universität koblenz
weiter:denken

Web Information Retrieval

Language Models for IR

(SOSE 2023, 17.05.2023)

Frank Hopfgartner, Stefania Zourlidou
Institute for Web Science and Technologies

Credit for these slides

These slides have been adapted from

- Web IR (Zeyd Boukhers-WeST, SOSE 2020)

Recapitulation

- Boolean Model: Pros and Cons
- Ranked retrieval model
- Documents scoring
  - TF-IDF
- Query-document matching
  - Jaccard
  - Cosine
- Vector Space Model
- Relevance feedback

Objectives of this lecture

- Language models for Information Retrieval
  - Language Models
  - Query likelihood model
  - Document likelihood model
  - Comparison model

universität koblenz
weiter:denken

1. Language Models

Language Models (LM)?

- Formal language
  - Alphabet: finite set of symbols
  - Word: finite concatenation of symbols
  - Language: subset of all possible words over alphabet
- Language Model (LM)
  - Generative model for formal languages
  - Used to generate / recognize words of a language
  - Probabilistic model
- Applications (typically)
  - Speech recognition
  - POS tagging
  - Digitization of hand writing

What is a language model ?

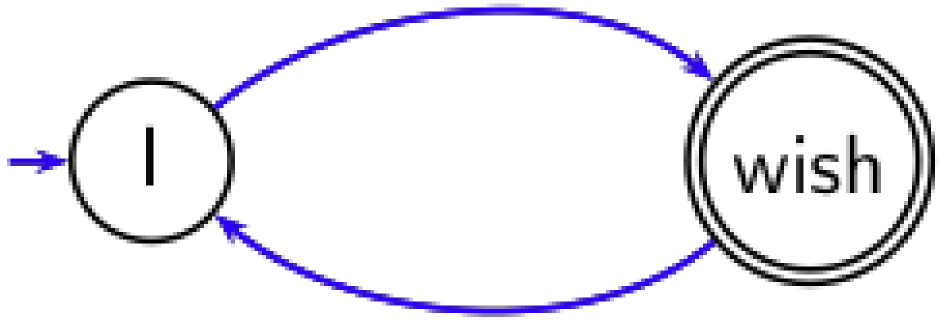
- Probability distribution over strings of text
  - How likely is a given string in a given “language”
  - E.g., consider probability for the following four strings
    - p1 = P(“a very warm day”)
    - p2 = P(“day warm a very”)
    - p3 = P(“Sehr warm day”)
    - p4 = P(“теплый день”)

– English: p1 > p2 > p3 > p4

- Depends on what “language” we are modeling
  - In IR, mostly it is assumed that p1 == p2

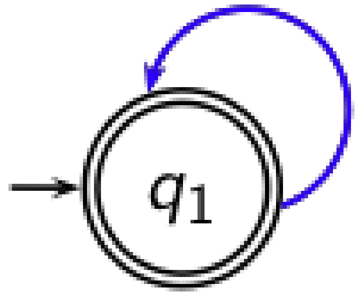
What is a language model ?

- We can view a finite state automaton as a deterministic language model
  - I wish
  - I wish I wish
  - I wish I wish 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

