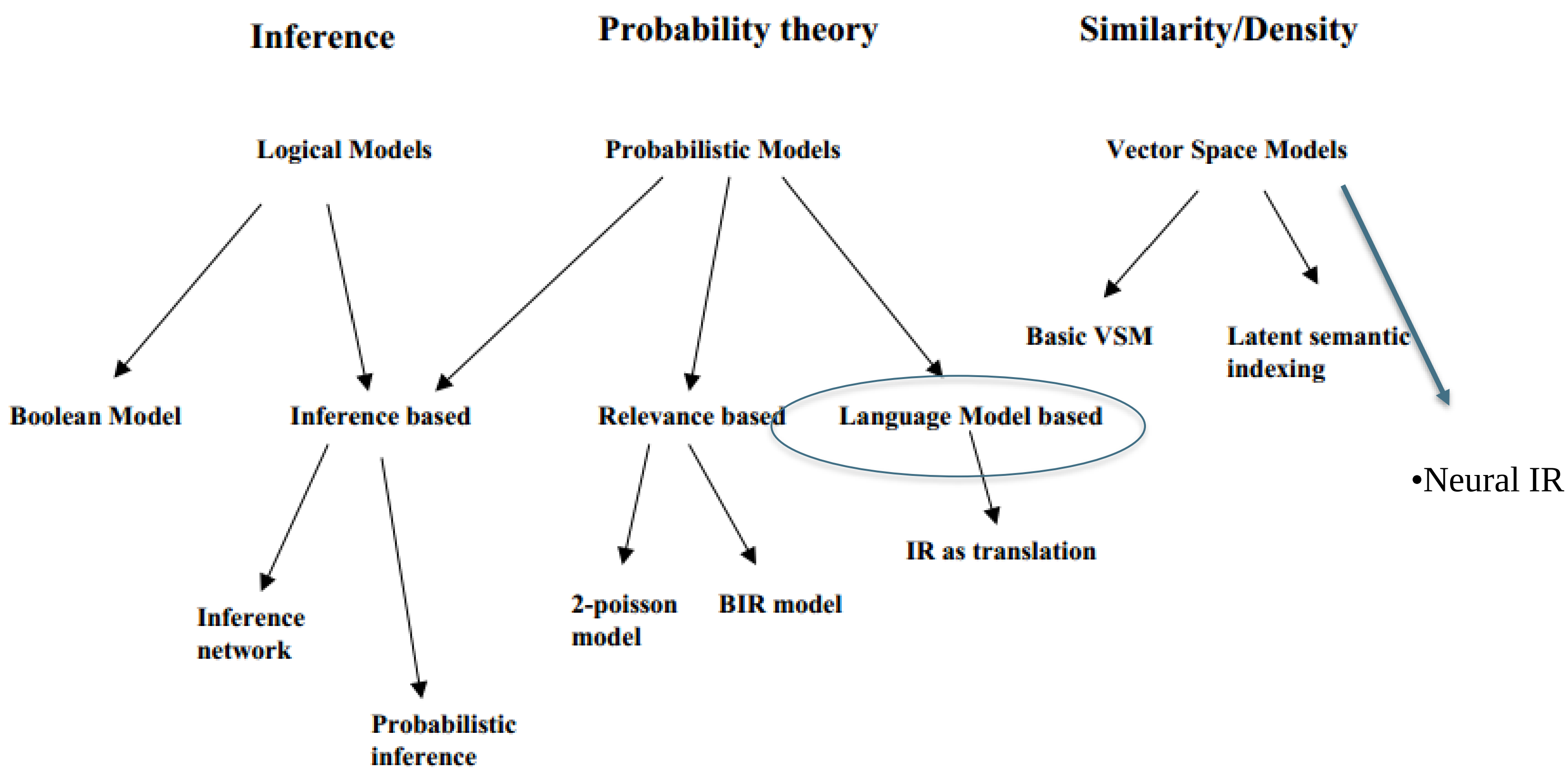


Figure 2.3. Taxonomy of IR models

Introduction to Generative Language Models

What is a language model?

