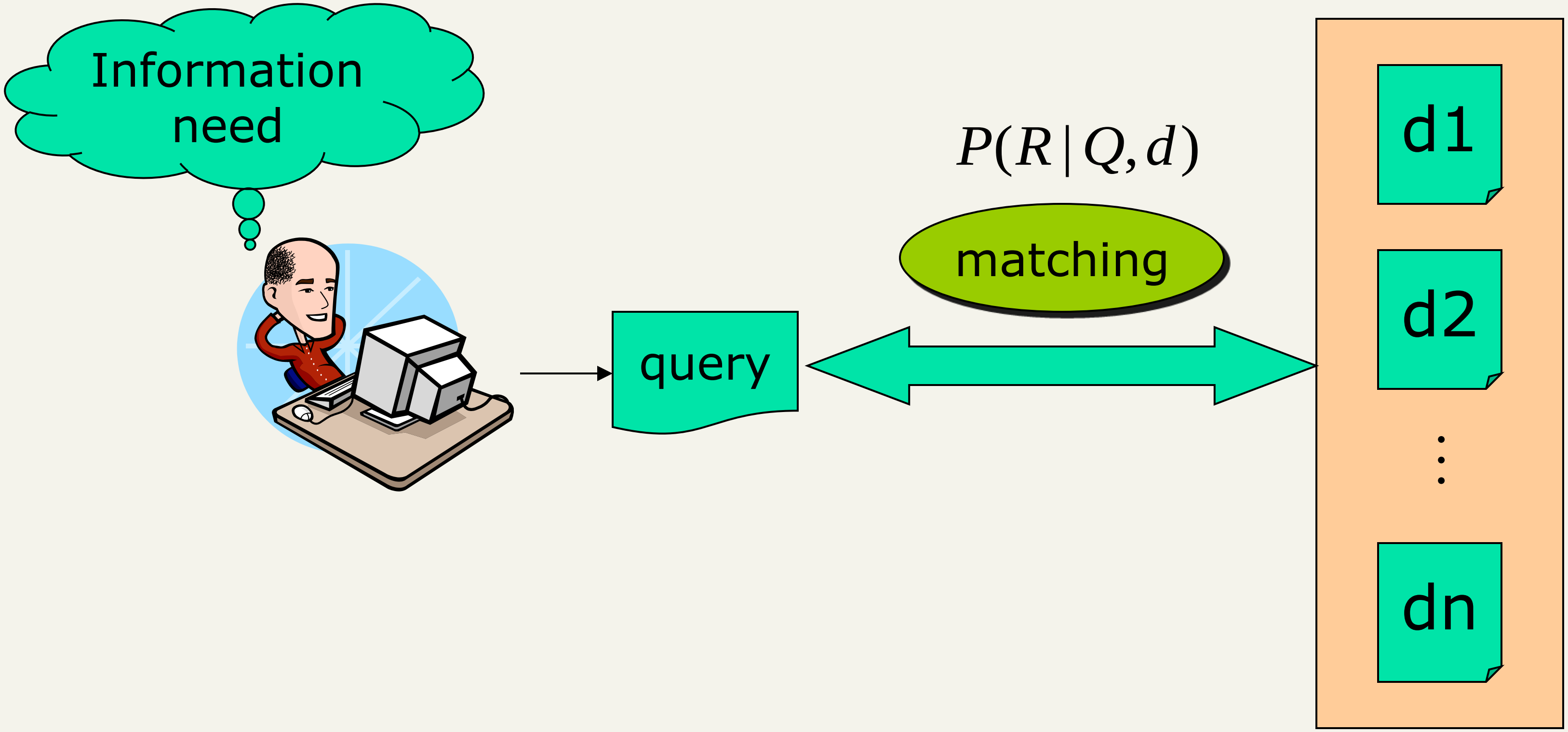
Language Models for Information Retrieval and Web Search

[Slides by Chris Manning, Prabhakar Raghavan and Hinrich Schütze. Some slides from Viktor Lavrenko and Chengxiang Zhai]

- The Language Model Approach to IR
  - Basic query generation model
  - Alternative models

