Probabilistic Retrieval Models

Many slides courtesy Javed A. Aslam@Northeastern

What is a Retrieval Model?

- Model is an idealization or abstraction of an actual process
  - Mathematical models used to study the properties of the process, draw conclusions, make predictions
- Conclusions derived from a model depend on whether the model is a good approximation of the actual situation
- Retrieval models can attempt to describe the human process
  - e.g. the information need, interaction
  - Few do so meaningfully
- Retrieval models can describe the computational process
  - e.g. how documents are ranked

Retrieval Models Overview

- Older models
  - Boolean retrieval
  - Vector Space model
- Probabilistic Models
  - Language models
  - BM25
- Combining evidence
  - Inference networks
  - Learning to Rank

Retrieval Models Overview

- Older models
  - Boolean retrieval
  - Vector Space model
- Probabilistic Models
  - Language models
  - BM25
- Combining evidence
  - Inference networks
  - Learning to Rank

Language Models

- A *language model* is a probability distribution over strings of text.
  - How likely is a given string in a given language?

- Consider probabilities for the following strings:
  - p1 = P(“language model”)
  - p2 = P(“probability distribution”)
  - p3 = P(“a language model is a probability distribution”)

- p2 > p1 > p3?

- Every possible string has some probability.
  - Depends on the language we are modelling

Language Models in Information Retrieval (IR)

- Every document in a collection defines a “language”
  - subject to topic, writing style, language …
- Consider all possible sentences (strings) that the author could have written down when creating some given document
  - some are perhaps more likely to occur than others
  - $P(s|M_D)$ = probability that author would write down string “s”
    - M_D is the language model
- Now suppose “Q” is the user’s query
  - what is the probability that author would write down “Q” ?
- Rank documents D in the collection by $P(Q|M_D)$
  - probability of observing “Q” during random sampling from the language model of document D

Unigram Language Models

- A *unigram language model* is a probability distribution over words.
  - P("language"), P("model")
  - P("probability"), P("distribution")
  - P("a"), P("is")

- Each word has a probability, probabilities sum to 1.
- Assumption: words are independent, order doesn't matter.
  - P("language model") = P("language")P("model") = P("model language")
  - P("a language model is a probability distribution") = P("a probability distribution is a language model")
- "bag of words" model.

Bigram Language Models

- A *bigram language model* is a probability distribution over pairs of words.
  - P(“language model”)
  - P(“probability distribution”)

- Or a collection of unigram language models, each conditioned on a word.
  - P(“model” | “language”)
  - P(“distribution” | “probability”)

- P(“a language model is a probability distribution”) = P(“a”)P(“language” | “a”)P(“model” |“language”)…P(“distribution” | “probability”)

- In general, n-gram models: models that consider n words

- Our main focus in this lecture: Unigram language models

Language Models (LM) for IR

- Each document D is a sample of language.
  - Consider all possible strings the author could have written while producing the document.
  - Some strings are more likely than others
  - $P(s \mid M_D)$ is the probability that the author would write string s.

- Three main approaches to LM in IR: Given a query Q and document D,
  - Query likelihood model: $P(Q|M_D)$
    - What is the probability to generate the given query, given a document language model?
  - Document likelihood model: $P(D|M_Q)$
    - What is the probability to generate the given document, given a query language model?
  - Divergence model: $D(P(M_D) \,||\, P(M_Q))$
    - How ”close” are 2 statistical models?

Query-Likelihood Model

- Standard language modelling approach

- Rank documents by the probability that the query could be generated by the document model
  - estimate a language model M_D for every document D in the collection
  - rank documents by the probability of “generating” the query

$$P(Q \mid M_D) = P(q_1 \ldots q_k \mid M_D) = \prod_{i=1}^{k} P(q_i \mid M_D)$$

- Drawbacks:
  - no notion of relevance in the model: everything is random sampling
  - user feedback / query expansion not part of the model
    - examples of relevant documents cannot help us improve the language model

Document-Likelihood Model

- Flip the direction of the query-likelihood approach
  - estimate a language model M_Q for the query Q
  - rank docs D by the likelihood of being a random sample from M_Q

$$P(D \mid M_Q) = \prod_{w \in D} P(w \mid M_Q)$$

- M_Q expected to “predict” a typical relevant document

- Drawbacks:
  - different doc lengths, probabilities not comparable
  - favours documents that contain frequent (low content) words

Divergence Model

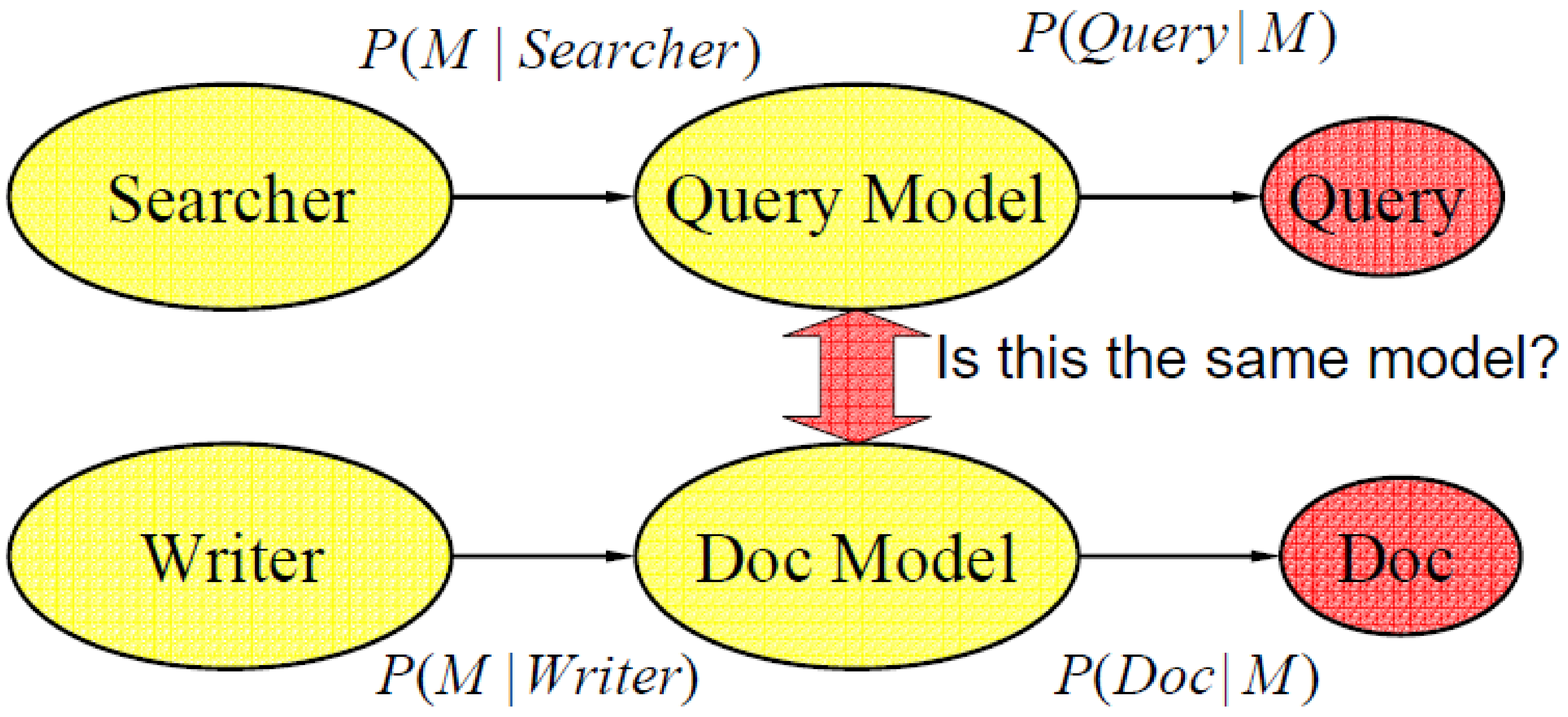
- Combine advantages of query-likelihood and document-likelihood models
  - estimate a model of both the query M_Q and the document M_D
  - directly compare similarity of the two models
  - natural measure of similarity is cross-entropy (Kullback-Leiblar divergence)

  - $H(M_Q||M_D) = -\sum_w P(w|M_Q) \log(P\,(w|M_D)/\,P(w|M_Q))$
  - $\qquad\qquad\qquad \underset{\text{RANK}}{=} -\sum_w P(w|M_Q) \log P\,(w|M_D)$

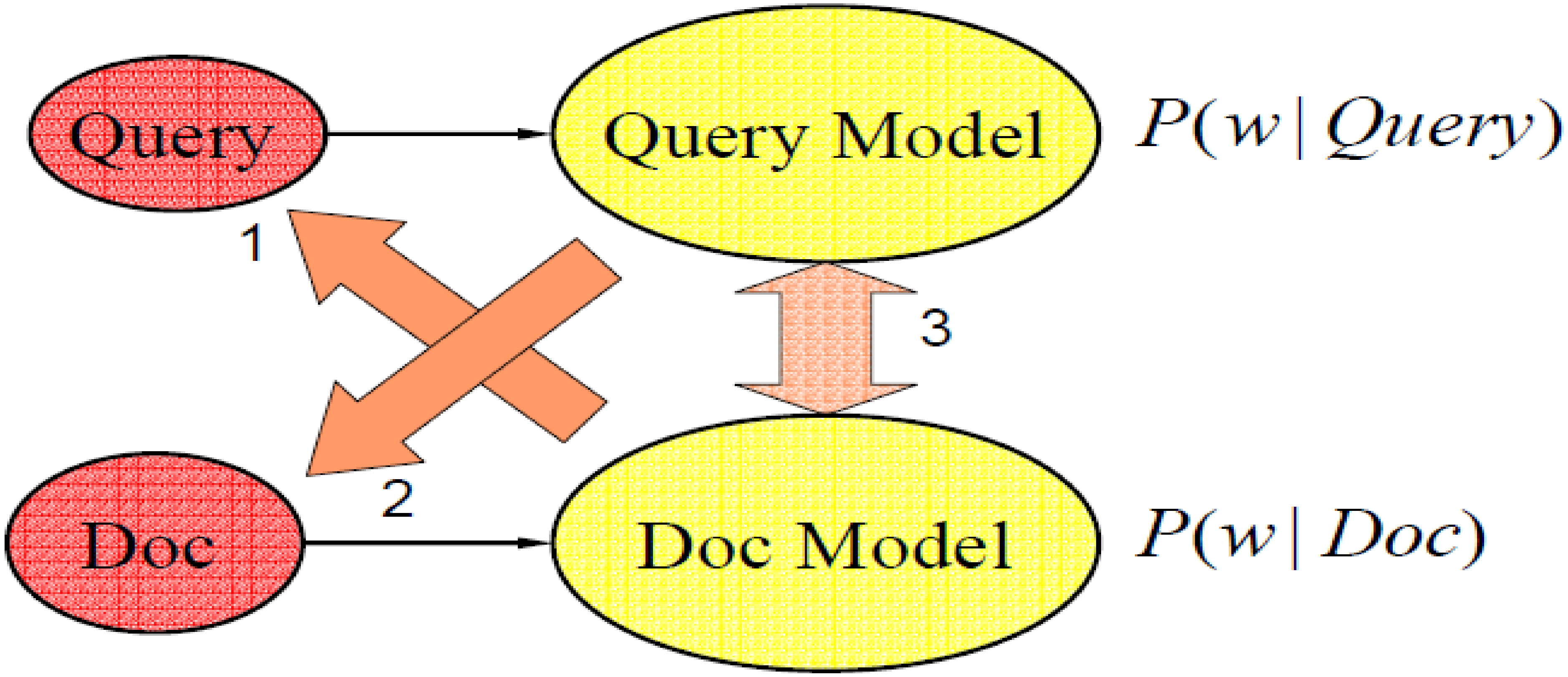
  - number of bits needed to encode M_Q using M_D