- Simplified statistical model of text
  - Data driven, as opposed to rule based, symbolic models of text.
  - Local (preceding) context predicts following words
    - Ex: "`For he is a jolly good ..`"
    - Ex: "`Barcelona scored 1-0 in the 2`nd `..`"

- LM can be used to compute the <u>probability of observing a sentence</u> given a model of a language (fragment) as opposed to <u>syntactical well-formedness</u> of that string.

Comparing probabilities

- What's the use of these probabilities?
- We can compare the degree of well formedness
  - P('Obama walks in the park' | $M_{English}$) > P('Obama walk in the park' | $M_{English}$)
- We can also compare which model has the best fit for the data
  - P('Ajax won the cup' | M_{sports}) > P('Ajax won the cup' | $M_{politics}$)
- This means that a *topic* of a document or query can also be represented as a language model
  - i.e., words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model

How can we compute P?

- Starting point: generative model
  - Sentence is a series of ordered terms <$t_{0,} t_{1,}$-t_n>
  - Probability of observing term t_i depends on previous terms
  - P("for he is a") = P("for").P("he"|"for").P("is"|"for he").P("a"|"for he is")
  - $P(t_1, t_2, t_3, t_4)$=P(t_1)P($t_2|t_1$)P($t_3|t_1\ t_2$)P($t_4|t_1\ t_2\ t_3$)(chain rule)

$$P(S) = \prod_{i=0}^{n} P(t_i | t_0 .. t_{i-1})$$

Memory

- "Memory" of a practical generative model is usually restricted. Why?
  - E.g. Memory=1: First order Markov model

Application in IR

- Intuition: each document is represented by a language model *D.*

- A user constructs a query *Q* by choosing some terms of which he *assumes* that they occur in relevant documents.
- Rank documents according to

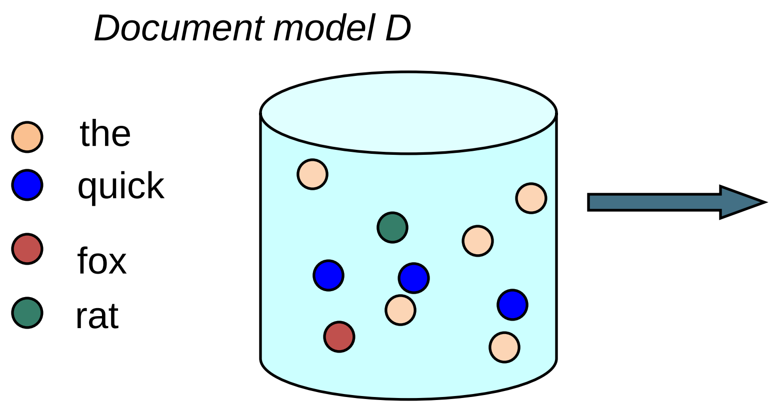
$$\boxed{P(D \mid Q) = P(Q \mid D)P(D) / P(Q) \cong P(Q \mid D)P(D)}$$

  - How probable is Q, when taking a random text sample from D's language model.

- Simple model (memory=0) works surprisingly well!
  - This means that we assume that all terms are chosen independently, which is clearly wrong.
  - dependency models (memory>0) give a small improvement.

Unigram language models: example

- Words are generated independent of the “history”.
  - Urn model: sampling with replacement.



What is P(Q|D)?

P(Q|D)=P(the|D)P(quick|D)P(fox|D) = 0.5x0.3x0.1

P(the|D)=5/10
P(quick|D)=3/10
P(fox|D)=1/10
P(rat|D)=1/10

$$P(q_0 \ldots q_n \mid D) = \prod_{i=0}^{n} P(q_i \mid D)$$

Bag of Words

- For unigram language models, word order is ignored
  - Also known as bag-of-words, hence this is a
- *Multinomial* distribution over vocabulary *M*
  - There are multiple ways to create the same bag of words by generating different sequences (=>factorials)

In case we view a query as a B.O.W.

