IS2140 Information Storage and Retrieval

Unit 5: Statistical Language Model

Daqing He
School of Computing and Information
University of Pittsburgh

October 1, 2018

Muddiest Points

- Query processing
  - How does a wildcard match work in terms of document frequency? I failed to see any clue in that.
- Boolean model
  - "How should I implement a boolean query with only one search box? I mean when I type in something like 'You and Me', it may be a book's name, 'and' in this phrase may be also represented the logistic connection. How should I differentiate them?"
  - "The processing order method could also apply to three and four words, or there would be a better way for longer terms?"

Muddiest Points

- Term Weighting
  - "While calculating tf-idf weighting why do we take 'log to the base 10', why can't we just take 'log to the base e' i.e natural log. How do different bases affect the tf-idf weights? ”
  - "The point is about tf-idf. Since there are only the term-document relationship used in tf-idf, does it mean we will ignore or lose the information about the position or other additional information when users do query? "
  - "Among those weighting schemes mentioned in class: tf-idf weighting, SMART and Lucene sytems, what are the standards to generate different schemes and how to decide which one to use_”
  - Is there any circumstance that we use TF-IDF instead of cosine similarity?

Muddiest Points

- Term Weighting
  - "Since we can calculate the document frequency(df), why do we need to process it into inverse document frequency with the formula(p41)? Can we just simply reverse the document frequency? Furthermore, in the idf formula, it supposed N=1000,000, can the N be arbitrary const number? "

Muddiest Points

- Vector Space Model
  - "PPT p52: When calculating similarity, why just compute the angle between two vectors but not length difference or weighting of each dimention?"
  - "On the lsat lecture, why did you mention that query do not need to normalize in vector based model?"
  - "The tf-idf treats the terms of queries as equally important. However, the users may what to give some weights to the terms. Like if you type the ""Shrimp Ramen"" in the Yelp search box, it gives you out some Ramen restaurants rather than the sea food restaurants. How does the search engine give your terms in the query different weights so that to find what you need?"

Muddiest Points

- Vector Space model
  - How can we compute cosine ranking if vectors have different dimensions?
  - How to rank two high similarity scores when they are derived from two different key words in a query?
  - "The query terms are always very small compare with the index. The query terms may only contain one or two words. Therefore, the cosine value may be same for many documents , how to rank these documents when such collision occurring ?"
  - "About the vector space mode, if a system that indexes documents based on vector space model and the user has a query that contains two parts: a phrase and a simple word. When the system find one document that contains the phrase and another that contains the single word. Which one should be ranked higher?"

Muddiest Points

- Vector Space Model
  - "When using cosine of the angle to measure similarity between the document and the query, on some level we are ignoring the length of the document. Is it a little conflicting with tf-idf idea that short documents are usually more relevant?"

Agenda

- Efficiency in Vector Space Model
- Probability basics
- Statistical Language Models

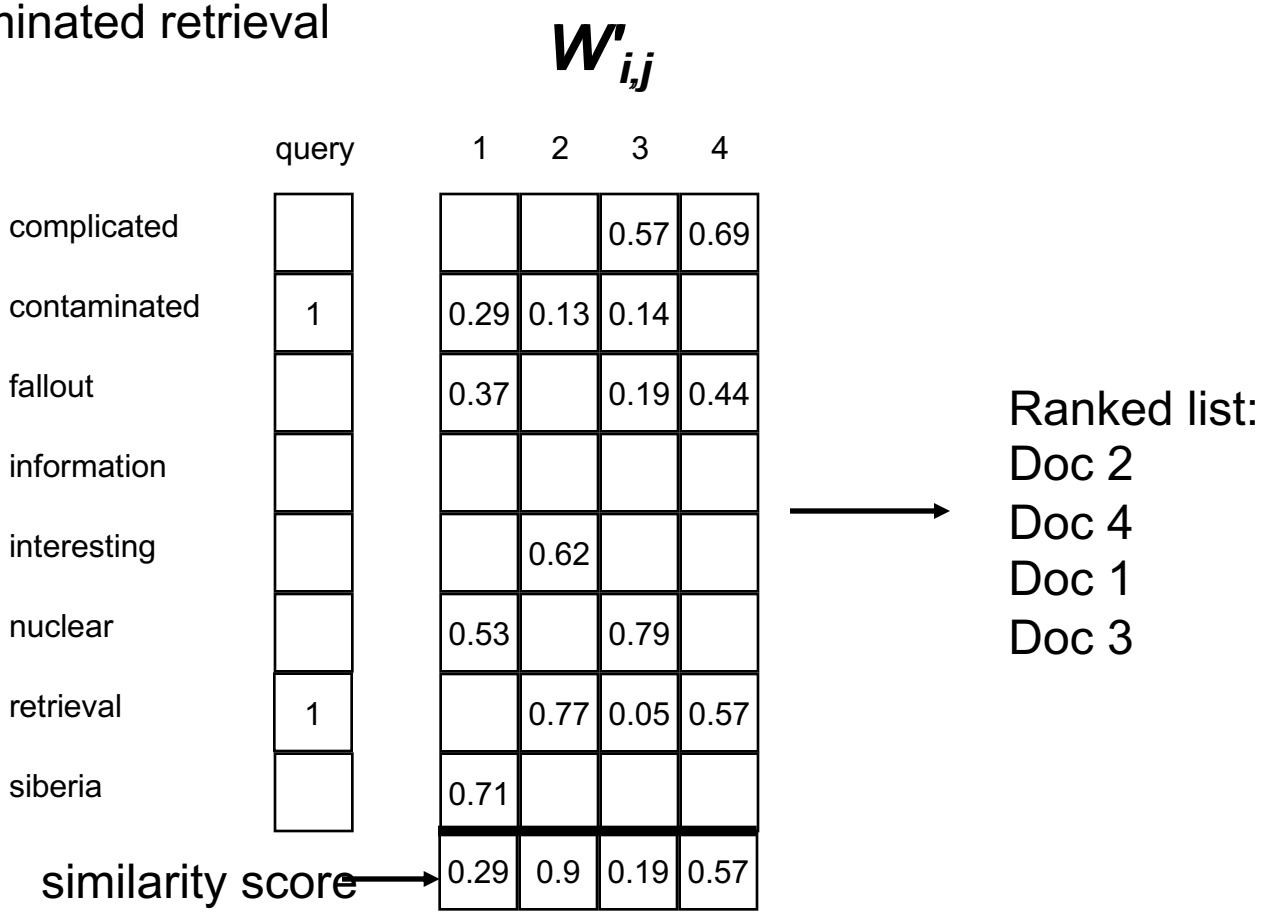
Vector Space Model - Efficiency

Efficient Cosine Ranking - I

- Find the K docs in the collection "nearest" to the query ⇒ K largest query-doc cosines.
- Special Case: unweighted queries
  - Opportunities: No weighting on query terms
  - Solution: Assume each query term occurs only once
  - Solution: Then for ranking, don't need to normalize query vector
    - Slight simplification of standard cosine algorithm

Retrieval Examples

Query: contaminated retrieval



Faster Cosine: Unweighted Query

```
FastCosineScore(q)
 1   float Scores[N] = 0
 2   for each d
 3   do Initialize Length[d] to the length of doc d
 4   for each query term t
 5   do calculate w_{t,q} and fetch postings list for t
 6       for each pair(d, tf_{t,d}) in postings list
 7       do add wf_{t,d} to Scores[d]
 8   Read the array Length[d]
 9   for each d
10   do Divide Scores[d] by Length[d]
11   return Top K components of Scores[]
```

Figure 7.1 A faster algorithm for vector space scores.

Efficient Cosine Ranking - II

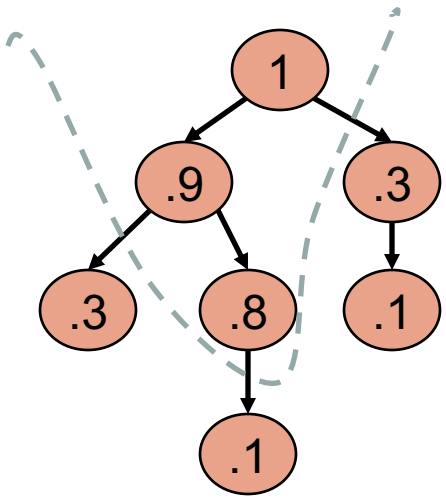
- Efficient ranking:
  - Computing a single cosine efficiently.
  - Choosing the *K* largest cosine values efficiently.
    - Key question: Can we do this without comparing all *N* cosines?

Comparing the *K* largest cosines: selection vs. sorting

- Typically we want to retrieve the top *K* docs (in the cosine ranking for the query)
  - not to totally order all docs in the collection
- Can we pick off docs with *K* highest cosines?
- Idea 1: Let *J* = number of docs with nonzero cosines
  - We seek the *K* best of these *J*

Use heap for selecting top *K*

- Binary tree in which each node's value > the values of children
- Takes *2J* operations to construct, then read *K* "winners" in 2log *J* steps.
- For *J*=1M, *K*=100, this is about 10% of the cost of sorting.



Pruning Postings

- Find a set A of *contenders*, with $K < |A| << N$
  - A does not necessarily contain the top K, but has many docs from among the top K
  - Return the top K docs in A
- Think of A as pruning non-contenders
- What could be methods for identifying A?
  - Idea 1: Index Elimination: Only consider high-idf query terms
  - Idea 2: Index Elimination: Only consider docs containing many query terms
  - Idea 3: Champion List
  - Many other ideas, say IIR chapter 7