- Cross-entropy is not symmetric: use H (M_Q || M_D)
  - reverse works consistently worse, favours different documents

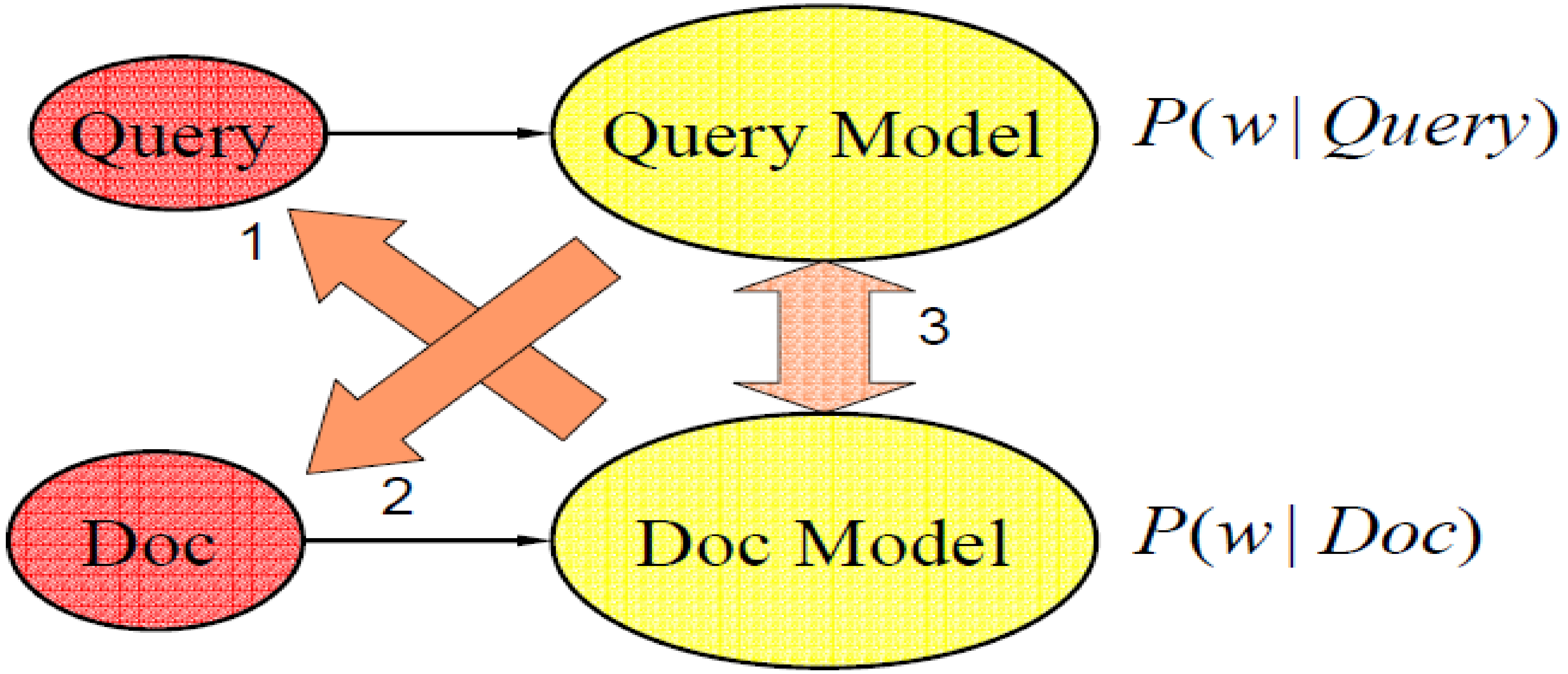
Models of Text Generation



Retrieval with Language Models



Retrieval with Language Models



1: Query likelihood approach
2: Document likelihood approach
3: Divergence approach

Constructing the Language Models

- We need to estimate M_D and/or M_Q from Q and/or D
  - Language models are distribution over words

- Estimating the distribution:

$$M_D = P(w \mid D) = \frac{f_{w,D}}{|D|}$$

  - Maximum likelihood estimate
  - Makes the observed value of $f_{w,D}$ most likely

- If query words are missing from document, score will be zero
  - Missing 1 out of 4 query words same as missing 3 out of 4

Example: Unigram Query-Likelihood Model

- Document:

The majority of Americans consider tobacco advertising a major influence in promoting the killer habit. Approximately 57% of the public thinks that cigarette advertising causes people to smoke. Also, 47% thinks that cigarette advertising makes it harder for smokers to give up the habit. If the tobacco industry didn't agree with these stats it wouldn't concentrate so heavily on using young models in its ads.

Query: ***"tobacco advertising"***

$P(Q|M_D) =$ *2/65* *3/65* $P(Q|D) = 6/4225$

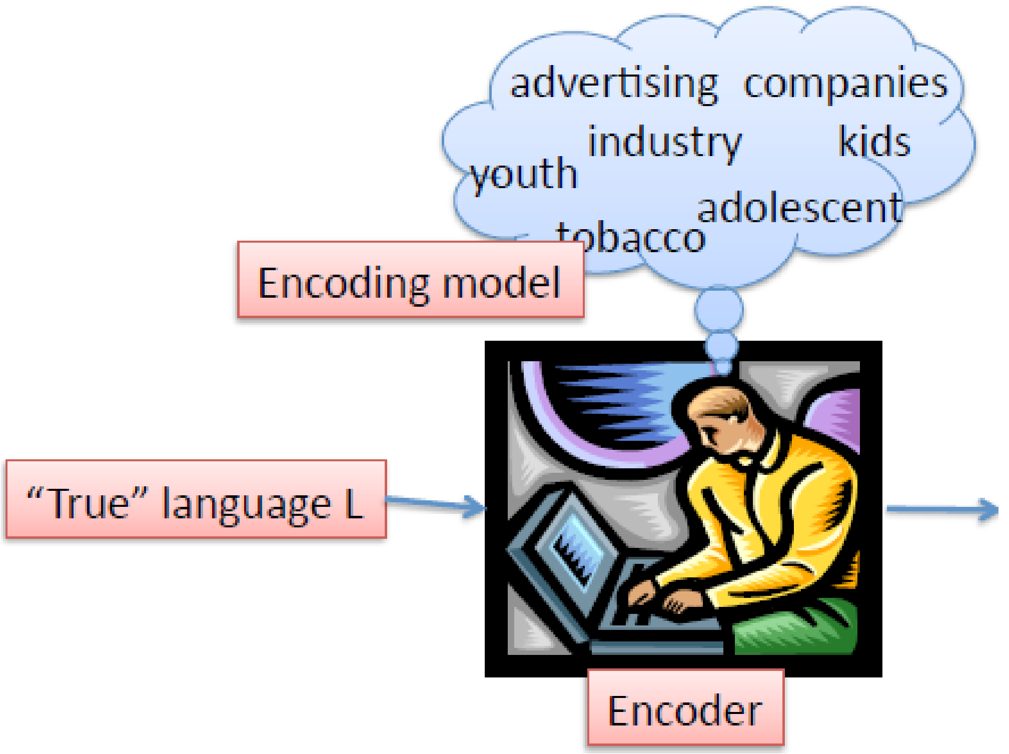
"tobacco companies"

$P(Q|M_D) =$ *2/65* *0/65* $P(Q|D) = 0$

"tobacco companies and the young"

$P(Q|D) = 0$

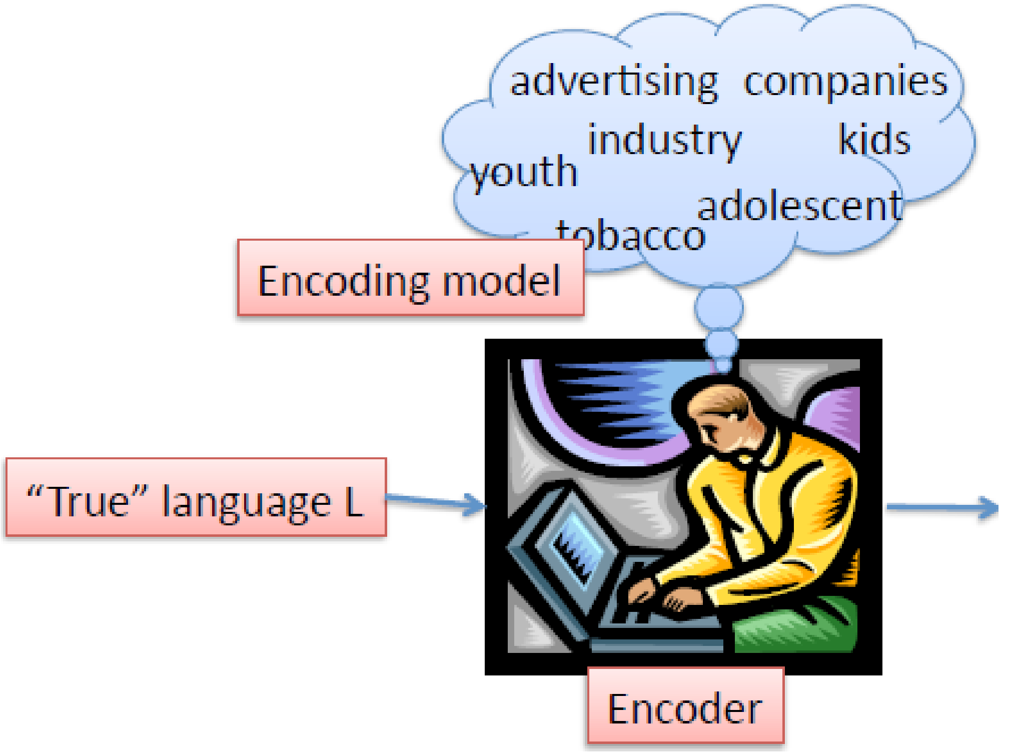
Language Generation as Encoding



The majority of Americans consider tobacco advertising a major influence in promoting the killer habit. Approximately 57% of the public thinks that cigarette advertising causes people to smoke. Also, 47% thinks that cigarette advertising makes it harder for smokers to give up the habit. If the tobacco industry didn't agree with these stats it wouldn't concentrate so heavily on using young models in its ads.

Encoded language

Language Generation as Encoding



The majority of Americans consider tobacco advertising a major influence in promoting the killer habit. Approximately 57% of the public thinks that cigarette advertising causes people to smoke. Also, 47% thinks that cigarette advertising makes it harder for smokers to give up the habit. If the tobacco industry didn't agree with these stats it wouldn't concentrate so heavily on using young models in its ads.

Encoded language

Problem: We only have the encoded language.
We need to estimate our model from that.

Solution: 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
- Gives every term (word) some probability of occurring in every document.