Multinomial constant:

$$P(d) = \frac{L_d!}{\mathrm{tf}_{t_1,d}!\mathrm{tf}_{t_2,d}!\cdots\mathrm{tf}_{t_M,d}!} P(t_1)^{\mathrm{tf}_{t_1,d}} P(t_2)^{\mathrm{tf}_{t_2,d}} \cdots P(t_M)^{\mathrm{tf}_{t_M,d}}$$

  - If we compare two models for the same document, constant can be left out.
  - In practice this constant is always left out => cross entropy formulation

More formally: Query-Likelihood Model

- Rank documents by the probability that the query could be generated by the document model (i.e. same topic)
- Point of departure: P(D|Q)
- Using Bayes' Rule

$$p(D|Q) \stackrel{rank}{=} P(Q|D)P(D)$$

- Assuming 1) prior is uniform, 2) unigram model

Query Likelihood $$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

Estimating Probabilities

- Obvious estimate for unigram probabilities is

$$P(q_i|D) = \frac{f_{q_i,D}}{|D|}$$

- *Maximum likelihood estimate*
  - makes the observed value of $f_{q_i;D}$ most likely
- *Example:*
  - *D1: "Election of Barack Obama in 2008."*
  - *D2: "US Election 2008: won by senator Obama"*
  - *D3: "Michelle Obama new first lady."*
  - *Q: "election Barack Obama"*
  - *what are the query likelihoods? Do they reflect relevance?*
- If query words are missing from document, score will be zero
  - Missing 1 out of 3 query terms same as missing 2 out of 3

Sparse data problem

- Feature space is large
  - ➔ the number of parameters is extremely high (all words in a language).

- Relatively small amount of data for estimation (just 1 document)
  - This explains why higher order models (bigrams and up) are hardly feasible for IR.

- Solution: “smoothing”

Smoothing

- Document texts are a *sample* from the language model
  - Missing words should not have zero probability of occurring
- *Smoothing* is a technique for estimating probabilities for missing (or unseen) words
  - lower (or *discount*) the probability estimates for words that are seen in the document text
  - assign that “left-over” probability to the estimates for the words that are not seen in the text

Smoothing by discounting

- Laplace

$$P_{laplace}(w) = \frac{c(w) + \alpha}{\sum_{w \in V} c(w) + \alpha |V|}$$

|V| is vocabulary size of language model, typically 10^5

- Is this a proper probability distribution?

- Typical values: α= 0.5

Limitation of discounting

- Problem: all unseen terms are assigned an equal probability
  - Why is this a problem?
- Solution:
  - Interpolation with a more general model
    - E.g. smooth a trigram model with a bigram model, which in turn is smoothed by a unigram model (ASR)
    - Or: smooth a document unigram model with a collection unigram model (*background model)*

Smoothing by linear interpolation

- Estimate for unseen words is $\alpha_D P(q_i|C)$
  - $P(q_i|C)$ is the probability for query word i in the *collection* language model for collection C (background probability)
  - α_D is a parameter

- Estimate for words that occur is

$$(1 - \alpha_D)\ P(q_i|D) + \alpha_D\, P(q_i|C)$$

- Different forms of estimation come from different α_D

Basic ranking formula: JM smoothing

$$P(q_1, q_2, \cdots q_n \mid D) = \prod_{j=1}^{n} P(q_j \mid D)$$

Generative model, term independence

- Add smoothing to P(Q|D)
- α_D is a constant: λ (Jelinek Mercer smoothing)

$$P(q_1, q_2, \cdots q_n \mid D) = \prod_{j=1}^{n} (1-\lambda) P(q_j \mid D) + \lambda P(q_j \mid C)$$

$$\Downarrow$$

$$\log P(q_1, q_2, \cdots q_n \mid D) = \sum_{j=1}^{n} \log\left[(1-\lambda) P(q_j \mid D) + \lambda P(q_j \mid C)\right]$$

- A good value for λ is e.g. 0.85, is this light or heavy smoothing?
- How does the model behave with λ approaching 0 and λ =1?

Where is *tf.idf* Weight?