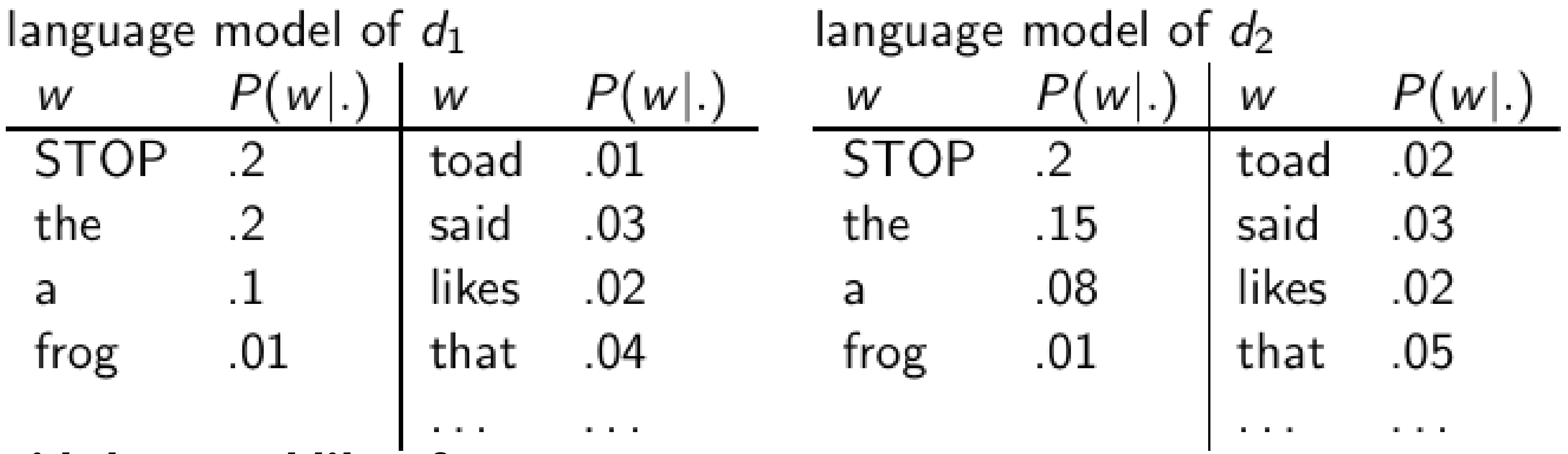


w	$P(w\|q_1)$	w	$P(w\|q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
		...	...

- This is an 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”***

$$P(string) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02$$
$$= 0.0000000000048$$

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***
  - $P(string|Md_1) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 0.0000000000048 = 4.8 \cdot 10^{-12}$
  - $P(string|M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 0.0000000000120 = 12 \cdot 10^{-12}$
  - $P(string|Md_1) < P(string|Md_2)$
- Thus, document d_2 is “more relevant” to the string “frog said that toad likes frog STOP” than d_1 is

Types of language models

- $P(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_1 t_2)P(t_4|t_1 t_2 t_3)$
- $P_{\text{uni}}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2)P(t_3)P(t_4)$
- $P_{\text{bi}}(t_1 t_2 t_3 t_4) = P(t_1)P(t_2|t_1)P(t_3|t_2)P(t_4|t_3)$

- Remember the chain rule
  - $P(A, B) = P(A \cap B) = P(A|B) * P(B) = P(B|A) * P(A)$

Language model for IR: basic concept

- Users have a reasonable idea of terms that are likely to occur in document of interest