Simple Smoothing Methods: Laplace Smoothing

- Count events in observed data
- Add 1 to every count
- Renormalize to obtain probabilities
- Corresponds to having uniform priors over words

- If number of occurrences of words in D are (m_1, m_2, …, $m_{|V|}$) with $\sum_i m_i = N$, where |V|: number of unique words in the entire collection (vocabulary size)
    - Max likelihood estimates: $\left(\frac{m_1}{|D|}, \frac{m_2}{|D|}, \ldots, \frac{m_{|V|}}{|D|}\right)$
    - Laplace estimates: $\left(\frac{m_1+1}{|D|+|V|}, \frac{m_2+1}{|D|+|V|}, \ldots, \frac{m_{|V|}+1}{|D|+|V|}\right)$

Discounting Methods

- Laplace smoothing gives too much weight to unseen terms

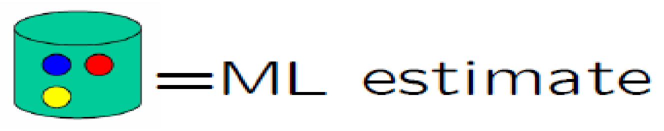
- Lindstone correction:
  - Add small number ε to every term count, renormalize:

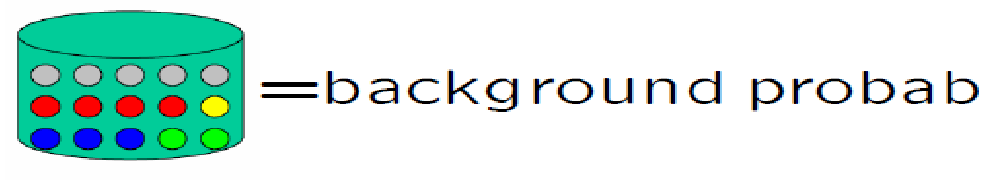
$$P(w|D) = \frac{tf_{w,D} + \varepsilon}{|D| + \varepsilon |V|}$$

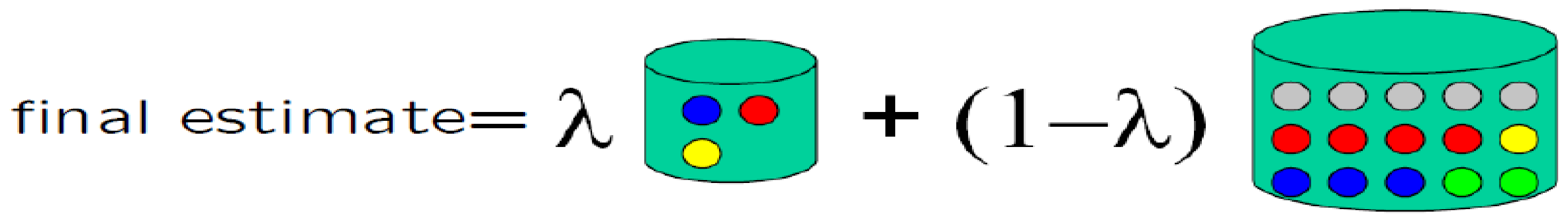
- Absolute discounting:
  - Subtract ε, re-distribute the probability mass

Interpolation Methods

- Problem with all discounting methods:
  - discounting treats unseen words equally (add or subtract **ε**)
  - some words are more frequent than others
- Idea: use background probabilities
  - "interpolate" ML estimates with the estimates from the collection
    - E.g. General English expectations (computed as relative frequency of a word in a large collection)
  - reflects expected frequency of events







Smoothing with Background

- Estimate for unseen words is $P(w|C)$, where $P(w|C)$ is the probability for word w in the *collection* language model for collection (background probability)

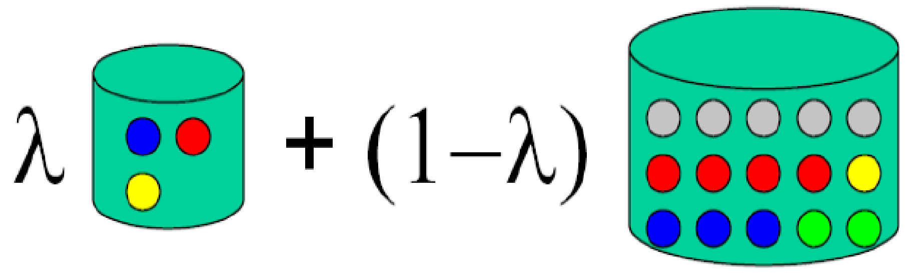
- Estimate for words that occur is
  - $\lambda P(w|D) + (1 - \lambda) D P(w|C)$

- λ is a parameter
  - Correctly setting λ is very important
  - Different forms of estimation come from how λ *is set*

Jelinek-Mercer Smoothing

- Start simple
  - set **λ** to be a constant, independent of document, query

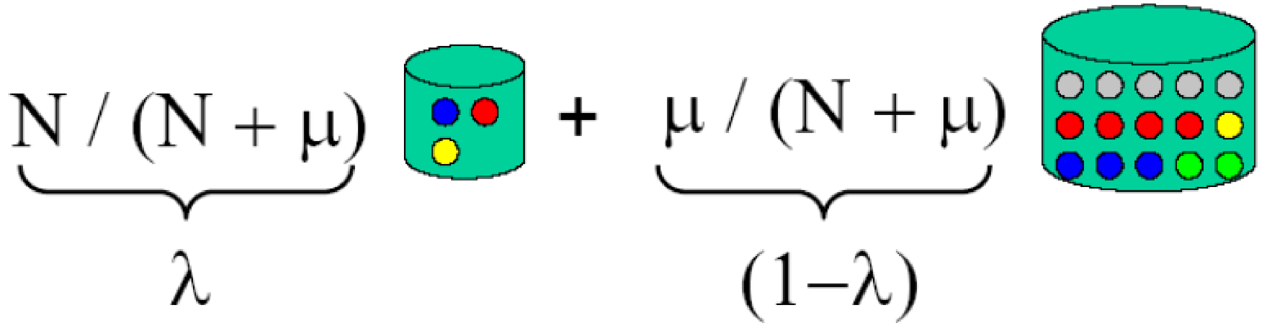
- Tune to optimize retrieval performance
  - optimal value of **λ** varies with different databases, query sets, etc.



Dirichlet Smoothing

- Problem with Jelinek-Mercer:
  - longer documents provide better estimates
    - could get by with less smoothing

- Make smoothing depend on sample size
  - N is length of sample = document length
  - μ is a constant


$\underbrace{N / (N + \mu)}_{\lambda}$
+
$\underbrace{\mu / (N + \mu)}_{(1-\lambda)}$

Example: Query-likelihood Example with Dirichlet Smoothing

- Query: “president lincoln”

- For the term “president”
  - *fqi,D* = 15, *cqi* = 160,000
- For the term “lincoln”
  - *fqi,D* = 25, *cqi* = 2,400

- Number of word occurrences in the 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
- μ = 2,000

Example: Query-likelihood Example with Dirichlet Smoothing

- $P(Q|D_M) \propto \log P(Q|D_M) = \log \prod_{\forall q_i \in Q} P(q_i \mid D) = \sum_{\forall q_i \in Q} \log P(q_i \mid D)$

$$= \log P(\text{"president"}| D) + \log P(\text{"lincoln"}| D)$$

$$= \log[\frac{1800}{1800+2000} \cdot \frac{15}{1800} + \frac{2000}{1800+2000} \cdot \frac{160000}{10^9}] +$$

$$\log[\frac{1800}{1800+2000} \cdot \frac{25}{1800} + \frac{2000}{1800+2000} \cdot \frac{2400}{10^9}]$$

$$= -10.55$$

Example: Query-likelihood Example with Dirichlet Smoothing

Frequency of "president"	Frequency of "Lincoln"	Query Likelihood Score
15	25	-10.55
15	1	-13.75
15	0	-19.05
1	25	-12.99
0	25	-14.40

What is the best smoothing?

- Don’t make too many assumptions.
- Don’t smooth too much.

- Dirichlet works well for short queries (need to tune the parameter)
- Jelinek-Mercer works well for longer queries (also needs tuning)
- In general, Dirichlet smoothing seems to provide the best “happy-medium”.

Language Models: Summary

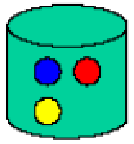
- Goal: estimate a model M from a sample text S
  - Use maximum-likelihood estimator
  - Count the number of times each word occurs in S, divide by length

- Smoothing to avoid zero frequencies
  - Discounting methods: add or subtract a constant, redistribute mass
  - Better: interpolate with background probability of a word
  - Smoothing has a role similar to IDF in classical models
  - Smoothing parameters very important

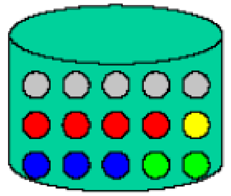
Example: Query-likelihood Example with Dirichlet Smoothing

- Make smoothing depend on sample size
  - N is length of sample = document length
  - μ is a constant

$$\underbrace{N / (N + \mu)}_{\lambda}$$



$$+ \quad \underbrace{\mu / (N + \mu)}_{(1-\lambda)}$$



Where is tf-idf weight?

$$P(Q \mid D) \alpha \log P(Q \mid D) = \sum_{i=1}^{n} \log\left(\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}\right)$$

$$= \sum_{i: f_{q_i,D}>0} \log\left(\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}\right) + \sum_{i: f_{q_i,D}=0} \log\left((1-\lambda)\frac{c_{q_i}}{|C|}\right)$$

$$= \sum_{i: f_{q_i,D}>0} \log \frac{\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log\left((1-\lambda)\frac{c_{q_i}}{|C|}\right)$$

$$\overset{rank}{=} \sum_{i: f_{q_i,D}>0} \log\left(\frac{\lambda \frac{f_{q_i,D}}{|D|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + 1 \right)$$

Where is tf-idf weight?

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

$$= \sum_{i: f_{q_i,D}>0} \log(\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}) + \sum_{i: f_{q_i,D}=0} \log((1-\lambda)\frac{c_{q_i}}{|C|})$$

$$= \sum_{i: f_{q_i,D}>0} \log \frac{\lambda \frac{f_{q_i,D}}{|D|} + (1-\lambda)\frac{c_{q_i}}{|C|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + \sum_{i=1}^{n} \log((1-\lambda)\frac{c_{q_i}}{|C|})$$

$$\overset{rank}{=} \sum_{i: f_{q_i,D}>0} \log\left(\frac{\lambda \frac{f_{q_i,D}}{|D|}}{(1-\lambda)\frac{c_{q_i}}{|C|}} + 1 \right)$$

- Proportional to the term frequency
- Inversely proportional to the collection frequency