Standard Probabilistic IR



Information need

query

$P(R|Q,d)$

matching

d1

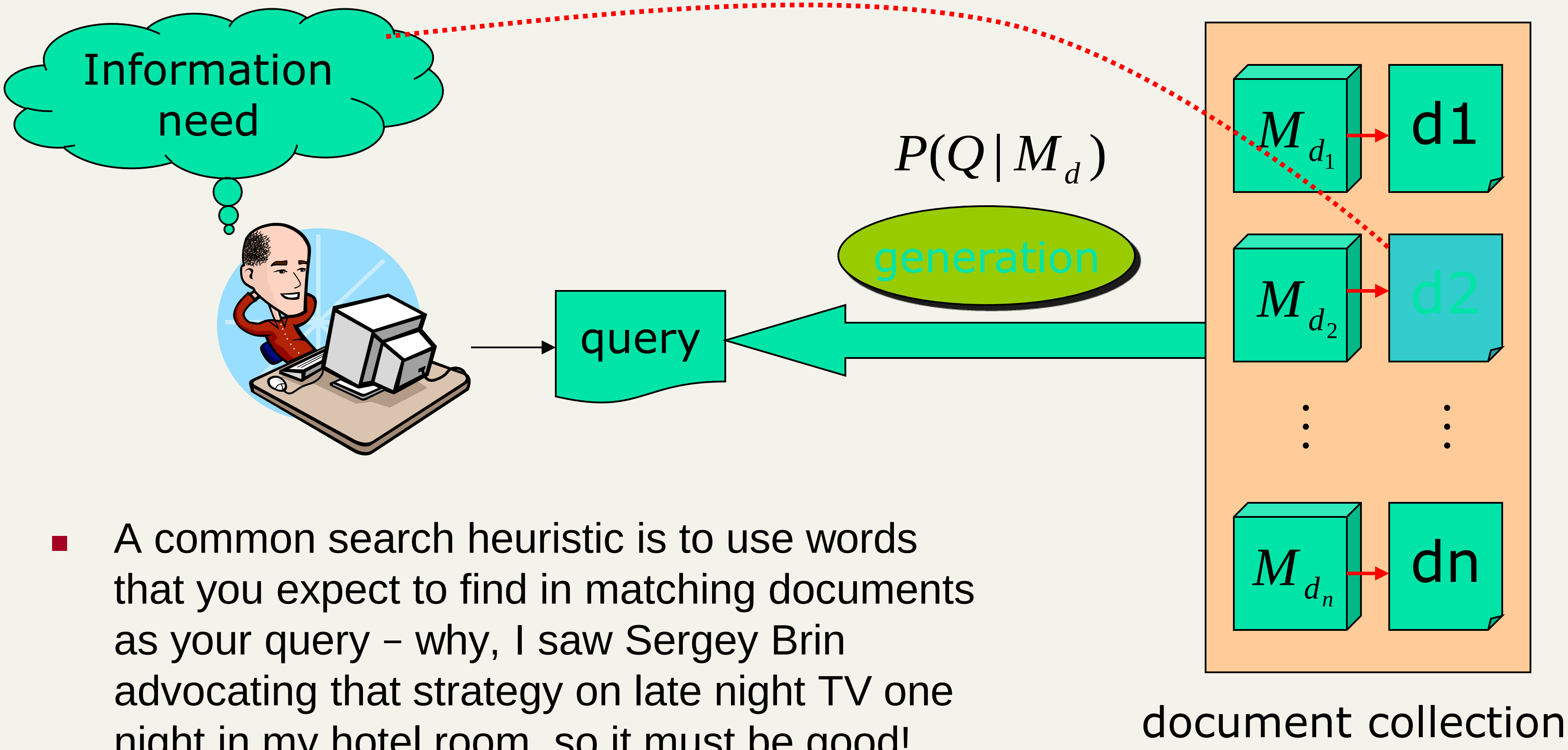
d2

⋮

dn

document collection

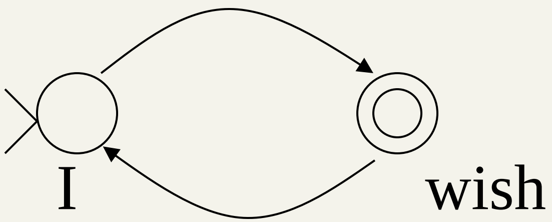
IR based on Language Model (LM)



- A common search heuristic is to use words that you expect to find in matching documents as your query – why, I saw Sergey Brin advocating that strategy on late night TV one night in my hotel room, so it must be good!
- The LM approach directly exploits that idea!

Formal Language (Model)

- Traditional generative model: generates strings
  - Finite state machines or regular grammars, etc.
- Example:



I wish

I wish I wish

I wish I wish I wish

I wish I wish I wish I wish

…

*wish I wish

Stochastic Language Models

- Models *probability* of generating strings in the language (commonly all strings over alphabet ∑)

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
…	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

P(s | M) = 0.00000008

Stochastic Language Models

- Model *probability* of generating any string

Model M1

0.2	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

Model M2

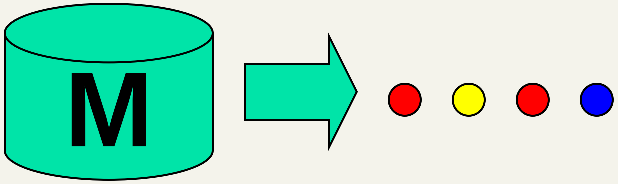
0.2	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$$P(s|M2) > P(s|M1)$$

Stochastic Language Models

- A statistical model for generating text
  - Probability distribution over strings in a given language



P (🔴 🟡 🔴 🔵 | M) = P (🔴 | M)

P (🟡 | M, 🔴)

P (🔴 | M, 🔴 🟡)

P (🔵 | M, 🔴 🟡 🔴)

Unigram and higher-order models

P (🔴 🟡 🔴 🔵)

= P (🔴) P (🟡 | 🔴) P (🔴 | 🔴 🟡) P (🔵 | 🔴 🟡 🔴)

- Unigram Language Models

  P (🔴) P (🟡) P (🔴) P (🔵)

Easy.
Effective!

- Bigram (generally, *n*-gram) Language Models

  P (🔴) P (🟡 | 🔴) P (🔴 | 🟡) P (🔵 | 🔴)

- Other Language Models
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

Using Language Models in IR

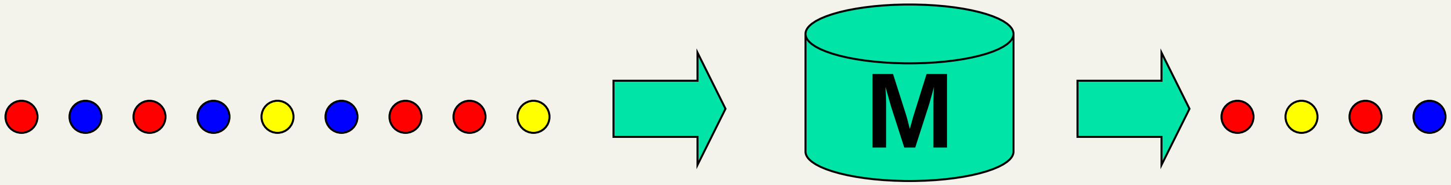
- Treat each document as the basis for a model (e.g., unigram sufficient statistics)
- Rank document d based on P(d | q)
- P(d | q) = P(q | d) x P(d) / P(q)
  - P(q) is the same for all documents, so ignore
  - P(d) [the prior] is often treated as the same for all d
    - But we could use criteria like authority, length, genre
  - P(q | d) is the probability of q given d’s model
- Very general formal approach

The fundamental problem of LMs

- Usually we don't know the model **M**
  - But have a sample of text representative of that model

P (🔴 🟡 🔴 🔵 **| M (** 🔴 🔵 🔴 🔵 🟡 🔵 🔴 🔴 🟡 **))**

- Estimate a language model from a sample
- Then compute the observation probability



Language Models for IR

- Language Modeling Approaches
    - Attempt to **model query generation process**
    - Documents are ranked by **the probability that a query would be observed as a random sample from the respective document model**
        - Multinomial approach

$$P(Q|M_D) = \prod_w P(w|M_D)^{q_w}$$

Retrieval based on probabilistic LM

- Treat the generation of queries as a random process.
- Approach
  - Infer a language model for each document.
  - Estimate the probability of generating the query according to each of these models.
  - Rank the documents according to these probabilities.
  - Usually a unigram estimate of words is used
    - Some work on bigrams, paralleling van Rijsbergen

Retrieval based on probabilistic LM

- Intuition
  - Users …
    - Have a reasonable idea of terms that are likely to occur in documents of interest.
    - They will choose query terms that distinguish these documents from others in the collection.
- Collection statistics …
  - Are integral parts of the language model.
  - Are not used heuristically as in many other approaches.
    - In theory. In practice, there's usually some wiggle room for empirically set parameters

Query generation probability (1)

- Ranking formula

$$p(Q,d) = p(d)p(Q|d)$$

$$\approx p(d)p(Q|M_d)$$

- The probability of producing the query given the language model of document d using MLE is:

$$\hat{p}(Q|M_d) = \prod_{t \in Q} \hat{p}_{mle}(t|M_d)$$

$$= \prod_{t \in Q} \frac{tf_{(t,d)}}{dl_d}$$

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

M_d : language model of document d

$tf_{(t,d)}$: raw tf of term t in document d

dl_d : total number of tokens in document d

Insufficient data

- Zero probability $p(t \mid M_d) = 0$
  - May not wish to assign a probability of zero to a document that is missing one or more of the query terms [gives conjunction semantics]
- General approach
  - A non-occurring term is possible, but no more likely than would be expected by chance in the collection.
  - If $tf_{(t,d)} = 0$, $p(t \mid M_d) = \frac{cf_t}{cs}$

cf_t : raw count of term t in the collection

cs : raw collection size (total number of tokens in the collection)