- They use words that they expect to find in matching documents as their query

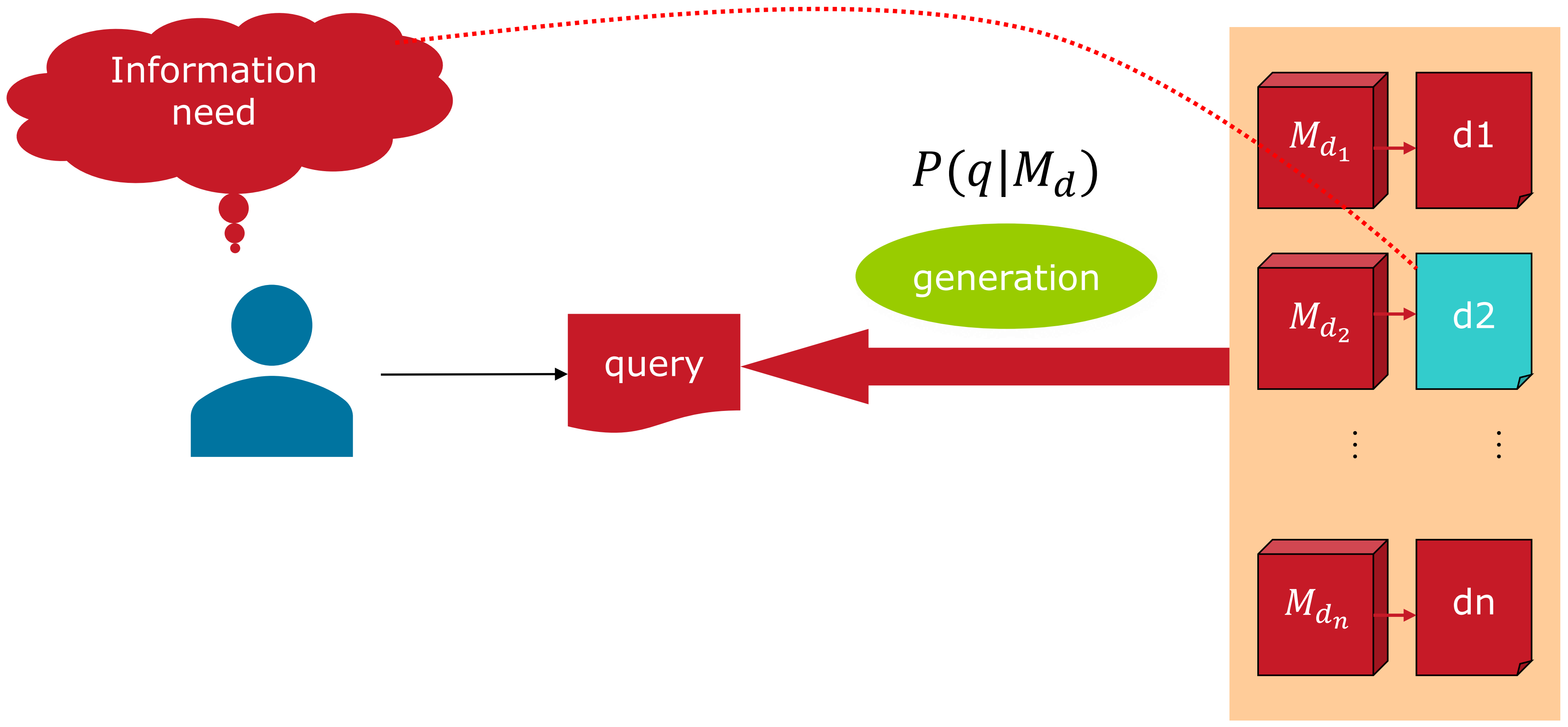
- The LM approach directly exploits this idea!

Language models in IR

- IR views documents as models and considers queries as strings of texts randomly sampled from models

- Documents are ranked according to the probability of observing a query q in repeated random samples from the document model $P(q|M_d)$

Language modelling for IR



document collection

Reformulate

- Use Bayes‘ Theorem

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

- Easier to handle
  - $P(q)$: constant for a given query → does not alter ranking

    ➔ can be ignored
  - $P(d)$: probabilty for a document
    - All documents equal → treat as uniform distribution
    - Use static quality measures to define a document prior

      ➔ can be computed independent of query
  - $P(q|d)$: Probability of query given a document
    - Document describes the query
    - Use LM for document to generate the query

      ➔ needs to be computed when retrieving documents

Query generation probability

The probability of producing the query given the language model of document d

$$P(q|M_d) = \prod_{t \in q} P(t|M_d)$$

$$= \prod_{t \in q} \frac{\mathrm{tf}_{(t,d)}}{dl_d}$$

Unigram assumption
Given a particular language model, the query terms occur independently

M_d: language model of document d

$\mathrm{tf}_{(t,d)}$: raw tf of term t in document d

dl_d: total number of tokens in document d

Need of smoothing

- Zero probability
  - May not wish to assign a probability of zero to a document in which one or more query terms are missing

$$P(t|M_d) = 0$$

- There is a number of approaches for smoothing probability distributions to deal with this problem

2. Smoothing

Smoothing

- Jelinek-Mercer smoothing
  - Linear combination of corpus and document model

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

– $0 < \lambda < 1$: parameter to adjust the influence of the document model

- Bayesian Update smoothing
  - Used for updating estimates in probabilistic relevance feedback
  - Adjusts estimates

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

– α: parameter to adjust the influence of the document model

Example

- Query: „cup jar“
- Estimate term probabilities for document models, e.g.

$$P(\text{"jar"}|d_2) = \frac{2}{5}$$

t_j	d_1	d_2	d_3	d_4	d_5
P(coffee)	1	0	0.25	0.3	0
P(cup)	0	0.2	0.50	0.3	0
P(jar)	0	0.4	0.25	0.3	0.5
P(tea)	0	0.4	0	0.1	0
P(water)	0	0	0	0	0.5

1. coffee, coffee
2. cup, jar, jar, tea, tea
3. coffee, cup, cup, jar
4. coffee, coffee, coffee, cup, cup, cup, jar, jar, jar, tea
5. jar, jar, water, water