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

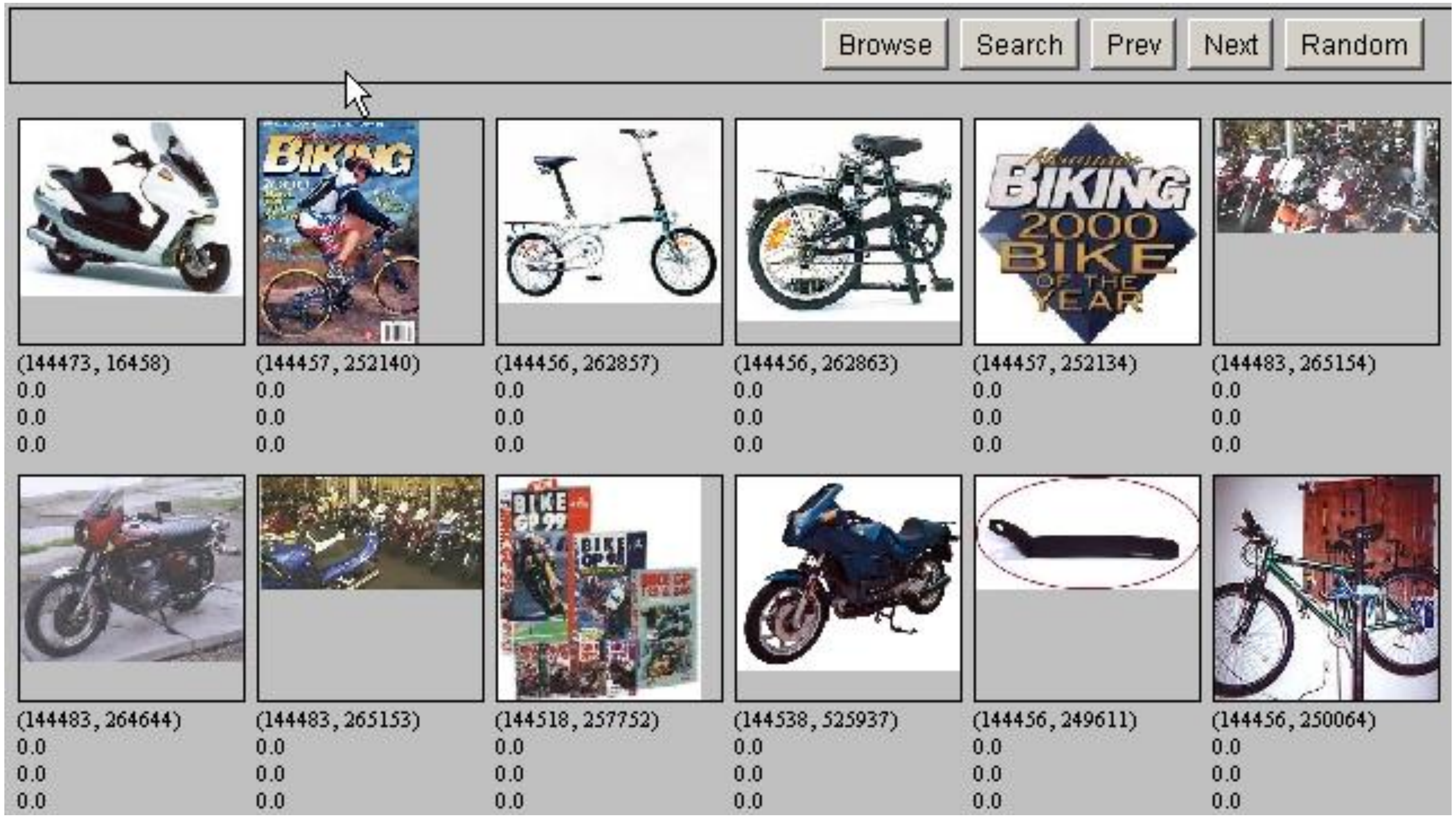
- Different in others
  - Based on probability rather than similarity
    - Intuitions are probabilistic rather than geometric
  - Details of use of document length and term, document, and collection frequency differ

Relevance Feedback

- Relevance feedback: user feedback on relevance of docs in initial set of results
  - User issues a (short, simple) query
  - The user marks some results as relevant or non-relevant.
  - The system computes a better representation of the information need based on feedback.
  - Relevance feedback can go through one or more iterations.

- Idea: it may be difficult to formulate a good query when you don’t know the collection well, so iterate

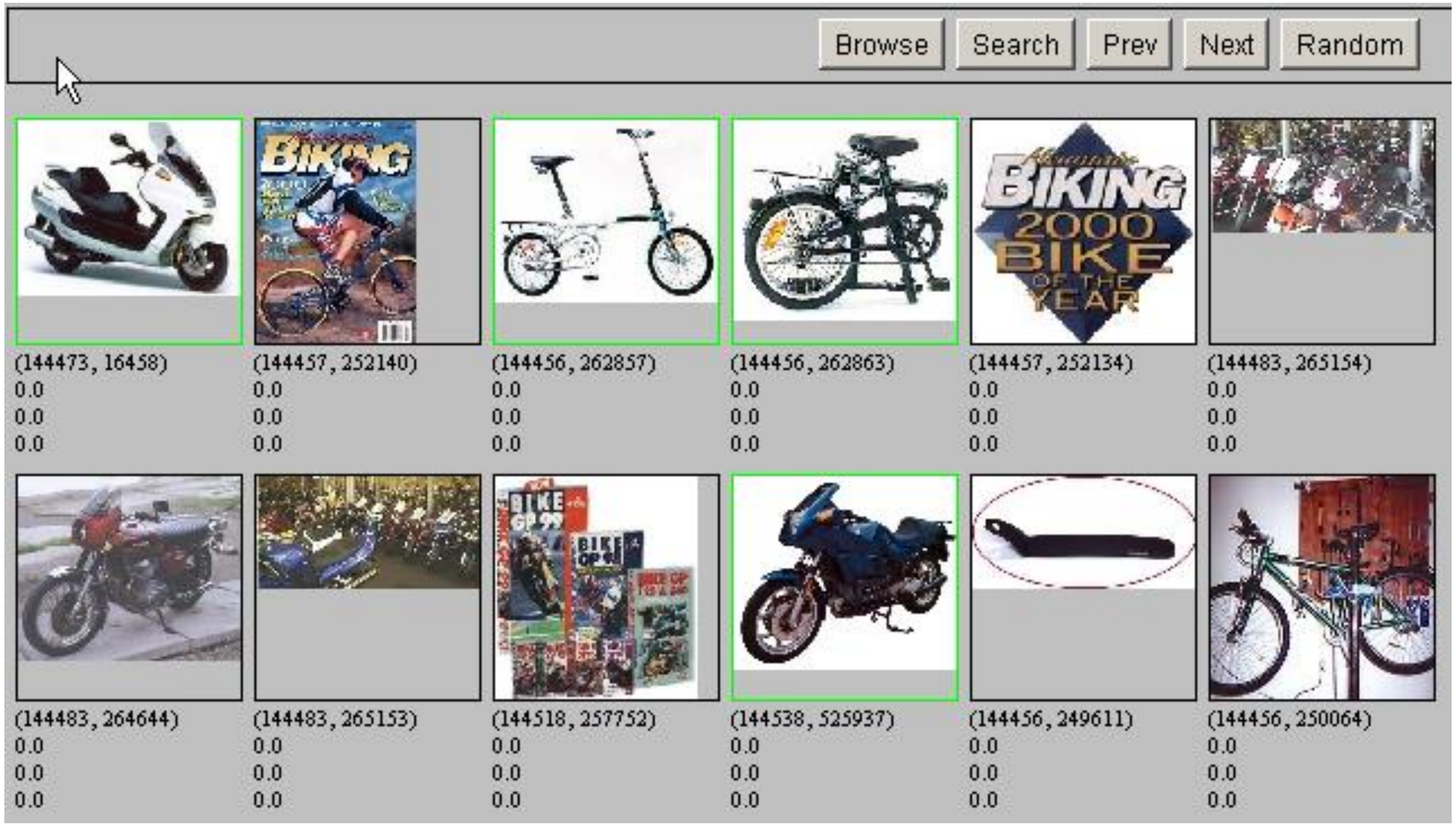
Results for Initial Query



```
(144473, 16458)   (144457, 252140)  (144456, 262857)  (144456, 262863)  (144457, 252134)  (144483, 265154)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0

(144483, 264644)  (144483, 265153)  (144518, 257752)  (144538, 525937)  (144456, 249611)  (144456, 250064)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
```

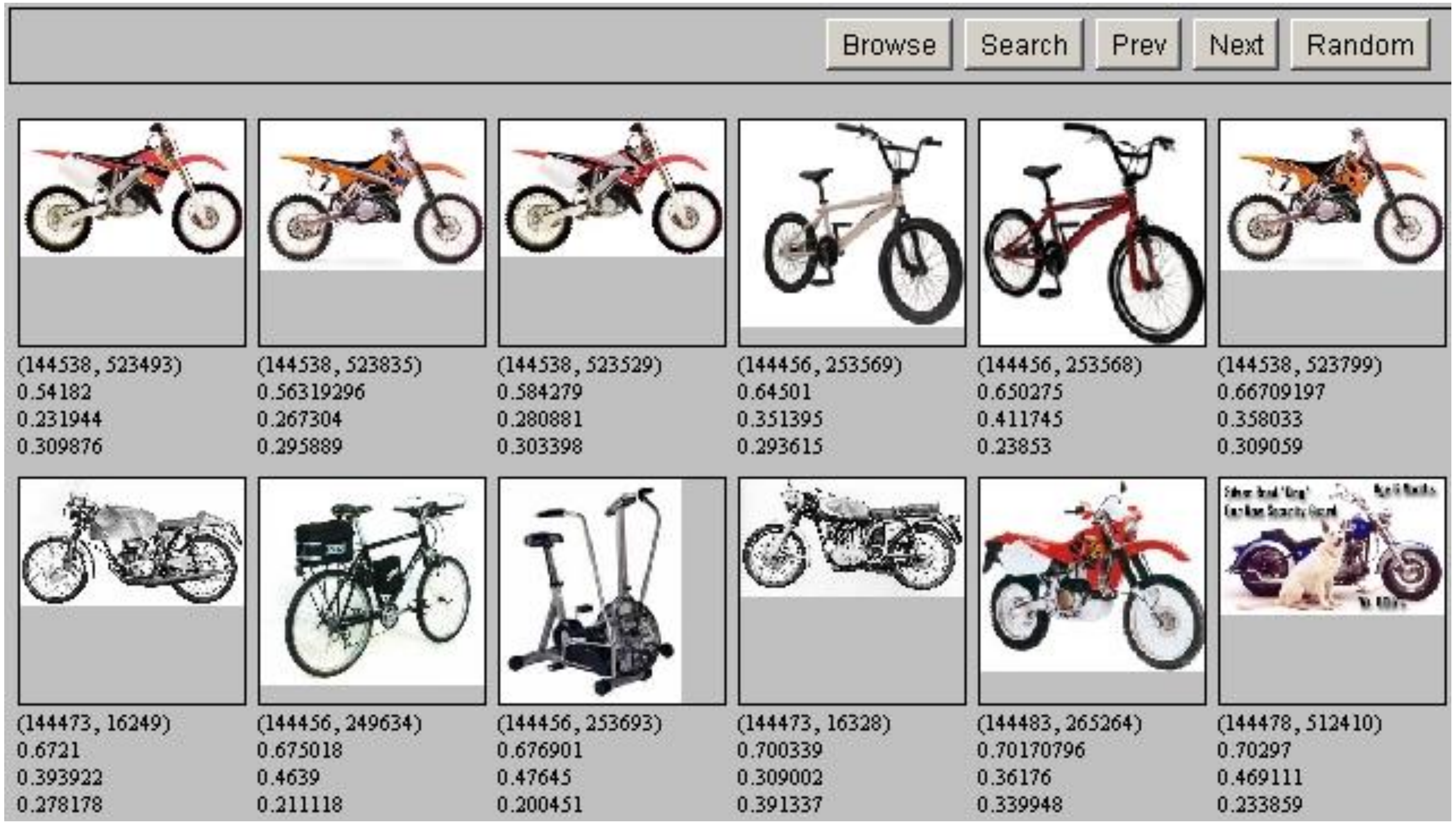
Relevance Feedback



```
(144473, 16458)   (144457, 252140)  (144456, 262857)  (144456, 262863)  (144457, 252134)  (144483, 265154)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0

(144483, 264644)  (144483, 265153)  (144518, 257752)  (144538, 525937)  (144456, 249611)  (144456, 250064)
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
0.0               0.0               0.0               0.0               0.0               0.0
```

