CS 589 Fall 2020

Probability ranking principle

Probabilistic retrieval models

Instructor: Susan Liu
TA: Huihui Liu

Stevens Institute of Technology

Random variables

- Random variables

sequence = 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0 **Observation**

$p(up) = \alpha, p(down) = 1 - \alpha$ α : **parameter**

$$p(sequence) = \alpha \times (1-\alpha) \cdots \times (1-\alpha) \times (1-\alpha)$$

Bernoulli distribution

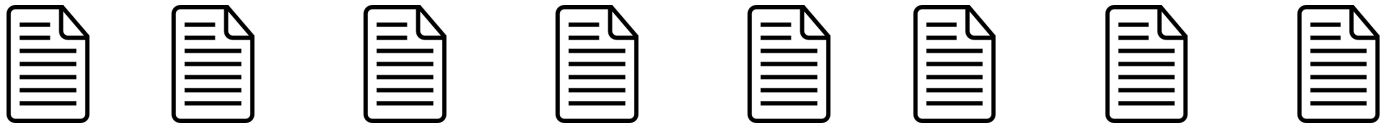
$$= \alpha^{\#up} \times (1-\alpha)^{\#down}$$

$$\Rightarrow \alpha = \frac{\#up}{\#up + \#down}$$

Maximum likelihood estimation

Random variables in information retrieval

query **=** *“artificial intelligence”*



rel = 0, 1, 0, 0, 0, 0, 0, 0

d = artificial, intelligence, machine, intelligence, information, retrieval

d = 56, 100, 120, 100, 50, 221

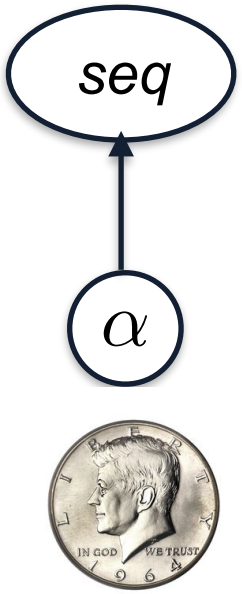
q = artificial, intelligence

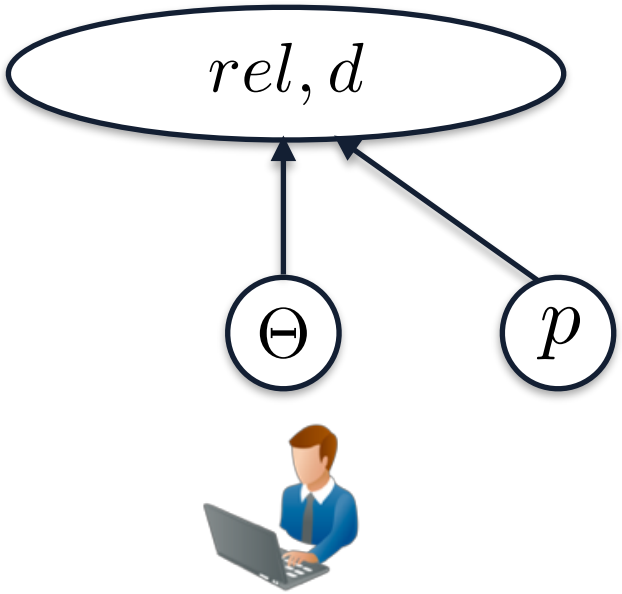
q = 56, 100

Observation

Notations: *in future slides, q denotes the query, d denotes the document, rel denotes the relevance judgment*

Probabilistic graphical model





parameter	α	Θ
distribution	Bernoulli	Multinomial-Dirichlet, 2-Poisson, etc.
parameter estimation	$\alpha = \frac{\#up}{\#up + \#down}$	maximum likelihood estimation maximum a posterior estimation

Bayes' rules

Chain rule:

$$P(A,B) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

Bayes' rule:

posterior likelihood prior

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \left[\frac{P(B|A)}{\sum_{X\in\{A,\overline{A}\}} P(B|X)P(X)}\right] P(A)$$

$$P(A|B) \propto P(B|A)P(A)$$

$$\sum_{A} P(A|B) = 1$$

skipping estimating $P(B)$

trick for estimating the posterior

Probability ranking principle

- Assume documents are labelled by 0/1 labels (i.e., the relevance judgement is either 0 or 1), given query q, documents should be ranked on their probabilities of relevance *(van Rijsbergen 1979)*:

PRP: rank documents by $p(rel = 1|q, d)$

- ***Theorem***. *The PRP is optimal, in the sense that it minimizes the expected loss (Ripley 1996)*

Notations: *in future slides, q denotes the query, d denotes the document, rel denotes the relevance judgment*

Estimating $p(rel = 1|q, d)$

$$p(rel = 1|q, d) = \frac{p(rel = 1, q, d)}{p(q, d)}$$

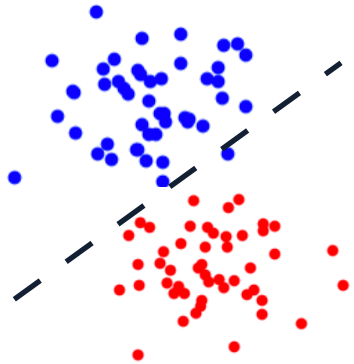
$$= \frac{count(rel = 1, q, d)}{count(q, d)}$$

$$p(rel = 1|q, d) \propto p(d|rel = 1, q)p(rel = 1)$$

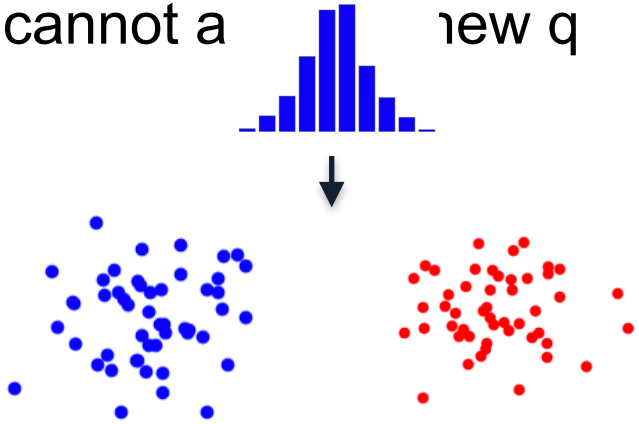
Problems with this estimation?