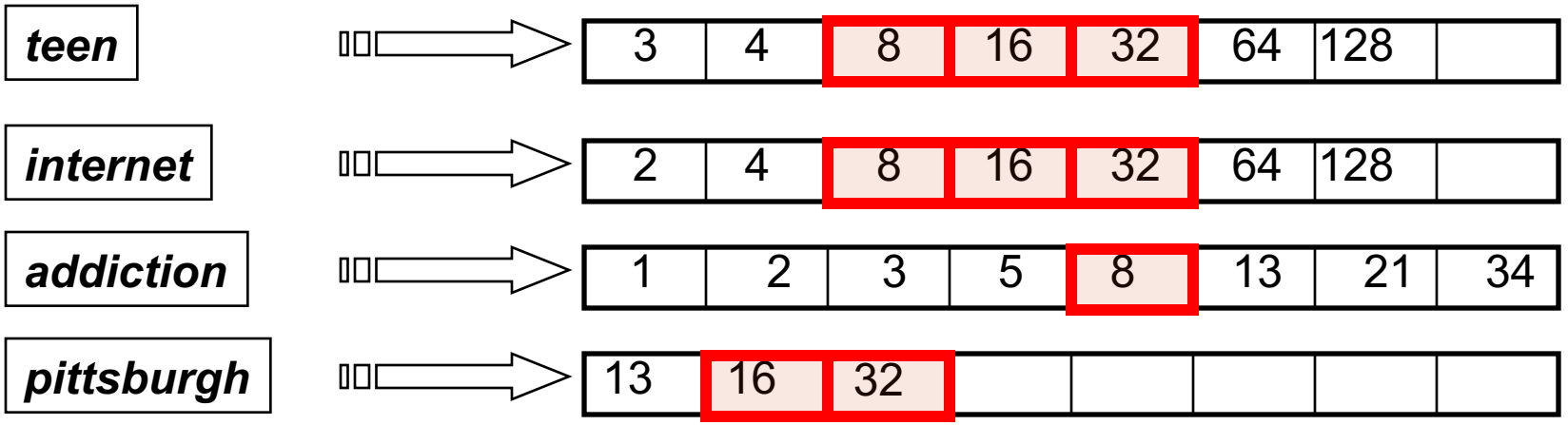
Idea 1 High-idf query terms only

- For a query such as *novel the catcher in the rye*
- Only accumulate scores from *catcher* and *rye*
- Intuition: *novel, in* and *the* contribute little to the scores and don't alter rank-ordering much
- Benefit:
  - Postings of low-idf terms have many docs → these (many) docs get eliminated from *A*

Idea 2: Docs with Many Query Terms

- Any doc with at least one query term is a candidate for the top *K* output list
- For multi-term queries, only compute scores for docs containing several of the query terms
  - Say, at least 3 out of 4
  - Imposes a "soft conjunction" on queries seen on web search engines (early Google)
- Easy to implement in postings traversal

Example: 3 of 4 query terms

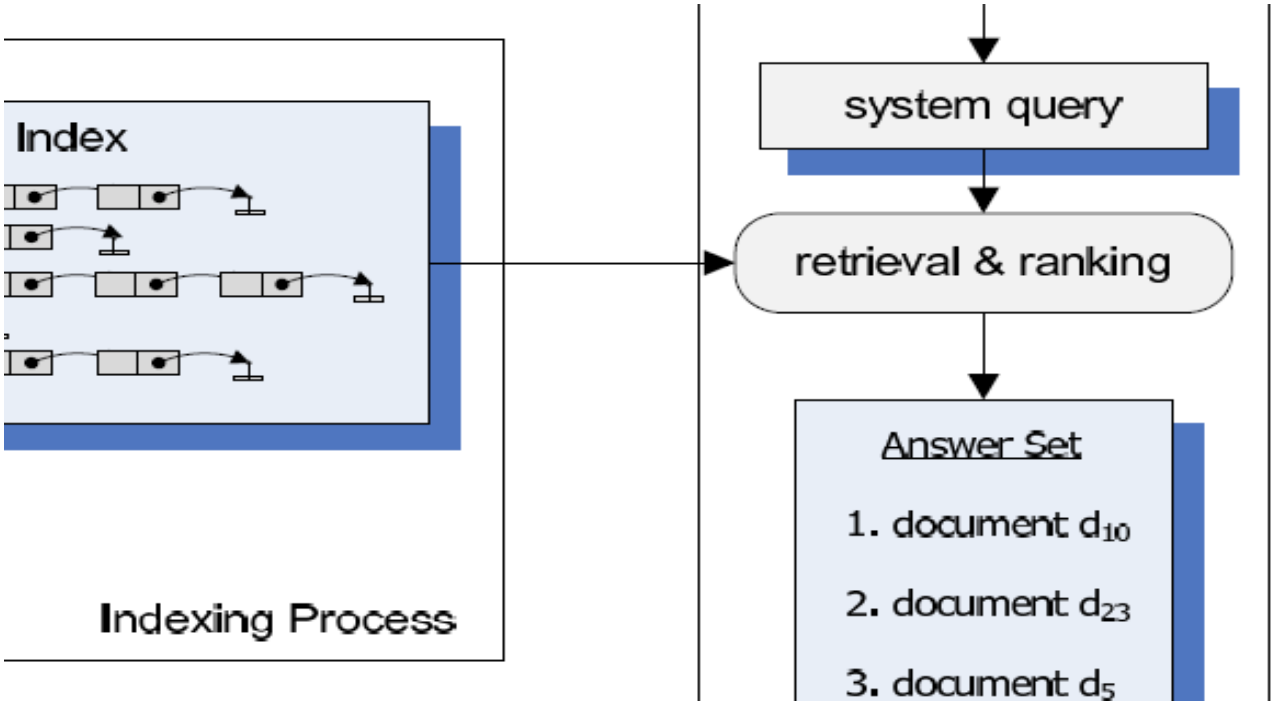


Scores only computed for 8, 16 and 32.

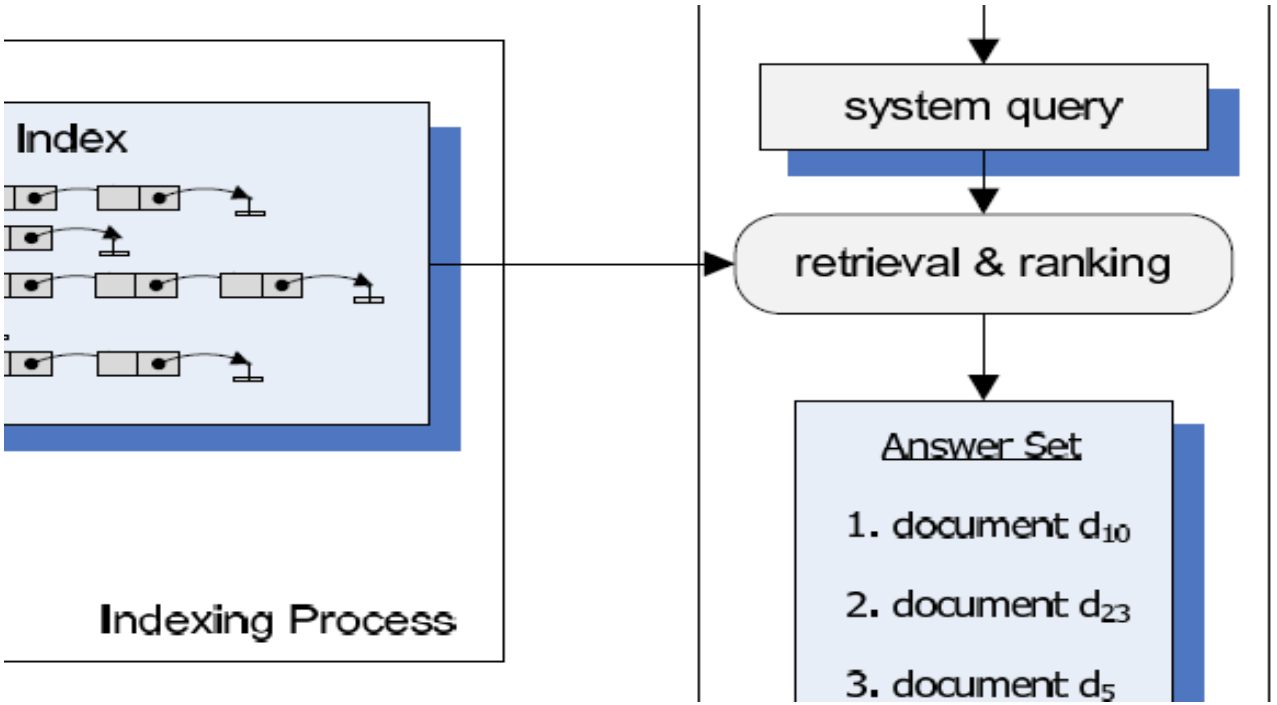
Idea 2.3: Champion lists

- Precompute for each term *t in vocabuarly* , the *r* docs of highest weight in *t*'s postings
  - Call this the champion list for *t*
  - (aka fancy list or top docs for *t*)
- Note that *r* has to be chosen at index time
- At query time, only compute scores for docs in the champion list of some query term
  - Pick the *K* top-scoring docs from amongst these

Retrieval Model in IR



Why Probabilities in IR?



Understanding of user need is uncertain

How to match?

Uncertain guess of whether document has relevant content

In traditional IR systems, matching between each document and query is attempted in a semantically imprecise space of index terms.

Probabilities provide a principled foundation for uncertain reasoning.
Can we use probabilities to quantify our uncertainties?

The Probability Ranking Principle

"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."

- [1960s/1970s] S. Robertson, W.S. Cooper, M.E. Maron; van Rijsbergen (1979:113); Manning & Schütze (1999:538)

The document ranking problem

- We have a collection of documents
- User issues a query
- A list of documents needs to be returned
- **Ranking method is core of an IR system:**
  - **In what order do we present documents to the user?**
  - We want the "best" document to be first, second best second, etc….
- **Idea: Rank by probability of relevance of the document w.r.t. information need**
  - P(relevant | document$_i$, query)

Probabilistic IR topics

- Classical probabilistic retrieval model
  - Probability ranking principle, etc.
- (Naïve) Bayesian Text Categorization
- Bayesian networks for text retrieval
- Language model approach to IR
  - An important emphasis in recent work

- *Probabilistic methods are one of the oldest but also one of the hottest topics in IR.*
  - *Traditionally: neat ideas, but they've never won on performance. It may be different now.*

Probability Basics

Discrete Random Variable

- A is a discrete random variable if:
  - A describes an event with a finite number of possible outcomes
    - So A is discrete
  - A describes an event whose outcome has some degree of uncertainty
    - So A is random
- Examples
  - A: it will snow in the coming Saturday
  - A: a startup company will go public in five years
  - A: the number of heads in a coin tossed three times
  - A: a student will study at Pittsburgh after being accepted

Probability and Probability Distribution

- Probability
  - The likeliness that an outcome happens
  - All probability is between 0 and 1 (inclusive)
  - P(head) = 1/2
  - P(dice take value 6) = 1/6
- Probability distribution of a discrete random variable A
  - The probability of each outcome of a discrete random variable A
  - Often can be obtained through the number of sampling $n \to \infty$
  - Sum of all probabilities for a variable A have to be 1

Examples

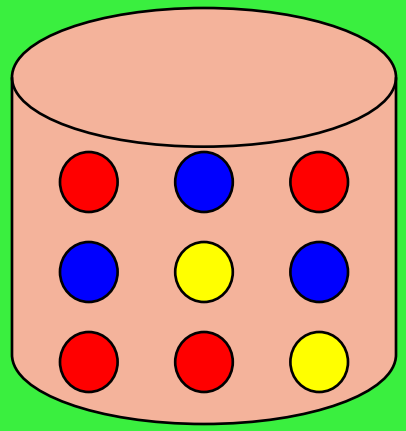
- P(RED) = probability of reaching to this bag and pulling out a RED ball
- P(BLUE) = probability of reaching to this bag and pulling out a BLUE ball
- P(YELLOW) = probability of reaching to this bag and pulling out a YELLOW ball

$$0 \le P(\text{RED}) \le 1$$

$$0 \le P(\text{BLUE}) \le 1$$

$$0 \le P(\text{YELLOW}) \le 1$$

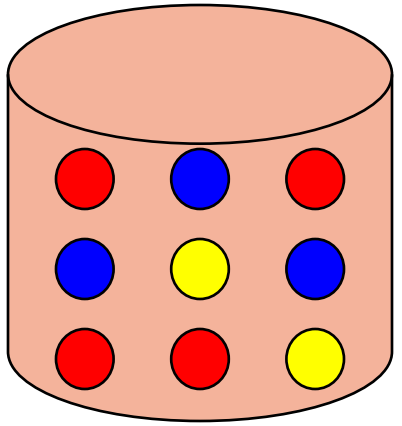
$$P(\text{RED}) + P(\text{BLUE}) + P(\text{YELLOW}) = 1$$



Probability Distribution Estimation

- Estimate the probabilities based on what we know

P(RED) = ?
P(BLUE) = ?
P(YELLOW) = ?