Results after Relevance Feedback

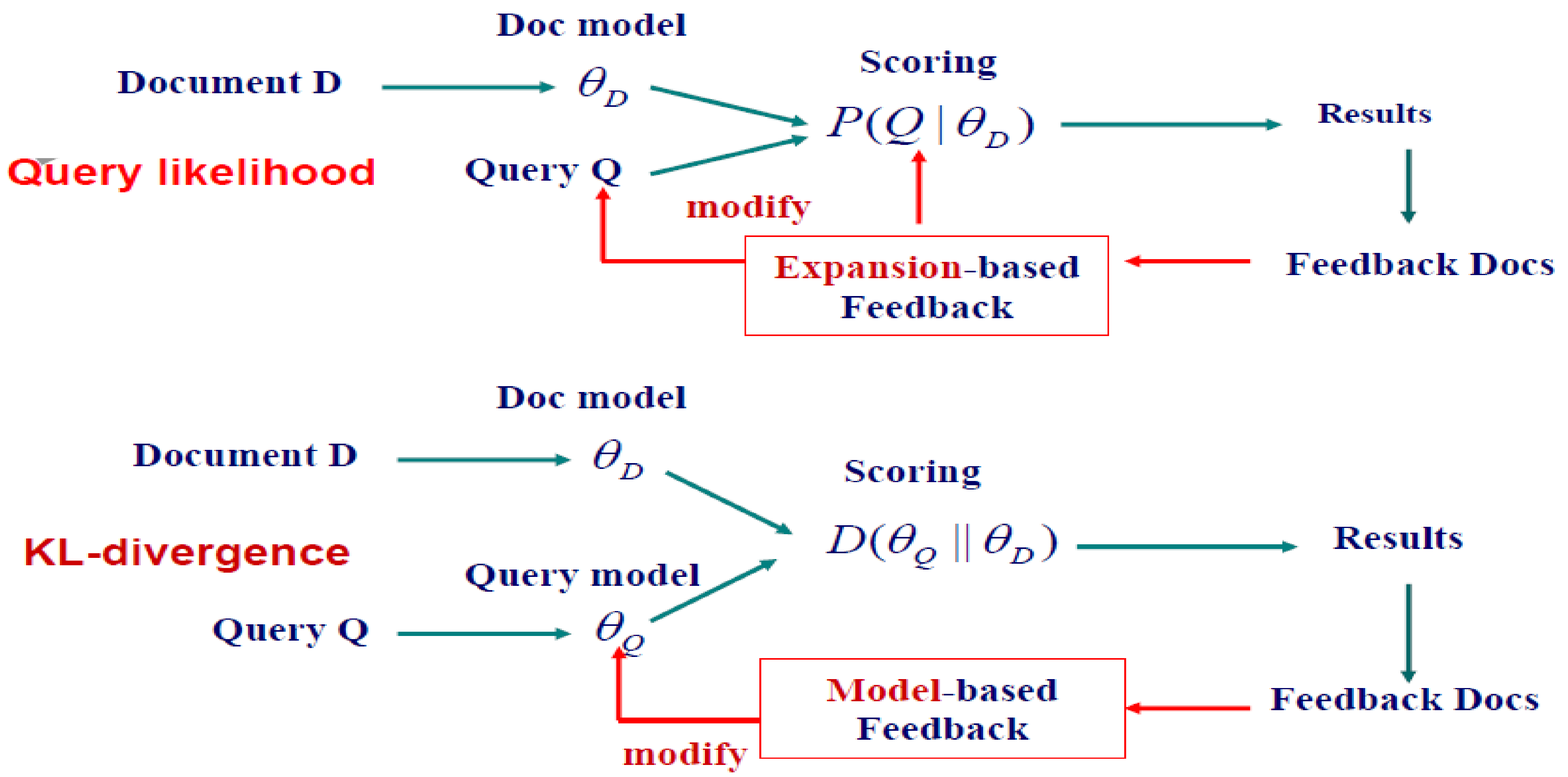


(144538, 523493)	(144538, 523835)	(144538, 523529)	(144456, 253569)	(144456, 253568)	(144538, 523799)
0.54182	0.56319296	0.584279	0.64501	0.650275	0.66709197
0.231944	0.267304	0.280881	0.351395	0.411745	0.358033
0.309876	0.295889	0.303398	0.293615	0.23853	0.309059
(144473, 16249)	(144456, 249634)	(144456, 253693)	(144473, 16328)	(144483, 265264)	(144478, 512410)
0.6721	0.675018	0.676901	0.700339	0.70170796	0.70297
0.393922	0.4639	0.47645	0.309002	0.36176	0.469111
0.278178	0.211118	0.200451	0.391337	0.339948	0.233859

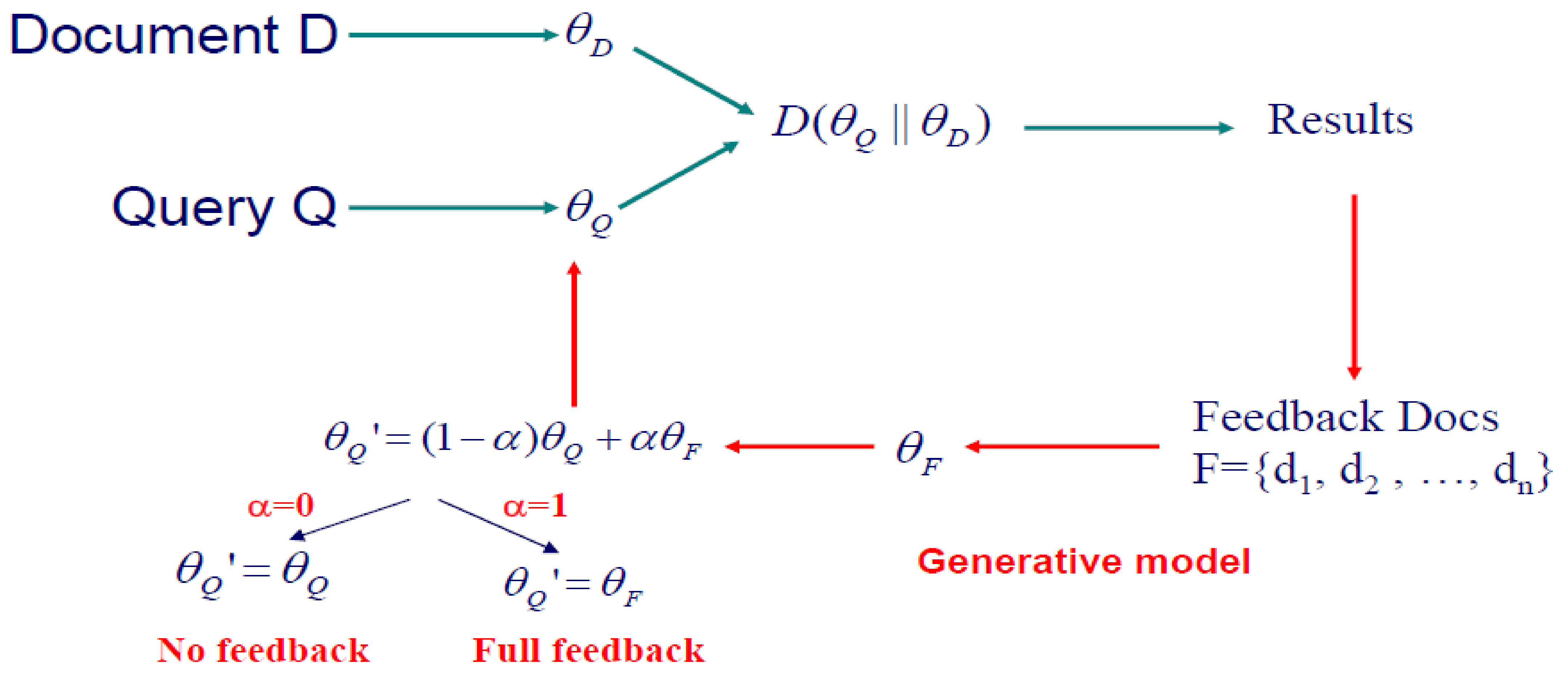
Relevance Feedback and Language Modelling

- Introduce a query model & treat feedback as query model updating
- Feedback:
  - Expansion-based =>Model-based

Expansion-based vs. Model-based Feedback



Feedback as Model Interpolation



Summary

- Three different approaches to language modelling
  - Query likelihood model
  - Document likelihood model
  - Divergence model
- Different approaches to smoothing
  - Important role in performance
- Relevance feedback and language modelling

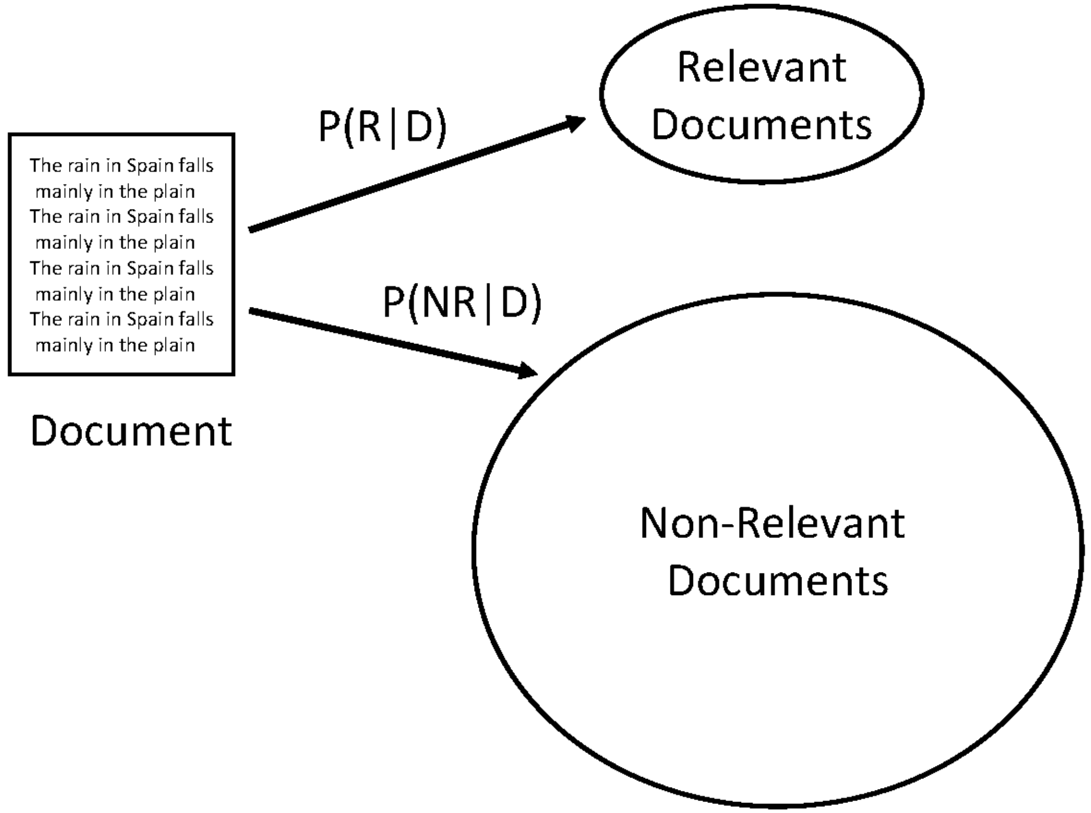
Retrieval Models Overview

- Older models
    - Boolean retrieval
    - Vector Space model
- Probabilistic Models
    - Language models
    - BM25
- Combining evidence
    - Inference networks
    - Learning to Rank

Probability Ranking Principle

- Robertson (1977)
  - “If a reference retrieval system’s response to each request is a ranking of the documents in the collection in order of decreasing probability of relevance to the user who submitted the request,
  - where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose,
  - the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data.”