1. not enough data
2. cannot a new q

generative model



discriminative model



generative model

Estimating $p(rel = 1|q, d)$

$$p(rel = 1|q, d) \propto p(d|rel = 1, q)p(rel = 1)$$

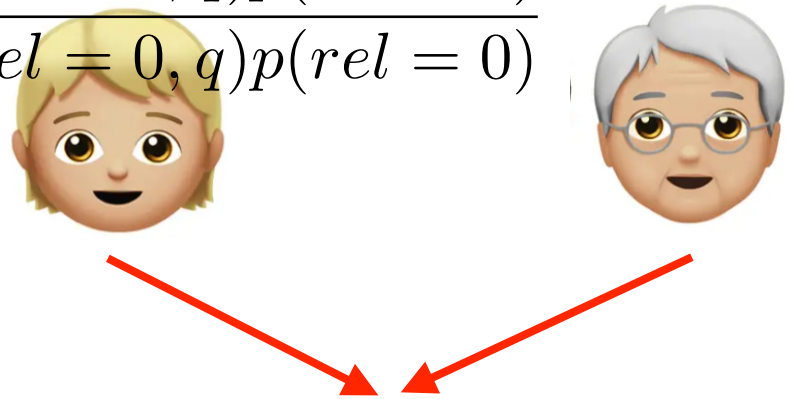
Problems with this estimation

$$O(rel = 1|q, d) = \frac{p(rel = 1|q, d)}{p(rel = 0|q, d)}$$

Difference in absolute probability of relevance

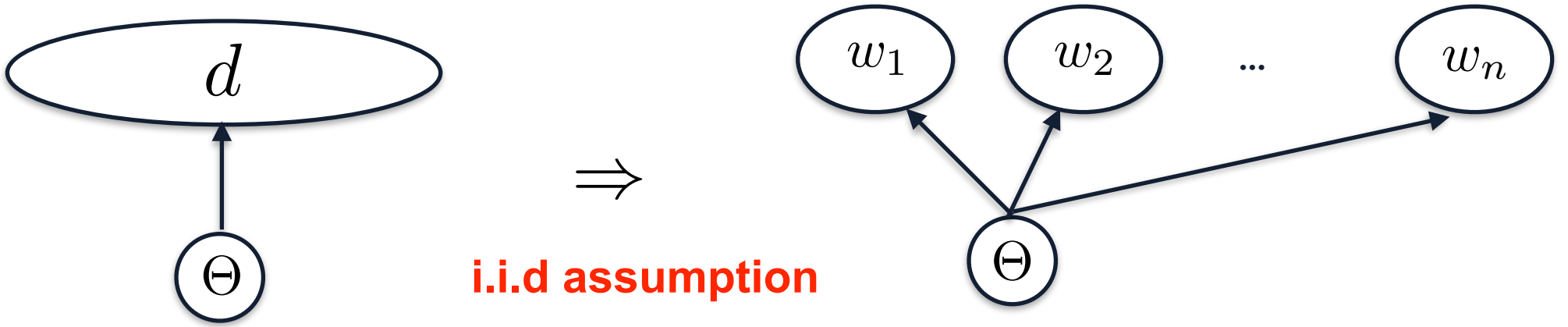
$$= \frac{p(d|rel = 1, q)p(rel = 1)}{p(d|rel = 0, q)p(rel = 0)}$$

odds



agree on the relative order

Estimating the generative model $p(d|rel=1,q)$



$$p(d|rel=1,q) = \prod_i p(w_i|rel=1,q)$$

$$O(rel=1|q,d) = \prod_i \frac{p(w_i|rel=1,q)}{p(w_i|rel=0,q)} \times \frac{p(rel=1)}{p(rel=0)}$$

$$\stackrel{rank}{=} \prod_{w_i=1} \frac{\alpha_i}{\beta_i} \times \prod_{w_i=0} \frac{(1-\alpha_i)}{(1-\beta_i)}$$

$$\stackrel{rank}{=} \prod_{w_i=1} \frac{\alpha_i}{\beta_i} \times \prod_{w_i=1} \frac{(1-\beta_i)}{(1-\alpha_i)} \times const$$

$$= \prod_{w_i=1} \frac{\alpha_i(1-\beta_i)}{\beta_i(1-\alpha_i)}$$

$$\stackrel{rank}{=} \sum_{w_i=1} \log \frac{\alpha_i(1-\beta_i)}{\beta_i(1-\alpha_i)}$$

RSJ model

$$O(rel=1|q,d) \overset{rank}{=} \sum_{w_i=1} \log \frac{\alpha_i(1-\beta_i)}{\beta_i(1-\alpha_i)}$$

(Robertson & Sparck Jones 76)

$$\begin{aligned}\alpha_i =& p(w_i=1|q,rel=1) \\ =& \frac{count(w_i=1,rel=1)+0.5}{count(rel=1)+1}\end{aligned}$$

Probability for a word to appear in a relevant doc

$$\begin{aligned}\beta_i =& p(w_i=0|q,rel=0) \\ =& \frac{count(w_i=0,rel=0)+0.5}{count(rel=0)+1}\end{aligned}$$

Probability for a word to appear in a non-relevant doc

RSJ model: Summary

- Uses only **binary word** occurrence (binary inference model), does not leverage TF information
  - RSJ model was designed for retrieving short text and abstract!

- Requires relevance judgment
  - No-relevance judgment version: [Croft & Harper 79]

- Performance is not as good as tuned vector-space model

How to improve RSJ based on these desiderata?

Desiderata of retrieval models

- Recall the desiderata of a retrieval models:
  - The importance of TF is sub-linear
  - Penalizing term with large document frequency using IDF
  - Document length normalization

How to improve RSJ based on these desiderata?

