
University of
Southampton

University of Southampton

Information Retrieval

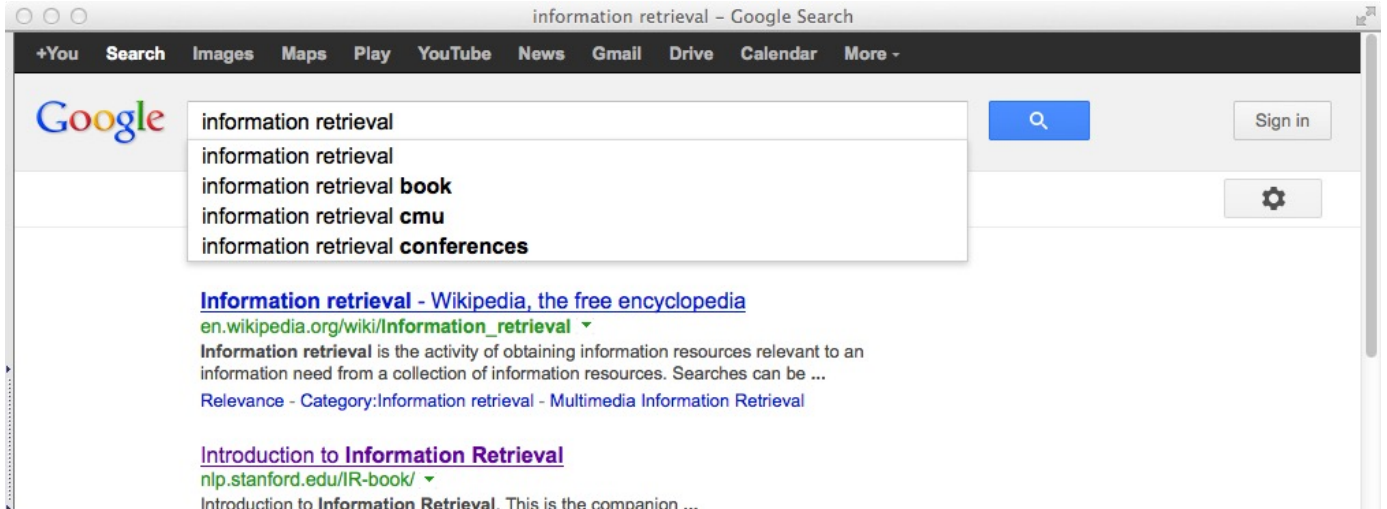
COMP3211 Advanced Databases

Dr Nicholas Gibbins – nmg@ecs.soton.ac.uk

A Definition

Information retrieval (IR) is the task of finding relevant documents in a collection

Best known modern examples of IR are Web search engines



University of Southampton

Overview

- Information Retrieval System Architecture
- Implementing Information Retrieval Systems
- Indexing for Information Retrieval
- Information Retrieval Models
  - Boolean
  - Vector
- Evaluation

The Information Retrieval Problem

The primary goal of an information retrieval system is to retrieve all the documents that are relevant to a user query while retrieving as few non-relevant documents as possible

Terminology

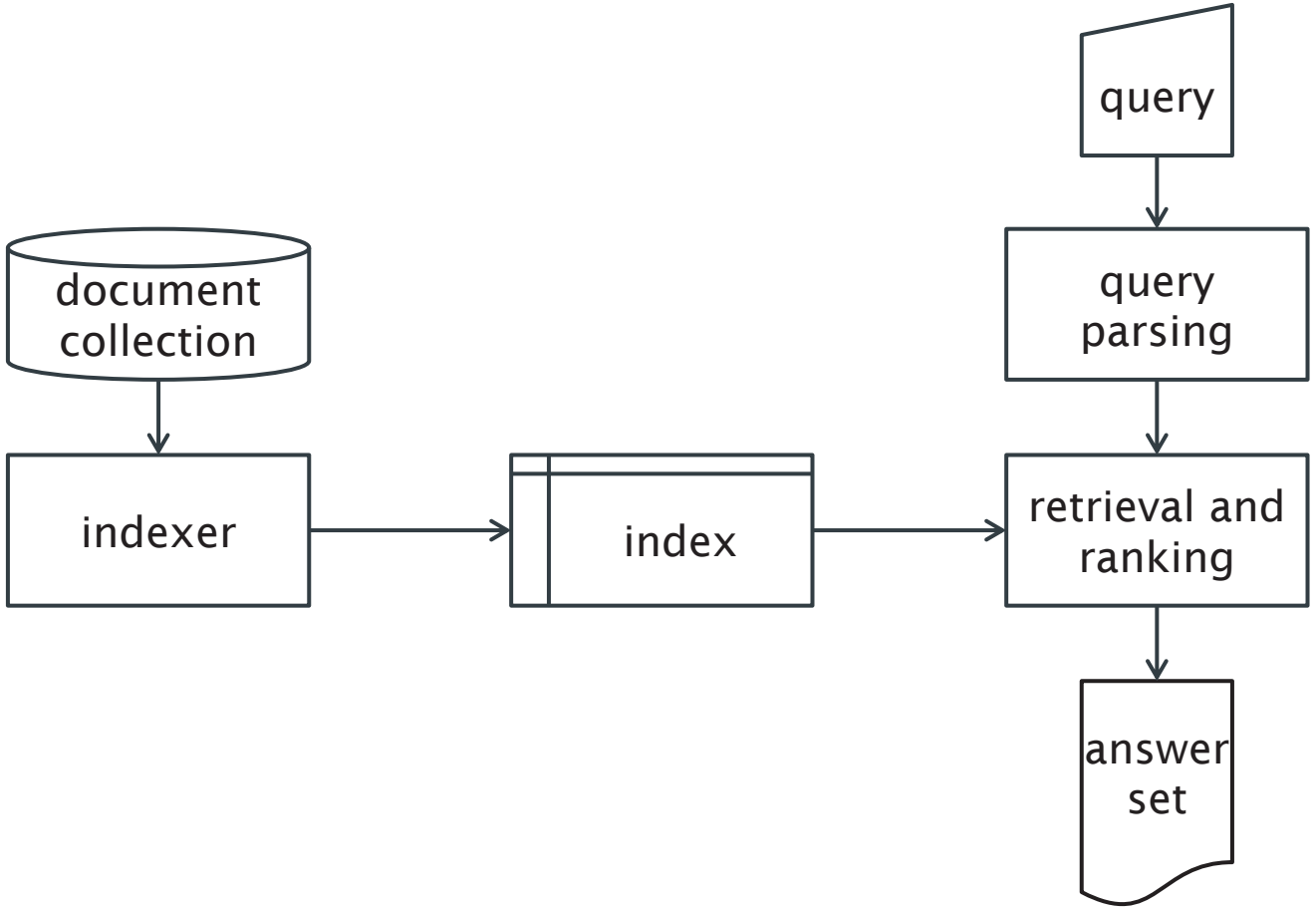
An *information need* is a topic about which a user desires to know more

A *query* is what the user conveys to the computer in an attempt to communicate their information need.

A document is *relevant* if it is one that the user perceives as containing information of value with respect to their personal information need.

University of Southampton

High-Level System Architecture



Characterising IR Systems

Document collection

- Document granularity: paragraph, page or multi-page text

Query

- Expressed in a query language
- List of words [AI book], a phrase [“AI book”], contain boolean operators [AI AND book] and non-boolean operators [AI NEAR book]

Answer Set

- Set of documents judged to be relevant to the query
- Ranked in decreasing order of relevance