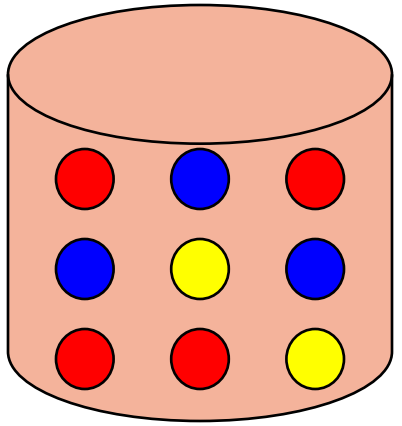


Probability Distribution Estimation

- Estimate the probabilities based on what we know

P(RED) = 4/9
P(BLUE) = 3/9
P(YELLOW) = 2/9

P(RED) + P(BLUE) + P(YELLOW) = 1



Joint Probability P(A,B)

- A and B are two discrete random variables
  - if A and B are independent, P(A,B) = P(A)P(B)

Example 1:

P("get lost during driving") = 1/1000, P("being bald") = 34/500

P("get lost during driving and being bald")

= P("get lost during driving") P("being bald")

= 1/1000 x 34/500 = 34/500000

Example 2: If P(RED, BLUE) means the probability of picking up RED ball first, putting it back and then picking up BLUE ball

P(RED, BLUE) = P(RED)P(BLUE) = 4/9 x 3/9=12/81

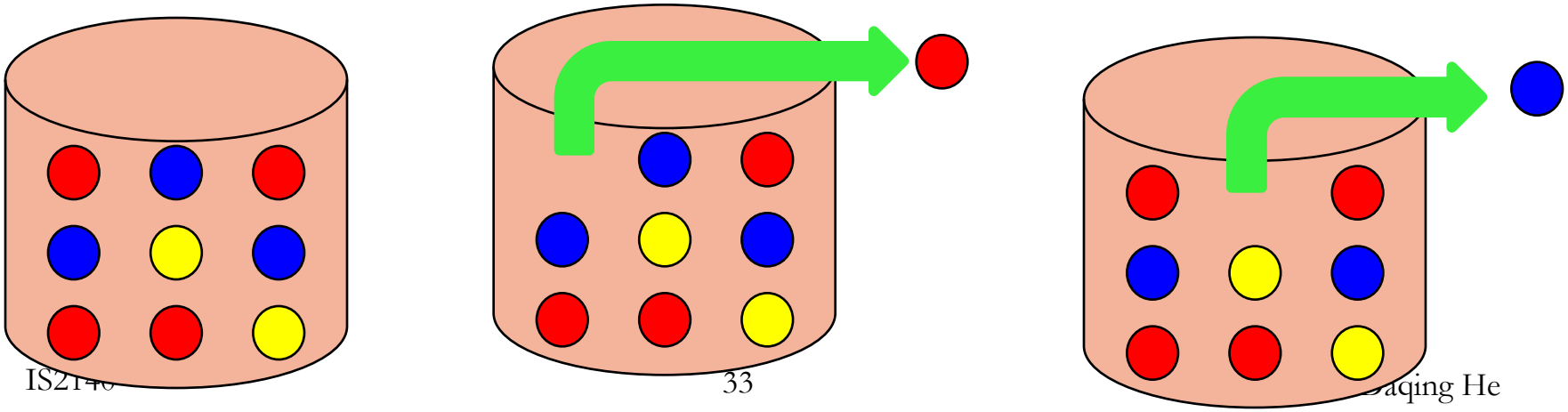
Joint Probability P(A,B)

- A and B are two discrete random variables
  - if A and B are not independent: $P(A,B) = P(A|B)P(B) = P(B|A)P(A)$

Example 1: If P(RED, BLUE) means the probability of picking up a RED ball and a BLUE ball together

$P(RED, BLUE) = P(BLUE|RED)P(RED) = 3/8 \times 4/9 = 12/72$

$P(RED, BLUE) = P(RED|BLUE)P(BLUE) = 4/8 \times 3/9 = 12/72$



Example

P("get lost during driving") = 1/1000
P("having GPS") = 64/100
P("get lost during driving"|"having GPS") = 1/10000

P("get lost during driving and having GPS")
= P("get lost during driving"|"having GPS") P("having GPS")
= 1/10000 x 64/100 = 64/1000000

Conditional Probability

- Conditional Probability P(A | B): Given even B has happened, the likeliness of event A happens

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$

- Bayes' Rule

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

P(A) is called prior probability

P(A | B) is called posterior probability after seeing event B

- If A can be divided into smaller events $a_1, a_2, \ldots, a_k$, and they are independent of each other

$$P(A \mid B) = P(a_1 \ldots a_k \mid B) = \prod_{i=1}^{k} P(a_i \mid B)$$

Statistical Language Models

Many slides are based on James Allan , Jimmy Lin and Lavrenko’s related courses

What is a Language Model (LM)?

- Language Model represents a probability distribution of a language
- Based on language model, we assign a probability to a string of text

p_1 = P("a happy running dog")

p_2 = P("dog running a happy")

p_3 = P("一条 happy running dog")

p_4 = P("一条快乐地奔跑的狗")

- Probabilities depend on what language we're modeling

In a language model for English: $p_1 > p_2 > p_3 > p_4$

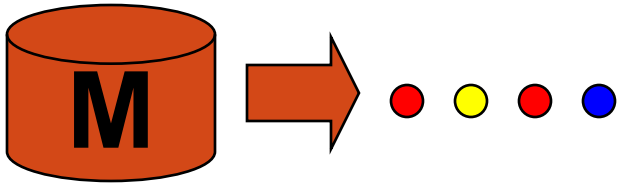
In a language model for Chinese: $p_4 > p_3 > p_1 > p_2$

Conventions

- M …. "language" we are trying to model
- s … observation (string of tokens from some vocabulary)
- P(s|M) … probablility of observing "s" in language M
- M can be thought of as a "source" or a generator
  - A mechanism that can spit out strings that are legal in the language
  - P(s|M) … probability of getting "s" from random sampling in M

Stochastic Language Models

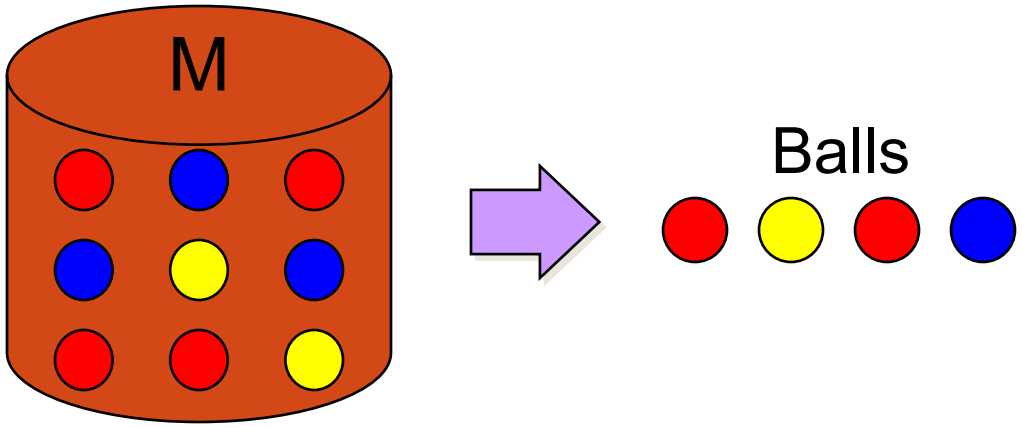
- A statistical model for generating text
  - Probability distribution over strings in a given language



P(● ○ ● ● | M) = P(● | M) P(○ | M, ●) P(● | M, ● ○) P(● | M, ● ○ ●)

Unigram Language Model

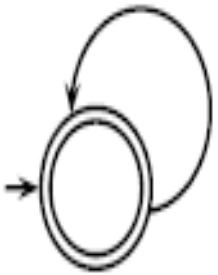
- Balls are randomly drawn from the box (with replacement)



P(● ● ● ● |M) = P(● |M) × P(● |M) × P(● |M) × P(● |M)

= (4/9) × (2/9) × (4/9) × (3/9)

Unigram Language Model



the	0.2
a	0.1
frog	0.01
toad	0.01
said	0.03
likes	0.02
that	0.04

▶ **Figure 12.2** A one-state finite automaton that acts as a unigram language model. We show a partial specification of the state emission probabilities.