Okapi/BM25

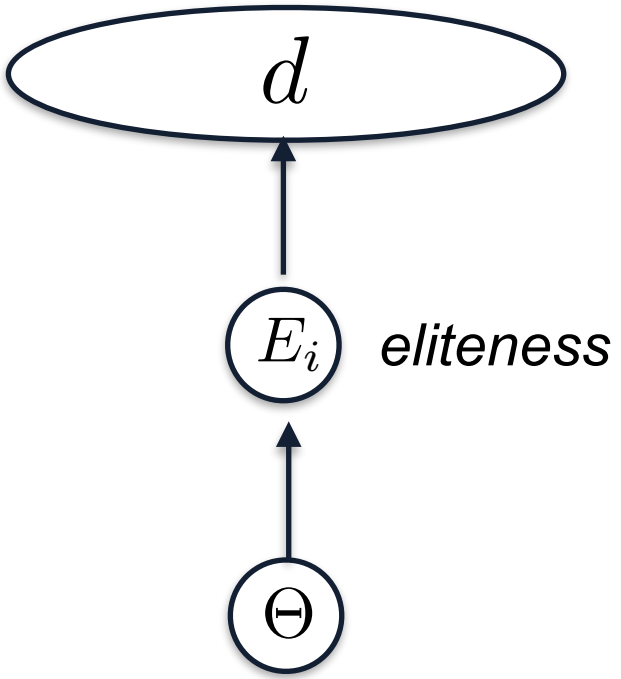
- Introduced in 1994
  - SOTA retrieval model that does not required training

- $$score(q,d) = \sum_i c_i^{elite}(tf_i)$$

$$c_i^{elite}(tf_i) = \log \frac{p(w_i = tf_i|q, rel = 1)p(w_i = 0|q, rel = 0)}{p(w_i = 0|q, rel = 1)p(w_i = tf_i|q, rel = 0)}$$

$$\begin{aligned} p(w_i = tf_i|q, rel = 1) &= p(w_i = tf_i|E_i = 1)p(E_i = 1|q, rel) \\ &+ p(w_i = tf_i|E_i = 0)p(E_i = 0|q, rel) \\ &= \pi \frac{\lambda^{tf_i}}{tf_i!} e^{-\lambda} + (1-\pi)\frac{\mu^{tf_i}}{tf_i!} e^{-\mu} \end{aligned}$$

(2 Poisson model)

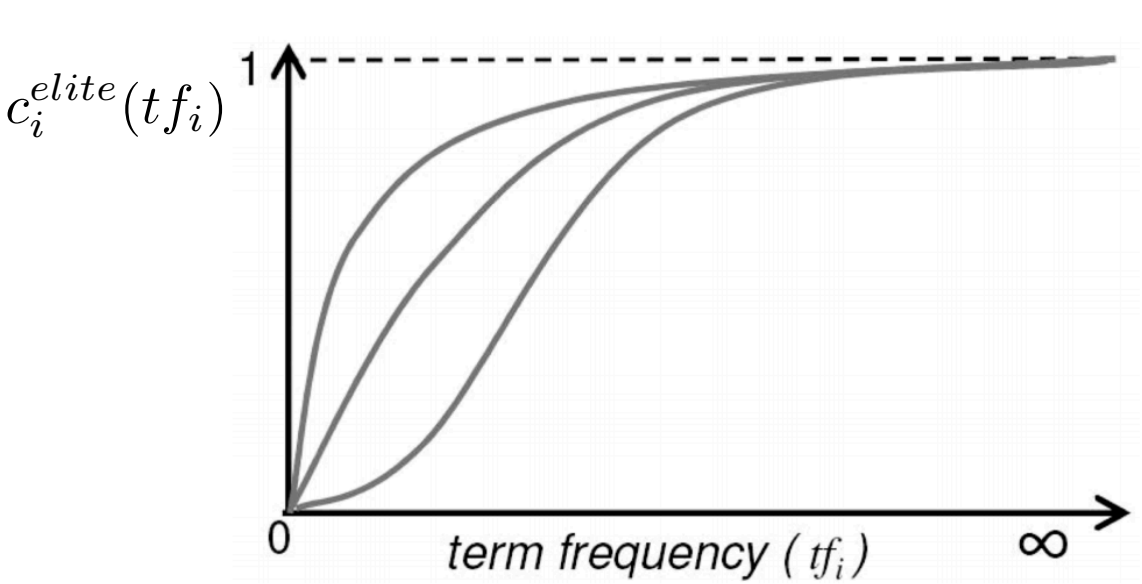


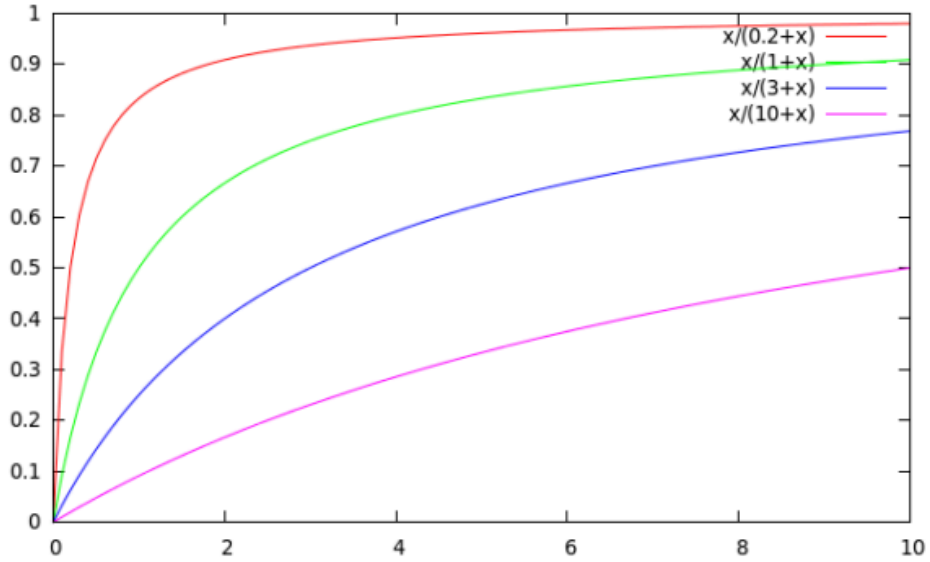
Okapi/BM25

$$p(w_i = tf_i|q, rel = 1) = \pi \frac{\lambda^{tf_i}}{tf_i!} e^{-\lambda} + (1 - \pi) \frac{\mu^{tf_i}}{tf_i!} e^{-\mu}$$

- We do not know λ, μ, π
- Can we estimate λ, μ, π ? Difficulty to estimate
- Designing a parameter-free model such that it simulates $p(w_i = tf_i|q, rel = 1)$