Implementing IR Systems

User expectations of information retrieval systems are exceedingly demanding

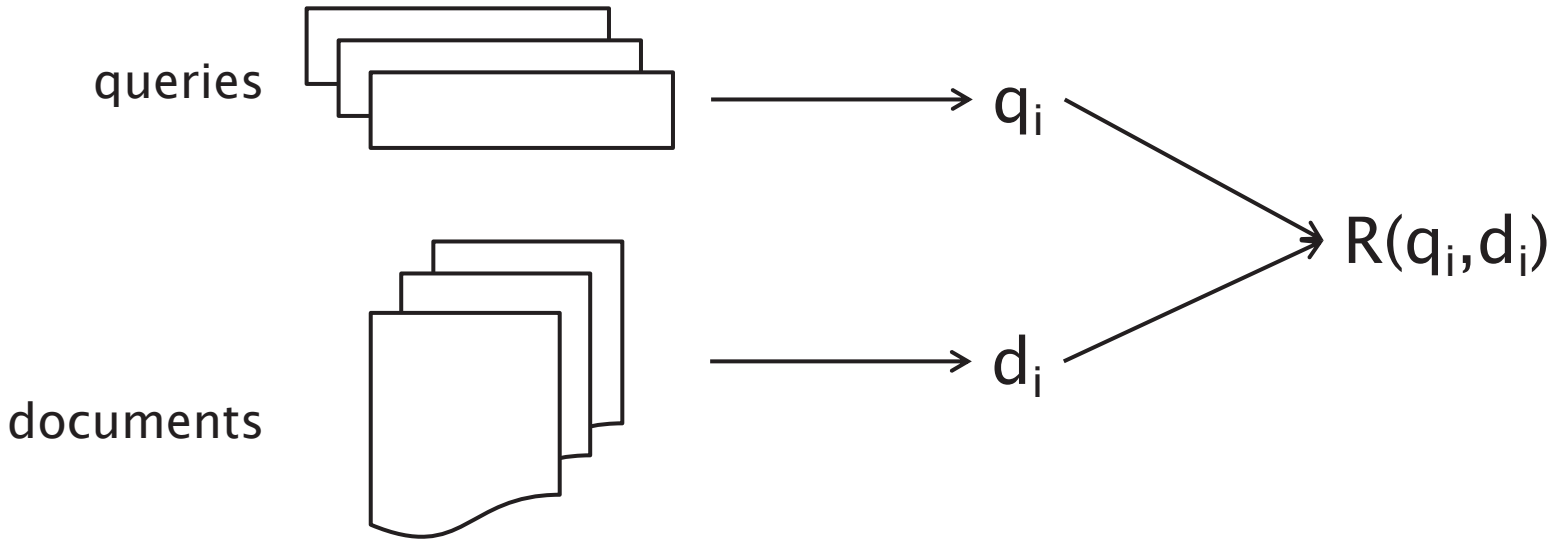
- Need to index billion document collections
- Need to return top results from these collections in a fraction of a second

Design of appropriate document representations and data structures is critical

Information Retrieval Models

An information retrieval model consists of

- A set D of representations of the documents in the collection
- A set Q of representations of user information needs (queries)
- A ranking function $R(d_i,q_i)$ that associates a real number with a query representation $q_i \in Q$ and a document representation $d_i \in D$



Information Retrieval Models

Information retrieval systems are distinguished by how they represent documents, and how they match documents to queries

Several common models:

- Boolean (set-theoretic)
- Algebraic (vector spaces)
- Probabilistic

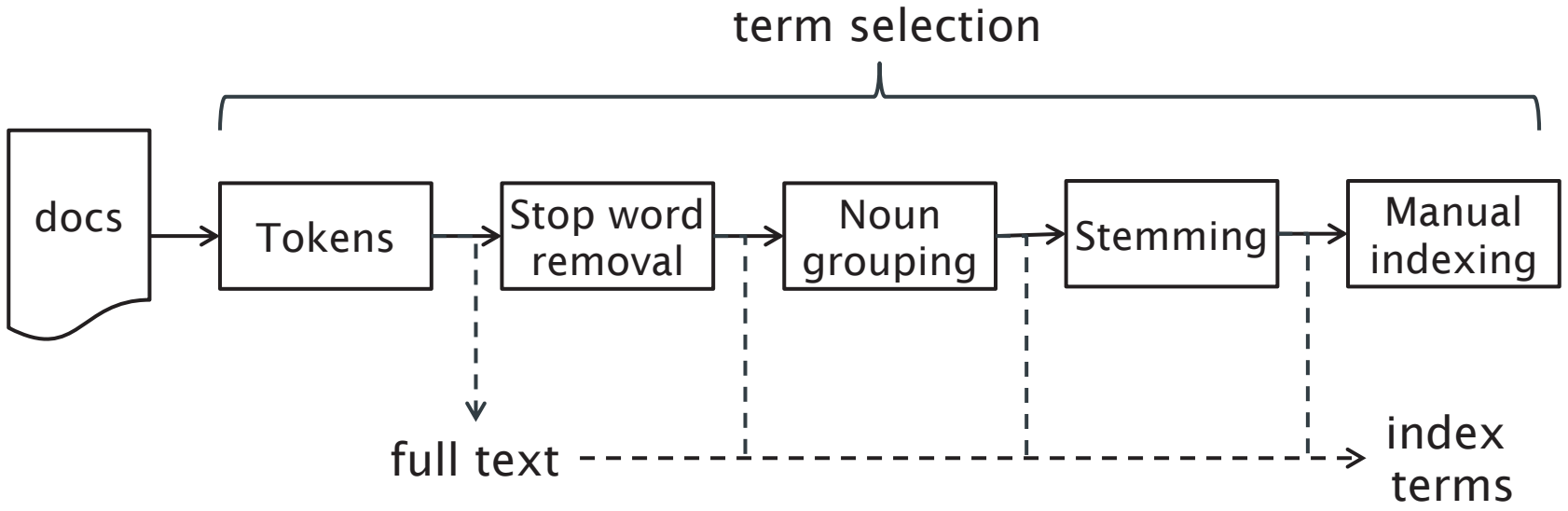
University of Southampton

Implementing Information Retrieval

Term Selection

Information retrieval systems either:

- index all words contained in documents (*full-text indexing*)
- selectively extract *index terms* from the documents that are to be indexes



Tokenisation

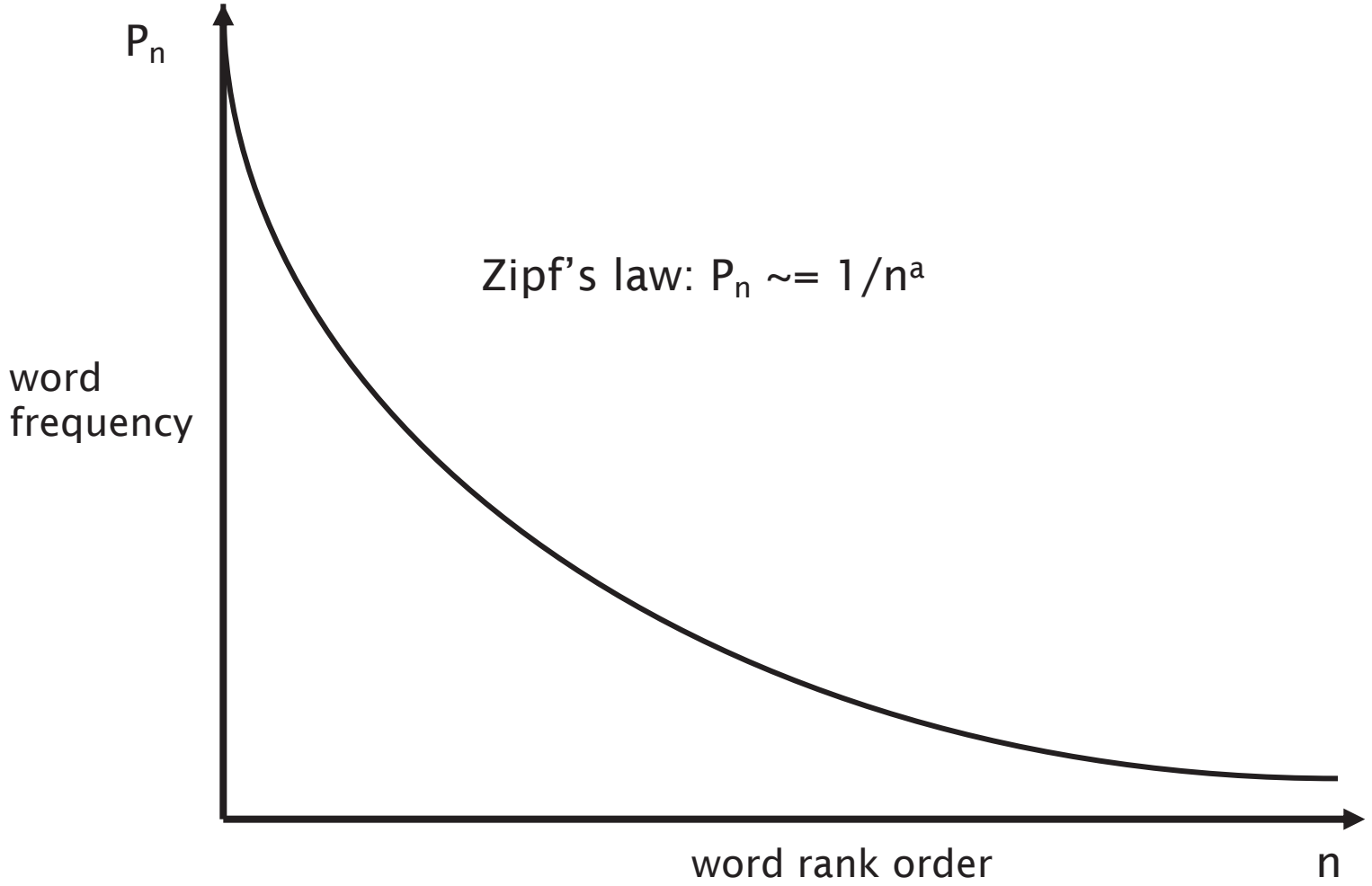
Identify distinct words

- Separate on whitespace, turn each document into list of tokens
- Discard punctuation (but consider tokenisation of O'Neill, and hyphens)
- Fold case, remove diacritics (normalisation)