Example

- Jelinek-Mercer smoothing

  $\lambda = 0.5$

- Estimate term probabilities for corpus
- models, e.g.

$$P(\text{"jar"}|M_c) = \frac{8}{25}$$

- Smooth document models

$$P(\text{"jar"}|d_2) = 0.5 \cdot 0.4 + 0.5 \cdot 0.32 = 0.36$$

1. coffee, coffee
2. cup, jar, jar, tea, tea
3. coffee, cup, cup, jar
4. coffee, coffee, coffee, cup, cup, cup, jar, jar, jar, tea
5. jar, jar, water, water

t_j	C
P(coffee)	0.24
P(cup)	0.24
P(jar)	0.32
P(tea)	0.12
P(water)	0.08

Example

t_j	d_1	d_2	d_3	d_4	d_5
P(coffee)	0.62	0.12	0.25	0.27	0.12
P(cup)	0.12	0.22	0.37	0.27	0.12
P(jar)	0.16	0.36	0.28	0.31	0.41
P(tea)	0.06	0.26	0.06	0.11	0.06
P(water)	0.04	0.04	0.04	0.04	0.29

1. coffee, coffee
2. cup, jar, jar, tea, tea
3. coffee, cup, cup, jar
4. coffee, coffee, coffee, cup, cup, cup, jar, jar, jar, tea
5. jar, jar, water, water

Rank	Document	ρ
1	d_3	0.105
2	d_4	0.084
3	d_2	0.079
4	d_5	0.049
5	d_1	0.019

- Compute retrieval values, e.g.

$$P(q|d_2) = 0.22^1 \cdot 0.36^1 = 0.0792$$

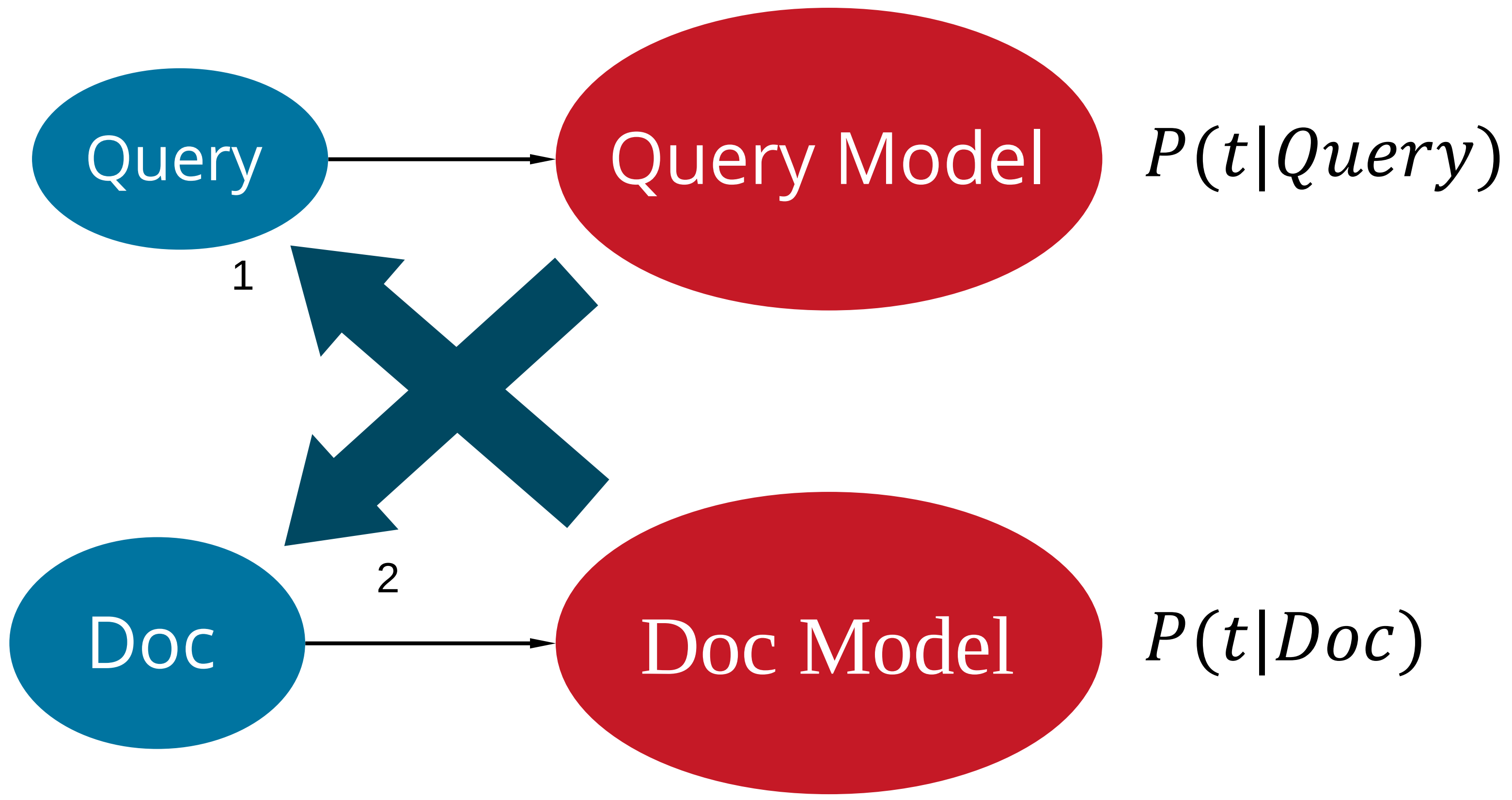
Note: d_1 at rank 5 does not contain any search term!