Insufficient data

- Zero probabilities spell disaster
  - We need to smooth probabilities
    - Discount nonzero probabilities
    - Give some probability mass to unseen things
- There’s a wide space of approaches to smoothing probability distributions to deal with this problem, such as adding 1, ½ or ε to counts, discounting, and interpolation
- A simple idea that works well in practice is to use a mixture between the document multinomial and the collection multinomial distribution

Mixture model

- $P(w|d) = \lambda P_{mle}(w|M_d) + (1 - \lambda)P_{mle}(w|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
- Correctly setting λ is very important
- A high value of lambda makes the search "conjunctive-like" – suitable for short queries
- A low value is more suitable for long queries
- Can tune λ to optimize performance
  - Perhaps make it dependent on document size

Basic mixture model summary

- General formulation of the LM for IR

$$p(Q,d) = p(d)\prod_{t \in Q}((1-\lambda)\,p(t) + \lambda p(t \mid M_d))$$

general language model

individual-document 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

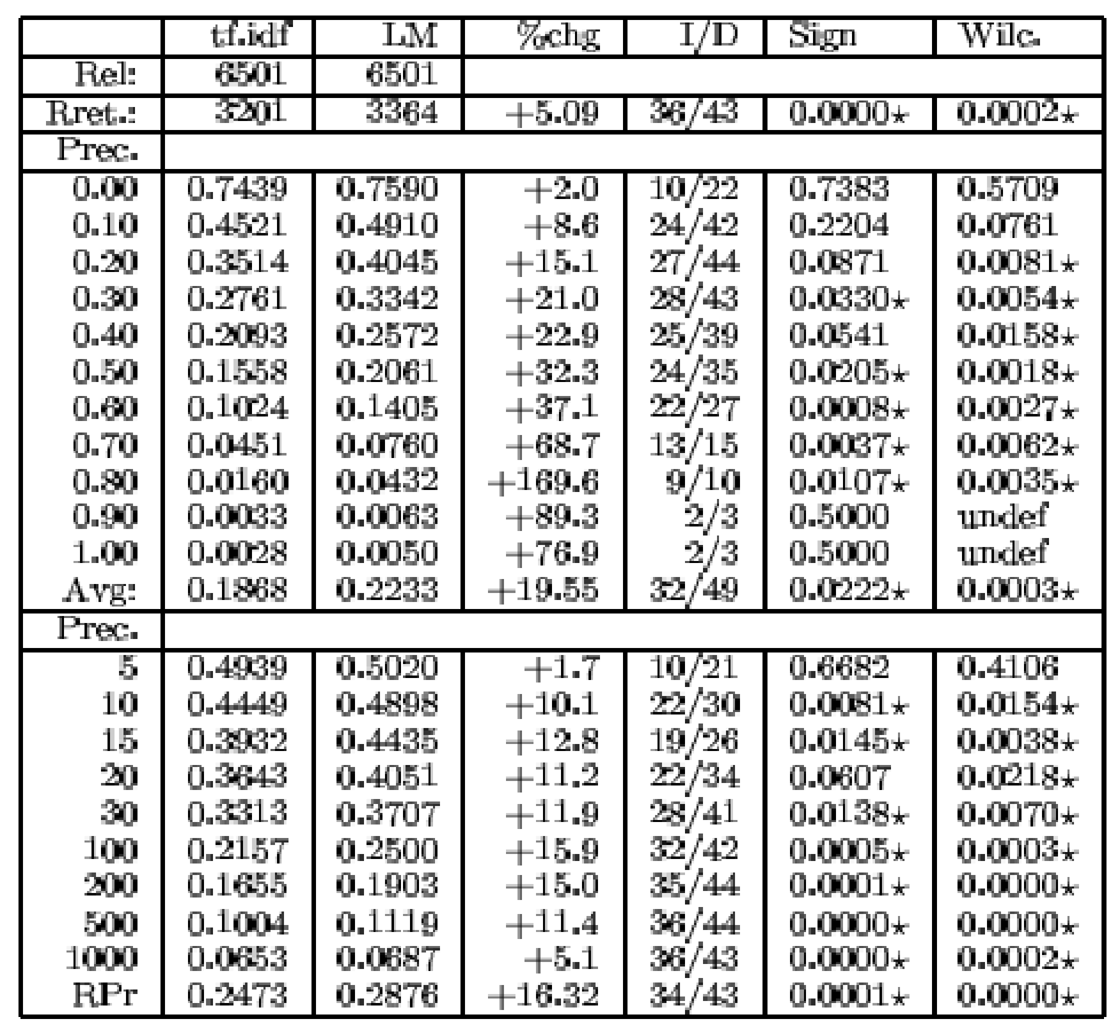
- Document collection (2 documents)
  - d_1: Xerox reports a profit but revenue is down
  - d_2: Lucent narrows quarter loss but revenue decreases further
- Model: MLE unigram from documents; $\lambda = ½$
- Query: *revenue down*
  - $P(Q|d_1) = [(1/8 + 2/16)/2] \text{ x } [(1/8 + 1/16)/2]$
    $= 1/8 \text{ x } 3/32 = 3/256$
  - $P(Q|d_2) = [(1/8 + 2/16)/2] \text{ x } [(0 + 1/16)/2]$
    $= 1/8 \text{ x } 1/32 = 1/256$
- Ranking: $d_1 > d_2$

Ponte and Croft Experiments

- Data
  - TREC topics 202-250 on TREC disks 2 and 3
    - Natural language queries consisting of one sentence each
  - TREC topics 51-100 on TREC disk 3 using the concept fields
    - Lists of good terms