Language Models

- Modeling *probability* of generating strings in the language (commonly all strings over alphabet Σ)
  - This is called **Prediction** based on the model

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

Unigram Language Model

- Assume each word is generated independently
  - This is why it is called unigram
  - It is consistent with the bag of word representation
- The probability of a string, given a model, then becomes

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

The probability of a sequence of words decomposes into a product of the probabilities of individual words

Unigram Language Model

Model M	
P(w)	w
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

P("the man likes the woman"|M)
= P(the|M) × P(man|M) × P(likes|M) × P(the|M) × P(man|M)
= 0.00000008

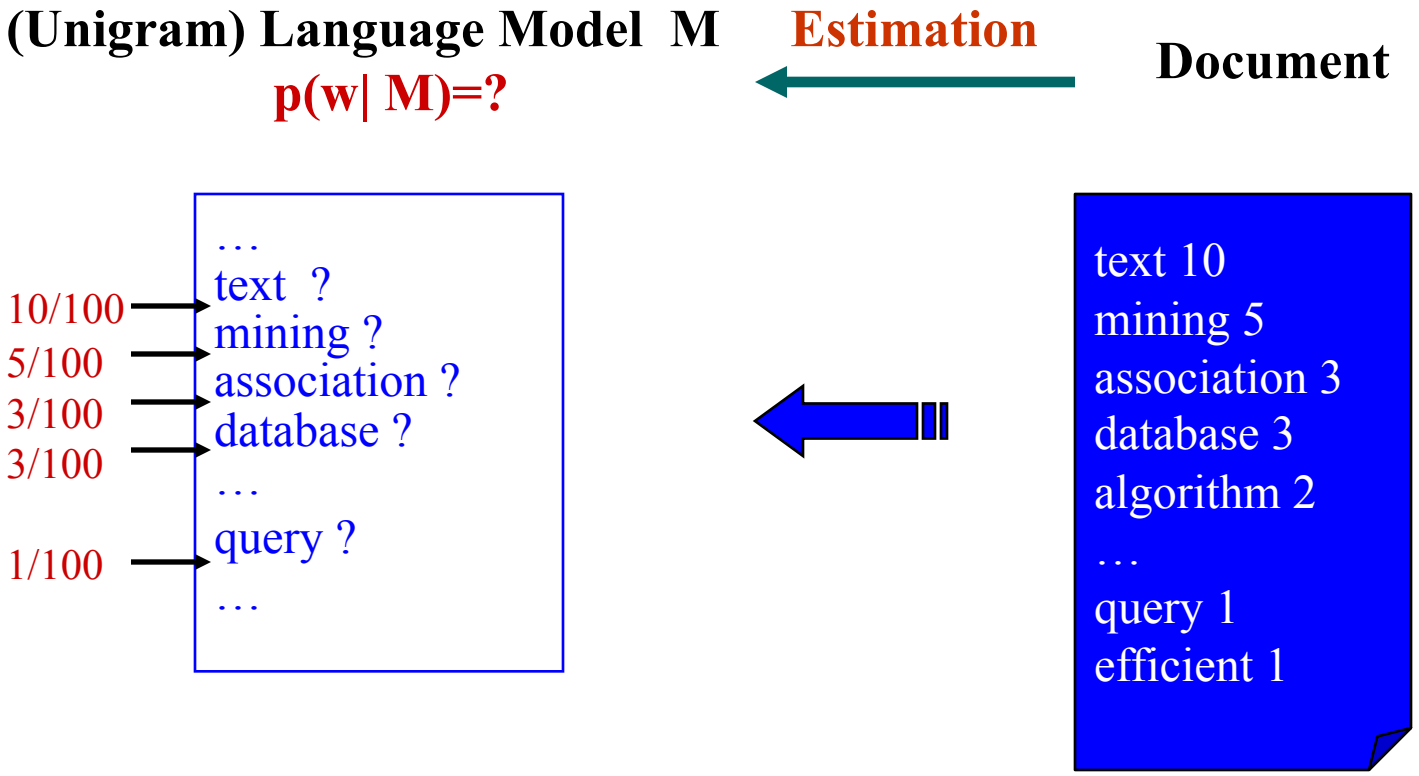
Estimation of Language Model

- Two important steps in language modeling
  - **Step 2: Predition**: using the language model to assign a probability to a text
    - We have talked about
  - **Step 1: Estimation**: observing relevant text and estimating the probability of each word
    - Relevant text can be as short as a word, a sentence to a document, and as long as a corpus, the entire web, or all text written for a language
    - The more text is there, the more accurate is the language model estimation

Unigram Language Model Estimation

- General estimation steps
  - Obtain relevant text for a language model to be estimated
    - General English, text about information retrieval, documents on teen health
  - Perform tokenization on the collected text
    - Similar to the text pre-processing steps for constructing index
    - Could perform case folding, phrase identification, stemming
    - But do not remove stop words
  - Counting
    - Count the total number of term occurrences (N)
    - Count the number of occurrences of each term t ➔ collection frequency CF(t)
    - Assign term t a probability equal to $$P(t) = \frac{CF(t)}{N}$$

Estimation of Unigram LM



Exercise

- Suppose we have 3 document collection

Doc1: wrecked roads and bridges in chile hinder rescue effort in chile earthquake

Doc2: test for chile's coalition in presidential election in chile

Doc3: frantic rescue efforts in chile as troops seek to keep order for rescue

- We know the length of doc 1 is 12, that of doc 2 is 9, that of doc 3 is 13, we build a language model M based on these three docs
- What is the probability p(chile | M)?

Comparing Predictions of LMs

Model M_1	
P(w)	w
0.2	the
0.0001	yon
0.01	class
0.0005	maiden
0.0003	sayst
0.0001	pleaseth
…	

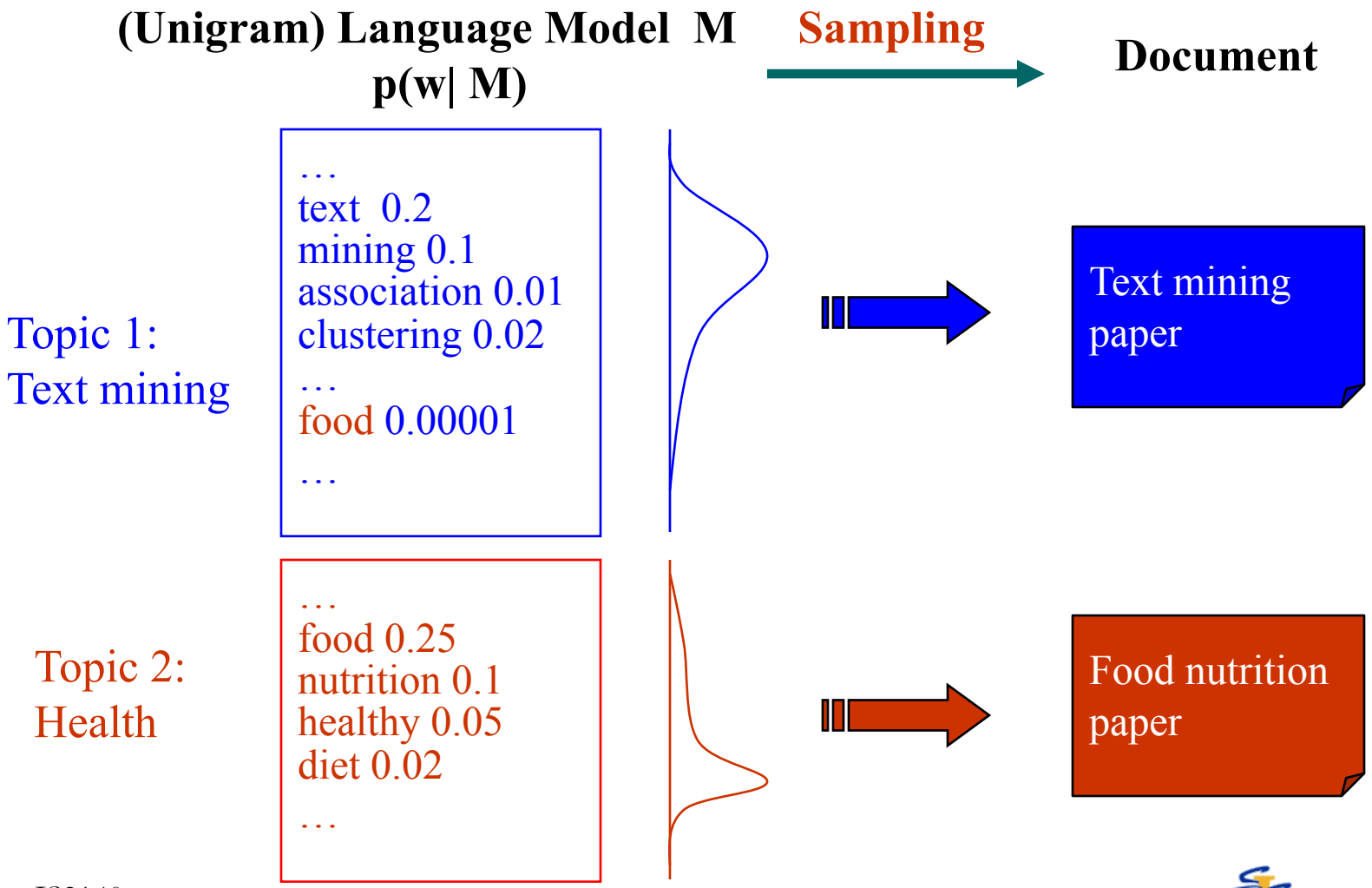
Model M_2	
P(w)	w
0.2	the
0.1	yon
0.001	class
0.01	maiden
0.03	sayst
0.02	pleaseth
…	

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.001	0.02	0.1	0.01

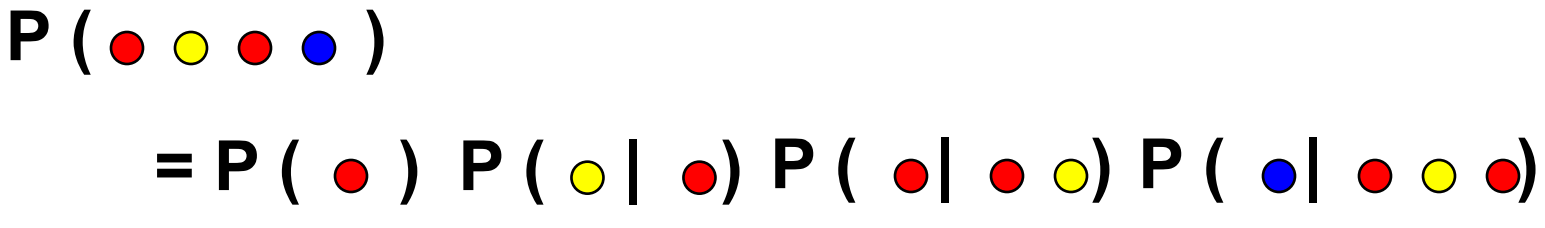
$$P(s|M_2) > P(s|M_1)$$

What exactly does this mean?

