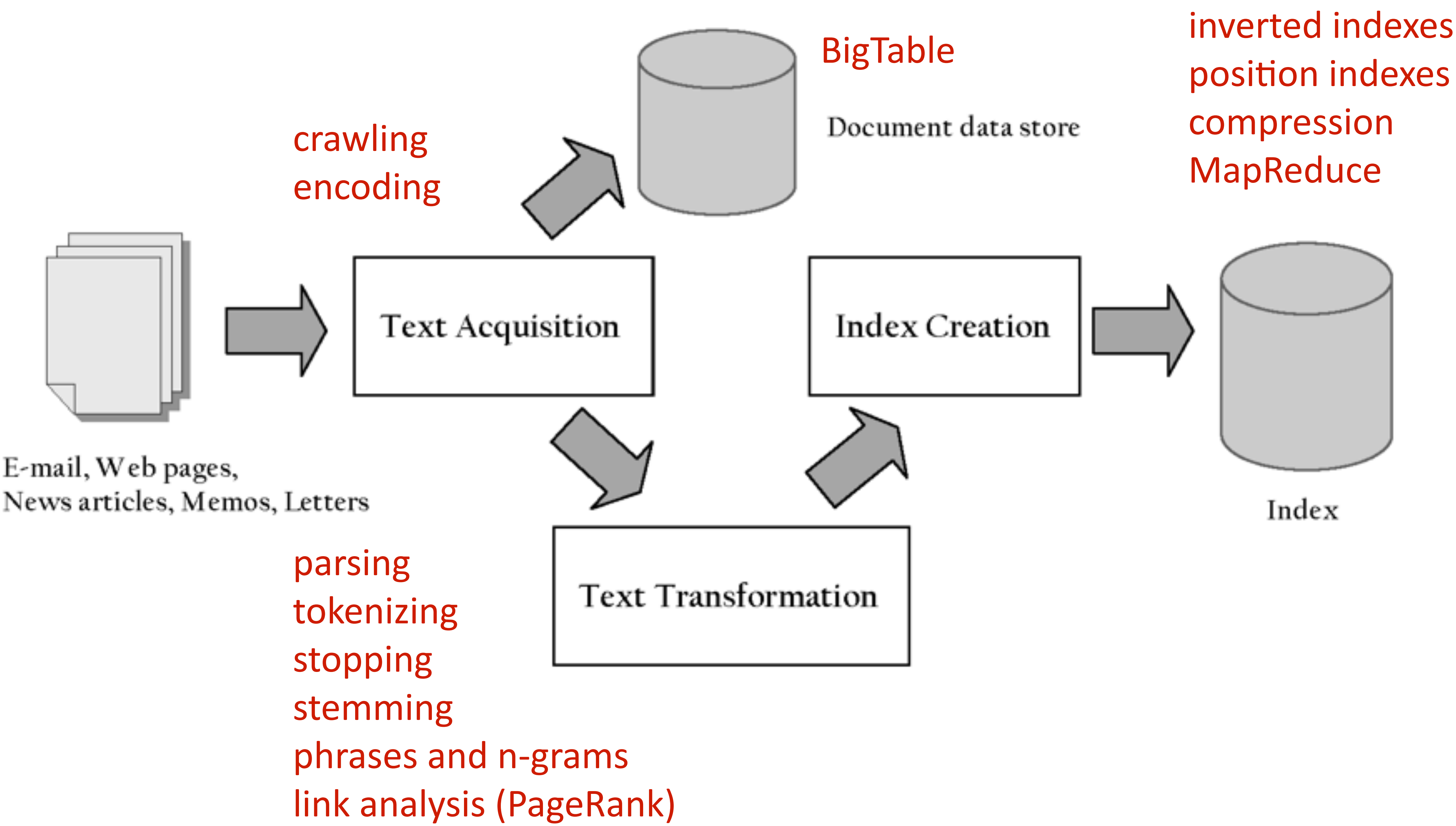
A Brief Review

CS6200
Information Retrieval

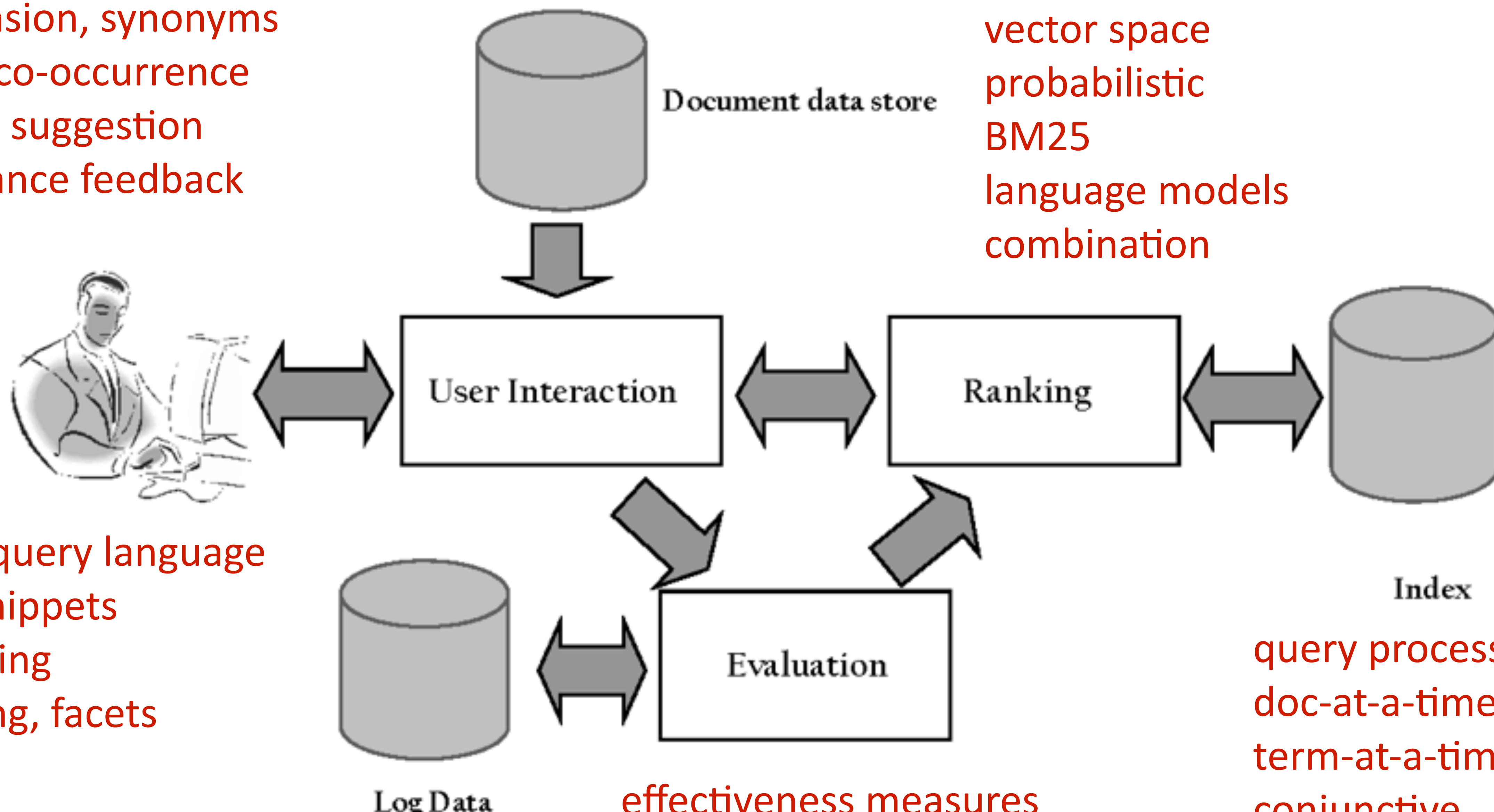
Indexing Process



Query Process

Information needs
query transformation
spelling correction
expansion, synonyms
term co-occurrence
query suggestion
relevance feedback

retrieval models
Boolean
vector space
probabilistic
BM25
language models
combination