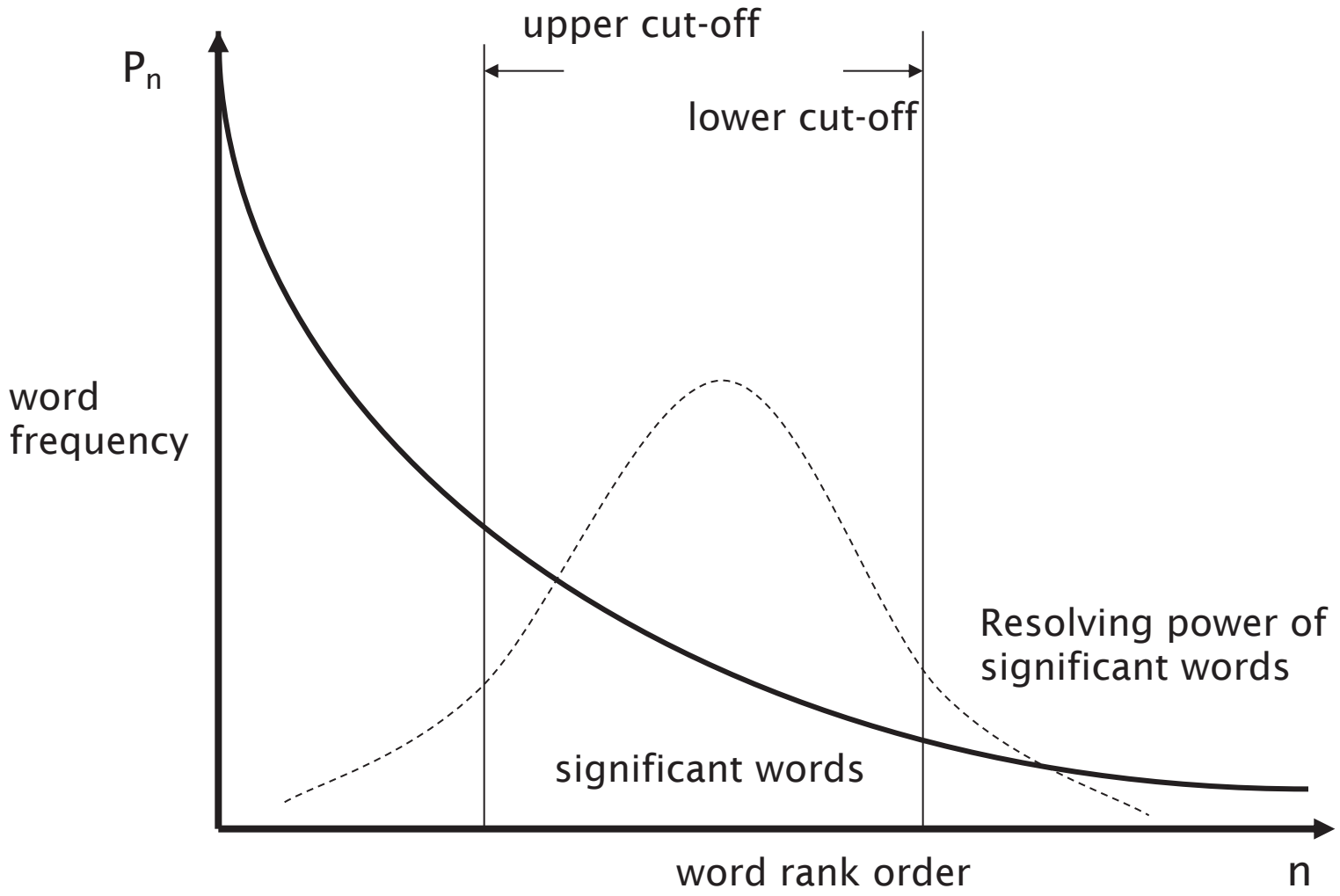
Tokenisation is language-specific

- Identify language first (using character n-gram model, Markov model)
- Languages with compound words (e.g. German, Finnish) may require special treatment (*compound splitting*)
- Languages which do not leave spaces between words (e.g. Chinese, Japanese) may require special treatment (*word segmentation*)

Word Significance



Word Significance



University of Southampton

Stop Word Removal

Extremely common words have little use for discriminating between documents

- e.g. the, of, and, a, or, but, to, also, at, that...

Construct a *stop list*

- Sort terms by collection frequency
- Identify high frequency *stop words* (possibly hand-filtering)
- Use stop list to discard terms during indexing

Noun Grouping

Most meaning is carried by nouns

Identify groups of adjacent nouns to index as terms

- e.g. railway station
- Special case of biword indexing

University of Southampton

Stemming

Terms in user query might not exactly match document terms

- query contains ‘computer’, document contains ‘computers’

Terms have wide variety of syntactic variations

- Affixes added to word stems, that are common to all inflected forms
- For English, typically suffixes
- e.g. connect is the stem of: connected, connecting, connects

Use stemming algorithm:

- to remove affixes from terms before indexing
- when generating query terms from query

University of Southampton

Stemming Algorithms

Lookup table

- Contains mappings from inflected forms to uninflected stems

Suffix-stripping

- Rules for identifying and removing suffixes
- e.g. ‘-sses’→‘-ss’
- Porter Stemming Algorithm

Lemmatisation

- Identify part of speech (noun, verb, adverb)
- Apply POS-specific normalisation rules to yield lemmas

Stemming

Yields small (~2%) increase in recall for English

- More important in other languages

Can harm precision

- ‘stock’ versus ‘stocking’

Difficult in agglomerative languages (German, Finnish)

Difficult in languages with many exceptions (Spanish)

- Use a dictionary

Manual Indexing

Identification of indexing terms by a (subject) expert

Indexing terms typically taken from a *controlled vocabulary*

- may be a flat list of terms, or a hierarchical thesaurus
- may be structured terms
  (e.g. names or structures of chemical compounds)

Result Presentation

Probability ranking principle

- List ordered by probability of relevance
- Good for speed
- Doesn't consider utility
- If two copies of most relevant document, second has equal relevance but zero utility once you've seen the first.

Many IR systems eliminate results that are too similar

University of Southampton

Result Presentation

Results classified into pre-existing taxonomy of topics

- e.g. News as world news, local news, business news, sports news
- Good for a small number of topics in collection

Document clustering creates categories from scratch for each result set

- Good for broader collections (such as WWW)

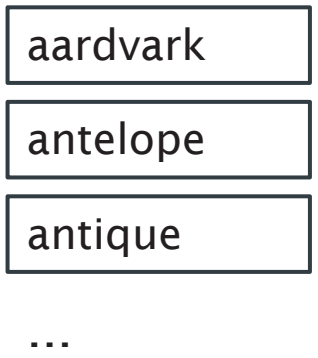
University of Southampton

Indexing

Lexicon

Data structure listing all the terms that appear in the document collection

- Lexicon construction carried out after term selection
- Usually hash table for fast lookup

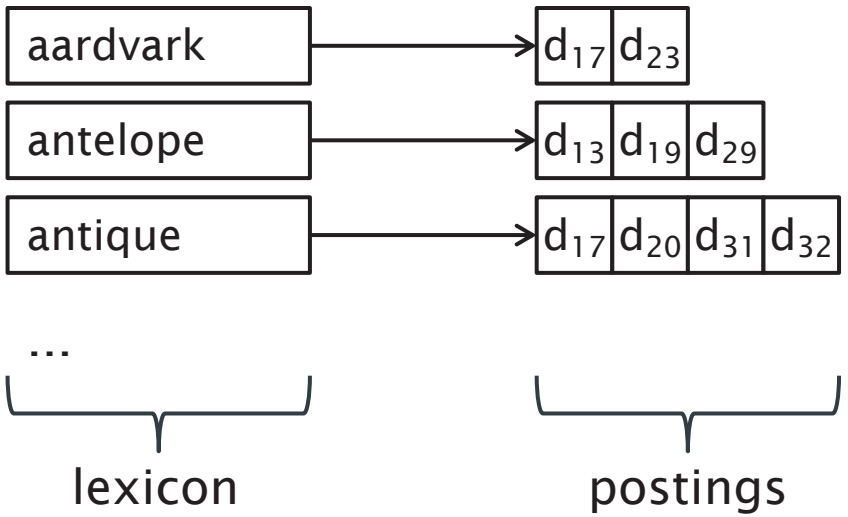


Inverted Index

Data structure that relates a word in the lexicon to a document in which it appears (a *posting*)

