Dublin City University

School of Computing

CA4009: Search Technologies

Section 3: Text Retrieval

Gareth Jones

October 2016

Introduction

The purpose of an *information retrieval* (IR) system is *to satisfy a user's information need*.

The IR system seeks to locate document *relevant* to this information need.

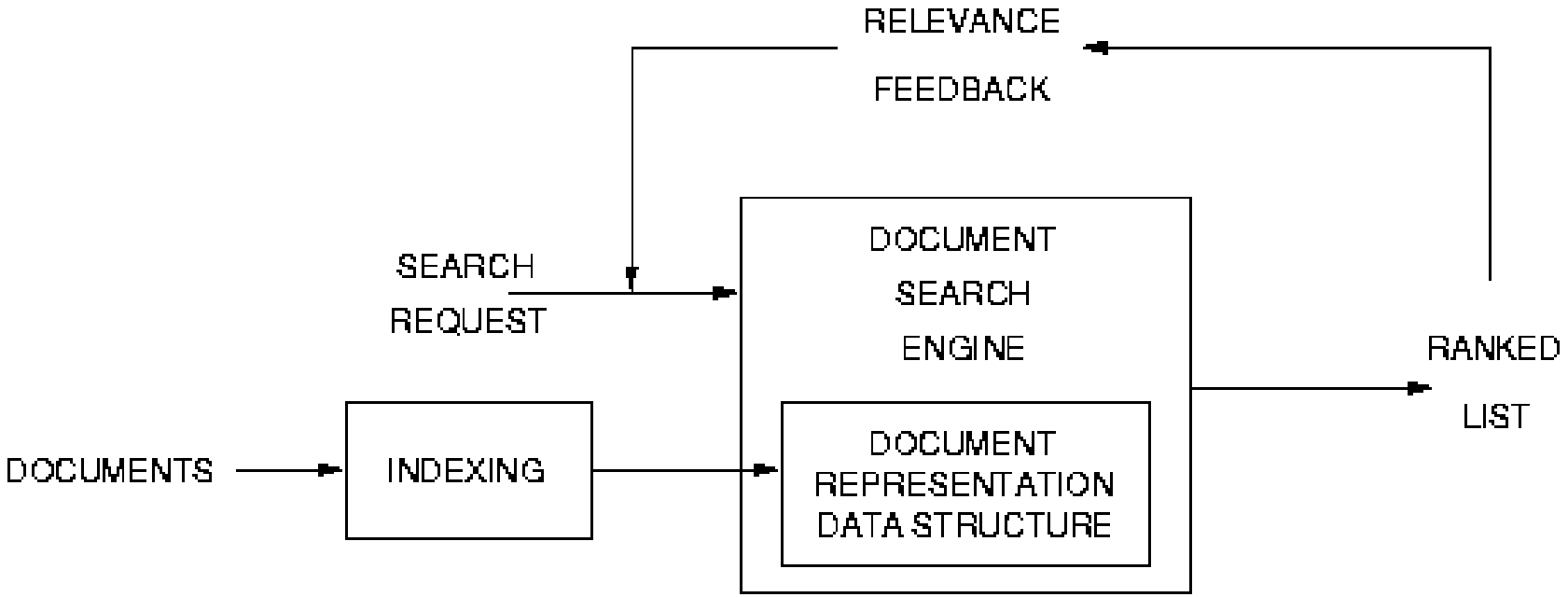
An IR system attempts to do this based on the relationship between the contents of the user's query and each potentially relevant document which is available in a collection.

Many other factors can be used as well to determine which document the users might find most relevant or useful.

Use of some of these factors will be introduced later in the module, including hypertext structures, popularity with other searchers, quality of content, recency of updates, diversity of the contents delivered to the user, ...

Introduction

In this section, we focus on the fundamental challenge of document retrieval based on using a user search request to identify potentially relevant documents based on the contents of the documents.



Components of a standard text information retrieval system.