Simulating the 2-Poisson model

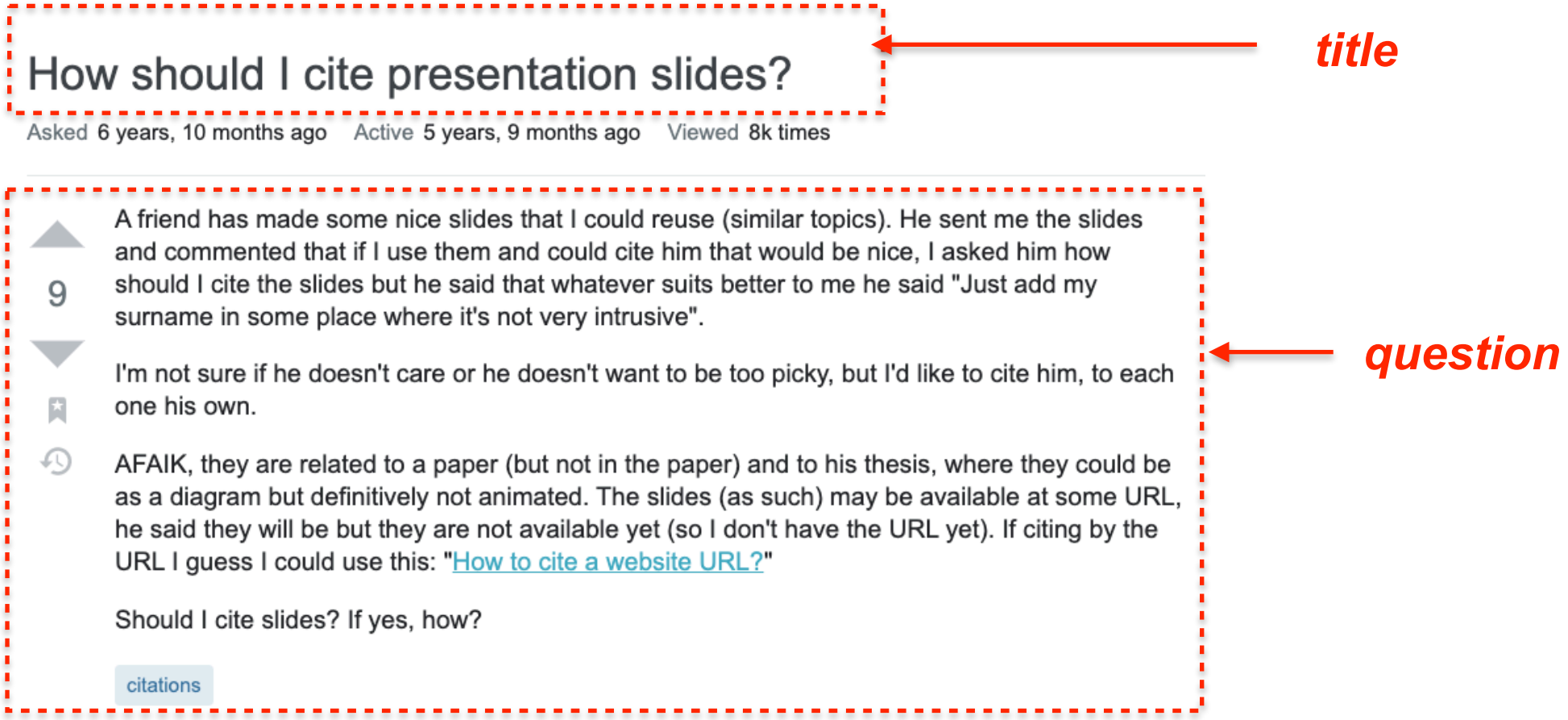




$$c_i^{BM25}(tf_i) \approx \log \frac{N}{df_i} \times \frac{tf_i(k_1+1)}{k_1(1-b+b\frac{|dl|}{|avgdl|}) + tf_i} \qquad b = 0.75, k_1 \in [1.2, 2.0]$$

slides from Stanford CS276 Information retrieval

Multi-field retrieval

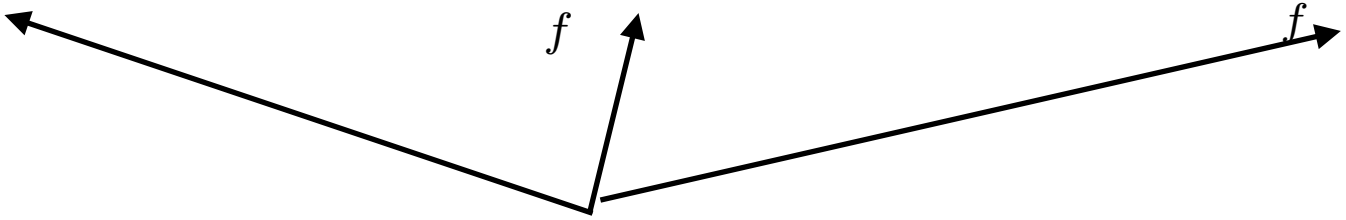


BM25F

$$score^{BM25F}(q,d) = \log \frac{N}{df_i} \times \frac{tf_i^F(k_1+1)}{k_1(1-b+b\frac{|dl|^F}{|avgdl|^F}) + tf_i^F}$$

- Each variable is estimated as the weighted sum of its field value

$$tf_i = \sum_f \alpha_f \times tf_{i,f} \qquad dl = \sum_f \alpha_f \times dl_f \qquad avgdl = \sum_f \alpha_f \times avgdl_f$$



parameter estimation using grid search

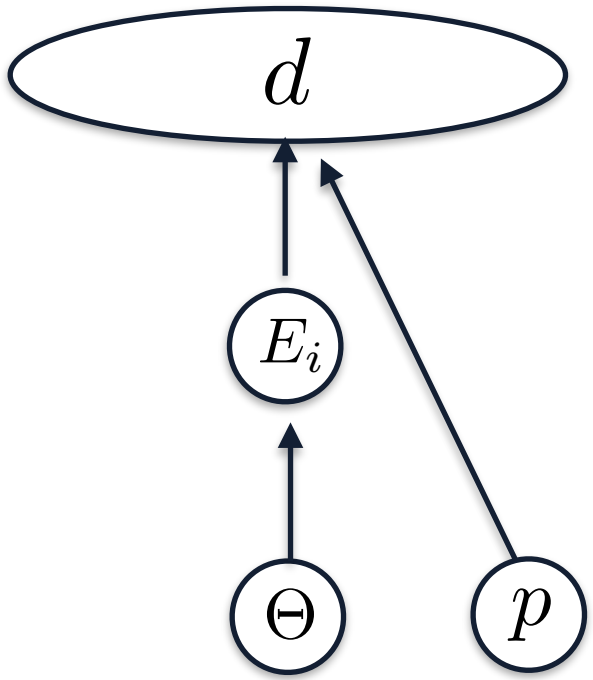
Multi-field retrieval

- BM25 outperforms TF-IDF in every field & combined

	[1.0, 0.0, 0.0]	[0.0, 1.0, 0.0]	[0.0, 0.0, 1.0]	[1.0, 0.5, 0.5]
Python, bm2, ndcg@10	0.319	0.322	0.293	0.378
Python, tfidf, ndcg@10	0.317	0.274	0.276	0.355
Java, bm2, ndcg@10	0.327	0.287	0.254	0.376
Java, tfidf, ndcg@10	0.315	0.258	0.238	0.349
Javascript, bm2, ndcg@10	0.349	0.330	0.267	0.407
Javascript, tfidf, ndcg@10	0.346	0.289	0.247	0.374