May also include:

- occurrence count within document
- pointer to location of word within document



University of Southampton

Searching for single words, attempt 1

If postings consist only of document identifiers, no ranking of results is possible

1. Lookup query term in lexicon
2. Use inverted index to get address of posting list
3. Return full postings list

Searching for single words, attempt 2

If postings also contain term count, we can do some primitive ranking of results:

1. Lookup query term in lexicon
2. Use inverted index to get address of posting list
3. Create empty priority queue with maximum length R
4. For each document/count pair in posting list
    1. If priority queue has fewer than R elements, add (doc, count) pair to queue
    2. If count is larger than lowest entry in queue, delete lowest entry and add the new pair

University of Southampton

Boolean Model

Boolean Model

Lookup each query term in lexicon and inverted index

Apply boolean set operators to posting lists for query terms to identify set of relevant documents

- AND - set intersection
- OR - set union
- NOT - set complement

Example

Document collection:

- d1 = “Three quarks for Master Mark”
- d2 = “The strange history of quark cheese”
- d3 = “Strange quark plasmas”
- d4 = “Strange Quark XPress problem”

Example

Lexicon and inverted index:

three	→	$\{d_1\}$
quark	→	$\{d_1, d_2, d_3, d_4\}$
master	→	$\{d_1\}$
mark	→	$\{d_1\}$
strange	→	$\{d_2, d_3, d_4\}$
history	→	$\{d_2\}$
cheese	→	$\{d_2\}$
plasma	→	$\{d_3\}$
xpress	→	$\{d_4\}$
problem	→	$\{d_4\}$

Example

Query:

"strange" AND "quark" AND NOT "cheese"

Result set:

$\{d_2, d_3, d4\} \cap \{d_1, d_2, d_3, d_4\} \cap \{d_1, d_3, d_4\} = \{d_3, d_4\}$

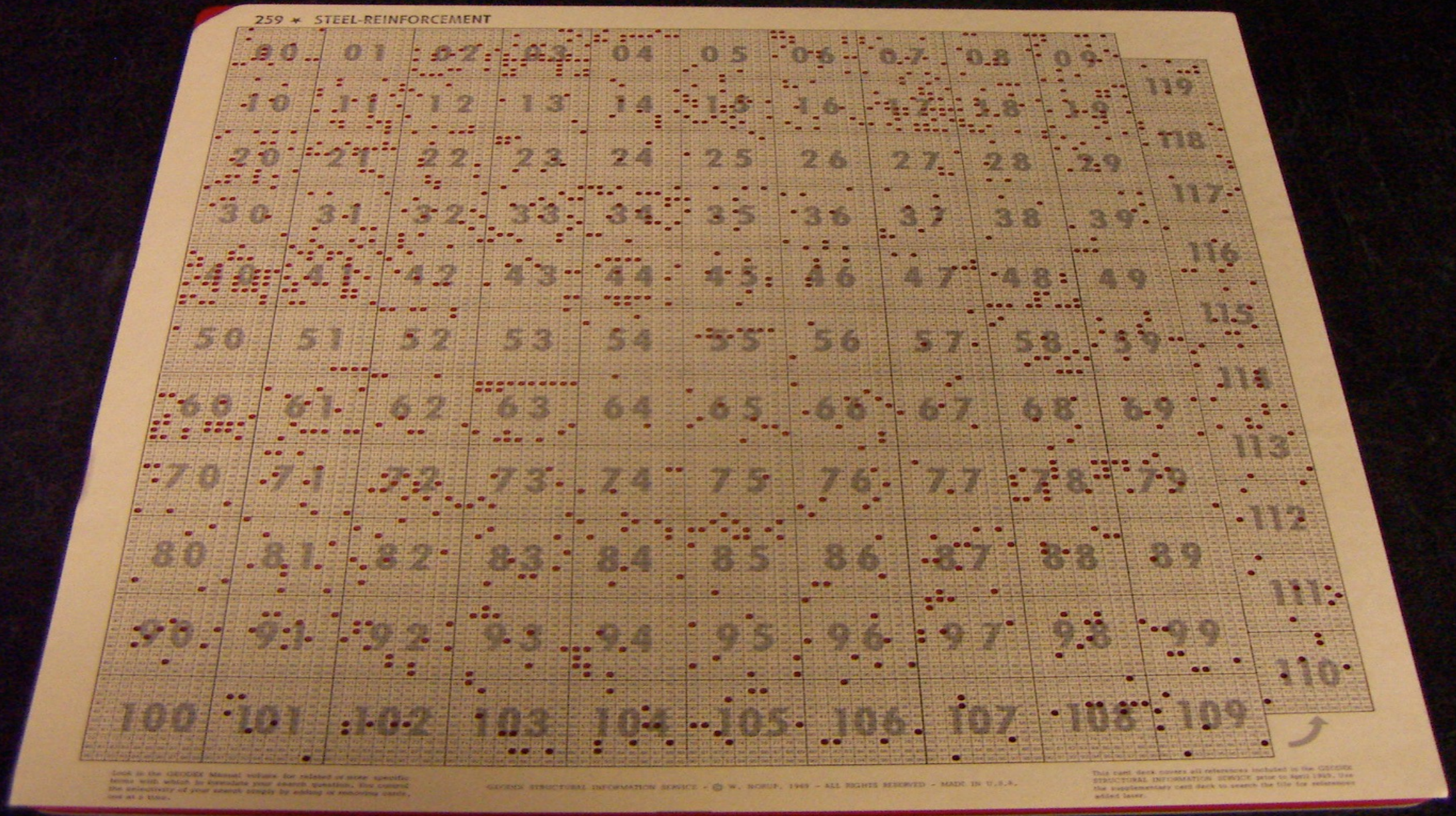


Image courtesy of Deborah Fitchett

Skip Pointers

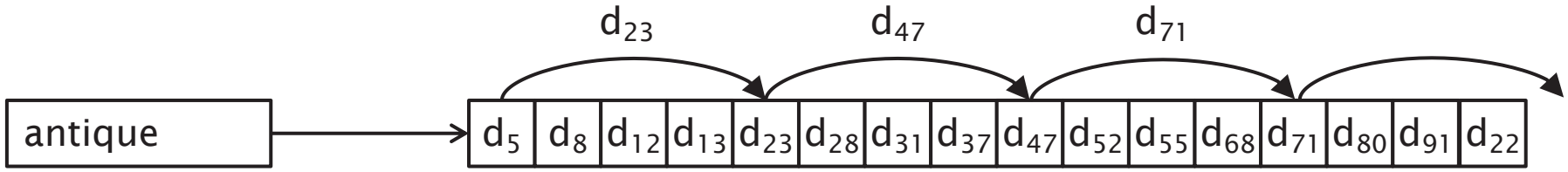
Calculating the intersections of postings lists is critical to efficient query evaluation

If postings lists are ordered by document, we can walk through postings lists simultaneously (cf. merge sort)

- For postings lists A and B, $O(|A| + |B|)$ operations, where $|A|$ is the number of entries in A

Can improve on this by providing short cuts that allow irrelevant postings to be ignored – *skip pointers*

A reasonable heuristic: for a list A of size $|A|$, provide $\sqrt{|A|}$ skip pointers



Document Length Normalisation

In a large collection, document sizes vary widely

- Large documents are more likely to be considered relevant

Adjust answer set ranking – divide rank by document length

- Size in bytes
- Number of words
- Vector norm

University of Southampton

Positional Indexes

If postings include term locations, we can also support a term proximity operator

- term1 /k term2
- effectively an extended AND operator
- term1 occurs in a document within k words of term2

When calculating intersection of posting lists, examine term locations

Biword Indexes

Can extend boolean model to handle phrase queries by indexing biwords (pairs of consecutive terms)

Consider a document containing the phrase “electronics and computer science”

- After removing stop words and normalising, index the biwords “electronic computer” and “computer science”
- When querying on phrases longer than two terms, use AND to join constituent biwords

k-gram indexes

Approaches considered so far assume that we're only allowing queries on full words

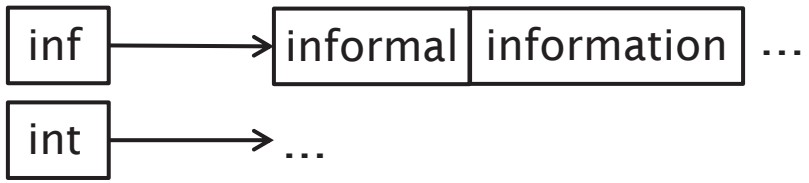
- What about wild card or partial queries: info*