Galago query language
result snippets
advertising
clustering, facets

query logs

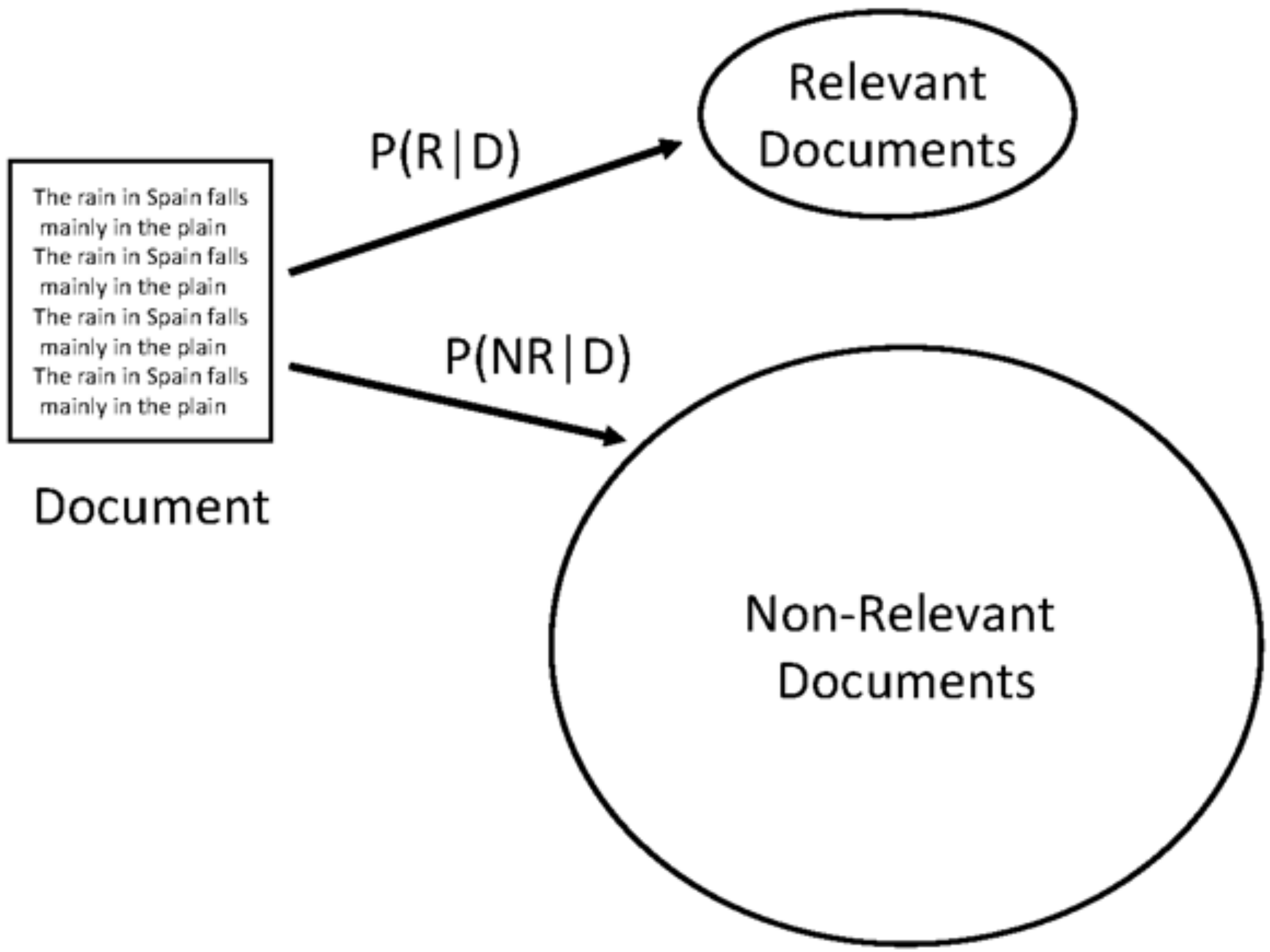
effectiveness measures
recall, precision, MAP
NDCG, significance

query processing
doc-at-a-time
term-at-a-time
conjunctive
optimization

Retrieval Model Overview

- Older models
  - Boolean retrieval
  - Vector Space model
- Probabilistic Models
  - BM25
  - Language models
- Combining evidence
  - Inference networks
  - Learning to Rank
  - *Term dependence and other feature-based models*