Analysis on the n-Poisson model

- **Advantage:** BM25 is based on the 2-Poisson model



eliteness**: d satisfies q's information need, when q is a **single term

- **Disadvantages:**
  - For single term, documents will not fall cleanly into elite/non-elite set
  - For multiple term, requires a combinatorial explosion of elite set
  - Requires explicit indexing of the 'elite' words

Language model-based retrieval

- A language model-based retrieval method [Ponte and Croft, 1998]

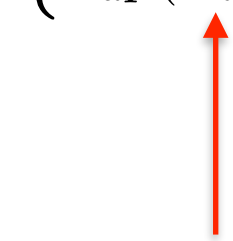
$$score(q,d) = \log p(q|d) = \prod_{i, w_i \in q} p(w_i = 1|d) \prod_{i, w_i \notin q} (1.0 - p(w_i = 1|d))$$

- Bernoulli -> multinomial

$$score(q,d) = \log p(q|d) = \prod_{w_i=1}^{V} p(w_i|d)^{c(w_i,q)}$$

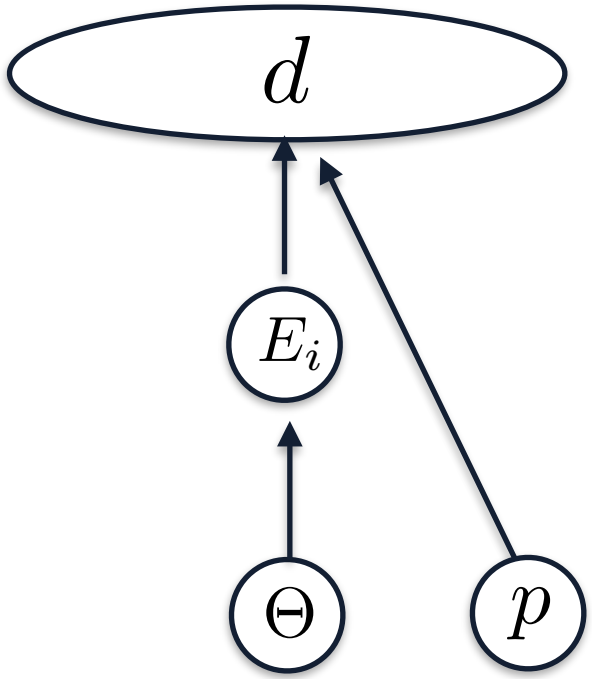
$$= \sum_{w_i=1}^{V} c(w_i,q) \log p(w_i|d)$$

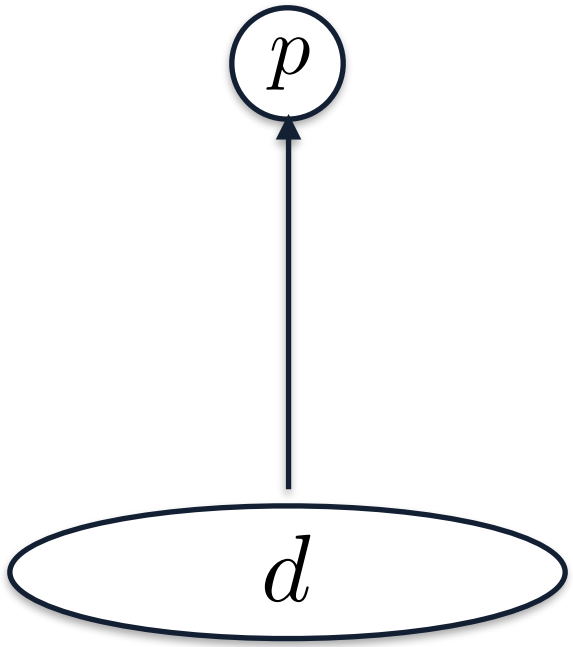
$$p(w_i|d) = \begin{cases} p_{seen}(w_i|d) & \text{if } w_i \text{ is seen in d} \\ \alpha_d p(w_i|C) & o.w. \end{cases}$$



corpus unigram LM

Language model-based retrieval





Disclaimer: the right figure is a schematic model, not a rigorous graphical model

Language model-based retrieval

$$\log p(q|d) = \sum_{w_i}^{V} c(w_i, q) \log p(w|d)$$

$$= \sum_{w_i, w_i \in d} c(w_i, q) \log p_{seen}(w_i|d) + \sum_{w_i, w_i \notin d} c(w_i, q) \log \alpha_d p(w_i|C)$$

. . .

$$= \sum_{w_i, w_i \in d} c(w_i, q) \log \frac{p_{seen}(w_i|d)}{p(w_i|C)} + |q| \log \alpha_d + \sum_{w_i=1}^{V} c(w_i, q) \log p(w_i|C)$$

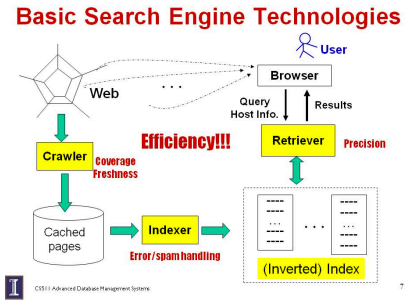
$$score^{LM}(q, d) \overset{rank}{=} \sum_{w_i, w_i \in d} c(w_i, q) \log \frac{p_{seen}(w_i|d)}{\alpha_d p(w_i|C)} + |q| \log \alpha_d$$

constant

efficient to compute, general formulation

Different senses of ‘model’ [Ponte and Croft, 98]

- First sense: an abstraction of the retrieval task itself



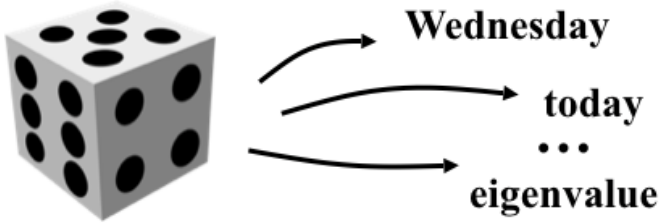
decouple → **retrieval model + other problems (e.g., indexing)**