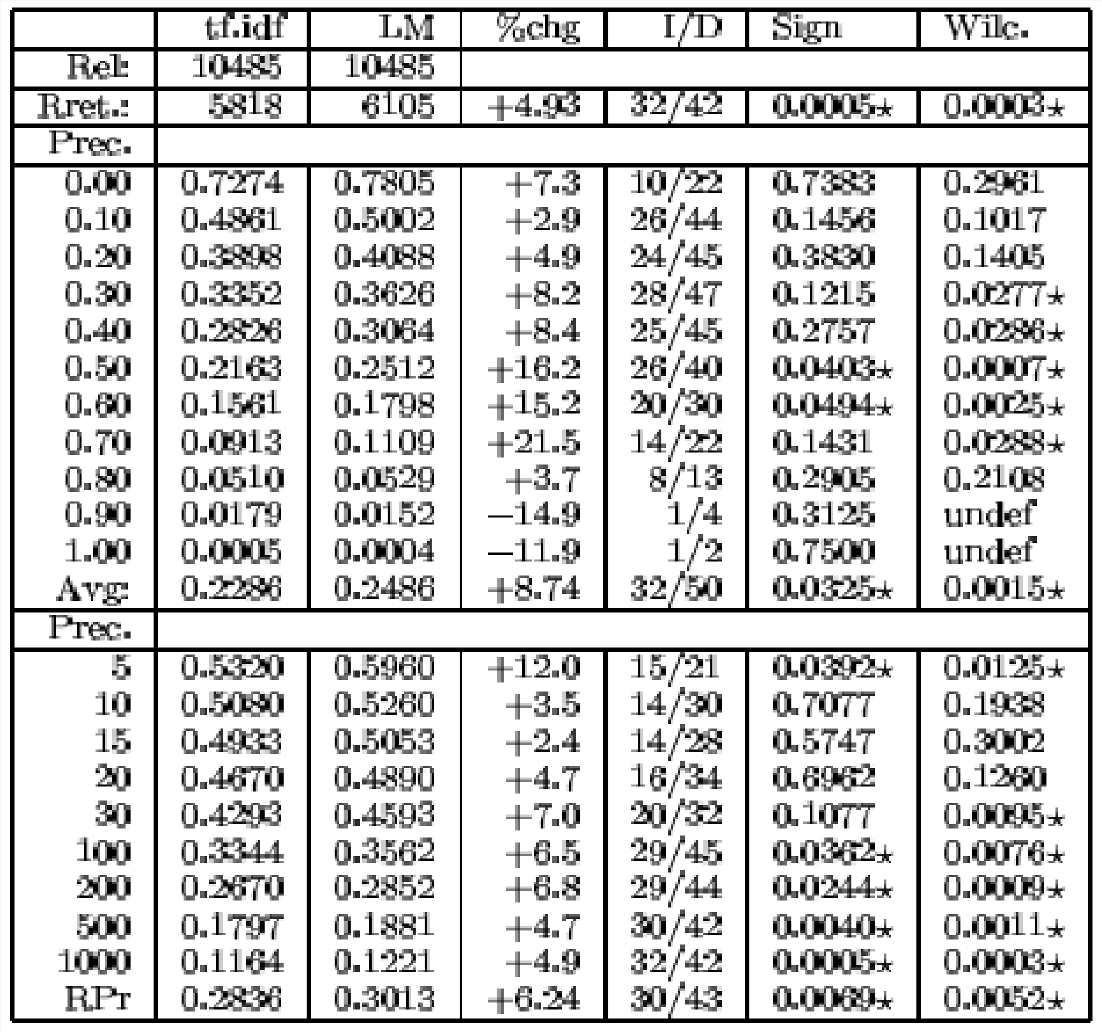
```
<num>Number: 054
<dom>Domain: International Economics
<title>Topic: Satellite Launch Contracts
<desc>Description:
… </desc>

<con>Concept(s):
1.    Contract, agreement
2.    Launch vehicle, rocket, payload, satellite
3.    Launch services, …  </con>
```

Precision/recall results 202-250



	tf.idf	LM	%chg	I/D	Sign	Wilc.
Rel:	6501	6501				
Rret.:	3201	3364	+5.09	36/43	0.0000⋆	0.0002⋆
Prec.						
0.00	0.7439	0.7590	+2.0	10/22	0.7383	0.5709
0.10	0.4521	0.4910	+8.6	24/42	0.2204	0.0761
0.20	0.3514	0.4045	+15.1	27/44	0.0871	0.0081⋆
0.30	0.2761	0.3342	+21.0	28/43	0.0330⋆	0.0054⋆
0.40	0.2093	0.2572	+22.9	25/39	0.0541	0.0158⋆
0.50	0.1558	0.2061	+32.3	24/35	0.0205⋆	0.0018⋆
0.60	0.1024	0.1405	+37.1	22/27	0.0008⋆	0.0027⋆
0.70	0.0451	0.0760	+68.7	13/15	0.0037⋆	0.0062⋆
0.80	0.0160	0.0432	+169.6	9/10	0.0107⋆	0.0035⋆
0.90	0.0033	0.0063	+89.3	2/3	0.5000	undef
1.00	0.0028	0.0050	+76.9	2/3	0.5000	undef
Avg:	0.1868	0.2233	+19.55	32/49	0.0222⋆	0.0003⋆
Prec.						
5	0.4939	0.5020	+1.7	10/21	0.6682	0.4106
10	0.4449	0.4898	+10.1	22/30	0.0081⋆	0.0154⋆
15	0.3932	0.4435	+12.8	19/26	0.0145⋆	0.0038⋆
20	0.3643	0.4051	+11.2	22/34	0.0607	0.0218⋆
30	0.3313	0.3707	+11.9	28/41	0.0138⋆	0.0070⋆
100	0.2157	0.2500	+15.9	32/42	0.0005⋆	0.0003⋆
200	0.1655	0.1903	+15.0	35/44	0.0001⋆	0.0000⋆
500	0.1004	0.1119	+11.4	36/44	0.0000⋆	0.0000⋆
1000	0.0653	0.0687	+5.1	36/43	0.0000⋆	0.0002⋆
RPr	0.2473	0.2876	+16.32	34/43	0.0001⋆	0.0000⋆

Precision/recall results 51-100



	tf.idf	LM	%chg	I/D	Sign	Wilc.
Rel:	10485	10485				
Rret.:	5818	6105	+4.93	32/42	0.0005⋆	0.0003⋆
Prec.						
0.00	0.7274	0.7805	+7.3	10/22	0.7383	0.2961
0.10	0.4861	0.5002	+2.9	26/44	0.1456	0.1017
0.20	0.3898	0.4088	+4.9	24/45	0.3830	0.1405
0.30	0.3352	0.3626	+8.2	28/47	0.1215	0.0277⋆
0.40	0.2826	0.3064	+8.4	25/45	0.2757	0.0286⋆
0.50	0.2163	0.2512	+16.2	26/40	0.0403⋆	0.0007⋆
0.60	0.1561	0.1798	+15.2	20/30	0.0494⋆	0.0025⋆
0.70	0.0913	0.1109	+21.5	14/22	0.1431	0.0288⋆
0.80	0.0510	0.0529	+3.7	8/13	0.2905	0.2108
0.90	0.0179	0.0152	−14.9	1/4	0.3125	undef
1.00	0.0005	0.0004	−11.9	1/2	0.7500	undef
Avg:	0.2286	0.2486	+8.74	32/50	0.0325⋆	0.0015⋆
Prec.						
5	0.5320	0.5960	+12.0	15/21	0.0392⋆	0.0125⋆
10	0.5080	0.5260	+3.5	14/30	0.7077	0.1938
15	0.4933	0.5053	+2.4	14/28	0.5747	0.3002
20	0.4670	0.4890	+4.7	16/34	0.6962	0.1260
30	0.4293	0.4593	+7.0	20/32	0.1077	0.0095⋆
100	0.3344	0.3562	+6.5	29/45	0.0362⋆	0.0076⋆
200	0.2670	0.2852	+6.8	29/44	0.0244⋆	0.0009⋆
500	0.1797	0.1881	+4.7	30/42	0.0040⋆	0.0011⋆
1000	0.1164	0.1221	+4.9	32/42	0.0005⋆	0.0003⋆
RPr	0.2836	0.3013	+6.24	30/43	0.0069⋆	0.0052⋆