Parameter choice for smoothing

- How to choose λ or α?
- Exploit reference corpus
  - Use (comparable) evaluation corpus
  - Optimise retrieval performance on goldstandard
- Incorporate knowledge about query
  - Query length
    - Long queries: strong smoothing
      - ➢ Missing terms have a less strong influence
    - Short queries: weak smoothing
      - ➢ All/most terms should be present

3. Other Models

IR with language models



Retrieval: Query likelihood (1), Document likelihood (2)

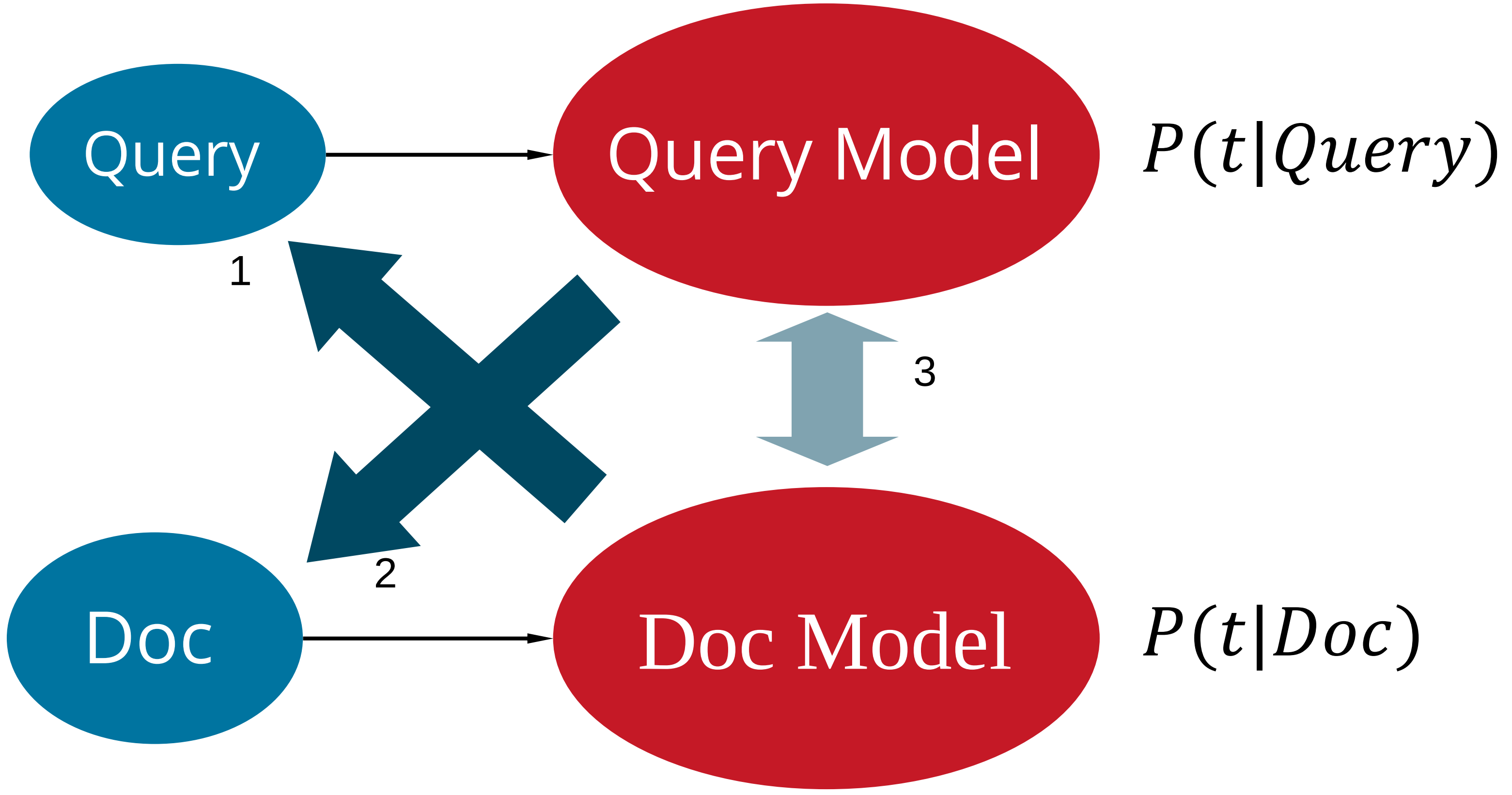
Lafferty and Zhai, 2001a

Document likelihood approach

- Flip the direction of the query-likelihood approach
- Rank docs D by the likelihood of being a random sample query
- Problems
  - different doc lengths, probabilities not comparable
  - query too small to estimate a good model
- Try to fix document likelihood
  - related to Probability Ranking Principle

    $P(D|R)\ /\ P\ (D|N)$ allows relevance feedback, query expansion, etc.
  - can benefit from complex estimation of the query model

IR with language models



Retrieval: Query likelihood (1), Document likelihood (2), Model comparison (3)

Lafferty and Zhai, 2001b

Model comparison approach

- Estimate query model
- Estimate document models
- Compare both models
- Suitable measure is KL divergence

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

- Better results than query-likelihood or document-likelihood approaches

Feedback as model interpolation

$$\begin{tikzcd}
\text{Document d} \arrow[r] & M_d \arrow[dr] & & \\
& & D(M_q \,\|\, M_d) \arrow[r] & \text{Results} \arrow[dd] \\
\text{Query q} \arrow[r] & M_q \arrow[ur] & & \\