IR as Classification



Bayes Classifier

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

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

  - classify a document as relevant if

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

    - This is *likelihood ratio*

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)
  - 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
$d_i = 1$ $d_i = 0$	r_i $R - r_i$	$n_i - r_i$ $N - n_i - R + r_i$	n_i $N - r_i$
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 doc length
  - Typical TREC value for k_1 is 1.2, k_2 varies from 0 to 1000, b = 0.75

Language Model

- Language model
  - Probability distribution over strings of text
- **Unigram** language model
  - generation of text consists of pulling words out of a "bucket" according to the probability distribution and replacing them
- **N-gram** language model
  - some applications use bigram and trigram language models where probabilities depend on previous words

Language Model

- A *topic* in a document or query can be represented as a language model
  - i.e., words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model
- *Multinomial* distribution over words
  - text is modeled as a finite sequence of words, where there are *t* possible words at each point in the sequence
  - commonly used, but not only possibility
  - doesn’t model *burstiness*

LMs for Retrieval

- 3 possibilities:
  - probability of generating the query text from a document language model
  - probability of generating the document text from a query language model
  - comparing the language models representing the query and document topics
- Models of topical relevance

Query-Likelihood Model

- Rank documents by the probability that the query could be generated by the document model (i.e. same topic)
- Given query, start with P(D|Q)
- Using Bayes' Rule

$$p(D|Q) \stackrel{rank}{=} P(Q|D)P(D)$$

- Assuming prior is uniform, unigram model

$$P(Q|D) = \prod_{i=1}^{n} P(q_i|D)$$

Estimating Probabilities

- Obvious estimate for unigram probabilities is

$$P(q_i|D) = \frac{f_{q_i,D}}{|D|}$$

- *Maximum likelihood estimate*
  – makes the observed value of $f_{q_i;D}$ 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

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
  - What does this do to the likelihood of the document?

Estimating Probabilities

- Estimate for unseen words is $\alpha_D P(q_i|C)$
  - $P(q_i|C)$ is the probability for query word *i* in the *collection* language model for collection *C* (background probability)
  - α_D is a parameter
- Estimate for words that occur is

  $$(1 - \alpha_D)\ P(q_i|D) + \alpha_D P(q_i|C)$$

- Different forms of estimation come from different α_D

Effectiveness Measures

A is set of relevant documents,
B is set of retrieved documents

	Relevant	Non-Relevant
Retrieved	$A \cap B$	$\overline{A} \cap B$
Not Retrieved	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$