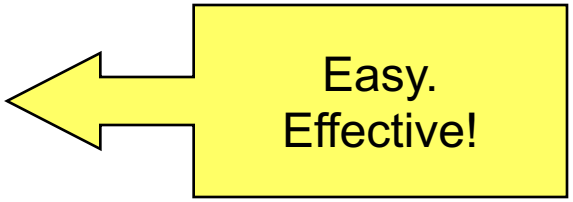
Text Generation with Unigram LM



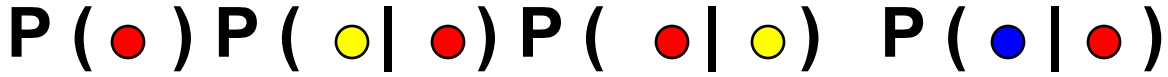
Summary of Various Language Models



- Unigram Language Models



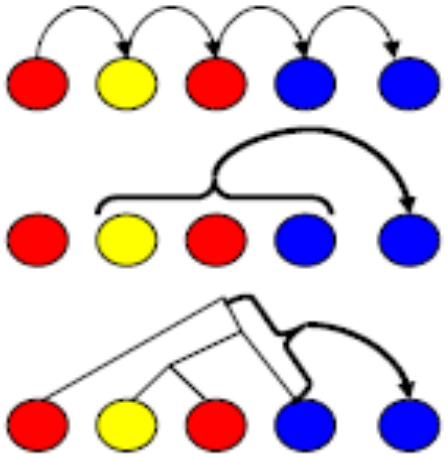
- Bigram (generally, *n*-gram) Language Models



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

More on Higher-order Models

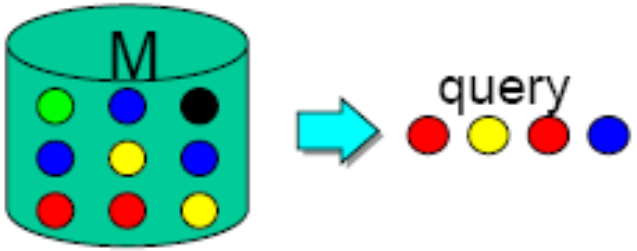
- Unigram model assumes word independence
  - cannot capture surface form: P("brown dog") == P("dog brown")
- Higher-order models
  - n-gram: condition on preceding words:
  - cache: condition on a window (cache):
  - grammar: condition on parse tree



- Are they useful?
  - no improvements from n-gram, grammar-based models
  - some research on cache-like models (proximity, passages, etc.)
  - parameter estimation is prohibitively expensive

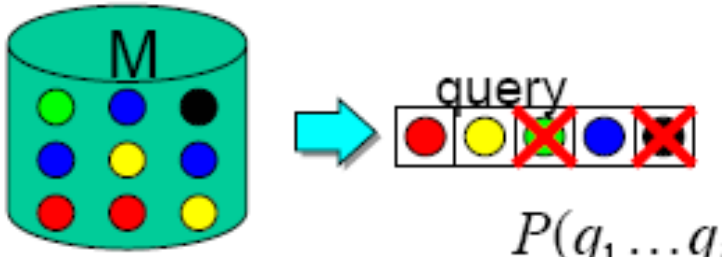
Multinomial and Multiple-Bernoulli

- Predominant model is the multinomial: [3,4,5]
  - fundamental event: *what is the identity of the i'th query token?*
  - observation is a sequence of events, one for each query token



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

- Original model is multiple-Bernoulli: [1,2]
  - fundamental event: *does the word w occur in the query?*
  - observation is a vector of binary events, one for each possible word

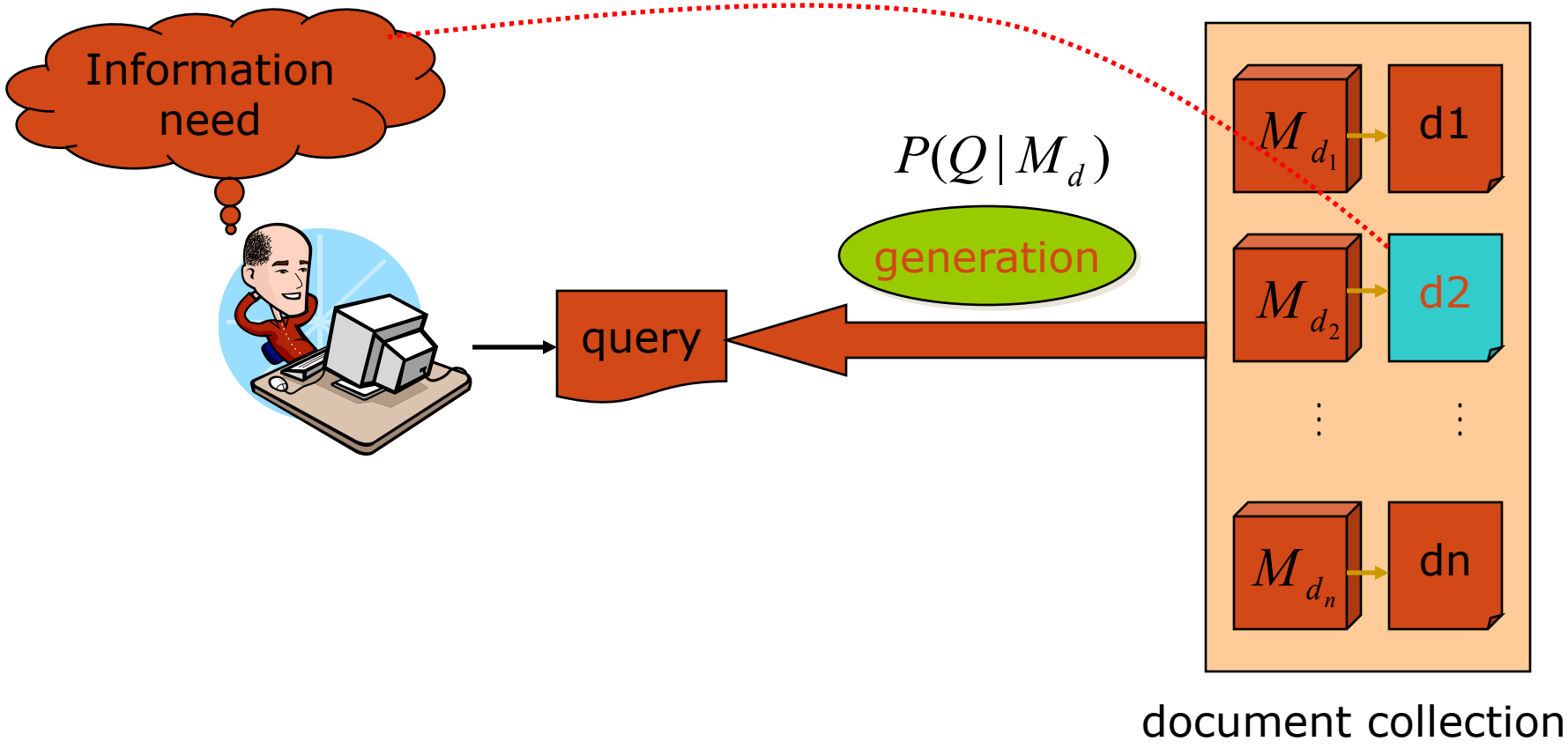


$$P(q_1 \ldots q_k \mid M) = \prod_{w \in q_1 \ldots q_k} P(w \mid M) \prod_{w \notin q_1 \ldots q_k} [1 - P(w \mid M)]$$

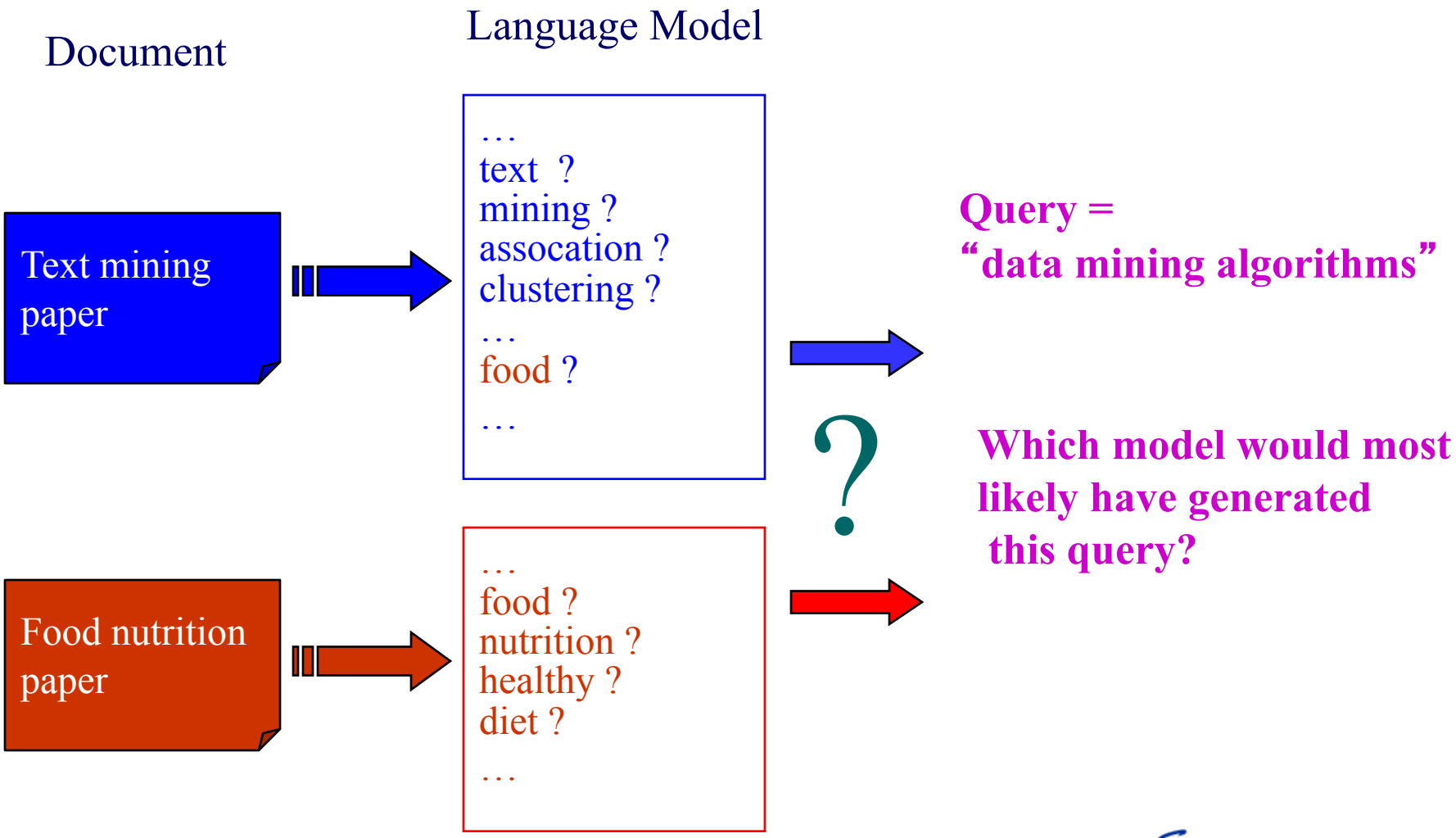
Multinomial or Multiple-Bernoulli

- Two models are fundamentally different
  - Entirely different events spaces
  - But both assume word independence (though it has different meanings)
- Multinomial
  - Can account for multiple word occurrences in the query
  - Widely used in many NLP areas
  - Possibility for integration with ASR, MT, NLP (same event space)
- Multiple-Bernoulli
  - May suit to IR (directly check on the presence of query terms)
  - Provisions for explicit negation of query terms ("A AND NOT B")
  - increasingly less popular than multinomial method