$P(q_i \mid D)$ $P(q_i \mid C)$

$$
\begin{aligned}
\log P(Q|D) &= \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) \\
&= \sum_{i:f_{q_i,D}>0} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) + \sum_{i:f_{q_i,D}=0} \log(\lambda\frac{c_{q_i}}{|C|}) \\
&= \sum_{i:f_{q_i,D}>0} \log\frac{((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|})}{\lambda\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log(\lambda\frac{c_{q_i}}{|C|}) \\
&\stackrel{rank}{=} \sum_{i:f_{q_i,D}>0} \log\left(\frac{((1-\lambda)\frac{f_{q_i,D}}{|D|}}{\lambda\frac{c_{q_i}}{|C|}} + 1\right)
\end{aligned}
$$

- *proportional to the term frequency,*
- *inversely proportional to the collection frequency*

Dirichlet Smoothing

- α_D depends on document length

$$\alpha_D = \frac{\mu}{|D|+\mu}$$

- Gives probability estimation of

$$p(q_i|D) = \frac{f_{q_i,D} + \mu\frac{c_{q_i}}{|C|}}{|D|+\mu}$$

- and document score

$$\log P(Q|D) = \sum_{i=1}^{n} \log \frac{f_{q_i,D} + \mu\frac{c_{q_i}}{|C|}}{|D|+\mu}$$

Query Likelihood Example

- For the term “president”
  - $f_{qi,D} = 15$, $c_{qi} = 160{,}000$
- For the term “lincoln”
  - $f_{qi,D} = 25$, $c_{qi} = 2{,}400$
- length (number of word occurrences) of document $|d|$ is assumed to be 1,800
- number of word occurrences in the collection is 10^9
  - 500,000 documents times an average of 2,000 words
- $\mu = 2{,}000$

Query Likelihood Example (2)

LOG!

$$
\begin{aligned}
QL(Q,D) &= \log \frac{15 + 2000 \times (1.6 \times 10^5/10^9)}{1800 + 2000} \\
&\quad + \log \frac{25 + 2000 \times (2400/10^9)}{1800 + 2000} \\
&= \log(15.32/3800) + \log(25.005/3800) \\
&= -5.51 + -5.02 = -10.53
\end{aligned}
$$

Negative number because summing logs of small numbers

Query Likelihood Example (5 docs)

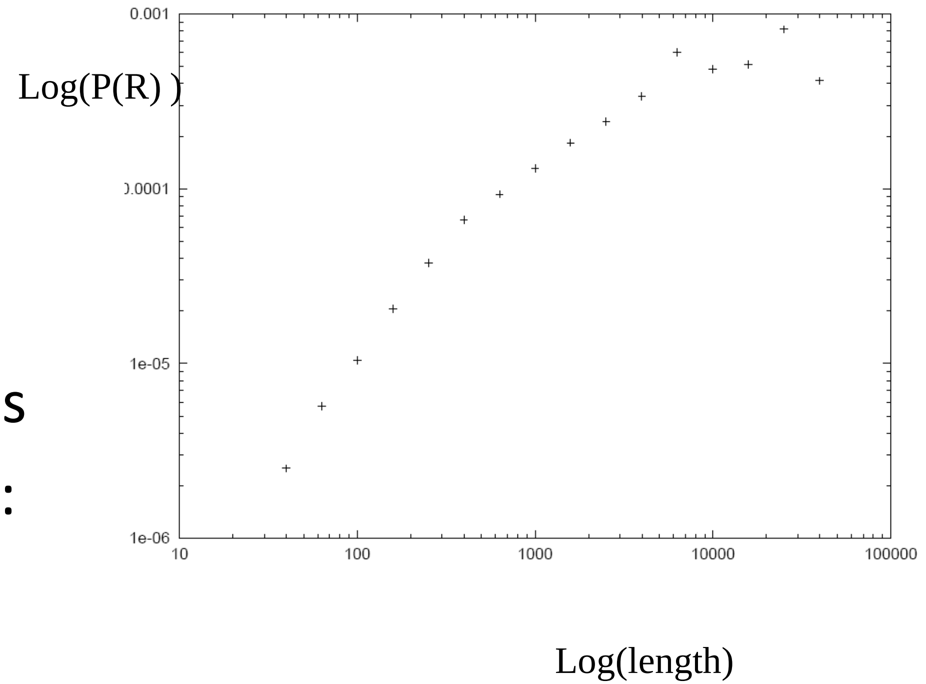
Frequency of "president"	Frequency of "lincoln"	QL score
15	25	-10.53
15	1	-13.75
15	0	-19.05
1	25	-12.99
0	25	-14.40