Components of an IR System

- Document Collection:
  - published or professional documents, e.g. business reports, newspaper articles.
  - web documents, collected using a web spider, crawler,
  - soical media contents, e.g. tweets.

  The documents usually need to be preprocessed to standard format, e.g. case conversion, removal of SGML/HTML or other markup, stop word removal, stemming.
- Document Indexing:
  - convert documents into file structure for rapid access, e.g. inverted file.

Components of an IR System

- Search Request:
  - enter search request expressing user information need - apply similar text preprocessing as used for the documents .
- Document Searching:
  - calculate set of potentially relevant documents, and return to user, usually ranked by some score indicating likelihood of relevance, based on of the query with each document.
- Relevance Feedback:
  - modify search, e.g. expand query using extra search items, based on relevance data; and run the search again.

Introduction

Topics to be Covered in this Section

This section will introduce the following topics in information retrieval:

- indexing - text preprocessing
- file structures
- retrieval models: Boolean, best-match (ranked retrieval)
- relevance feedback
- distributed information retrieval
- multilingual information retrieval
- evaluation of information retrieval systems

The Nature of Text

Before considering how to perform IR, it is important to consider the general nature of text.

- Text is composed of **word tokens** taken from a surprisingly small vocabulary, e.g.
  - 10GB of web documents typically has around 160,000 unique words.

  Each token independently conveys some (often ambiguous!) meaning.
- Word **morphology** (e.g. tense, single/plural) is changed as words are combined into phrases and further combined into sentences when used in natural language.

  For each language, there is a grammar of syntactically allowable word combinations to which phrases and sentences should conform.

The Nature of Text

- Properly formed sentences are constrained to the grammar to which they should conform.
- There is no way of enforcing the use of the accepted grammar of the language.
- In formal publications an editor can seek to ensure this, but may fail to do so or may deliberately choose to break rules to create an effect with the use of language.
- There will also (almost inevitably!) be a number of typographical errors (spelling mistakes), even in published texts.

The Nature of Text

- Sentences are in turn concatenated to make longer elements of **prose** which make up **documents**.
- Sufficiently large documents may be organised with structural elements, e.g. sections, chapters.
- These can be organised using hierarchies to ease navigation through the document.
- Documents may be linked via **hypertext** structures, as described in Section 2..
  - The linking of documents in this way has important implications for search of web and other content types.

The Nature of Text

For IR, we are concerned with the retrieval of elements of content from a collection of content items.

An important question is what should constitute a search item.

Content may consist of:

- one single large structured document, e.g. an encyclopedia
- very many independent documents, e.g. news articles, legal or medical reports.
- very many connected documents,e.g. *wikipedia*, the web.

The Nature of Text

Many collection of content items may not fit easier into of these categories, e.g. a collection of documents will often be partially connected.

People may disagree over the most appropriate connections that should be included.

In many situations it is not obvious what element should be retrieved,

- e.g. would you want to retrieve a complete encyclopedia or a whole website or a single webpage from the site?

Variations in Text Style

Texts are not the same:

- classical literature vs. modern prose
  - changes in vocabulary over time
  - standardised spellings are actually quite a modern feature of many languages
- spoken vs written English

There are many sub-languages, e.g.

- weather reports
- legal documents
- emails, blogs, microblogs (tweets), SMS

Variations in Text Style

Written text can take make forms, e.g.

- *technical documentation*: terse, tight prose, complex phrases and sentences because topic is complex.
- *journalism*: newspaper articles, often simple short easy to read sentences (will depend to some extent on your newspaper!)
- *storybook prose*: can be complex, but makes it difficult to read ... should be easy to assimilate, read for entertainment.
- *email messages*: ungrammatical, full of abbreviations, dialects and slang ... `c u l8r!`

Variations in Text Style

- *office memos*: grammatically correct, but not as complex as technical documentation.
- *formal language*: e.g. wills, deeds, legal documents, agreed domain specific use of vocabulary and grammar is vital.
- *Web pages*: wide variations in style and quality ... basically all of the above styles may appear.
- *blogs, microblogs*: informal varied style, very terse short statements.