IR as Classification


The rain in Spain falls mainly in the plain The rain in Spain falls mainly in the plain The rain in Spain falls mainly in the plain The rain in Spain falls mainly in the plain
Document
P(R|D)
P(NR|D)
Relevant Documents
Non-Relevant Documents

Bayes Classifier

- Bayes Decision Rule
  - A document *D* is relevant if $P(R|D) > P(NR|D)$
- Estimating probabilities
  - Use Bayes Rule

$$P(R|D) = \frac{P(D|R)P(R)}{P(D)} \qquad P(NR|D) = \frac{P(D|NR)P(NR)}{P(D)}$$

  - Classify a document as relevant if

$$\frac{P(D|R)P(R)}{P(D)} > \frac{P(D|NR)P(NR)}{P(D)}$$

    - lhs is *likelihood ratio*

$$\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$$

Estimating P(D|R)

- Assume independence

$$P(D|R) = \prod_{i=1}^{t} P(d_i|R)$$

- *Binary independence model*
  - document represented by a vector of binary features indicating term occurrence (or non-occurrence)
  - Assume:
    - p_i is probability that term i occurs (i.e., has value 1) in relevant document
    - s_i is probability of occurrence in non-relevant document

Binary Independence Model

$$\frac{P(D|R)}{P(D|NR)} = \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i}$$

$$= \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \left(\prod_{i:d_i=1} \frac{1-s_i}{1-p_i} \cdot \prod_{i:d_i=1} \frac{1-p_i}{1-s_i}\right) \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i}$$

$$= \prod_{i:d_i=1} \frac{p_i(1-s_i)}{s_i(1-p_i)} \cdot \prod_{i} \frac{1-p_i}{1-s_i}$$

Binary Independence Model

- Scoring function is

$$\sum_{i:d_i=1} \log \frac{p_i(1-s_i)}{s_i(1-p_i)}$$

- Query provides information about relevant documents
- If we assume p_i constant, s_i approximated by entire collection, get *idf*-like weight

$$\log \frac{0.5(1-\frac{n_i}{N})}{\frac{n_i}{N}(1-0.5)} = \log \frac{N-n_i}{n_i}$$

Contingency Table

	Relevant	Non-relevant	Total
$d_i = 1$	r_i	$n_i - r_i$	n_i
$d_i = 0$	$R - r_i$	$N - n_i - R + r_i$	$N - n_i$
Total	R	$N - R$	N

$$p_i = (r_i + 0.5)/(R + 1)$$

$$s_i = (n_i - r_i + 0.5)/(N - R + 1)$$

Gives scoring function:

$$\sum_{i:d_i=q_i=1} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)}$$

BM25

- Popular and effective ranking algorithm based on binary independence model
  - adds document and query term weights

$$\sum_{i \in Q} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)} \cdot \frac{(k_1+1)f_i}{K+f_i} \cdot \frac{(k_2+1)qf_i}{k_2+qf_i}$$

  - *k_1, k_2* and *K* are parameters whose values are set empirically

$$K = k_1((1-b) + b \cdot \frac{dl}{avdl})$$

  - *dl* is document length
  - *avdl is* average document length
  - Typical value for *k_1* is 1.2, *k_2* varies from 0 to 1000, b = 0.75

BM25 Example

- Query with two terms, “president lincoln”, (*qf = 1)*
- No relevance information (*r and R are* zero)
- *N* = 500,000 documents
- *“president”* occurs in 40,000 documents (n_1 = 40, 000)
- *“lincoln”* occurs in 300 documents (n_2 = 300)
- “president” occurs 15 times in doc (f_1 = 15)
- *“lincoln”* occurs 25 times (f_2 = 25)
- document length is 90% of the average length (*dl/avdl* = .9)
- k_1 = 1.2, *b* = 0.75, and k_2 = 100
- $K = 1.2 \cdot (0.25 + 0.75 \cdot 0.9) = 1.11$

BM25 Example

BM25 (*Q*,*D*) = $\sum_{i \in Q} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)} \cdot \frac{(k_1+1)f_i}{K+f_i} \cdot \frac{(k_2+1)qf_i}{k_2+qf_i}$

$$\begin{aligned}
BM25(Q,D) \quad = & \\
& \log \frac{(0+0.5)/(0-0+0.5)}{(40000-0+0.5)/(500000-40000-0+0+0.5)} \\
& \times \frac{(1.2+1)15}{1.11+15} \times \frac{(100+1)1}{100+1} \\
& + \log \frac{(0+0.5)/(0-0+0.5)}{(300-0+0.5)/(500000-300-0+0+0.5)} \\
& \times \frac{(1.2+1)25}{1.11+25} \times \frac{(100+1)1}{100+1}
\end{aligned}$$

$$\begin{aligned}
= \quad & \log 460000.5/40000.5 \cdot 33/16.11 \cdot 101/101 \\
& + \log 499700.5/300.5 \cdot 55/26.11 \cdot 101/101 \\
= \quad & 2.44 \cdot 2.05 \cdot 1 + 7.42 \cdot 2.11 \cdot 1 \\
= \quad & 5.00 + 15.66 = 20.66
\end{aligned}$$

BM25 Example

- Effect of term frequencies

Frequency of "president"	Frequency of "lincoln"	BM25 score
15	25	20.66
15	1	12.74
15	0	5.00
1	25	18.2
0	25	15.66

Summary

- Probabilistic models for information retrieval
  - Language models
  - BM25

- More advanced models available
  - Learning to rank
  - BM25/language model based features are the strongest ones

Term embeddings for IR

Retrieval using vector representations

Generate vector representation of query

Generate vector representation of document

Estimate relevance from q-d vectors

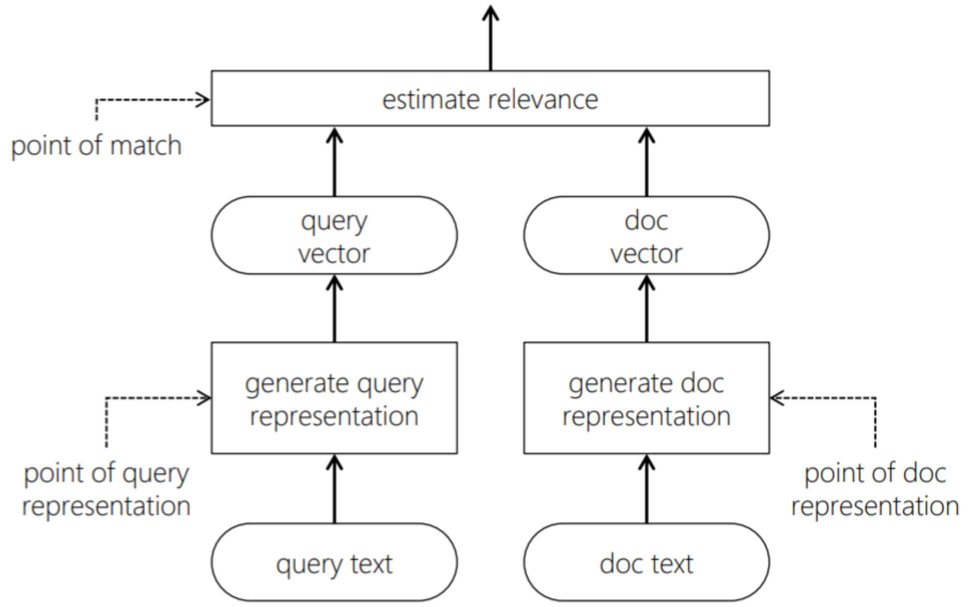
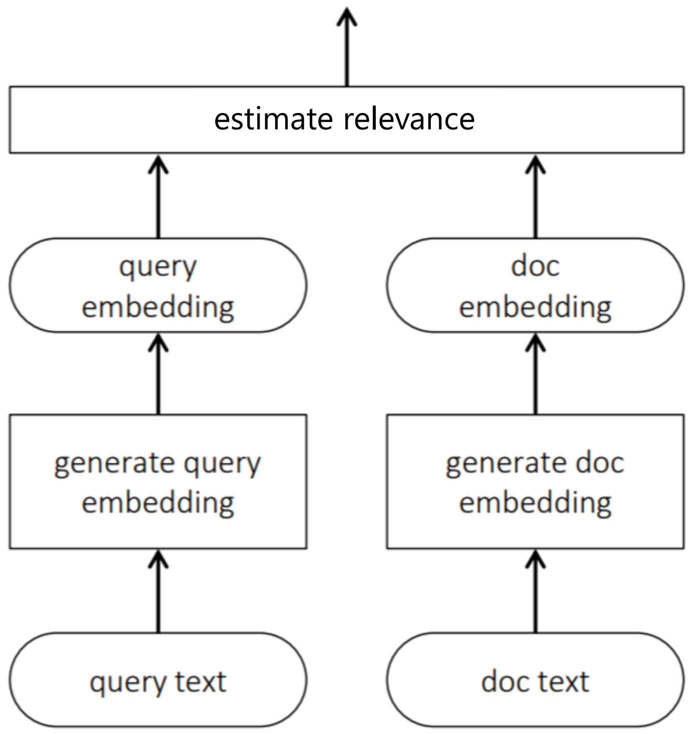
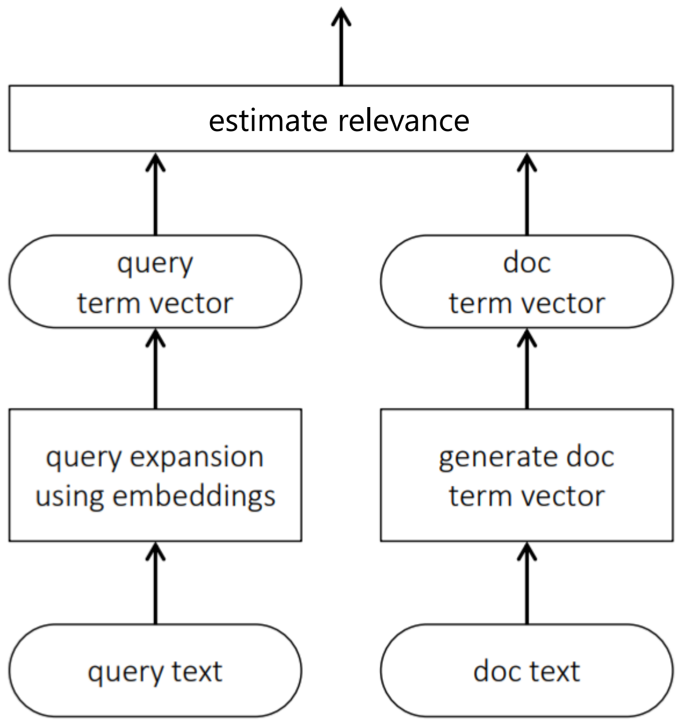

estimate relevance
point of match
query vector
doc vector
generate query representation
generate doc representation
point of query representation
point of doc representation
query text
doc text

Figure 2.3: Document ranking typically involves a query and a document representation steps, followed by a matching stage. Neural models can be useful either for generating good representations or in estimating relevance, or both.