A document that misses the term lincoln is penalized more than a document that misses the more general term president.

Using the prior for document length normalization

- Almost linear relation between P(R) and document length
- Document priors improve average precision, especially for short queries
- TREC7 ad hoc MAP (1999):
  - Okapi: 0.232
  - LM: 0.241 (JM smoothing)
  - LM+prior: 0.251

RSV += log(length)



Djoerd Hiemstra and Wessel Kraaij, ``Twenty-One at TREC-7: ad-hoc and cross-language track'', *Proceedings of the seventh Text Retrieval Conference TREC-7*, NIST Special Publication 500-242, pages 227-238, 1999

Cross-entropy for monolingual IR

$$P(q_1, q_2, \cdots q_n \mid D) = \prod_{j=1}^{n} P(q_j \mid D)$$

~ rank equiv.

From tokens to type counts in a bag of words*, taking logs

$$\log P(Q \mid D) = \sum_{w_i \in Q} c(w_i, Q) \log P(w_i \mid D)$$

~ rank equiv.

Normalizing by |Q|

$$-H(w; D) = \sum_{i=1}^{n} P(w_i \mid Q) \log P_{sm}(w_i \mid D)$$

Which in fact equals the

negated cross-entropy

*(*ignoring the multinomial coefficient, which is a constant that does not change the ranking)*

Summary of the lecture

- Statistical language modeling offers a *clean*, *competitive* and *extensible* framework for a range of (IR and NLP) tasks

- *Parameter estimation* techniques accommodating the *sparse data problem* are key to its success

Comparing models

	Effectiveness	Efficiency	Explain/Use	Parsimony
Boolean	No ranking	++ (presence only)	++/-	+
Vector Space (lnc.ntc)	Ranking	++ (presence only)	fair	fair
Lnu.ltu	Ranking ++	++ (presence only)	fair	- (more parameters)
Neural IR	Precision oriented ++ (needs BM25 1st stage)	-- (all terms)	Better than LSI	-
BIM	No Tf!	++ (presence only)	Theory is clear	0 hyperparameters
BM25	++	++ (presence only)	Complex derivation	2 hyperparameters
LM	++	++ (presence only)	Simple derivation	1 hyperparameter

More references

- **TREC Experiment and Evaluation in Information Retrieval**
  Edited by Ellen M. Voorhees and Donna K. Harman
  MIT Press
- **Language Modeling for Information Retrieval**
  Series: The Kluwer International Series on Information Retrieval, Vol. 13
  Croft, W. Bruce; Lafferty, John (Eds.)
  2003, 264 p., Hardcover
  ISBN: 1-4020-1216-0
- **Relevance-Based Language Models**, by Lavrenko, V. and Croft, W.B., in Proceedings of the 24th annual international ACM SIGIR conference, New Orleans, LA, September 7 - 12, 2001.
- **Using Language Models for Information Retrieval**, by Djoerd Hiemstra, Ph.D. Thesis, Centre for Telematics and Information Technology, University of Twente, January 2001, ISSN 1381-3617 (no. 01-32), ISBN 90-75296-05-3
- **Transitive probabilistic CLIR models**, by Wessel Kraaij and Franciska de Jong. In Proceedings of RIAO 2004, 2004 .
- **Variations on Language Modeling for Information Retrieval**, by Wessel Kraaij. PhD thesis, University of Twente, June 2004
- **Foundations of Statistical Natural Language Processing**, Manning and Schuetze, MIT press. http://nlp.stanford.edu/fsnlp/

End of Lecture