Written vs Spoken Text

The structure and content of spoken language is very different from manually written content.

Consider the difference in language use between your own writing and speaking.

This has implications for how we search and interaction with spoken content in a retrieval system.

What document unit should we search for?

What is the appropriate document unit to search for?

- Strictly only Individual files?
- What about an email file, consisting of many, sometimes chained. emails?
- What about email attachments?
- Converting a single Powerpoint file to HTML produces multiple webpages, what should we search for now?

What issues should we consider when deciding search units for a particular type of data?

What document unit should we search for?

The answers to these questions are often not straightforward, and may depend on the application in which the IR system is being used, or its context of operation.

The answer may be:

- a single file,
- multiple files dividing the content into smaller units,
- or perhaps both (dividing content in multiple ways a the same time in this way means that there can be more than one way of retrieving the same information).

Computational Processing of Natural Language Text

Text can be processed in many ways:

- word processing - concerned with presentation, NOT content!
- compression/encryption.
- acquisition - speech recognition, OCR.
- spell checking, stylistic checking.
- string searching.
- automatic natural language understanding
- machine translation.

Computational Processing of Natural Language Text

The style and complexity of electronic prose will affect the type and degree of automatic processing which can be applied.

For example, if the style is highly structured and predictable, it is much easier to perform high quality machine translation, than if the language is used in a highly creative way.

Text IR or search is another application of text processing.

Components of an IR System

Assume that the content collection to be searched has been divided into some agreed form of search units - the documents to be searched, e.g. news articles, web pages.

Assume also that we are building an IR system for English language documents.

Components of an IR System

A typical IR system will contain the following entities:

- Documents (or pointers to them!).
- List of *terms* – processed words used to index the Document contents
- Stored representations of the documents – generally representing full open vocabulary text.

  Note in early IR systems storage limitations restricted representations to document titles/abstracts and/or a restricted keyword vocabulary.

Components of an IR System

Note:

1. This model does not take into account the user, i.e. the response of the IR system is the same for all users.
2. Assumes that the contents and characteristics of the document archive are static.

In many operational IR systems the data collection will be changing frequently, e.g. the list of pages and their contents indexed by a web search engine will be changing very frequently.

Indexing - text preprocessing

- The effectiveness and reliability of IR can be improved by careful content preprocessing.
- The large volume of data to be preprocessed generally means that preprocessing must be entirely automatic.
- Requirement for consistency of preprocessing (see the discussion of the Cranfield experiments in Section 1) also favours automatic indexing.
- Multiple stages of language specific automatic indexing are needed to ensure accurate and efficient retrieval, i.e. they will be different for English, French, etc.

Automatic Indexing

Documents can only be accessed via information stored about them.

The most simple approach would be to store the document contents and search these for matches with the words in the search request.

There are various reasons why this approach is not suitable:

- The complete contents may be too large to store conveniently or the material may be copyright.
- Semantically identical words may be instantiated differently in requests and documents, e.g.
  - synonyms
  - different word forms,

  where this occurs a simple match between the words will fail.

Automatic Indexing

If it were possible to perform a *semantic* analysis of the documents and requests these problems might well not be significant.

In this scenario we would “understand” the contents of the documents and the request, i.e. we would know exactly what the documents are about and what the user is looking for.

We would then somehow compare this information between the documents and the requests to select the relevant documents.

Unfortunately, this is a) not currently possible in terms of natural language processing technology, and b) even if it were possible, would probably be computationally too expensive to be practical.

Automatic Indexing

Beyond this, requests typically do not provide enough information to know what the user is looking for.

IR is typically underspecified - the user's request doesn't give us a precise description of what they are looking for.

So we have to do as well as possible with the information available.

An exception to this is generally when the request is in the form of a question looking for a specific piece of information, "What is the capital of Ireland?"'

Fortunately, cheaper and simpler statistical methods better suited to standard text IR are available and functional effectively.