Language Model (LM) based IR



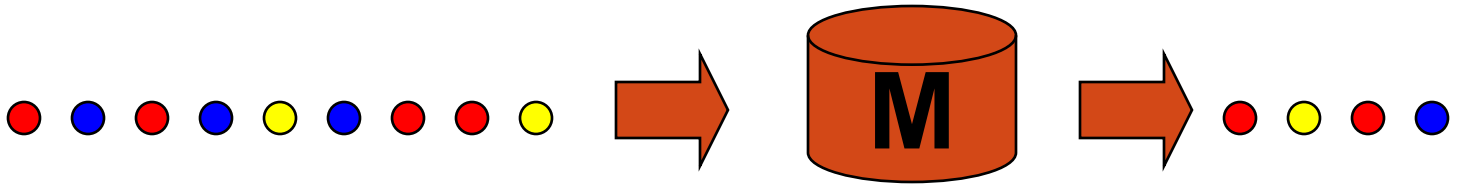
Language Models for Retrieval
(Ponte & Croft 98)



Language Model for IR

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

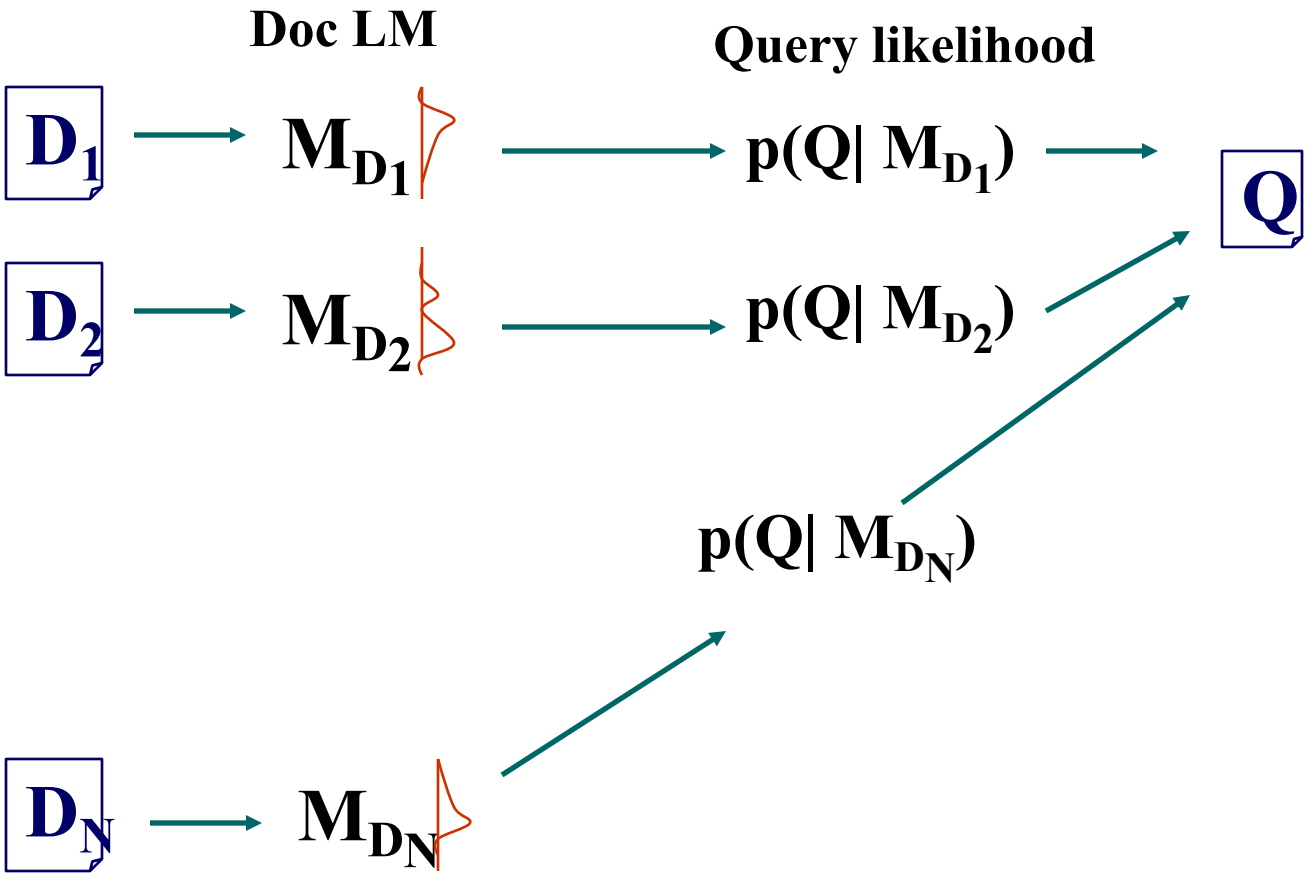
- Estimate a language model from a sample
- Then compute the observation probability



One Language Model Idea for IR

- Every document in a collection defines a "language"
  - Consider all possible sentences (strings) that author could have written drown when creating some given document
  - Some are perhaps more likely to occur than others
    - Subject to topic, writing style, language
    - P(s|M) … probability that author would write down string "s"
- Now suppose "q" is the user's query
  - What is the probability that author would written 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

Ranking Docs by Query Likelihood



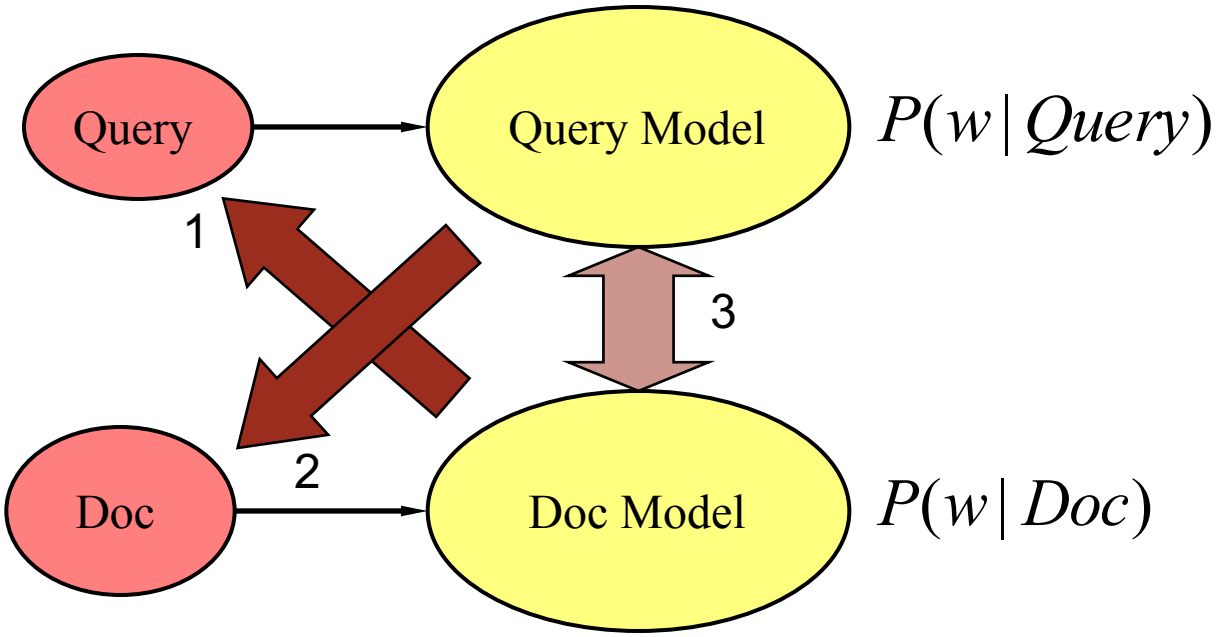
Query Likelihood Language Model

- Rank documents by the probability of generating the query

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

- Major issue is estimating the document model
- Drawbacks:
  - No notion of relevance in the model, everything is random sampling
  - Use feedback/query expansion is not part of the model
    - Examples of relevant documents cannot help in improving the document model Md
    - Only notion is to expand query Q with more terms
  - Does not directly allow structured or weighted queries.

Retrieval Using Language Models



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

Document Likelihood: Method 2

- Estimate a language model M_Q for query Q
- Rank documents by the likelihood of being a random sample from M_Q

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

- 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
  - But: different document length, probabilities are not comparable

Likelihood Ratio: Model 2'

- Using Bayes' likelihood that MQ is the source, given that we observed document D

$$P(M_Q \mid D) = \frac{P(M_Q)P(D \mid M_Q)}{P(D)} \approx \frac{c\prod_{w \in D} P(w \mid M_Q)}{\prod_{w \in D} P(w \mid GE)}$$

- Related to Probability Ranking principle: P(D|R)/P(D|NR)
- Allows relevance feedback, query expansion, etc.
- Can benefits from complex estimation of the query model MQ
- But does not provide modeling on document size

Model Comparison: Method 3

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

$$KL(M_Q \| M_D) = \sum_{w \in V} M_Q(w) \log \frac{M_Q(w)}{M_D(w)}$$

dependent to query only

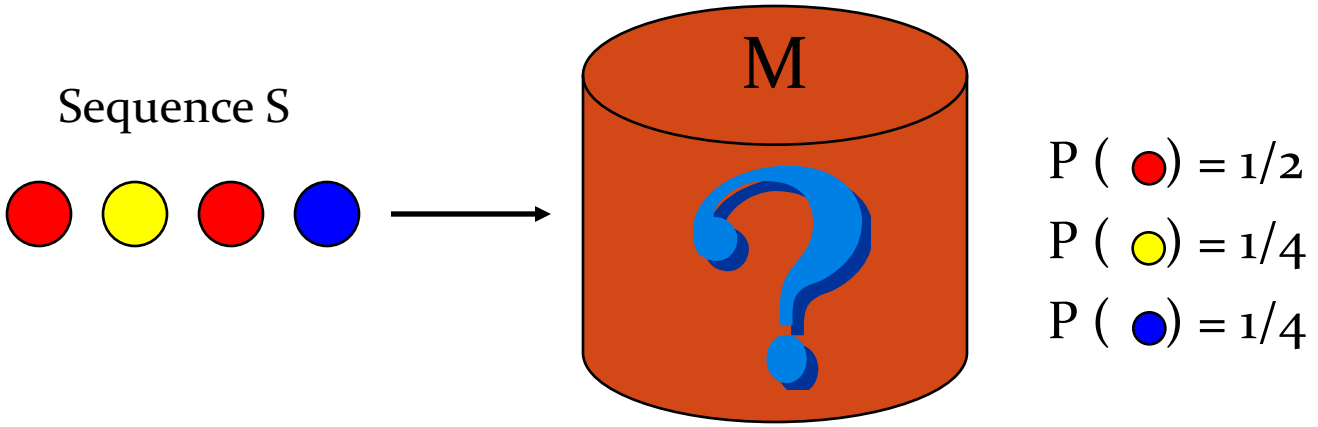
$$= -(\sum_{w \in V} M_Q(w) \log M_D(w)) + (-\sum_{w \in V} M_Q(w) \log M_Q(w)))$$