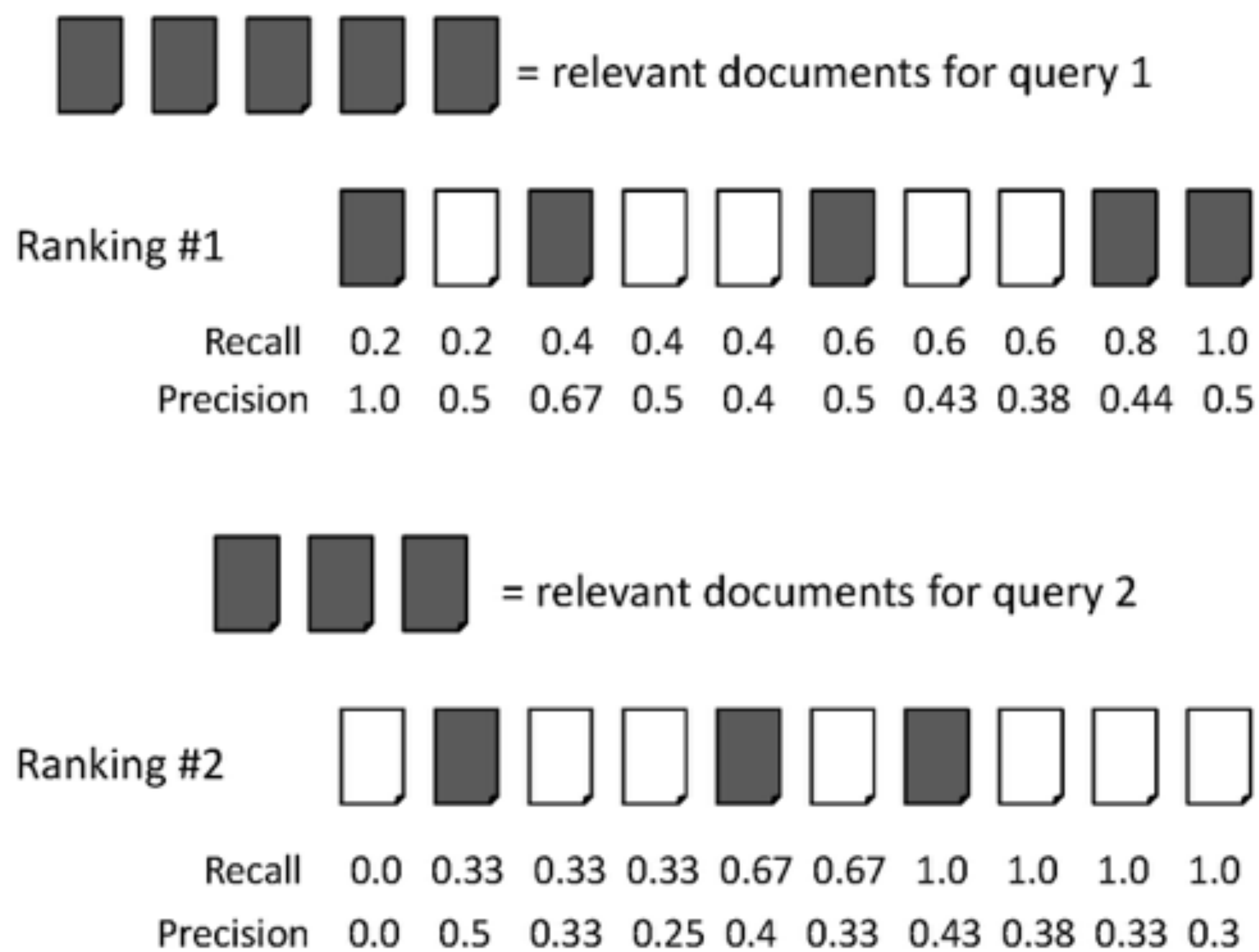
$$Recall = \frac{|A \cap B|}{|A|}$$

$$Precision = \frac{|A \cap B|}{|B|}$$

Averaging

- *Mean Average Precision* (MAP)
  - summarize rankings from multiple queries by averaging average precision
  - most commonly used measure in research papers
  - assumes user is interested in finding many relevant documents for each query
  - requires many relevant judgments in text collection
- Recall-precision graphs are also useful summaries

MAP



$$\textit{average precision query 1} = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$
$$\textit{average precision query 2} = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$\textit{mean average precision} = (0.62 + 0.44)/2 = 0.53$$

Focusing on Top Documents

- Precision at Rank R
  - R typically 5, 10, 20
  - easy to compute, average, understand
  - not sensitive to rank positions less than R
- Reciprocal Rank
  - reciprocal of the rank at which the first relevant document is retrieved
  - *Mean Reciprocal Rank (MRR)* is the average of the reciprocal ranks over a set of queries
  - very sensitive to rank position

Discounted Cumulative Gain

- Popular measure for evaluating web search and related tasks
- Two assumptions:
  - Highly relevant documents are more useful than marginally relevant document
  - the lower the ranked position of a relevant document, the less useful it is for the user, since it is less likely to be examined

Discounted Cumulative Gain

- *DCG* is the total gain accumulated at a particular rank *p*:

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

- Alternative formulation:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log(1+i)}$$

  – used by some web search companies
  – emphasis on retrieving highly relevant documents

Setting up a Classifier

Setting up a Classifier

- What we want:

  $p(\text{☺️} \mid w_1, w_2, ..., w_n) > p(\text{☹️} \mid w_1, w_2, ..., w_n)$?