LM vs. Prob. Model for IR

- The main difference is whether “Relevance” figures explicitly in the model or not
  - LM approach attempts to do away with modeling relevance
- LM approach asssumes that documents and expressions of information problems are of the same type
- Computationally tractable, intuitively appealing

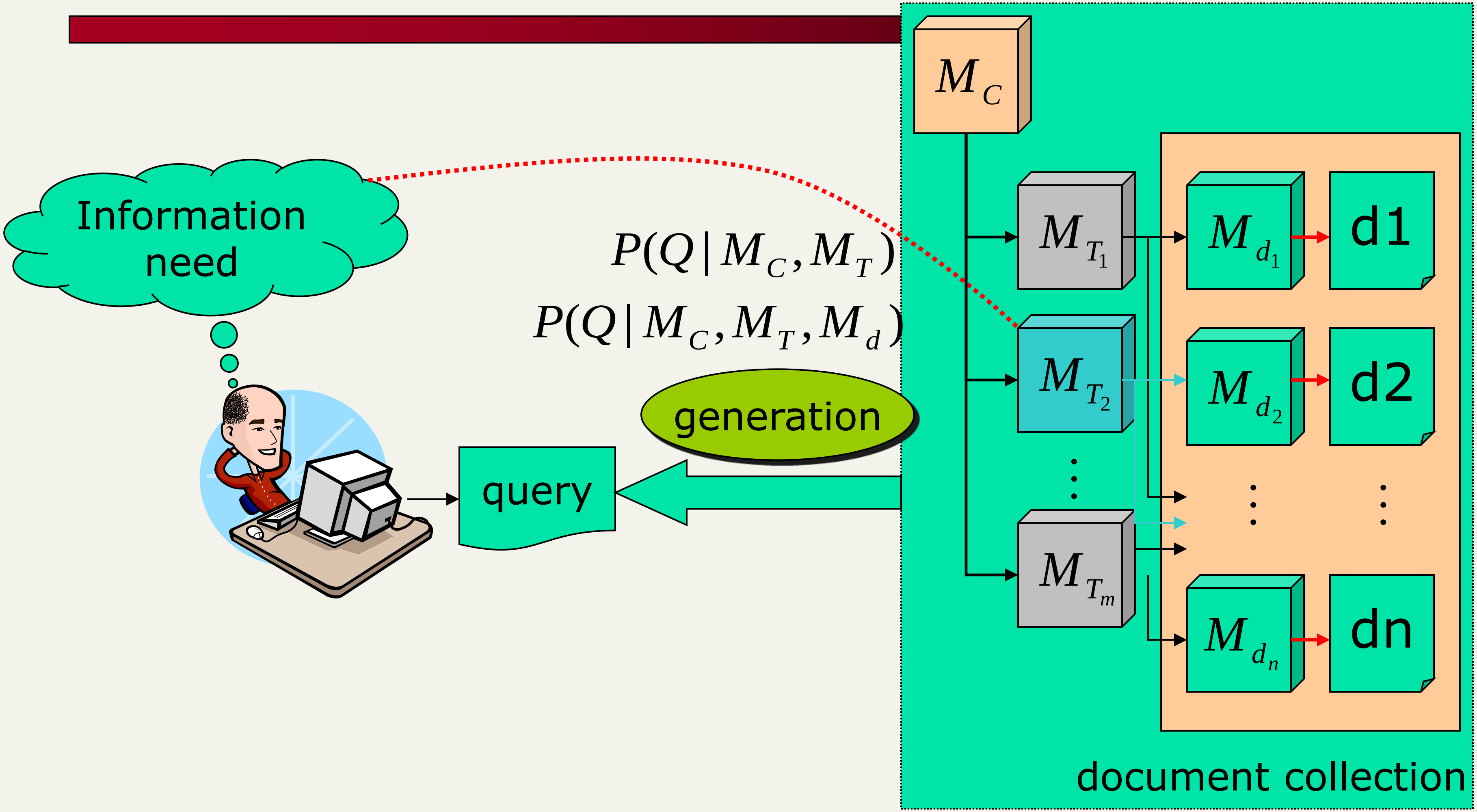
LM vs. Prob. Model for IR

- Problems of basic LM approach
  - Assumption of equivalence between document and information problem representation is unrealistic
  - Very simple models of language
  - Relevance feedback is difficult to integrate, as are user preferences, and other general issues of relevance
  - Can’t easily accommodate phrases, passages, Boolean operators
- Current extensions focus on putting relevance back into the model, etc.

Extension: 3-level model

- 3-level model
    1. Whole collection model (M_C)
    2. Specific-topic model; relevant-documents model (M_T)
    3. Individual-document model (M_d)
- Relevance hypothesis
    - A request(query; topic) is generated from a specific-topic model $\{M_C, M_T\}$.
    - Iff a document is relevant to the topic, the same model will apply to the document.
        - It will replace part of the individual-document model in explaining the document.
    - The probability of relevance of a document
        - The probability that this model explains part of the document
        - The probability that the $\{M_C, M_T, M_d\}$ combination is better than the $\{M_C, M_d\}$ combination