- Second sense: modeling the distribution, e.g., 2-Poisson model

- Thirds sense: which **statistical language model** is used in $p_{seen}(w_i|d)$

Statistical language model

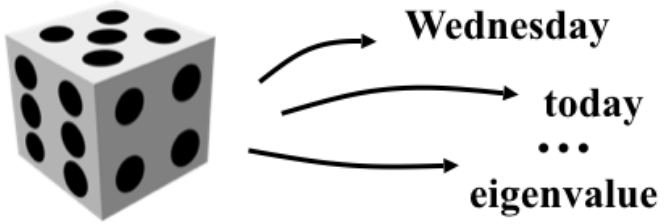
- A probability distribution over word sequences
  - p("*Today is Wednesday*") ≈ 0.001
  - p("*Today Wednesday is*") ≈ 0.0000000000001
  - p("*The eigenvalue is positive*") ≈ 0.00001
- Unigram language model
  - Generate text by generating each word INDEPENDENTLY
  - Thus, $p(w_1\ w_2\ ...\ w_n)=p(w_1)p(w_2)\ldots p(w_n)$
  - Parameters: $\{p(t_i)\}$ $p(t_1)+\ldots+p(t_N)=1$ (N is voc. size)



$$p(\text{"}today\ is\ Wed\text{"})$$
$$= p(\text{"}today\text{"})p(\text{"}is\text{"})p(\text{"}Wed\text{"})$$
$$= 0.0002 \times 0.001 \times 0.000015$$

Statistical language model

- A probability distribution over word sequences
  - p("*Today is Wednesday*") ≈ 0.001
  - p("*Today Wednesday is*") ≈ 0.0000000000001
  - p("*The eigenvalue is positive*") ≈ 0.00001
- Unigram language model
  - Generate text by generating each word INDEPENDENTLY
  - Thus, $p(w_1\ w_2\ ...\ w_n)=p(w_1)p(w_2)\ldots p(w_n)$
  - Parameters: $\{p(t_i)\}$ $p(t_1)+\ldots+p(t_N)=1$ (N is voc. size)



$$p(\text{"}today\ is\ Wed\text{"})$$
$$= p(\text{"}today\text{"})p(\text{"}is\text{"})p(\text{"}Wed\text{"})$$
$$= 0.0002 \times 0.001 \times 0.000015$$

Notes on language model-based retrieval

- **Advantages**:
  - Avoided the disadvantages in eliteness
  - Defines a general framework, more accurate $p_{seen}(w_i|d)$ can further improve the model
  - In some cases, has outperformed BM25

- **Disadvantages**:
  - The assumed equivalence between query and document is unrealistic
  - Only studied unigram language model
  - Performance is not always good

Equivalence to KL-divergence retrieval model

$$score^{LM}(q,d) \overset{rank}{=} \sum_{w_i, w_i \in d} c(w_i,q) \log \frac{p_{seen}(w_i|d)}{\alpha_d p(w_i|C)} + |q| \log \alpha_d$$

- KL divergence

$$D(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

$$-D(\hat{\theta}_q\|\hat{\theta}_d) = \sum_{w_i=1}^{V} p(w_i|\hat{\theta}_q) \log p(w_i|\hat{\theta}_d) + \left(-\sum_{w_i=1}^{V} p(w_i|\hat{\theta}_q) \log p(w_i|\hat{\theta}_d)\right)$$

why not the opposite?

smoothed ***constant***

$$\overset{rank}{=} \sum_{w_i, w_i \in d} p(w_i|\hat{\theta}_q) \log \frac{p_{seen}(w_i|d)}{\alpha_d p(w_i|C)} + \log \alpha_d \qquad \textit{(Eq. 1)}$$

derivation can be found in reading list

Estimating $p_{seen}(w_i|d)$

- Estimating $p_{seen}(w_i|d)$ based on the maximum likelihood estimation

$$p_{seen}(w_i|d) = \frac{count(w_i)}{|dl|}$$

- Disadvantage: if the word is unseen, probability will be 0

- Solution: language model smoothing:

$$p_s(w_i|d) = \frac{c(w_i,d) + \mu p(w_i|C)}{|d| + \mu} \qquad \alpha_d = \frac{\mu}{\mu + |d|}$$

(plug in Eq. 1)

$$= \frac{|d|}{|d| + \mu} p(w_i d) + \frac{\mu}{|d| + \mu} p(w_i|C)$$

Dirichlet smoothing

Estimating $p_{seen}(w_i|d)$

- Dirichlet smoothing

$$score^{Dir}(q,d) = \sum_{w_i, w_i \in d, p(w_i|\hat{\theta}_q)} p(w_i|\hat{\theta}_q) \log\left(1 + \frac{count(w_i,d)}{\mu p(w_i|C)}\right) + \log\frac{\mu}{\mu + |dl|}$$

- Jelinek-Mercer smoothing

$$score^{JM}(q,d) = \sum_{w_i, w_i \in d, p(w_i|\hat{\theta}_q)} p(w_i|\hat{\theta}_q) \log\left(1 + \frac{(1-\lambda)count(w_i,d)}{\lambda p(w_i|C)}\right)$$

Other smoothing methods

- Additive smoothing
- Good-Turing smoothing
- Absolute discounting
- Kneser-ney smoothing

Tuning parameters in smoothing models [Zhai and Lafferty 02]

$$score^{Dir}(q,d) = \sum_{w_i, w_i \in d, p(w_i|\hat{\theta}_q)} p(w_i|\hat{\theta}_q) \log\left(1 + \frac{count(w_i,d)}{\mu p(w_i|C)}\right) + \log\frac{\mu}{\mu + |dl|}$$

- Tuning parameter μ using "leave-one-out" method

$$\hat{\mu} = argmax_{\mu} \sum_{w_i=1}^{V} \sum_{d} \log p(w_i|d; w_i \notin d)$$

remove w_i

- Estimating parameter using Newton's method (2nd derivative)

Tuning parameters in smoothing models [Zhai and Lafferty 02]

$$score^{JM}(q,d) = \sum_{w_i, w_i \in d, p(w_i|\hat{\theta}_q)} p(w_i|\hat{\theta}_q) \log\left(1 + \frac{(1-\lambda) count(w_i, d)}{\lambda p(w_i|C)}\right)$$