Create new index that includes all k-character subsequences of the indexed words

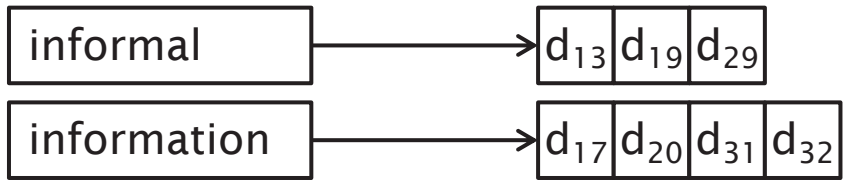
- k-gram index points at entries in the lexicon
- For k=3, "information" yields "$in", "inf", "nfo", "for", "orm", "rma", "mat", "ati", "tio", "ion", "on$"

Decompose query similarly: "info*" yields "$in" AND "inf" AND "nfo"

Postfiltering step on lexicon entries



k-gram index



inverted index

University of Southampton

Vector Model

University of Southampton

Beyond Boolean

In the basic boolean model, a document either matches a query or not

- Need better criteria for ranking hits (similarity is not binary)
- Partial matches
- Term weighting

Binary Vector Model

Documents and queries are represented as vectors of term-based features (occurrence of terms in collection)

$$\overrightarrow{d_j} = \langle w_{1,j}, w_{2,j}, \ldots, w_{t,j} \rangle$$
$$\vec{q} = \langle w_{1,q}, w_{2,q}, \ldots, w_{t,j} \rangle$$

Features may be binary:

$w_{m,j} = 1$ if term k_m is present in document d_j,
$w_{m,j} = 0$ otherwise

Weighted Vector Model

Not all terms are equally interesting

- ‘the’ vs ‘system’ vs ‘Berners-Lee’

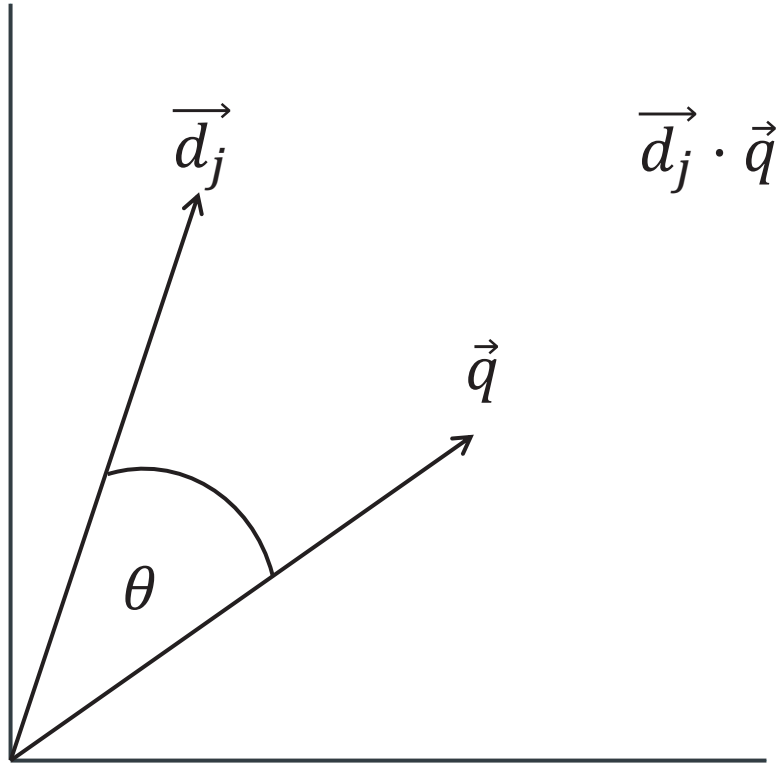
Replace binary features with weights

$$\overrightarrow{d_j} = \langle w_{1,j}, w_{2,j}, \ldots, w_{t,j} \rangle$$
$$\vec{q} = \langle w_{1,q}, w_{2,q}, \ldots, w_{t,j} \rangle$$
$$0 \leq w_{i,j} \leq 1$$

View documents and queries as vectors in multidimensional space

University of Southampton

Vector Model Similarity



$$\vec{d_j} \cdot \vec{q} = |\vec{d_j}||\vec{q}|cos\theta$$

Vector Model Similarity

Similarity can still be determined using the dot product

- Angle between vectors in multidimensional space
- Normalise by dividing by the Euclidean lengths of the vecctors

$$sim(\vec{q}, \overrightarrow{d_j}) = \frac{\vec{q} \cdot \overrightarrow{d_j}}{|\vec{q}||\overrightarrow{d_j}|}$$

$$sim(\vec{q}, \overrightarrow{d_j}) = \frac{\sum_{i=1}^{t} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

Term Weighting

Need a basis for assigning weights to terms in documents

How common is the term in the document?

- Within document measure
- Term frequency

How common is the term in the document collection?

- Collection frequency
- Inverse document frequency

TF-IDF

Term Frequency

$$tf_{i,j} = \frac{\#(t_i \; in \; d_j)}{\sum_k \#(t_k \; in \; d_j)}$$

Inverse Document Frequency

$$idf_i = \log_2 \frac{|D|}{|\{d \in D : t_i \; in \; d\}|}$$

Term Weighting

$$w_{i,j} = tf_{i,j} idf_i$$

TF-IDF Example

Consider the following document collection:

d_1 = “Introduction to Expert Systems”

d_2 = “Expert System Software: Engineering and Applications”

d_3 = “Expert Systems: Principles and Programming”

d_4 = “The Essence of Expert Systems”

d_5 = “Knowledge Representation and Reasoning”

d_6 = “Reasoning About Uncertainty”

d_7 = “Handbook of Knowledge Representation”

d_8 = “Expert Python Programming”

TF-IDF Example

What's the weight of "knowledge" in d_5?

$$idf_{"knowledge"} = \log_2(8/2) = \log_2 4 = 2$$
$$tf_{"knowledge",d_5} = 1/3$$
$$tf.idf_{"knowledge",d_5} = 2/3$$

(assuming that 'and' is removed as a stop word)

University of Southampton

Efficiency

Some observations:

- Most components of a query vector will be 0
- For the purposes of ranking, we're interested in relative rather than absolute similarity
- Computing the cosine similarity between a query and every document is expensive!

- Combine vector model with inverted index with term counts
- Calculate similarity only for documents with non-zero weights for terms

University of Southampton

Query Refinement

Query Refinement

Typical queries very short, ambiguous

- Add more terms to disambiguate, improve

Often difficult for users to express their information needs as a query

- Easier to say which results are/aren't relevant than to write a better query

Rocchio Method

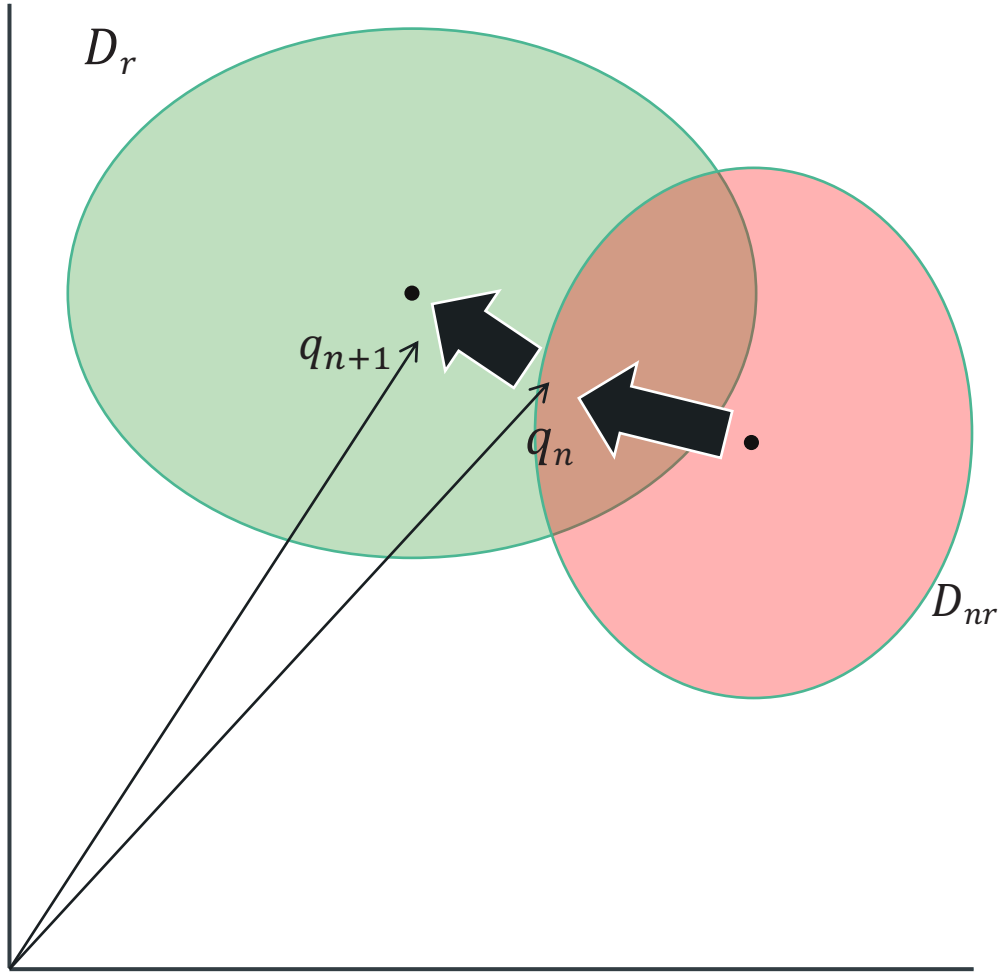
Query refinement through relevance feedback