3-level model



Information need

$P(Q \mid M_C, M_T)$

$P(Q \mid M_C, M_T, M_d)$

generation

query

M_C

M_{T_1}

M_{d_1}

d1

M_{T_2}

M_{d_2}

d2

$\vdots$

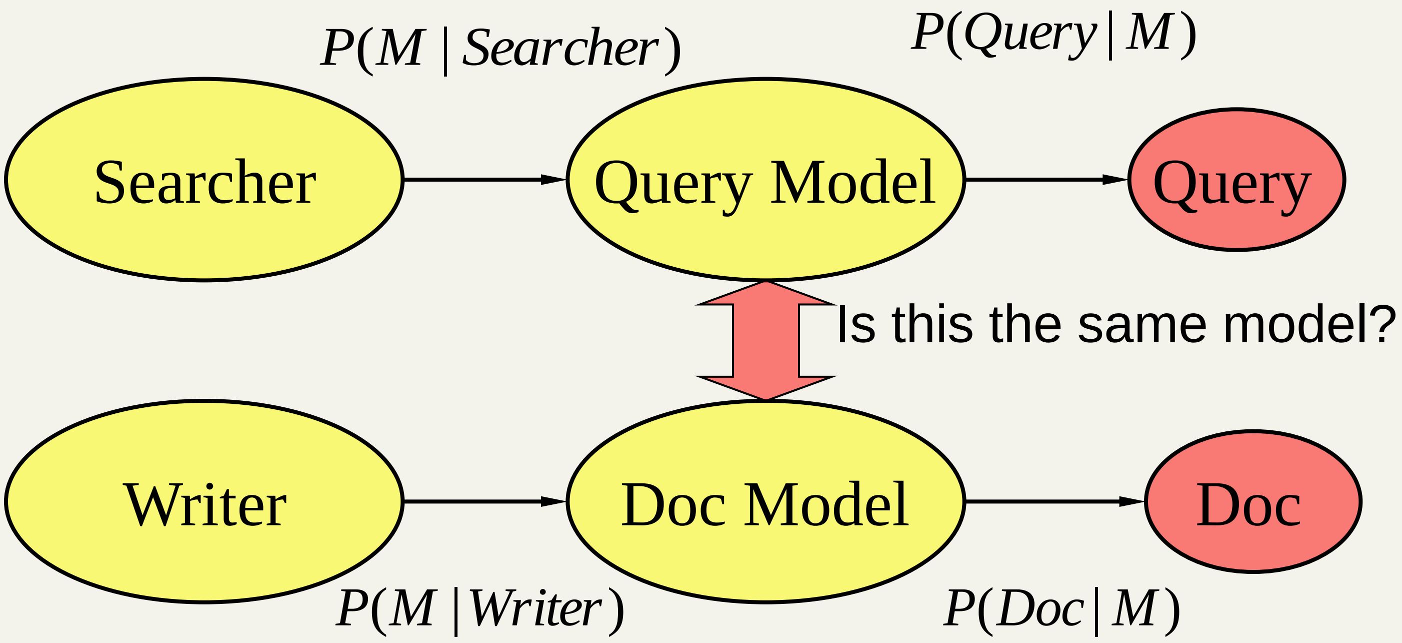
M_{T_m}

M_{d_n}

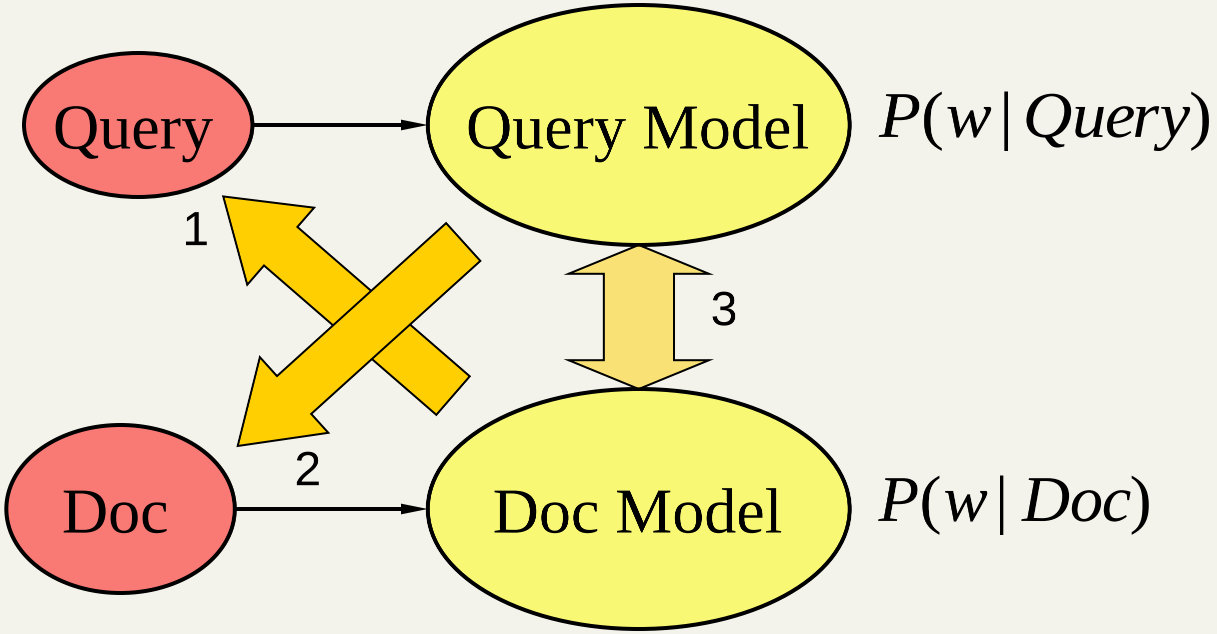
dn

document collection

Alternative Models of Text Generation



Retrieval Using Language Models



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

Query Likelihood

- $P(Q|D_m)$
- Major issue is estimating document model
  - i.e. smoothing techniques instead of tf.idf weights
- Good retrieval results
  - e.g. UMass, BBN, Twente, CMU
- Problems dealing with relevance feedback, query expansion, structured queries

Document Likelihood

- Rank by likelihood ratio P(D|R)/P(D|NR)
  - treat as a *generation* problem
  - P(w|R) is estimated by P(w|Q_m)
  - Q_m is the query or relevance model
  - P(w|NR) is estimated by collection probabilities P(w)
- Issue is estimation of query model
  - Treat query as generated by mixture of topic and background
  - Estimate relevance model from related documents (query expansion)
  - Relevance feedback is easily incorporated
- Good retrieval results
  - e.g. UMass at SIGIR 01
  - inconsistent with heterogeneous document collections

Model Comparison

- Estimate query and document models and compare
- Suitable measure is KL divergence $D(Q_m||D_m)$

$$D(Q_m \,\|\, D_m) = \sum_{x \in X} Q_m(x) \log \frac{Q_m(x)}{D_m(x)}$$

  - equivalent to query-likelihood approach if simple empirical distribution used for query model
- More general risk minimization framework has been proposed
  - Zhai and Lafferty 2001
- Better results than query-likelihood or document-likelihood approaches

Two-stage smoothing: Another Reason for Smoothing

Query	= “the	algorithms	for	data	mining”
d1:	**0.04**	**0.001**	**0.02**	**0.002**	**0.003**
d2:	**0.02**	**0.001**	**0.01**	**0.003**	**0.004**

p(“algorithms”|d1) = p(“algorithm”|d2)

p(“data”|d1) < p(“data”|d2)

p(“mining”|d1) < p(“mining”|d2)

But p(q|d1)>p(q|d2)!

We should make p(“the”) and p(“for”) less different for all docs.