Use this part for ranking

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

Estimation

- Want to estimate M_Q and/or M_D from Q and/or D
- Maximum Likelihood Estimate (MLE or ML): for example estimate M_D is to simply count the frequencies in the document



$$P_{MLE}(w|M_S) = \#(w,S) \,/\, |S|$$

$\#(w,S)$ = number of times *w* occurs in S
$|S|$ = length of S

Exercise

- Suppose we have 3 document collection

Doc1: wrecked roads and bridges in chile hinder rescue effort in chile earthquake

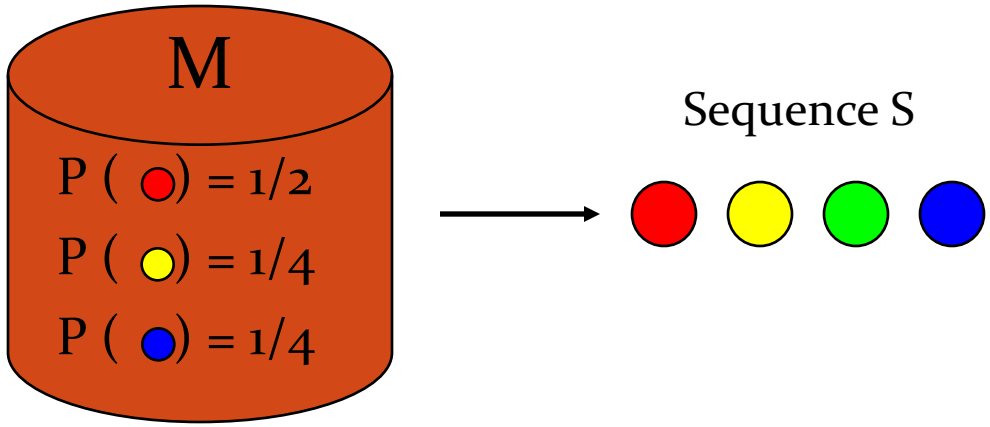
Doc2: test for chile's coalition in presidential election in chile

Doc3: frantic rescue efforts in chile as troops seek to keep order for rescue

- We know the length of doc 1 is 12, that of doc 2 is 9, that of doc 3 is 13
- Suppose the query is chile,
- What is the probability under query likelihood and MLE p(chile | doc 1)?

Zero-Frequency Problem

- Suppose some event is not in our observation S
  - Model will assign zero probability to that event



P ( ) = P ( ) × P ( ) × P ( ) × P ( )

= (1/2) × (1/4) × 0 × (1/4) = 0 !!

Why is this a Bad Idea?

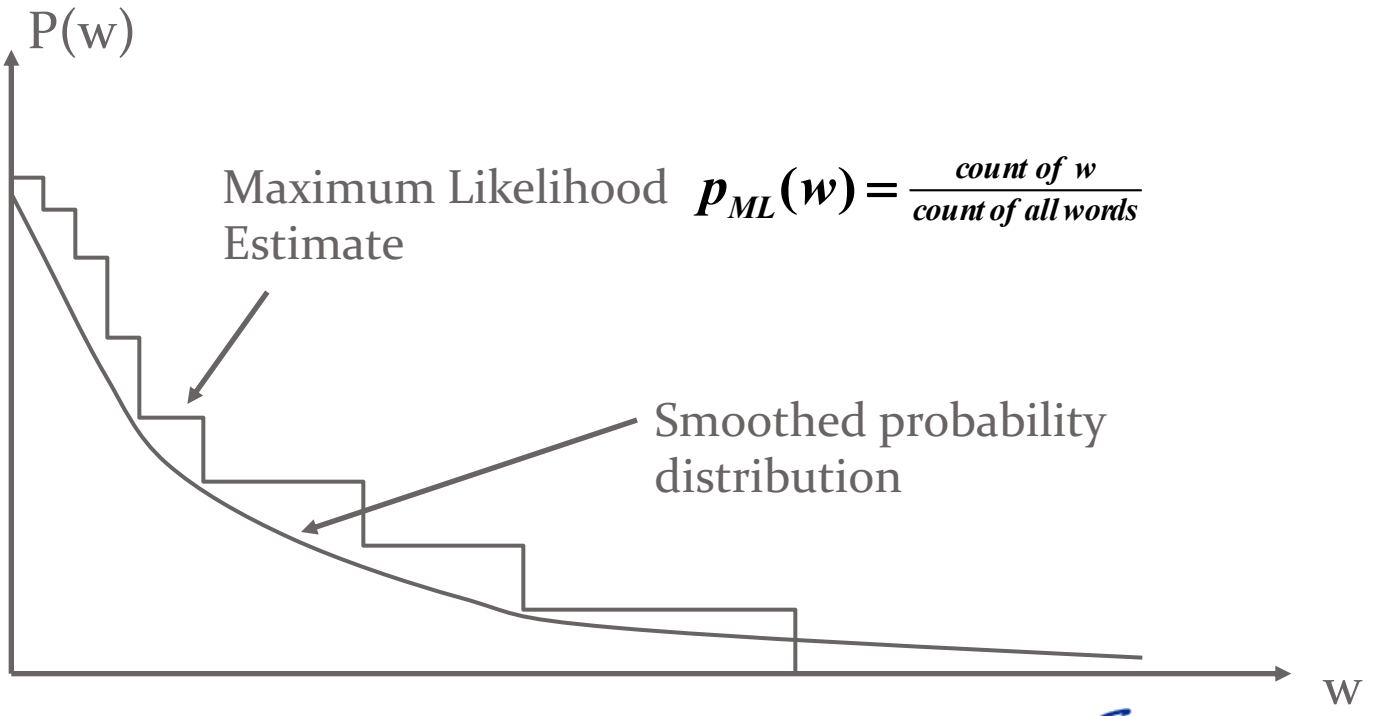
- Think of the document model as a topic
  - There are many documents that can be written about a single topic
  - We're trying to figure out what the model is based on just one document
- Modeling a document
  - Just because a word didn't appear doesn't mean it'll never appear…
  - But safe to assume that unseen words are rare

Analogy: fishes in the sea

- Practical effect: assigning zero probability to unseen words forces exact match
  - But partial matches are useful also!

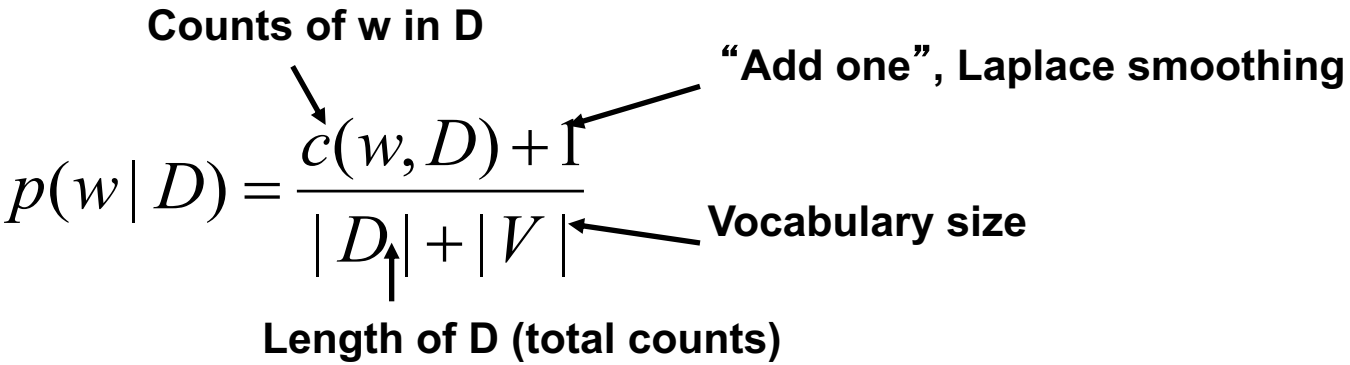
Smoothing

- All smoothing methods try to
  - discount the probability of words seen in a document
  - re-allocate the extra counts so that unseen words will have a non-zero count

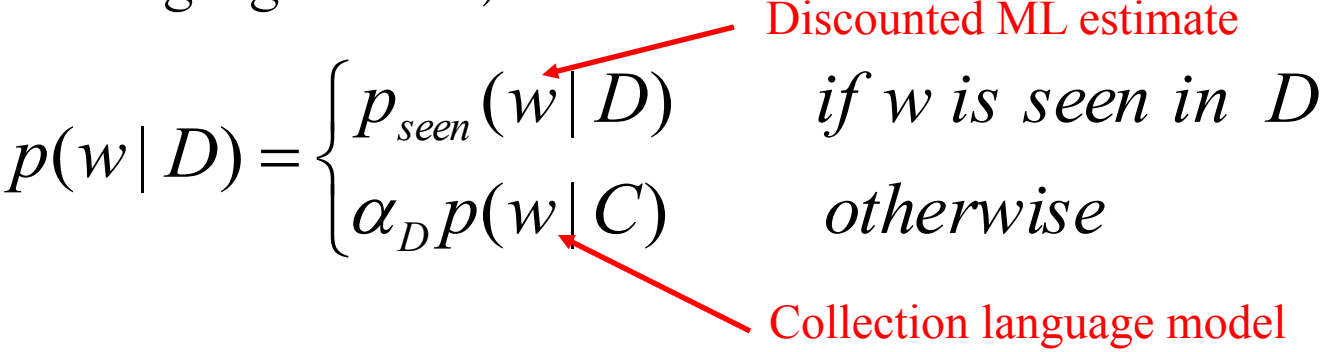


How to Smooth?