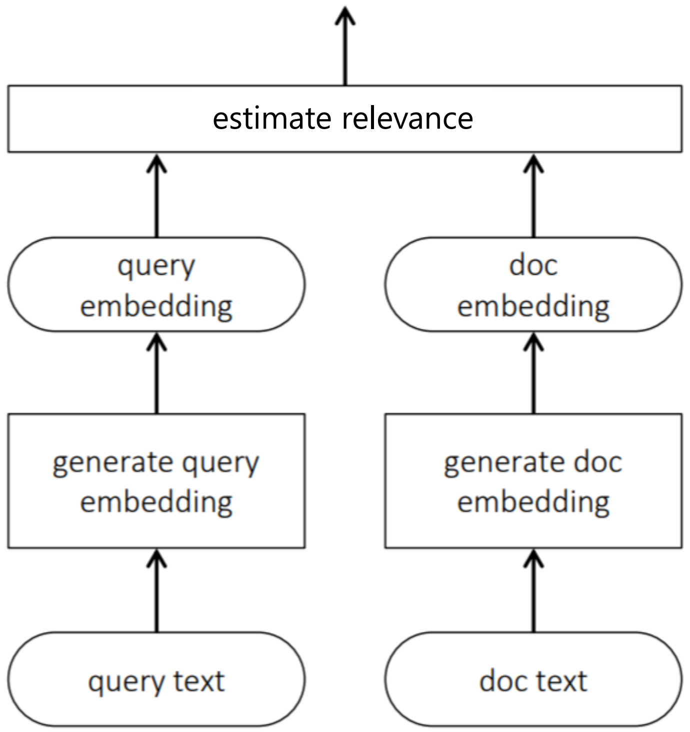
Popular approaches to incorporating term embeddings for matching



Compare query and document directly in the embedding space



Use embeddings to generate suitable query expansions



Compare query and document directly in the embedding space

E.g.,

Generalized Language Model [Ganguly et al., 2015]

Neural Translation Language Model [Zuccon et al., 2015]

Average term embeddings [Le and Mikolov, 2014, Nalisnick et al., 2016, Zamani and Croft, 2016, and others]

Word mover's distance

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. Word embedding based generalized language model for information retrieval. In SIGIR, 2015.
Guido Zuccon, Bevan Koopman, Peter Bruza, and Leif Azzopardi. Integrating and evaluating neural word embeddings in information retrieval. In ADCS, 2015.
Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In ICML, 2014.
Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In WWW, 2016.
Hamed Zamani and W Bruce Croft. Estimating embedding vectors for queries. In ICTIR, 2016.

Average Term Embeddings

Q-D relevance estimated by computing cosine similarity between centroid of q and d term embeddings

$$sim(q,d) = cos(\vec{v}_q, \vec{v}_d) = \frac{\vec{v}_q^\intercal \vec{v}_d}{\|\vec{v}_q\| \|\vec{v}_d\|}$$

$$\text{where,} \quad \vec{v}_q = \frac{1}{|q|} \sum_{t_q \in q} \frac{\vec{v}_{t_q}}{\|\vec{v}_{t_q}\|}$$

$$\vec{v}_d = \frac{1}{|d|} \sum_{t_d \in d} \frac{\vec{v}_{t_d}}{\|\vec{v}_{t_d}\|}$$

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In WWW, 2016.
Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

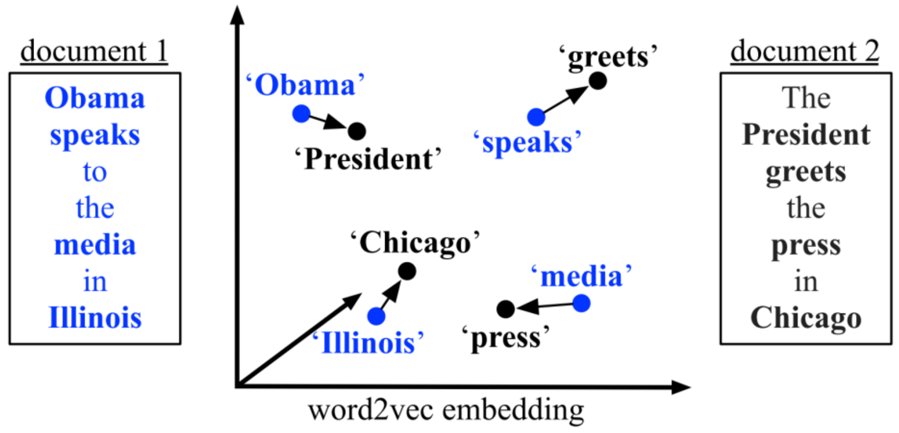
Word mover's distance

Based on the Earth Mover's Distance (EMD) [Rubner et al., 1998]

Originally proposed by Wan et al. [2005, 2007], but used WordNet and topic categories

Kusner et al. [2015] incorporated term embeddings

Adapted for q-d matching by Guo et al. [2016]



Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In CV, 1998.
Xiaojun Wan and Yuxin Peng. The earth mover's distance as a semantic measure for document similarity. In CIKM, 2005.
Xiaojun Wan. A novel document similarity measure based on earth mover's distance. Information Sciences, 2007.
Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In ICML, 2015.
Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. Semantic matching by non-linear word transportation for information retrieval. In CIKM, 2016.

A tale of two queries

“pekarovic land company”

Hard to learn good representation for the rare term *pekarovic*

But easy to estimate relevance based on count of exact term matches of *pekarovic* in the document

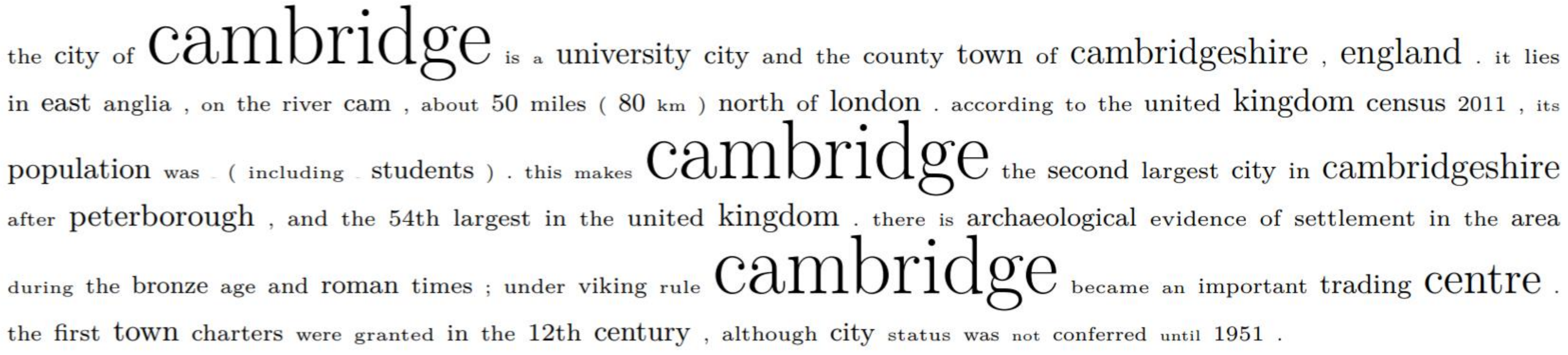
“what channel are the seahawks on today”

Target document likely contains *ESPN* or *sky sports* instead of *channel*

The terms *ESPN* and *channel* can be compared in a term embedding space

Matching in the term space is necessary to handle rare terms. Matching in the latent embedding space can provide additional evidence of relevance. Best performance is often achieved by combining matching in both vector spaces.

Query: Cambridge (Font size is a function of term-term cosine similarity)

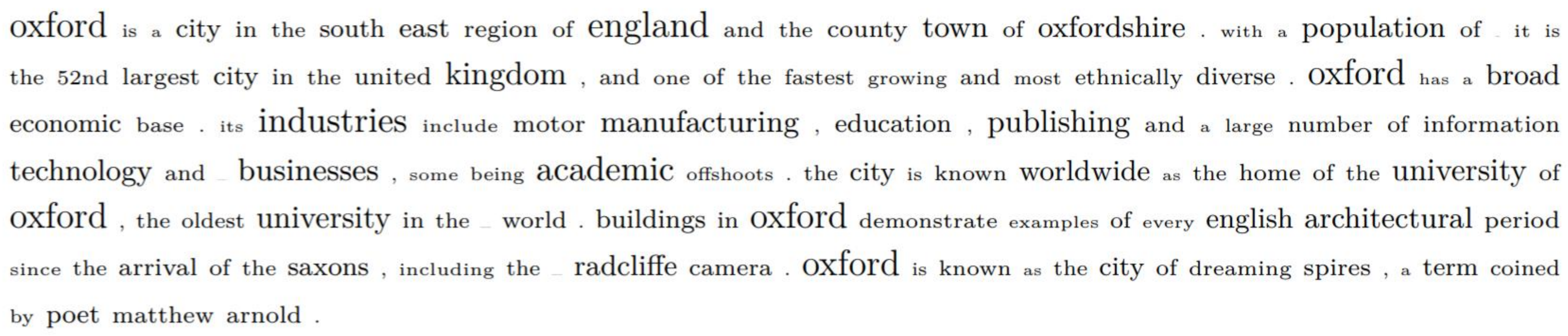


the city of cambridge is a university city and the county town of cambridgeshire , england . it lies in east anglia , on the river cam , about 50 miles (80 km) north of london . according to the united kingdom census 2011 , its population was (including students) . this makes cambridge the second largest city in cambridgeshire after peterborough , and the 54th largest in the united kingdom . there is archaeological evidence of settlement in the area during the bronze age and roman times ; under viking rule cambridge became an important trading centre . the first town charters were granted in the 12th century , although city status was not conferred until 1951 .

(a) Passage about the city of Cambridge

Besides the term "Cambridge", other related terms (e.g., "university", "town", "population", and "England") contribute to the relevance of the passage

Query: Cambridge (Font size is a function of term-term cosine similarity)



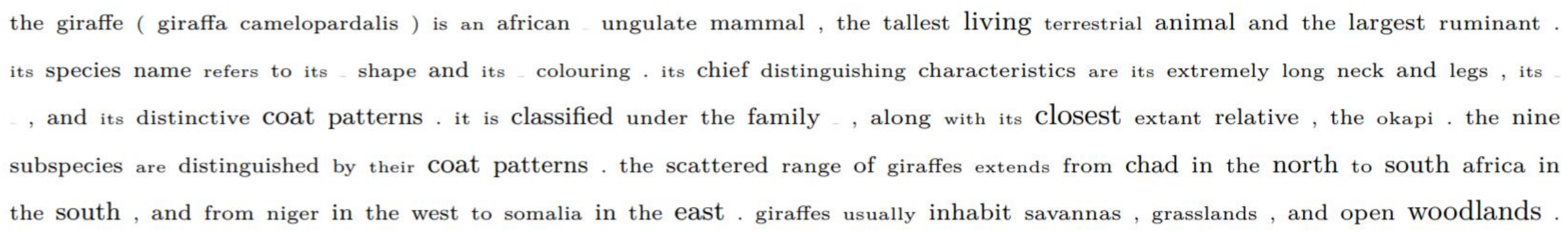
oxford is a city in the south east region of england and the county town of oxfordshire . with a population of it is the 52nd largest city in the united kingdom , and one of the fastest growing and most ethnically diverse . oxford has a broad economic base . its industries include motor manufacturing , education , publishing and a large number of information technology and businesses , some being academic offshoots . the city is known worldwide as the home of the university of oxford , the oldest university in the world . buildings in oxford demonstrate examples of every english architectural period since the arrival of the saxons , including the radcliffe camera . oxford is known as the city of dreaming spires , a term coined by poet matthew arnold .