Setting up a Classifier

- What we want:

  p(☺️ | $w_1, w_2, ..., w_n$) > p(☹️ | $w_1, w_2, ..., w_n$) ?

- What we know how to build:

Setting up a Classifier

- What we want:

  $p($☺️ $| w_1, w_2, \ldots, w_n) > p($☹️ $| w_1, w_2, \ldots, w_n)$?

- What we know how to build:
  - A language model for each class

Setting up a Classifier

- What we want:

  p(☺️ | $w_1, w_2, \ldots, w_n$) > p(☹️ | $w_1, w_2, \ldots, w_n$) ?

- What we know how to build:
  - A language model for each class
    - p($w_1, w_2, \ldots, w_n$ | ☺️)

Setting up a Classifier

- What we want:

  p(☺️ | $w_1, w_2, \ldots, w_n$) > p(☹️ | $w_1, w_2, \ldots, w_n$) ?

- What we know how to build:
  - A language model for each class
    - p($w_1, w_2, \ldots, w_n$ | ☺️)
    - p($w_1, w_2, \ldots, w_n$ | ☹️)

Bayes' Theorem

By the definition of conditional probability:

$$P(A, B) = P(B)P(A \mid B) = P(A)P(B \mid A)$$

we can show:

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

Seemingly trivial result from 1763;
interesting consequences...

REV. T. BAYES

A "Bayesian" Classifier

$$p(R \mid w_1, w_2, \ldots, w_n) = \frac{p(R)p(w_1, w_2, \ldots, w_n \mid R)}{p(w_1, w_2, \ldots, w_n)}$$