Automatic Indexing

- Extracting indexing units from text is referred to as *tokenization*.
- Tokenization is the process of chopping up and processing text strings into units that can form the basis of indexing units for retrieval.
- A *token* is a sequence of characters in a document that are grouped together into a meaningful semantic unit.
- For IR an indexing unit may be a real “word”, but it may not, e.g. it may be a phrase or a part of a word.

Automatic Indexing

Typically we process an extracted token in some way to form an indexing unit for IR.

A processed token to be used for search is referred to as a search *term*.

- A term unit used for retrieval is included in an IR system's dictionary.
- Tokens are preprocessed to form search terms. But note -
  - This preprocessing may involve no change in the unit.
  - Tokens may be deleted and not be indexed (stop word removal) - see later.

Automatic Indexing

Processing of tokens to form search terms is often non-trivial.

- For specific domains there are unusual tokens that we may wish to recognize as terms.

  For example, `C++`, `C♯`, aircraft names such as `A*)` or a television show such as `M*A*S*H`.

  Also, computer terminology such as email addresses `jblack@gmail.com`, URLS `http://www.dcu.ie`, numeric IP addresses (`142.32.48.231`), etc.

- What should we do about these?

  Delete them? Delete individual characters from them? Implications?

  Keep them? Implications?

Automatic Indexing

The key to token processing to form terms is to standardise the token in both the documents and queries to encourage matching of related terms with the objective of maximising retrieval performance.

The original token in the document and query may be identical, but often they are not.

Terms are intended to enable matching where the token is are closely related, but the tokens are different, which can occur for many reasons.

For clarity, we can distinguish between a search “request” entered by the user and the processed search “query” compsed of sesrch terms created by the preprocessing of the search request.

Term Utility in Retrieval

The objective of an IR system is to satisfy an user's information need, and it makes sense to seek to do this as effectively as possible.

This is most likely to be achieved if relevant documents are placed at the highest possible positions in a list of retrieved items, i.e. the ones that the user is most likely to look at first.

The ranking of documents depends on calculating the likely relevance of each document based on comparison of the query and the documents.

The question then arises: what terms should be used in this calculation and how they should be used?

The question really is, how can we used in terms to distinguish likely relevant documents from non-relevant ones.

Term Utility in Retrieval

We can refer back to early work on text analysis.

i) Considering how often a terms occurs in each document.

“It is hearby proposed that the frequency of word occurrence in an article furnishes a useful measurement of word significance.” (Luhn, 1957)

i.e. if a word occurs more often than expected in an article, this reflects emphasis on the part of the author about the topic of this word.

Term Utility in Retrieval

ii) Considering how frequent a term is in a set of documents.

A good indexing word helps us discriminate one subset of a collection as relevant to a request, e.g. consider the usefulness of the terms “artificial intelligence” in a collection of documents about artificial intelligence, vs a general collection on computer science.

Good indexing words reflect a relationship between an individual document and the collection from which it might be selected.

In practice, good keywords for retrieval are not the most frequent or the rarest, but rather those occurring a moderate number of times.

Term Utility in Retrieval

Let f be the frequency of occurrence of words in a document and r be the their rank order (the order of their frequency of occurrence).