(b) Passage about the city of Oxford

However, the same terms may also make a passage about Oxford look somewhat relevant to the query “Cambridge”

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

Query: Cambridge (Font size is a function of term-term cosine similarity)

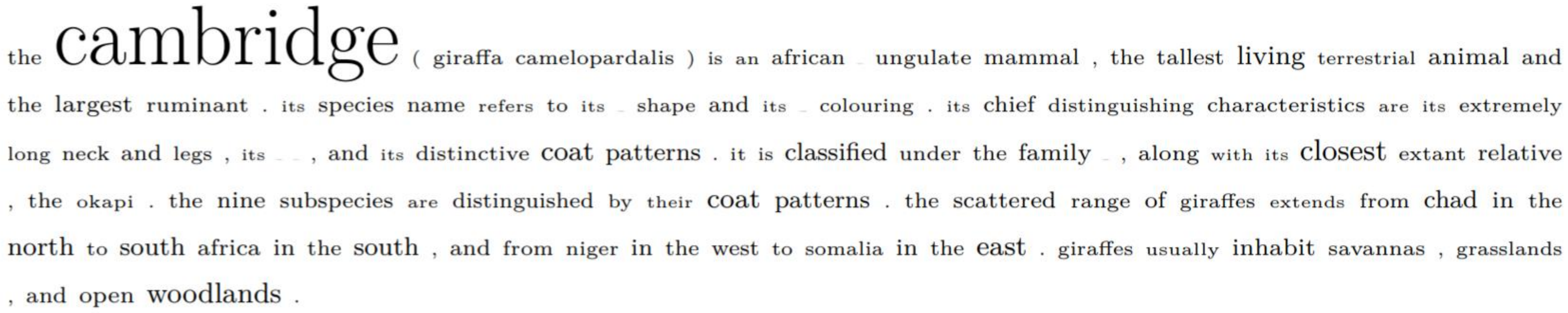

the giraffe (giraffa camelopardalis) is an african _ ungulate mammal , the tallest living terrestrial animal and the largest ruminant . its species name refers to its _ shape and its _ colouring . its chief distinguishing characteristics are its extremely long neck and legs , its _ _ , and its distinctive coat patterns . it is classified under the family _ , along with its closest extant relative , the okapi . the nine subspecies are distinguished by their coat patterns . the scattered range of giraffes extends from chad in the north to south africa in the south , and from niger in the west to somalia in the east . giraffes usually inhabit savannas , grasslands , and open woodlands .

(c) Passage about giraffes

A passage about giraffes, however, obviously looks non-relevant in the embedding space…

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.

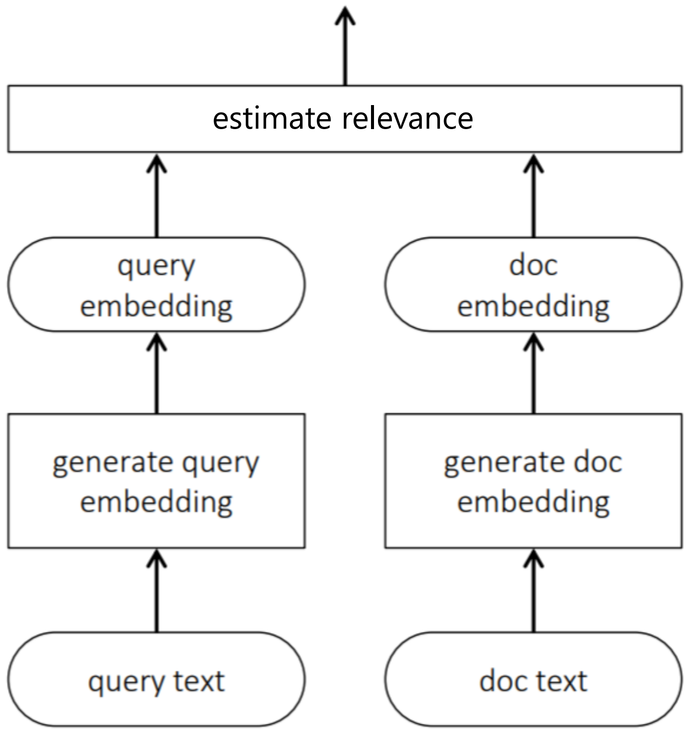
Query: Cambridge (Font size is a function of term-term cosine similarity)


the cambridge (giraffa camelopardalis) is an african ungulate mammal , the tallest living terrestrial animal and the largest ruminant . its species name refers to its shape and its colouring . its chief distinguishing characteristics are its extremely long neck and legs , its , and its distinctive coat patterns . it is classified under the family , along with its closest extant relative , the okapi . the nine subspecies are distinguished by their coat patterns . the scattered range of giraffes extends from chad in the north to south africa in the south , and from niger in the west to somalia in the east . giraffes usually inhabit savannas , grasslands , and open woodlands .

(d) Passage about giraffes, but ’giraffe’ is replaced by ’Cambridge’

But the embedding based matching model is more robust to the same passage when “giraffe” is replaced by “Cambridge”—a trick that would fool exact term based IR models. In a sense, the embedding based model ranks this passage low because Cambridge is not "an African even-toed ungulate mammal“.

Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. A dual embedding space model for document ranking. arXiv preprint arXiv:1602.01137, 2016.



Compare query and document directly in the embedding space

E.g.,

Generalized Language Model [Ganguly et al., 2015]

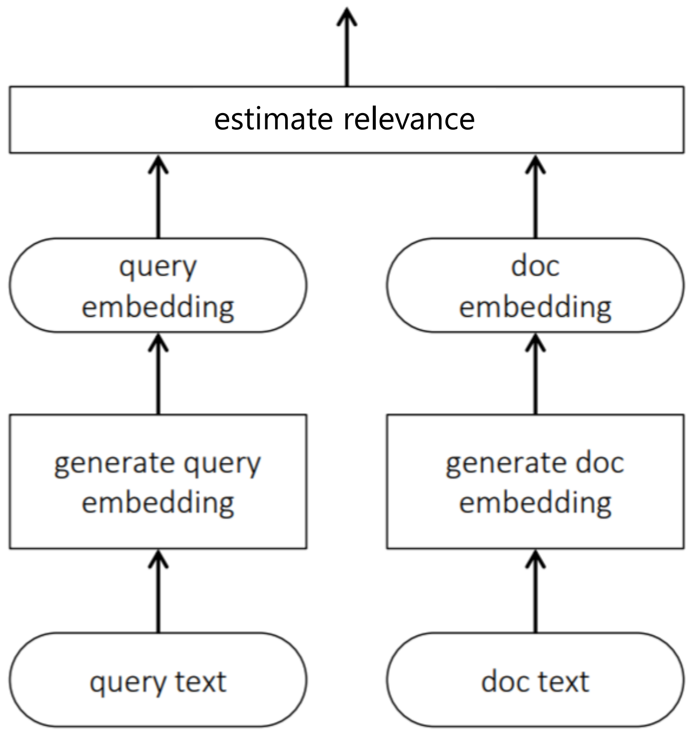
Neural Translation Language Model [Zuccon et al., 2015]

Average term embeddings [Le and Mikolov, 2014, Nalisnick et al., 2016, Zamani and Croft, 2016, and others]

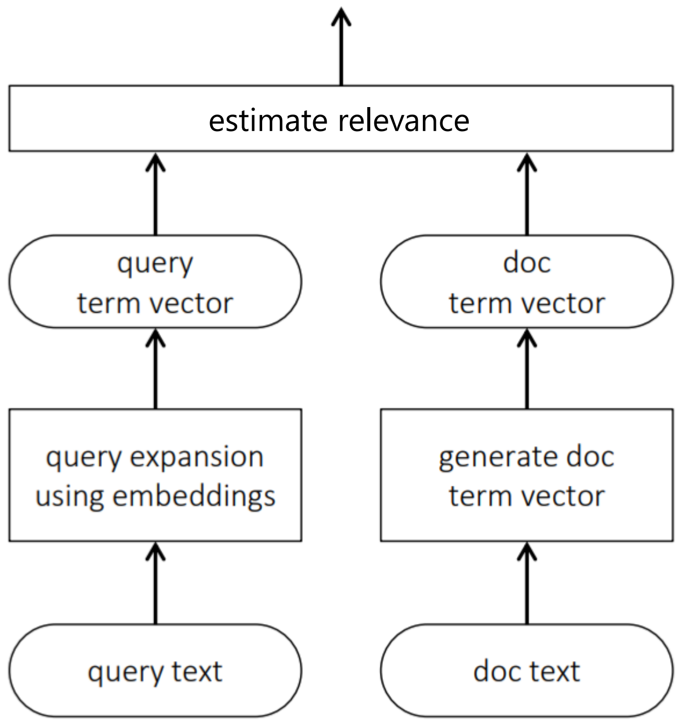
Word mover's distance

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. Word embedding based generalized language model for information retrieval. In SIGIR, 2015.
Guido Zuccon, Bevan Koopman, Peter Bruza, and Leif Azzopardi. Integrating and evaluating neural word embeddings in information retrieval. In ADCS, 2015.
Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In ICML, 2014.
Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In WWW, 2016.
Hamed Zamani and W Bruce Croft. Estimating embedding vectors for queries. In ICTIR, 2016.

Popular approaches to incorporating term embeddings for matching



Compare query and document directly in the embedding space



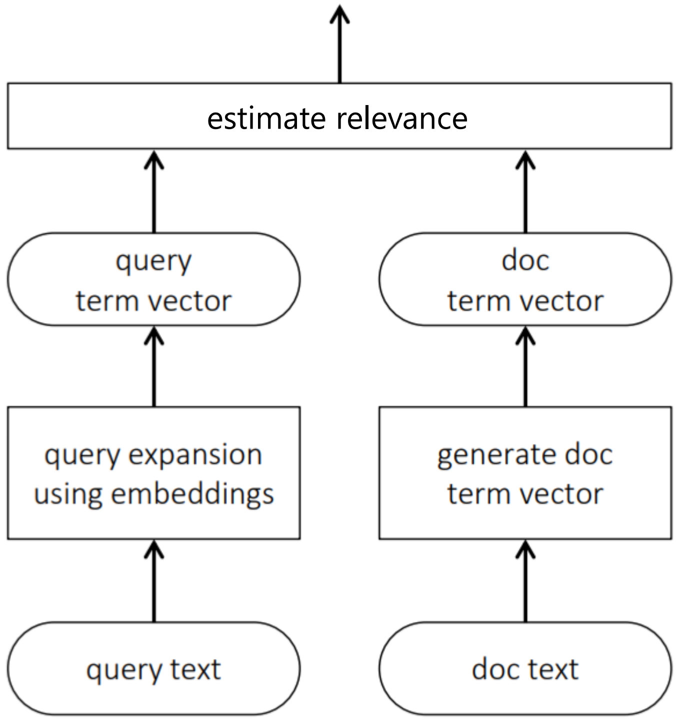
Use embeddings to generate suitable query expansions

Query expansion using term embeddings

Find good expansion terms based on nearness in the embedding space

$$score(t_c, q) = \frac{1}{|q|} \sum_{t_q \in q} cos(\vec{v}_{t_c}, \vec{v}_{t_q})$$

Better retrieval performance when combined with pseudo-relevance feedback (PRF) [Zamani and Croft, 2016] and if we learn query specific term embeddings [Diaz et al., 2016]



Use embeddings to generate suitable query expansions

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. Query expansion with locally-trained word embeddings. In ACL, 2016.
Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. Using word embeddings for automatic query expansion. arXiv preprint arXiv:1606.07608, 2016.
Hamed Zamani and W Bruce Croft. Embedding-based query language models. In ICTIR, 2016.

Summary

- Probabilistic retrieval models
  - Language modelling
  - BM25
- Term embeddings for IR
  - Embedding based retrieval
  - Query expansion using embeddings