Two-stage Smoothing

Stage-1

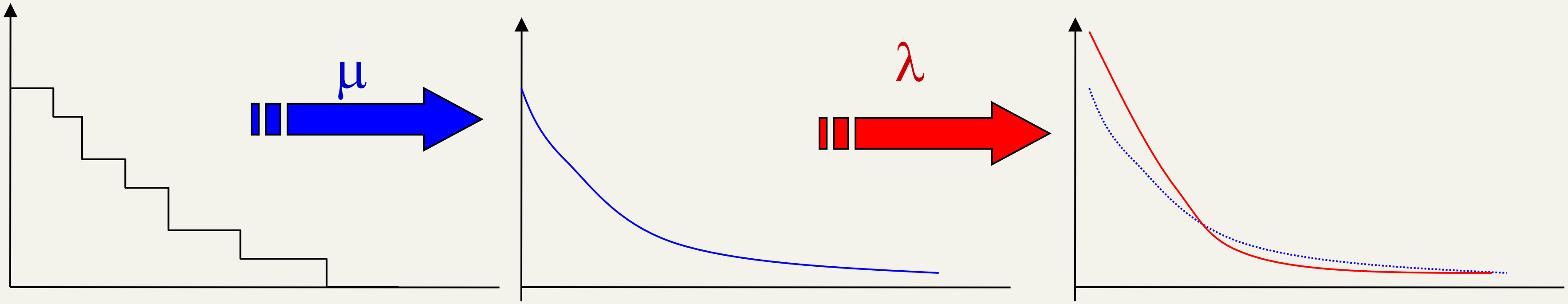
-Explain unseen words

-Dirichlet prior(Bayesian)

Stage-2

-Explain noise in query

-2-component mixture



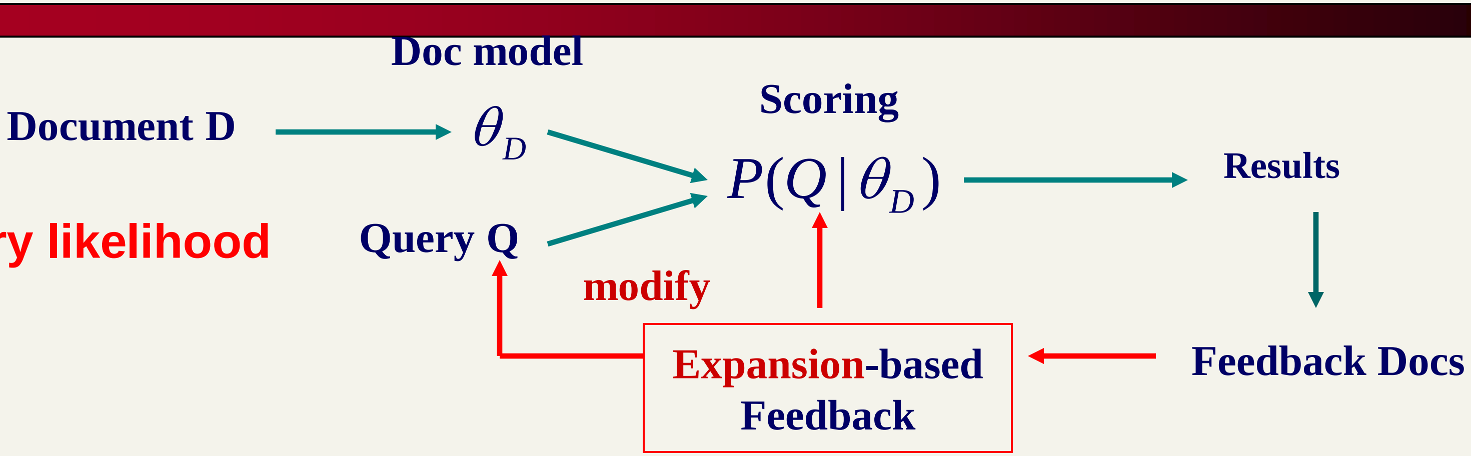
$$P(w|d) = (1-\lambda)\frac{c(w,d) + \mu p(w|C)}{|d| + \mu} + \lambda p(w|U)$$

How can one do relevance feedback if using language modeling approach?

- Introduce a query model & treat feedback as query model updating
  - Retrieval function:
    - Query-likelihood => KL-Divergence
  - Feedback:
    - Expansion-based => Model-based