- Retrieve with original queries
- Present results
- Ask user to tag relevant/non-relevant
- “push” toward relevant vectors, away from non-relevant

$$\overrightarrow{q_{i+1}} = \alpha\overrightarrow{q_i} + \frac{\beta}{|D_r|}\sum_{\overrightarrow{d_j}\in D_r}\overrightarrow{d_j} - \frac{\gamma}{|D_{nr}|}\sum_{\overrightarrow{d_k}\in D_{nr}}\overrightarrow{d_k}$$

$\gamma < \beta$ (typical values: $\alpha = 1, \beta = 0.75, \ \gamma = 0.25$)

Rocchio Method



University of Southampton

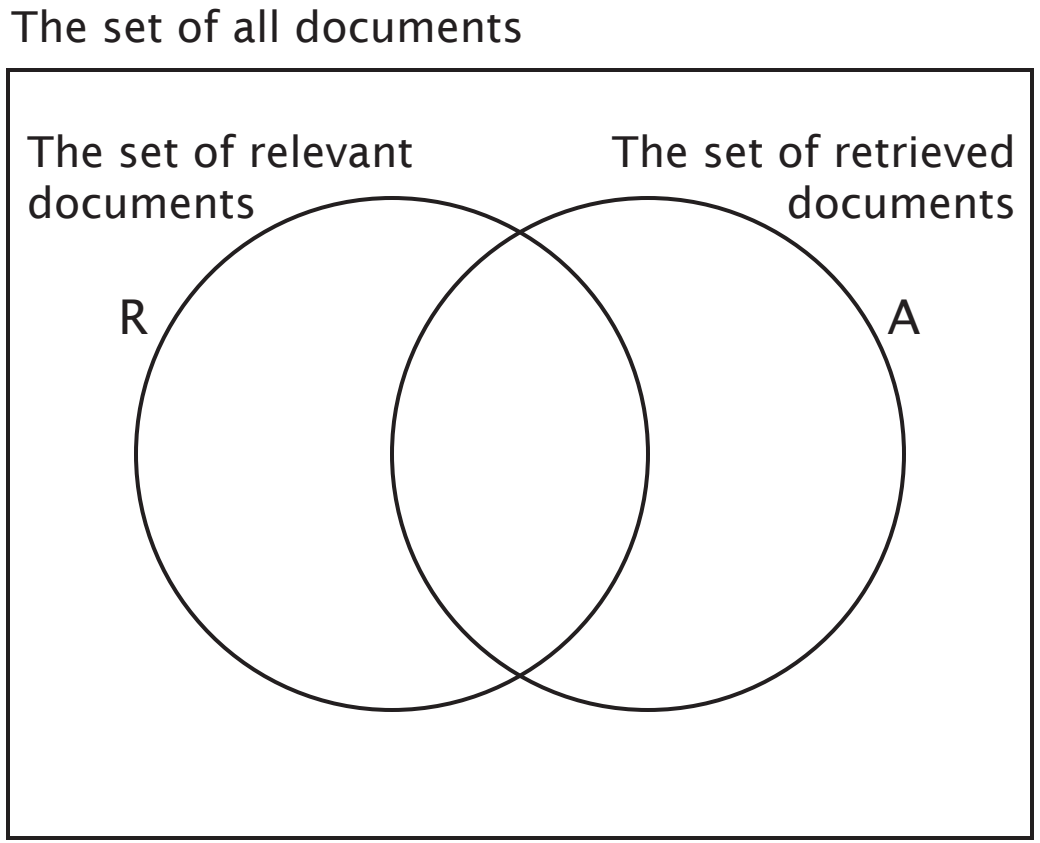
Evaluation

Relevance

Relevance is the key to determining the effectiveness of an IR system

- Relevance (generally) a subjective notion
- Relevance is defined in terms of a user's information need
- Users may differ in their assessment of relevance of a document to a particular query

Relevance versus Retrieved



Precision and Recall

Two common statistics used to determine effectiveness:

Precision: What fraction of the returned results are relevant to the information need?

Recall: What fraction of the relevant documents in the collection were returned by the system?

Precision and Recall

	Retrieved	Not Retrieved	
Relevant	$R \cap A$	$R \cap \neg A$	R
Not Relevant	$\neg R \cap A$	$\neg R \cap \neg A$	$\neg R$
	A	$\neg A$	

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

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

60

Precision and Recall, an alternative view

	Retrieved	Not Retrieved	
Relevant	$R \cap A$	$R \cap \neg A$	R
Not Relevant	$\neg R \cap A$	$\neg R \cap \neg A$	$\neg R$
	A	$\neg A$	

$$Precision = P(relevant|retrieved)$$

$$Recall = P(retrieved|relevant)$$

Positive and Negative Rates

	Retrieved	Not Retrieved
Relevant	True Positive	False Negative
Not Relevant	False Positive	True Negative

$$True\ Positive\ Rate = TP/(TP + FN) = Recall$$
$$True\ Negative\ Rate = TN/(FP + TN) = Specificity$$
$$False\ Positive\ Rate = FP/(FP + TN) = Fallout$$
$$Positive\ Predictive\ Rate = TP/(TP + FP) = Precision$$

Example

	Retrieved	Not Retrieved
Relevant	30	20
Not Relevant	10	40

Collection of 100 documents:

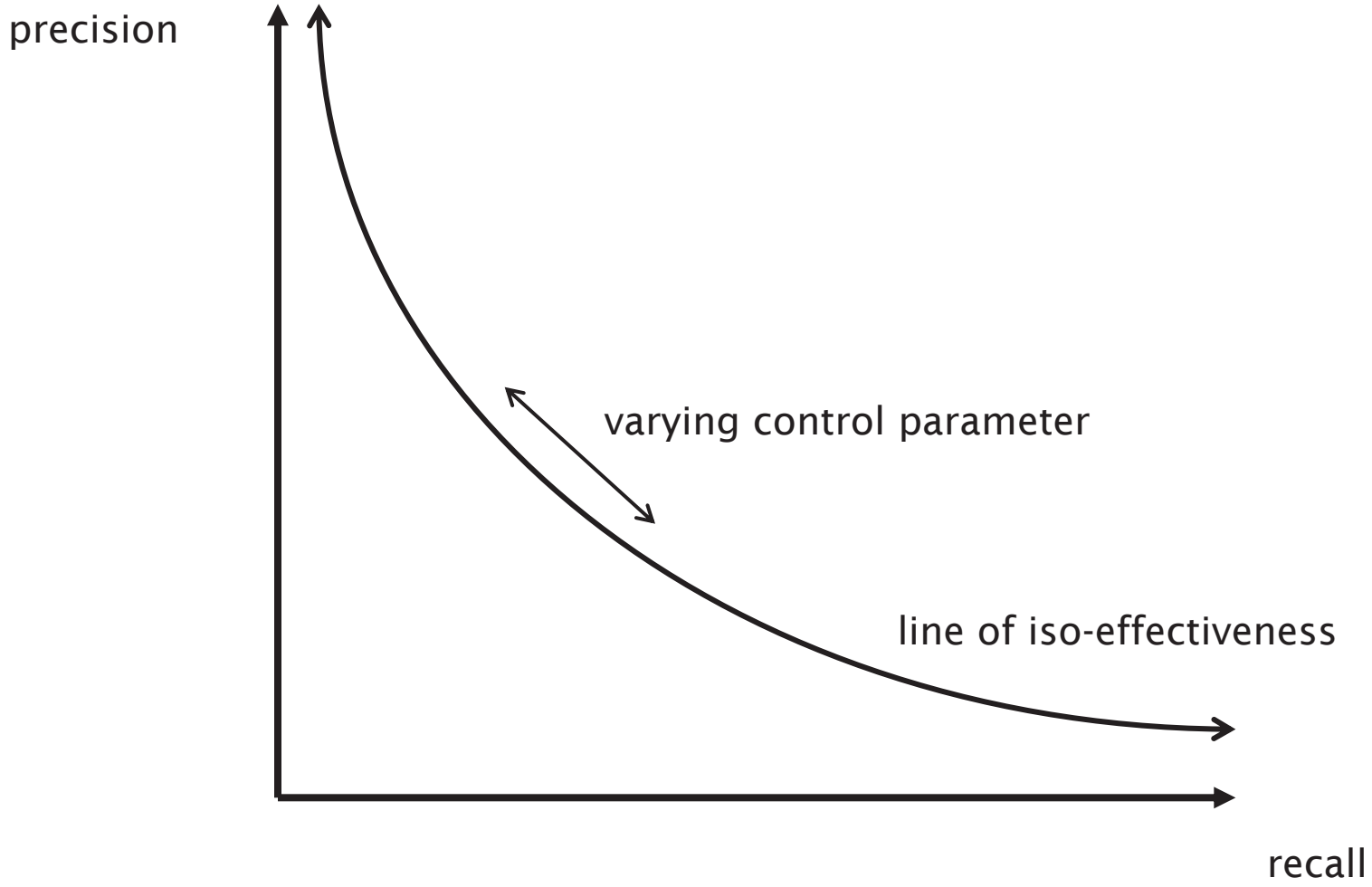
- $Precision = 30/(30 + 10) = 75\%$
- $Recall = 30/(30 + 20) = 60\%$