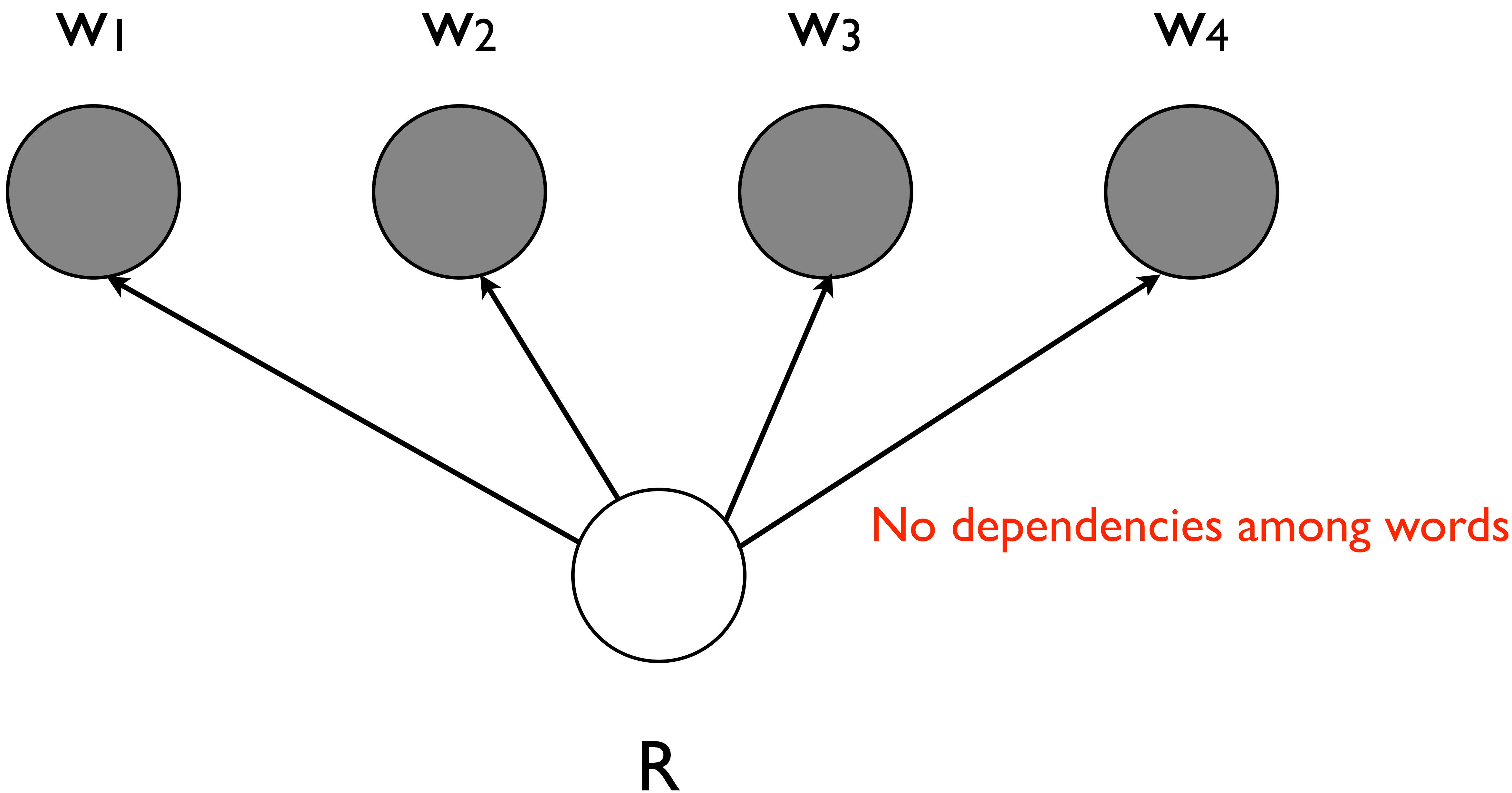
$$\max_{R \in \{\ddot\smile, \ddot\frown\}} p(R \mid w_1, w_2, \ldots, w_n) = \max_{R \in \{\ddot\smile, \ddot\frown\}} p(R)p(w_1, w_2, \ldots, w_n \mid R)$$

Posterior

Prior

Likelihood

Naive Bayes Classifier



NB on Movie Reviews

- Train models for positive, negative
- For each review, find higher posterior
- Which word probability ratios are highest?

```
>>> classifier.show_most_informative_features(5)

classifier.show_most_informative_features(5)
Most Informative Features
   contains(outstanding) = True              pos : neg    =     14.1 : 1.0
         contains(mulan) = True              pos : neg    =      8.3 : 1.0
        contains(seagal) = True              neg : pos    =      7.8 : 1.0
   contains(wonderfully) = True              pos : neg    =      6.6 : 1.0
         contains(damon) = True              pos : neg    =      6.1 : 1.0
```

Current Research Issues

- Understanding queries
  - NLP and queries, question answering, “semantic search”, query reformulation representations, query sessions, diversity, mapping queries to structure, rare queries, query similarity, query suggestion, genre classification
- Retrieval models
  - Learning to rank, Markov Random Field model, variations of language models, filtering models

Current Research Issues

- Evaluation
  - New metrics for new tasks (e.g., diversity, sessions), crowdsourcing, simulation, games
- New applications
  - Entity search, social search, personal search, multimedia search, aggregated search, opinion retrieval
- New architectures
  - Real-time search, mobile search, MapReduce

Careers and Study in IR

- Here: ML, NLP, data mining, independent study
- Graduate degrees: many possibilities, here, UMass Amherst, CMU, UIUC
- Careers:
  - Industry: Google, Microsoft, Yahoo, Amazon, Ebay, LinkedIn, Facebook, Twitter, etc. (all levels from B.S. to Ph.D.)
  - Academic: More in ML, “information” schools, Europe