Expansion-based vs. Model-based

Query likelihood



Doc model

Document D

θ_D

Scoring

Query Q

$P(Q|\theta_D)$

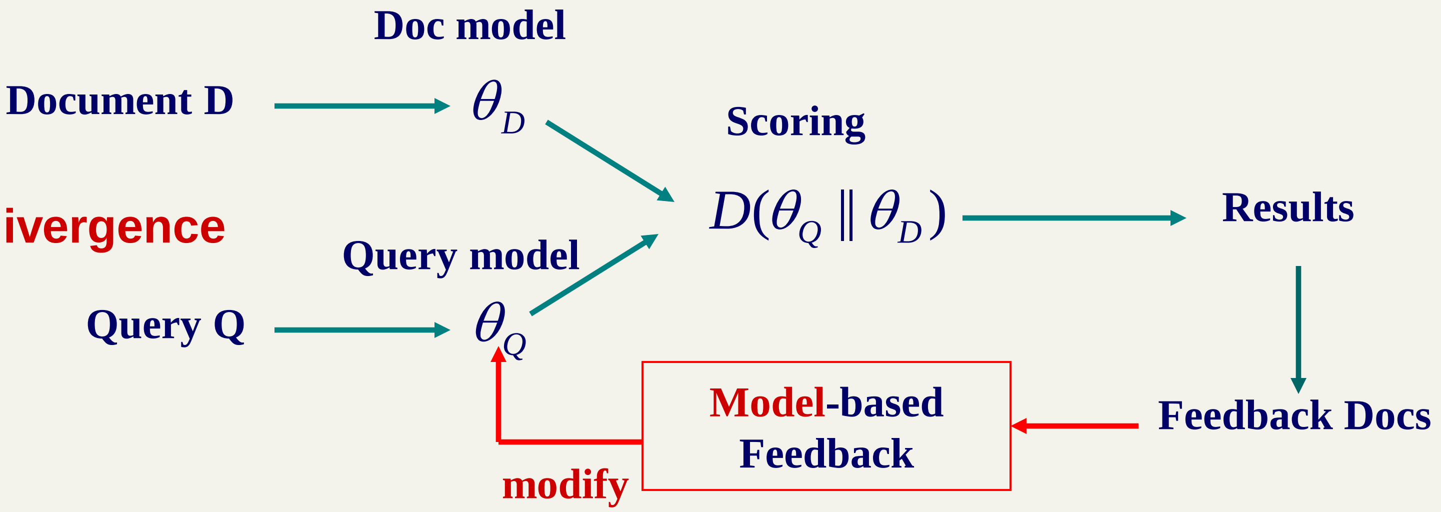
Results

modify

Expansion-based Feedback

Feedback Docs

KL-divergence



Doc model

Document D

θ_D

Scoring

$D(\theta_Q \| \theta_D)$

Results

Query model

Query Q

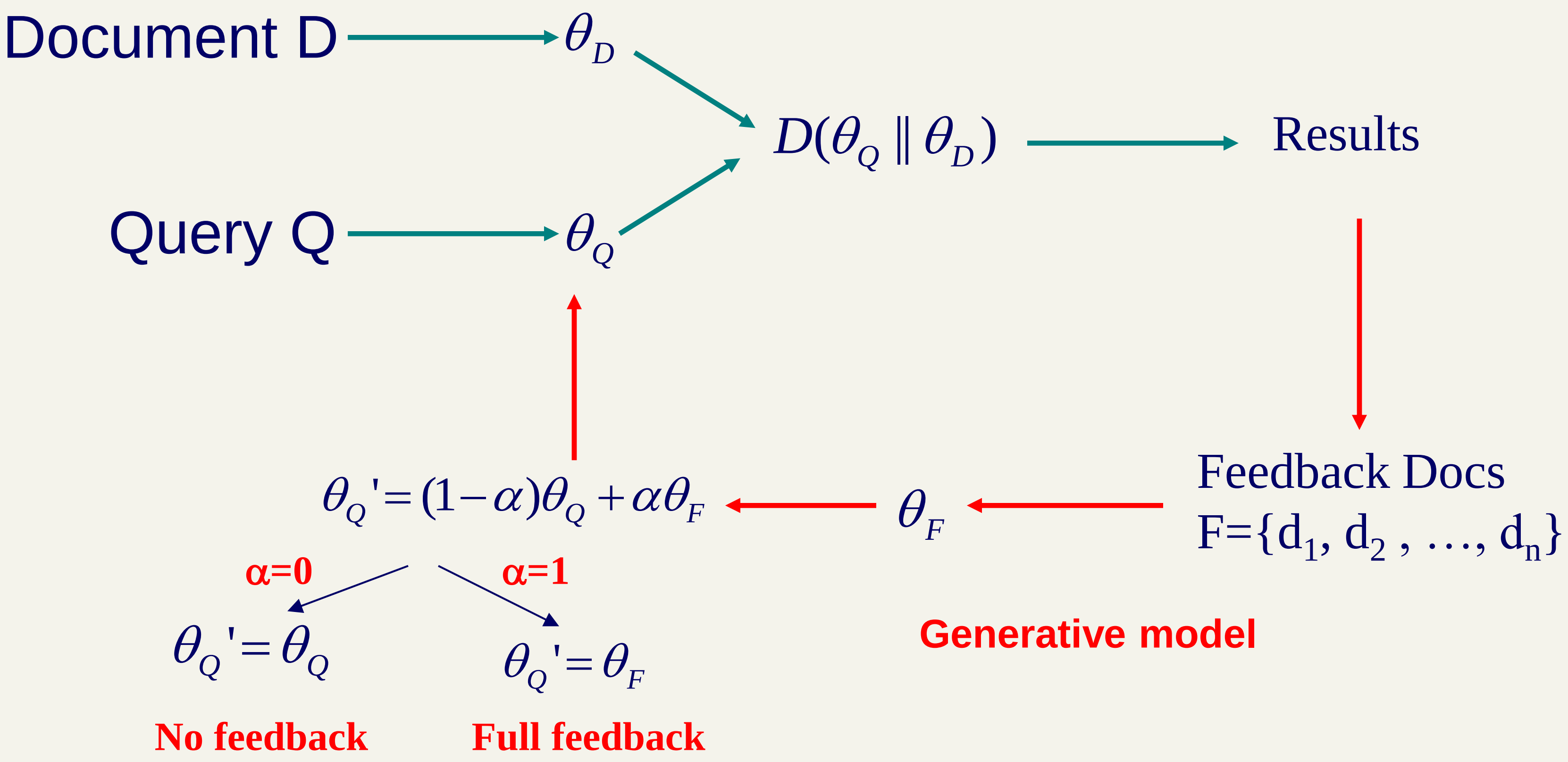
θ_Q

Model-based Feedback

Feedback Docs

modify

Feedback as Model Interpolation



Document D $\rightarrow \theta_D$

Query Q $\rightarrow \theta_Q$

$D(\theta_Q \| \theta_D)$ → Results

Feedback Docs
F={d_1, d_2 , …, d_n}

θ_F

Generative model

$$\theta_Q' = (1-\alpha)\theta_Q + \alpha\theta_F$$

α=0

$\theta_Q' = \theta_Q$

No feedback

α=1

$\theta_Q' = \theta_F$

Full feedback

Translation model (Berger and Lafferty)

- Basic LMs do not address issues of synonymy.
  - Or any deviation in expression of information need from language of documents
- A translation model lets you generate query words not in document via “translation” to synonyms etc.
  - Or to do cross-language IR, or multimedia IR

$$P(\vec{q} \mid M) = \prod_i \sum_{v \in Lexicon} P(v \mid M) T(q_i \mid v)$$

Basic LM Translation

  - Need to learn a translation model (using a dictionary or via statistical machine translation)

Language models: pro & con

- 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 (almost…)
- 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.*
        - *Or we get more sophisticated with translation model

Comparison With Vector Space

- There's some relation to traditional tf.idf models:
  - (unscaled) term frequency is directly in model
  - the probabilities do length normalization of term frequencies
  - the effect of doing a mixture with overall collection frequencies is a little like idf: terms rare in the general collection but common in some documents will have a greater influence on the ranking

Comparison With Vector Space

- Similar in some ways
  - Term weights based on frequency
  - Terms often used as if they were independent
  - Inverse document/collection frequency used
  - Some form of length normalization useful
- Different in others
  - Based on probability rather than similarity
    - Intuitions are probabilistic rather than geometric
  - Details of use of document length and term, document, and collection frequency differ

Resources

J.M. Ponte and W.B. Croft. 1998. A language modelling approach to information retrieval. In *SIGIR 21.*

D. Hiemstra. 1998. A linguistically motivated probabilistic model of information retrieval. *ECDL 2*, pp. 569–584.

A. Berger and J. Lafferty. 1999. Information retrieval as statistical translation. *SIGIR 22*, pp. 222–229.

D.R.H. Miller, T. Leek, and R.M. Schwartz. 1999. A hidden Markov model information retrieval system. *SIGIR 22*, pp. 214–221.

[Several relevant newer papers at *SIGIR 23–25,* 2000–2002.]

Workshop on Language Modeling and Information Retrieval, CMU 2001. http://la.lti.cs.cmu.edu/callan/Workshops/lmir01/ .

The Lemur Toolkit for Language Modeling and Information Retrieval. http://www-2.cs.cmu.edu/~lemur/ . CMU/Umass LM and IR system in C(++), currently actively developed.