Precision versus Recall

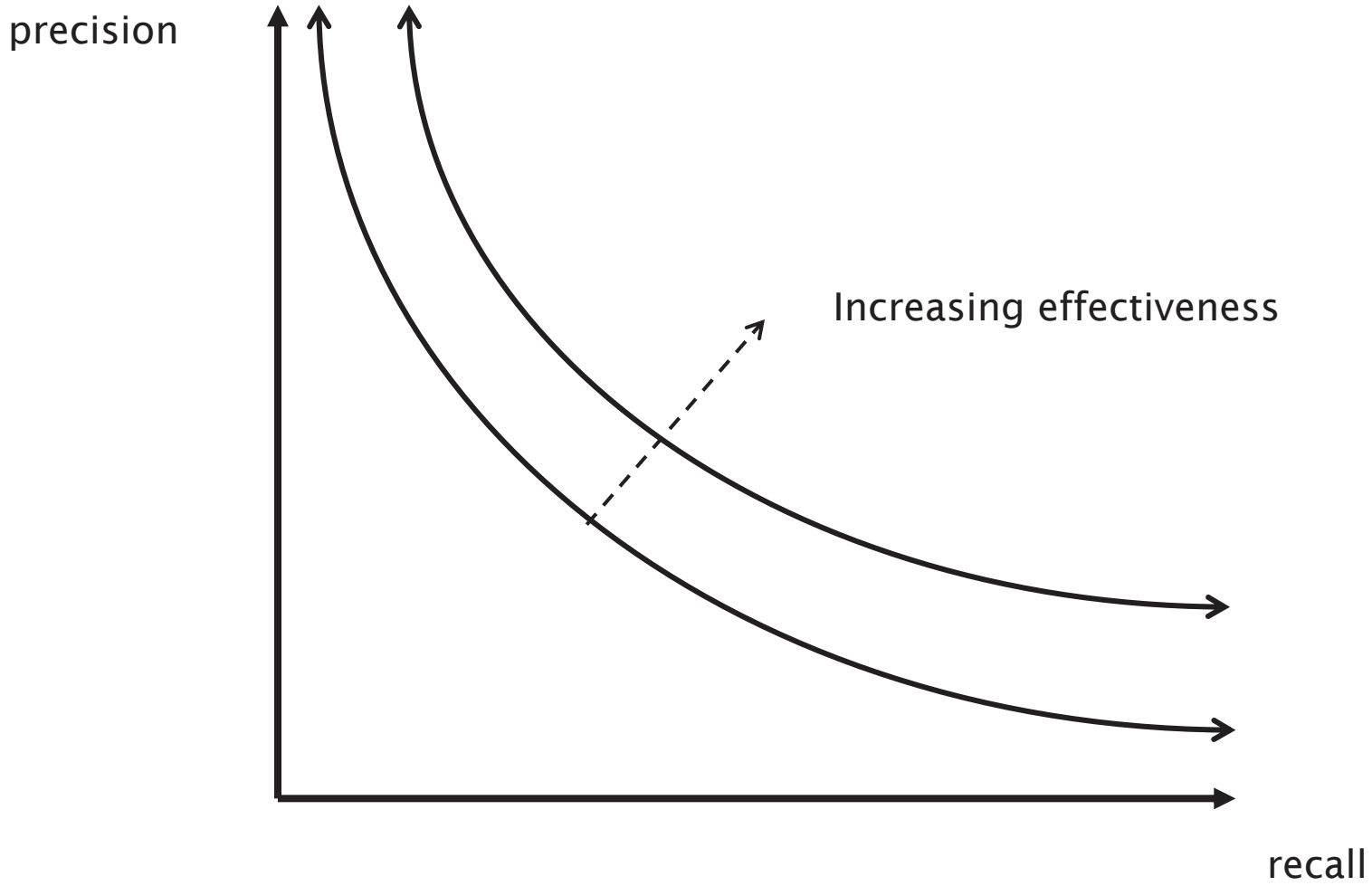
Can trade off precision against recall

- A system that returns every document in the collection as its result set guarantees recall of 100% but has low precision
- Returning a single document could give low recall but 100% precision

Precision versus Recall



Precision versus Recall



Precision versus Recall - Example

An information retrieval system contains the following ten documents that are relevant to a query q:

d1, d2, d3, d4, d5,
d6, d7, d8, d9, d10

In response to q, the system returns the following ranked answer set containing fifteen documents (**bold** are relevant):

d3, d11, **d1**, d23, d34,
d4, d27, d29, d82, **d5**,
d12, d77, d56, d79, **d9**

$$P(r_j) = \max_{\forall r > r_j} P(r)$$

University of Southampton

Precision versus Recall - Example

Calculate curve at eleven standard recall levels:
0%, 10%, 20%, ... 100%

Interpolate precision values as follows:

$$P(r_j) = \max_{\forall r > r_j} P(r)$$

where r_j is the j-th standard recall level

Precision versus Recall - Example

Calculate curve at eleven standard recall levels:
0%, 10%, 20%, ... 100%

If only **d3** is returned

- Precision = 1 / 1 = 100%
- Recall = 1 / 10 = 10%

Precision versus Recall - Example

Calculate curve at eleven standard recall levels:
0%, 10%, 20%, ... 100%

If **d3**, d11, **d1** are returned

- Precision = 2 / 3 = 67%
- Recall = 2 / 10 = 20%

Precision versus Recall - Example

Calculate curve at eleven standard recall levels:
0%, 10%, 20%, ... 100%

If **d3**, d11, **d1**, d23, d34, **d4** are returned

- Precision = 3 / 6 = 50%
- Recall = 3 / 10 = 30%

Precision versus Recall - Example

Calculate curve at eleven standard recall levels:
0%, 10%, 20%, ... 100%

If **d3**, d11, **d1**, d23, d34, **d4**, d27, d29, d82, **d5** are returned

- Precision = 4 / 10 = 40%
- Recall = 4 / 10 = 40%

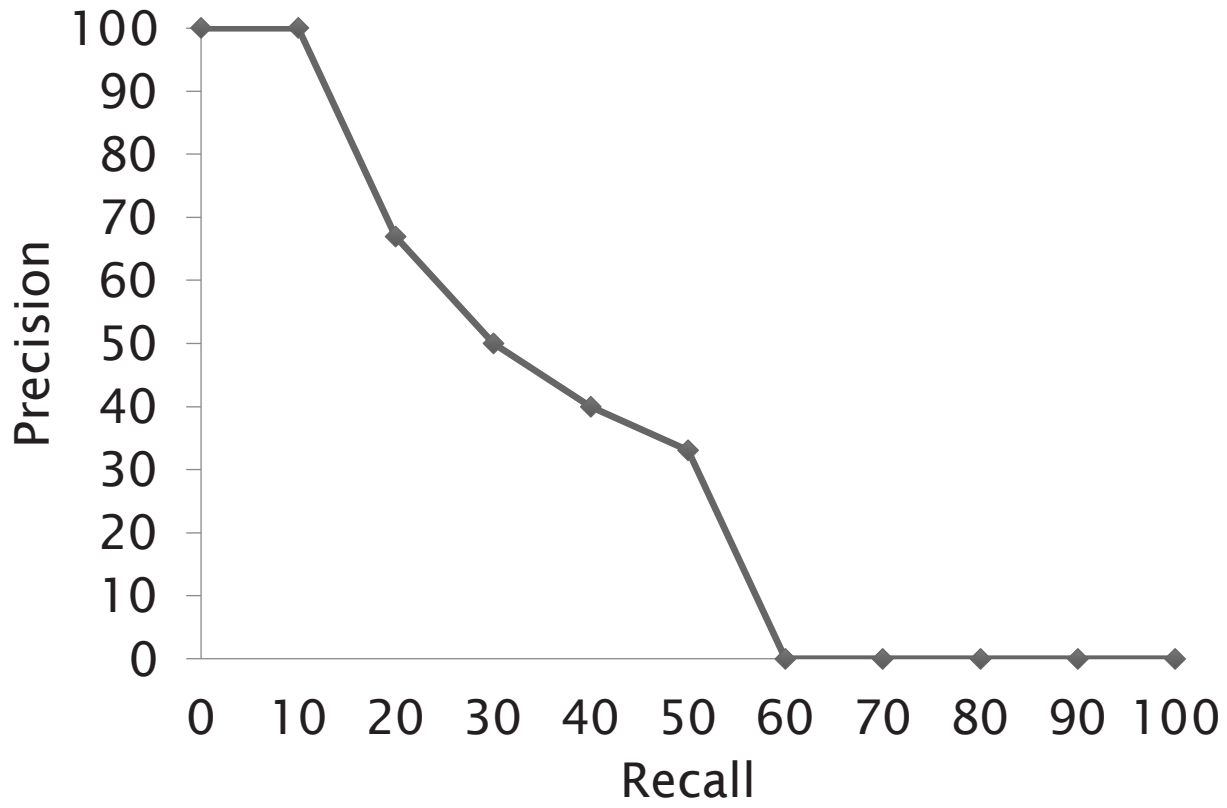
Precision versus Recall - Example

Calculate curve at eleven standard recall levels:
0%, 10%, 20%, ... 100%

If full answer set is returned

- Precision = 5 / 15 = 33%
- Recall = 5 / 10 = 50%

Precision versus Recall



Accuracy

$$Accuracy = (TP + TN)/(TP + FP + TN + FN)$$

- Treats IR system as a two-class classifier (relevant/non-relevant)
- Measures fraction of classification that is correct
- Not a good measure – most documents in an IR system will be irrelevant to a given query (*skew*)
- Can maximise accuracy by considering all documents to be irrelevant

F-Measure

Weighted harmonic mean of precision and recall

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

$$\beta^2 = \frac{1-\alpha}{\alpha}$$

Balanced F-Measure (F1) equally weights precision and recall

- $\alpha = 1/2, \beta = 1$
- $F1 = 2\,(precision\ x\ recall)\ /\ (precision\ +\ recall)$

Average Precision at N (P@N)

For most users, recall is less important than precision

- How often do you look past the first page of results?

Calculate precision for the first N results

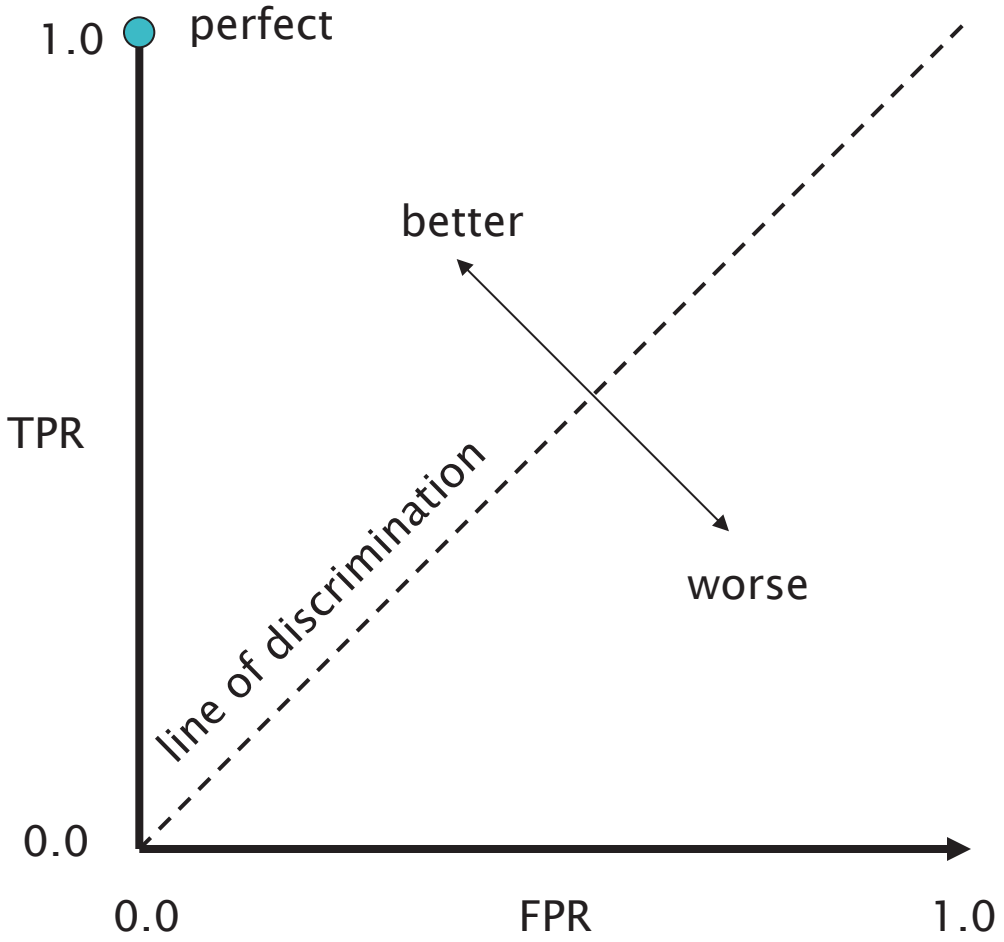
- Typical values for N: 5, 10, 20
- Average over a sample of queries (typically 100+)

Receiver Operating Characteristic

Plot curve with:

- True Positive Rate on y axis (Recall or Sensitivity)
- False Positive Rate on x axis (1 – Specificity)

Receiver Operating Characteristic Curve



Other Effectiveness Measures

Reciprocal rank of first relevant result:

- if the first result is relevant it gets a score of 1
- if the first two are not relevant but the third is, it gets a score of 1/3

Time to answer measures how long it takes a user to find the desired answer to a problem

- Requires human subjects

University of Southampton

Moving to Web Scale

University of Southampton

Scale

Typical document collection:

- 1E6 documents; 2-3 GB of text
- Lexicon of 500,000 words after stemming and case folding; can be stored in ~10 MB
- Inverted index is ~300 MB (but can be reduced to >100 MB with compression)

University of Southampton

Web Scale

Web search engines have to work at scales over four orders of magnitude larger (estimate of 1.1E10+ documents in 2005)

- Index divided into k segments on different computers
- Query sent to computers in parallel
- k result sets are merged into single set shown to user
- Thousands of queries per second requires n copies of k computers

Web search engines don't have complete coverage (Google was best at ~8E9 documents in 2005)

Extended Ranking

So far, we've considered the role that document *content* plays in ranking

- Term weighting using TF-IDF

Can also use document context – hypertext connectivity

- HITS (Hyperlink-Induced Topic Search)
- Google PageRank

Hyperlink-Induced Topic Search

Assign all documents a hub score and an authority score

- A document which links to many pages has a high hub score
- A document which is linked to by many pages has a high authority score

Calculated iteratively:

1. $\forall p \in P, hub(p) \leftarrow 1, auth(p) \leftarrow 1$
2. repeat
   1. $\forall p \in P, auth(p) \leftarrow \sum_{q \in p_{to}} hub(q)$ where p_{to} is the set of the pages that link to p
   2. $\forall p \in P, hub(p) \leftarrow \sum_{q \in p_{from}} auth(q)$ where p_{from} is the set of pages that p links to
   3. $\forall p \in P, auth(p) \leftarrow auth(p) / \sqrt{\sum_{q \in P} auth(q)^2}$
   4. $\forall p \in P, hub(p) \leftarrow hub(p) / \sqrt{\sum_{q \in P} hub(q)^2}$

Hub/auth scores are query-dependent – calculated at query time

PageRank

Unlike HITS, uses only a single score to estimate importance of document

- More important documents are linked to more often
- Links from more important documents carry more weight when calculating importance

$$pagerank(p) = \frac{1-d}{|P|} + d \sum_{q \in p_{to}} \frac{pagerank(q)}{|q_{from}|}$$

(d is a damping factor, typically 0.85)

PageRanks are query-independent – calculated at index time

University of Southampton

Next lecture: NoSQL Databases