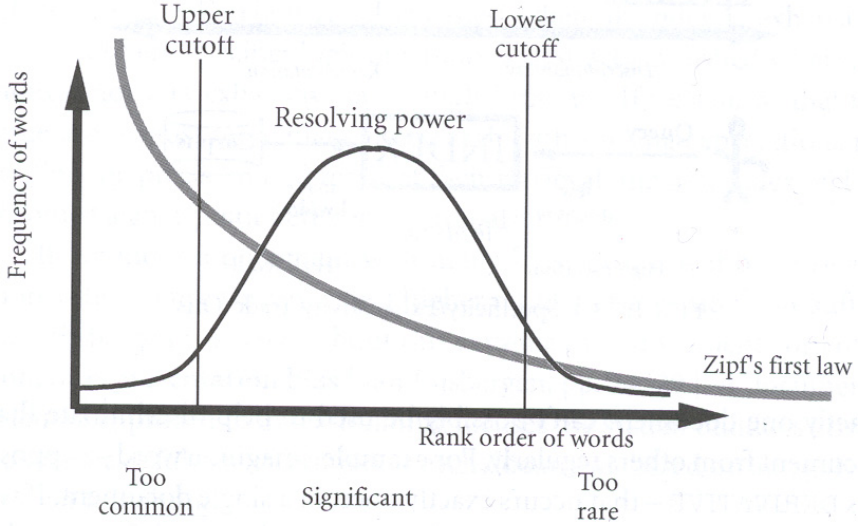


FIGURE 3.3 Resolving Power

Zipf's Law states that: $frequency(f) \times rank(r) = constant$.

Searching should use terms with maximum resolving power.

Stop Word Removal

Frequent words are shown to have low resolving power and intuitively are not of much benefit to the IR process,

They can thus be removed during indexing.

This is referred to as *stop word* removal.

Stop words are typically prepositions and conjunctions – *the*, *of*, *in*, *a*, *and*, *to*, *an*.

A list of stop words is usually chosen for a particular IR system by experiment.

Some standard lists are available (van Rijsbergen 1979).

Stop Word Removal

Removing stop words has two advantages:

- The computer memory requirement for document representations can often be reduced by more than 50% by removing only a small set of words.
- Can improve search efficiency since there is no matching of stop words which a user has included in their query.

Note: These words will not interfere with the retrieval process. Why not? Consider this point again later taking account of *term weighting*. Is this really important?

Conflation

A conflation procedure is designed to bring together words that are related to each other in some way.

We apply conflation methods to form standardised search terms.

The two main classes of conflation algorithm are:

- *stemming algorithms*
- *string-similarity measures*.

In both cases, the resemblances are based on measures of character and character-substring similarities.

After conflation search words are referred to as search *terms*.

Conflation

Some examples of the need for term conflation are as follows:

- Incorrect spellings: LINEAR and LYNEAR
- Alternative spellings: SULPHUR and SULFUR
- Multi-word concepts: DATABASE and DATA BASE
- Transliteration: TCHEBYSHEFF and CHEBYSHEFF
- Affixes: SINGLE and SINGULAR
- Abbreviations and contractions: APPROX and APPROXN
- Historical changes: ARITHMETIC and AERITHMATICKE

Most of these problems can be addressed by string-similarity measures.

If the strings are similar they are probably related. What rules might be used for the above examples?

Affixes

Affixes are often for matching different forms of words.

Some examples of common affix types are as follows:

Suffixes:

- CARRY, CARRIES
- ANALYSIS, ANALYTICAL
- INVERT, INVERSION
- FORGET, FORGETTING
- ABSORB, ABSORPTION

Prefixes:

- COGNITION, RECOGNITION
- POSSIBLE, IMPOSSIBLE
- COMPOSE, DECOMPOSE

While all these words are clearly related, we would generally regard only those differing in their suffix as semantically identical.

Stemming Algorithms

Stemming algorithms are designed to remove suffixes that inhibit matching.

A stemming algorithm operates by:

- matching the ending of a word against a suffix dictionary,
- removing any suffix that is identified,
- checking whether any context-sensitive rules apply.

The algorithm requires manual or automatic creation of a dictionary of suffixes, e.g. -ATION, -ING, -SES, etc.

In operation each word in an input text is compared against the suffix dictionary.

Stemming Algorithms

Two approaches to comparison:

- Longest-match: In a longest-match search, a word such as ALARMINGLY is stemmed to ALARM if both -INGLY and -LY are in the dictionary.
- Iterative search: In an iterative search, a word such a ALARMINGLY is first stemmed to ALARMING by removal of -LY and then to ALARM by removal of -ING.

Porter stemming dating from 1980 is an iterative methods and is still the most popular algorithm currently in use.