- A simple method (additive smoothing): **Add a small constant to the counts of each word**



- Another method using a reference model (collection language model) to discriminate unseen words



Linear interpolation Smoothing

- Also called Jelinek-Mercer smoothing
- Mixes the probability from the document with the general collection
  - "Shrink" uniformly toward p(w | Mc)

$$p(w|D) = \lambda p(w|M_D) + (1-\lambda) p(w|M_C)$$

- λ often is set around 0.8
- Methods for identifying optimal λ
  - Split data into training, held-out, and test
  - Train model on training set
  - Use held-out to test different values and pick the ones that works best (i.e., maximize the likelihood of the held-out data)
  - Test the model on the test data

Dirichlet Prior Smoothing

- Dirichlet prior smoothing is one particular smoothing method that follows the general smoothing scheme

$$p(w|D) = \frac{c(w,D) + \mu p(w|REF)}{|D| + \mu}$$

$$= \frac{|D|}{|D| + \mu} \frac{c(w,D)}{|D|} + \frac{\mu}{|D| + \mu} p(w|REF)$$

The optimal prior μ seems to vary from collection to collection, though in most cases, it is around 2,000.

Exercise

- Suppose we have 3 document collection

Doc1: wrecked roads and bridges in chile hinder rescue effort in chile earthquake

Doc2: test for chile's coalition in presidential election in chile

Doc3: frantic rescue efforts in chile as troops seek to keep order for rescue

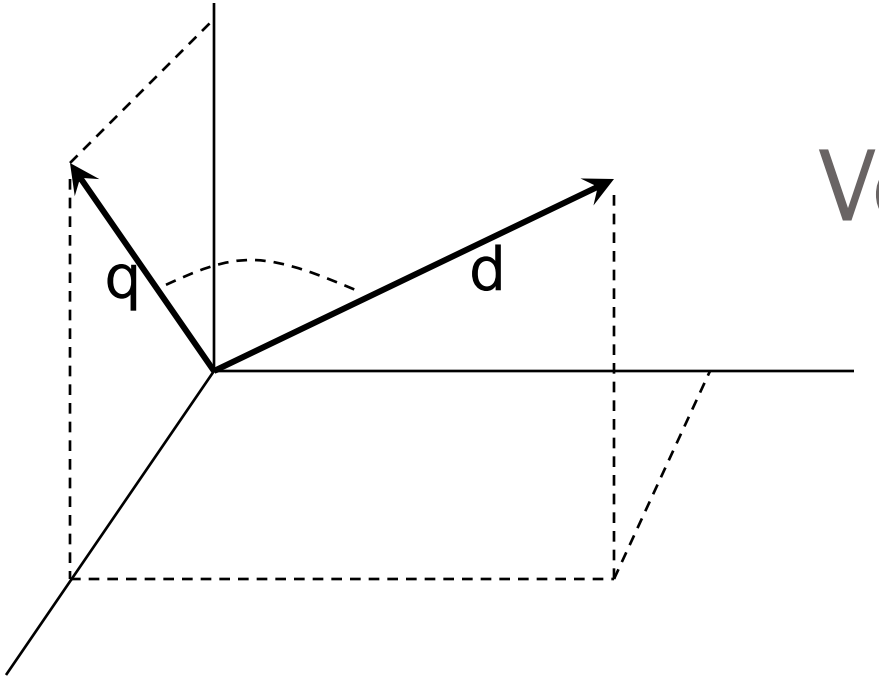
- If we use JM smoothing $p(t|M_D) = \lambda * p(t|D) + (1-\lambda) * p(t|C)$, where C is the whole collection and λ is 0.8, what is the probability p(earthquake | doc 3) and p(rescue | doc 3)?

Major Issues in Applying LM

- What kind of language model should we use?
  - Unigram or higher-order models
  - Multinomial or multiple-Bernoulli?
- How can we use the model for ranking?
  - Query-likelihood
  - Document-likelihood
  - Divergence of query and document models
- Many other issues, e.g.
  - how can we estimate model parameters?
  - How to model relevance
  - How to model relevance feedback

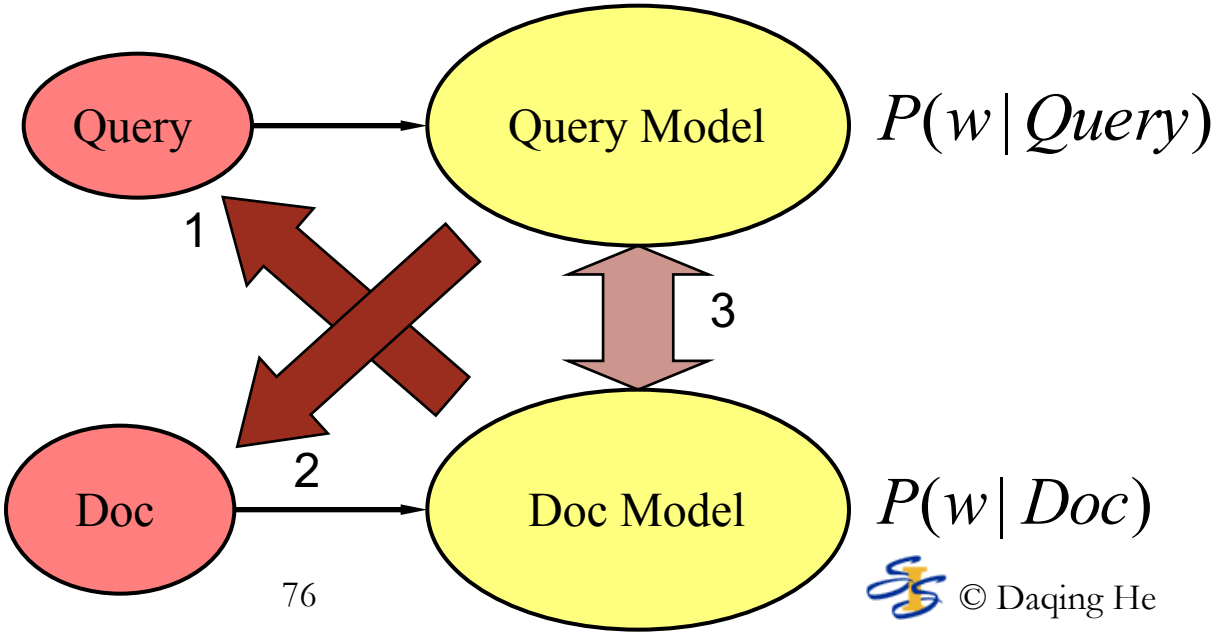
Language Modeling: pros and cons

- Pros
  - Formal mathematical model
  - Simple, well-understood framework
  - Integrates both indexing and retrieval models
  - Natural use of collection statistics, no heuristics
  - Avoid tricky issues of "relevance", "aboutness", etc
- Cons
  - Difficult to incorporate notions of "relevance", user preferences
  - Relevance feedback / query expansion not straightforward
  - Can't accommodate phrases, passages, Boolean operator
  - But there are recent LM works that address these issues



Vector Space

Language Modeling .



Language Modeling vs Vector Space

- 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
  - Details of use of document length and term, document, and collection frequency differ

Reference for Language Modelings

1. J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. Proceedings of ACM-SIGIR 1998, pages 275-281.
2. J. M. Ponte. A language modeling approach to information retrieval. Phd dissertation, University of Massachusets, Amherst, MA, September 1998.
3. D. Hiemstra. Using Language Models for Information Retrieval. PhD dissertation, University of Twente, Enschede, The Netherlands, January 2001.
4. D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden Markov model information retrieval system. Proceedings of ACM-SIGIR 1999, pages 214-221.
5. F. Song and W. B. Croft. A general language model for information retrieval. In Proceedings of Eighth International Conference on Information and Knowledge Management (CIKM 1999)
6. S. F. Chen and J. T. Goodman. An empirical study of smoothing techniques for language modeling. In Proceedings of the 34th Annual Meeting of the ACL, 1996.
7. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. Proceedings of the ACM-SIGIR 2001, pages 334-342.
8. V. Lavrenko and W. B. Croft. Relevance-based language models. Proceedings of the ACM SIGIR 2001, pages 120-127.
9. V. Lavrenko and W. B. Croft, Relevance Models in Information Retrieval, in Language Modeling for Information Retrieval, W. Bruce Croft and John Lafferty, ed., Kluwer Academic Publishers, chapter 2.

Classic Probabilistic Models

Many slides are based on James Allan's related courses and Manning's Introduction to IR book

A few words on BIR model and its extensions

- One of the oldest retrieval model
  - Give a firm theoretical foundations in 1970
  - Originally designed for short catalog records
  - Do not pay attention to term frequency and document length
    - So do not really suitable for full text retreival
- Revised in recent studies so it is among the best retrieval models
  - BM25 weighting scheme (also called Okapi weighting scheme)

$$RSV_d = \sum_{t \in q} \left[\log \frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1 - b) + b \times (L_d / L_{\mathrm{ave}})) + \mathrm{tf}_{td}} \cdot \frac{(k_3 + 1)\mathrm{tf}_{tq}}{k_3 + \mathrm{tf}_{tq}}$$

where k1 and k3 has a value between 1.2 and 2, and b = 0.75

Resources

S. E. Robertson and K. Spärck Jones. 1976. Relevance Weighting of Search Terms. *Journal of the American Society for Information Sciences* 27(3): 129–146.

C. J. van Rijsbergen. 1979. *Information Retrieval.* 2nd ed. London: Butterworths, chapter 6. [Most details of math] http://www.dcs.gla.ac.uk/Keith/Preface.html

N. Fuhr. 1992. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3),243–255. [Easiest read, with BNs]

F. Crestani, M. Lalmas, C. J. van Rijsbergen, and I. Campbell. 1998. Is This Document Relevant? … Probably: A Survey of Probabilistic Models in Information Retrieval. *ACM Computing Surveys* 30(4): 528–552.

http://www.acm.org/pubs/citations/journals/surveys/1998-30-4/p528-crestani/

[Adds very little material that isn't in van Rijsbergen or Fuhr]

Resources

H.R. Turtle and W.B. Croft. 1990. Inference Networks for Document Retrieval. *Proc. ACM SIGIR*: 1-24.

E. Charniak. Bayesian nets without tears. *AI Magazine* 12(4): 50-63 (1991). http://www.aaai.org/Library/Magazine/Vol12/12-04/vol12-04.html

D. Heckerman. 1995. A Tutorial on Learning with Bayesian Networks. Microsoft Technical Report MSR-TR-95-06 http://www.research.microsoft.com/~heckerman/

N. Fuhr. 2000. Probabilistic Datalog: Implementing Logical Information Retrieval for Advanced Applications. *Journal of the American Society for Information Science* 51(2): 95–110.

Summary

- Both LMs and BIR provide theoretical sound retrieval models based on probabilities
  - Explicitly model the uncertainty among the understanding of information need and that of the representation of documents and queries
  - But there are difference between LMs and BIR model
    - BIR explicitly model relevance, whereas LMs are not
- Both LMs and extensions of probabilistic retrieval models are among the state of art retrieval models