& M_q' = (1-\lambda)M_q + \lambda M_F \arrow[u] \arrow[dl, "\lambda=0"'] \arrow[d, "\lambda=1"] & M_F \arrow[l] & \text{Feedback Docs } F=\{d_1, d_2, \ldots, d_n\} \arrow[l] \\
M_q' = M_q & M_q' = M_F & &
\end{tikzcd}$$

No feedback **Full feedback**

Generative model

Language models: in brief

- Novel way of looking at the problem of text retrieval based on probabilistic language modeling
  - Conceptually simple and explanatory
  - Formal mathematical model
  - Natural use of collection statistics, not heuristics
  - LMs provide effective retrieval and can be improved to the extent that the following conditions can be met
    - Our language models are accurate representations of the data
    - Users have some sense of term distribution

Comparison With Vector Space Model

- Similarities
  - Term weights based on frequency
  - Terms often used as if they were independent
  - Inverse document/collection frequency used
  - Some form of length normalization useful
- Differences
  - Based on probability rather than similarity
    - Intuitions are probabilistic rather than geometric
  - Aspects such as usage of document length and term, as well as document and collection frequency differ

universität koblenz
weiter:denken

4. Summary

Summary

- At the end of this lecture you should understand the following concepts
  - Language models
  - Query likelihood model
  - Document likelihood model
  - Comparison model

References

[1] https://olat.vcrp.de/auth/RepositoryEntry/4071063853

[2] https://nlp.stanford.edu/IR-book/information-retrieval-book.html
Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.

- ► Chapter 11.1, Chapter 12