- Tuning parameter λ using MLE for the query probability

$$p(q|\lambda, C) = \sum_d \pi_d \prod_{w_i \in q} \left((1-\lambda) p(w_i|d) + \lambda p(w_i|C)\right)$$

- EM algorithm:

$$\pi_d^{(k+1)} = \frac{\pi_d^{(k)} \prod_{w_i \in q} \left(\left(1-\lambda^{(k)}\right) p\left(w_i \mid d\right) + \lambda^{(k)} p\left(w_i \mid C\right)\right)}{\sum_d \pi_d^{(k)} \prod_{w_i \in q} \left(\left(1-\lambda^{(k)}\right) p\left(w_i \mid d\right) + \lambda^{(k)} p\left(w_i \mid C\right)\right)}$$

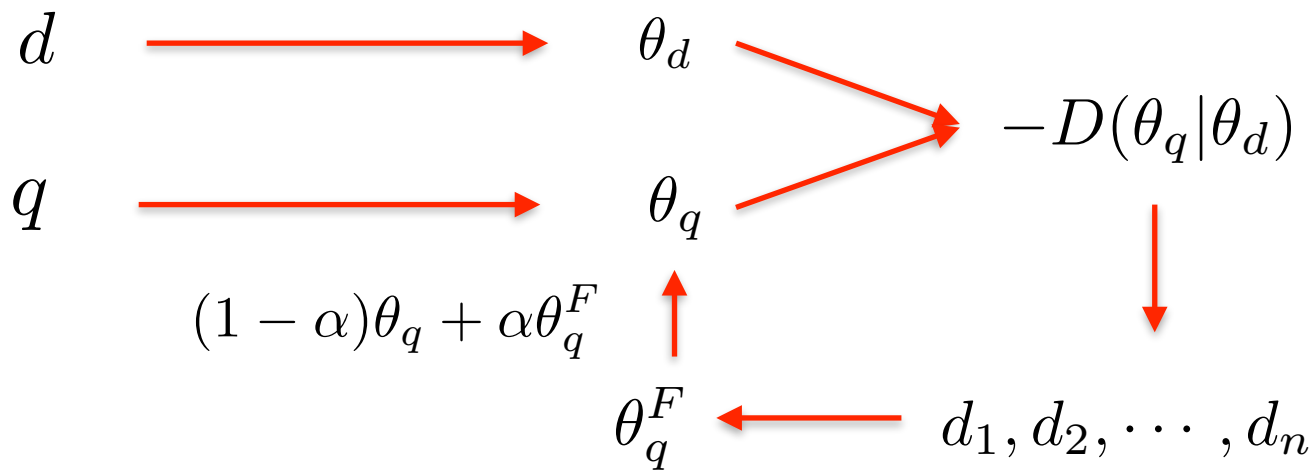
$$\lambda^{(k+1)} = \frac{1}{|q|} \sum_d \pi_d^{(k+1)} \sum_{w_i \in q} \frac{\lambda^{(k)} p\left(w_i \mid C\right)}{\left(1-\lambda^{(k)}\right) p\left(w_i \mid d\right) + \lambda^{(k)} p\left(w_i \mid C\right)}$$

Feedback language model [Zhai and Lafferty 02]

$$score^{JM}(q,d) = \sum_{w_i, w_i \in d, p(w_i|\hat{\theta}_q)} p(w_i|\hat{\theta}_q) \log\left(1 + \frac{(1-\lambda)count(w_i,d)}{\lambda p(w_i|C)}\right)$$

$$p(w_i|q) = \frac{count(w_i,q)}{|q|}$$

sparsity



Evaluation on smoothing methods [Zhai & Lafferty 02]

Collection	query	Optimal-JM	Optimal-Dir	Auto-2stage
AP88-89	SK	20.3%	23.0%	22.2%*
	LK	36.8%	37.6%	37.4%
	SV	18.8%	20.9%	20.4%
	LV	28.8%	29.8%	29.2%
WSJ87-92	SK	19.4%	22.3%	21.8%*
	LK	34.8%	35.3%	35.8%
	SV	17.2%	19.6%	19.9%
	LV	27.7%	28.2%	28.8%*
ZIFF1-2	SK	17.9%	21.5%	20.0%
	LK	32.6%	32.6%	32.2%
	SV	15.6%	18.5%	18.1%
	LV	26.7%	27.9%	27.9%*

Evaluation on smoothing methods [Zhai & Lafferty 01b]

collection		Simple LM	Mixture	Improv.	Div.Min.	Improv.
AP88-89	AvgPr	0.21	0.296	+41%	0.295	+40%
	InitPr	0.617	0.591	-4%	0.617	+0%
	Recall	3067/4805	3888/4805	+27%	3665/4805	+19%
TREC8	AvgPr	0.256	0.282	+10%	0.269	+5%
	InitPr	0.729	0.707	-3%	0.705	-3%
	Recall	2853/4728	3160/4728	+11%	3129/4728	+10%
WEB	AvgPr	0.281	0.306	+9%	0.312	+11%
	InitPr	0.742	0.732	-1%	0.728	-2%
	Recall	1755/2279	1758/2279	+0%	1798/2279	+2%

Comparison between BM25 and LM [Bennett et al. 2008]

Collection	Method	Parameter	MAP	R-Prec.	Prec@10
Trec8 T	Okapi BM25	Okapi	0.2292	0.2820	0.4380
	JM	$\lambda = 0.7$	0.2310 (p=0.8181)	0.2889 (p=0.3495)	0.4220 (p=0.3824)
	Dir	$\mu = 2,000$	**0.2470** (p=0.0757)	0.2911 (p=0.3739)	**0.4560** (p=0.3710)
	Dis	$\delta = 0.7$	0.2384 (p=0.0686)	0.2935 (p=0.0776)	0.4440 (p=0.6727)
	Two-Stage	auto	0.2406 (p=0.0650)	**0.2953** (p=0.0369)	0.4260 (p=0.4282)
Trec8 TD	Okapi BM25	Okapi	0.2528	0.2908	0.4640
	JM	$\lambda = 0.7$	0.2582 (p=0.5226)	0.3038 (p=0.1886)	0.4600 (p=0.8372)
	Dir	$\mu = 2,000$	**0.2621** (p=0.3308)	0.3043 (p=0.1587)	0.4460 (p=0.3034)
	Dis	$\delta = 0.7$	0.2599 (p=0.1737)	**0.3105** (p=0.0203)	**0.4880** (p=0.1534)
	Two-Stage	auto	0.2445 (p=0.2455)	0.2933 (p=0.7698)	0.4400 (p=0.1351)

However, BM25 outperforms LM in other cases

Feedback language model [Zhai and Lafferty 02]

slides from UVA CS4780 Introduction to information retrieval

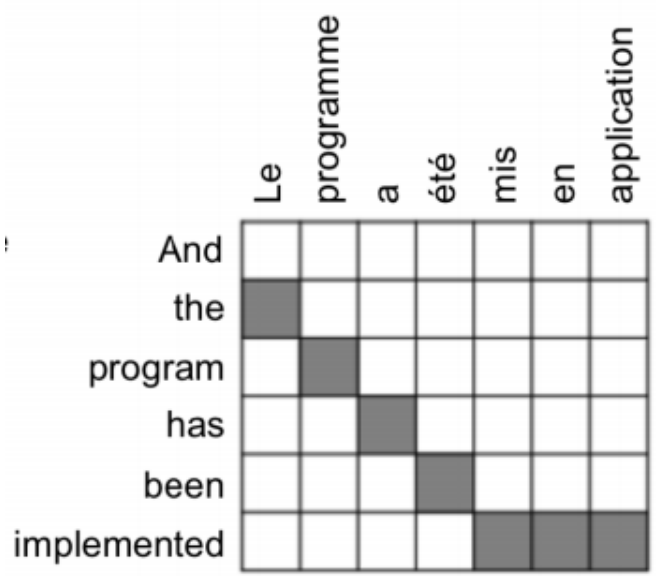
Translation-based language model

- The retrieval model can benefit from incorporating knowledge in the formulation

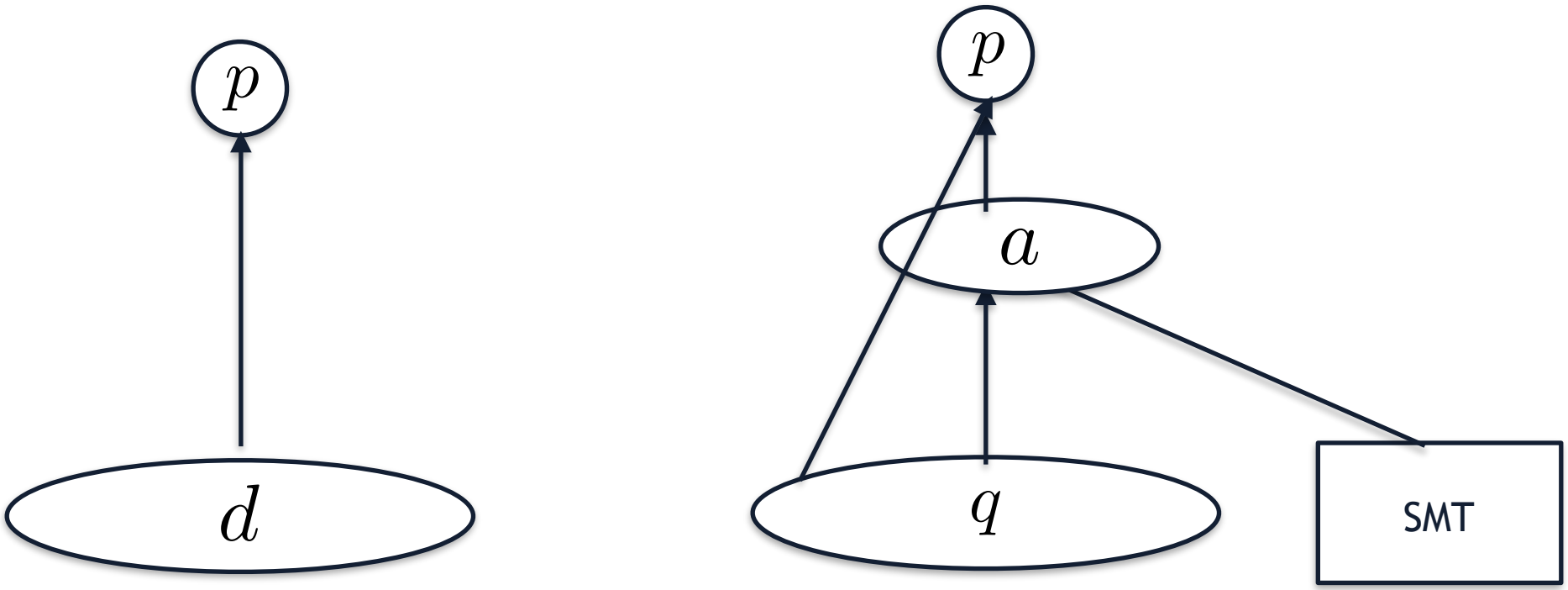
$$p_(w_i|d) = \frac{|dl|}{|dl|+\lambda} p_{mix}(w_i|d) + \frac{\lambda}{|dl|+\lambda} p(w_i|C)$$

$$p_{mix}(w_i|d) = (1-\beta) p(w_i|d) + \beta \sum_{t \in d} p_{tr}(w_i|t) p(t|d)$$

- Translation matrix:



Translation-based language model



Disclaimer: both figures are schematic models, not rigorous graphical models

Performance of translation based LM [Xue et al. 2008]

	Python			Java			JavaScript		
	R	@5	@10	R	@5	@10	R	@5	@10
TF-IDF	.299	.301	.360	.285	.282	.352	.305	.315	.378
BM25	.313	.320	.384	.311	.321	.382	.329	.344	.412
TransLM	.468	.502	.553	.455	.487	.544	.483	.528	.573

Type	Model	Trans Prob	Wondir	
			MAP	P@10
Type I	LM		0.3217	0.2211
	Okapi		0.3207	0.2158
	RM		0.3401	0.2395
Type II	LM-Comb		0.3791	0.2368
Type III	Murdock	$P(Q\|A)$	0.3566	0.25
	Murdock	$P(A\|Q)$	0.3658	0.2526
	Jeon	$P(Q\|A)$	0.3546	0.25
	Jeon	$P(A\|Q)$	0.3658	0.2526
	TransLM	$P(Q\|A)$	0.379	0.2658
	TransLM	$P(A\|Q)$	0.4059	0.2684

Discussion on query length

- What if the query is very long?
  - For example, the query is a paragraph or a document
  - The problem of retrieval is turned into a matching problem
    - i.e., semantic matching

Semantic matching