Implementations of Porter stemming in several programming languages are available for various natural languages, including English and French.

Stemming Algorithms

It is not always appropriate to apply stem removal.

When the word ends in a stem. For example, potential invalid stemming actions can be detected by means of a minimum stem length, e.g. -ATE and -ING would not be removed from CREATE and SING, respectively.

Alternatively context-sensitive rules may list exceptions to the general stem removal rules, e.g. rather than saying ”do not stem to less than four letters”. we could say to not stem the word SING.

Stemming Algorithms

Recoding can be used to cover changes in spelling that occur in addition to removal of a suffix in order to enable matching of different word forms, e.g. INVERT and INVERSION or FORGET and FORGETTING, could appear as stems as INVENT or FORGET.

Manual evaluation of stemming algorithms suggests that around 95% of words are processed to give reasonable stems.

Changing stemming rules to correct stemming errors tends to introduce different errors.

Retrieval experiments suggest that retrieval where word stemming is employed is generally more effective, but not greatly.

Stemming Problems

The following typical types of errors can be found in stemming algorithms:

Errors leading to unrelated words being stemmed to the same term.

Word	Stem
MEDICAL	MED
MEDIA	MED
MEDIAN	MED

Over- and under- stemming of words are semantically related.

Word	Stem
GASES	GAS (correct)
GAS	GA (over)
GASEOUS	GASE (under)

Information Retrieval for Other Languages

Most IR techniques were originally developed for English, since the majority of electronic texts were in English.

The amount of non-English electronic available has increased greatly in recent years, as a results of which interest emerged in developing effective retrieval for many other languages.

The IR needs for French, Spanish, German are probably obvious, but China and Japan are huge markets for IT as well and there are rapidly growing markets in India and the Arabic speaking world as well as other countries.

Also, consider lesser used languages, e.g. Basque in Spain, Flemish in Belgium, or Welsh or Irish.

Information Retrieval for Other Languages

In order to achieve IR in other languages, IR techniques developed for English need to be adapted to new languages.

Most IR algorithms actually make no underlying assumptions about the document language (see later in this section) and need little or no adaption for use with different languages.

However, the word level tokens and grammars of languages are highly varied, meaning that the preprocessing tokenisation and term creation need significant modification for IR in other languages.

Language Dependent Issues

Consider some examples of other languages.

French and Italian can be processed in a similar manner to English.

But we need to develop language dependent stop word lists and conflation algorithms.

French:

- PERSON, PERSONNES
- DEUX, DEUXIEME
- AL, AUX

Italian:

- VOTARE, VOTAZIONI
- DELLA, DELLE
- RISULTARE, RISULTATO

Language Dependent Issues

German, Dutch:

- Productive languages with respect to compounding.
  - rare (or unique!) compounds in documents and requests are unlikely to match.
  - useful search terms must be extracted by decompounding.
- Conflation their component words ideally dictionary-based.
- If there is no dictionary entry for a particular compound word, search the letter string of the compound for known constituent words.
  e.g. *Abendnachrichtensendungen*: Abend Nachricht Sendung
- Similar approach can be used for Dutch.

Language Dependent Issues

Chinese, Japanese:

- *agglutinating* languages, that is there are no spaces between words.
- Various indexing options:
  - extract single characters
  - n-gram character strings
  - segment using morphological analysis

Example of Japanese script

開始時間が午前１０時の日経ビジネススクール

日経 = nikkei (Chinese characters (kanji))

ビジネススクール = bizinesu sukuuru (phonetic transliteration)

Japanese has no notion of person (*I am* = *you are* = *they are* = *desu*) or singular or plural.

Language Dependent Issues

Arabic:

- Semitic language writing is right-to-left.
- alphabet of 28 letters; extended to about 90 using feature marks.
- complex inflection and morphology, suffix stripping is difficult.
- no space between some words and pronouns.
- prefixes and suffixes can be conjunction of two, three or four grammatical tokens.
- inconsistent transliteration of proper nouns.

Example of Arabic script

1. لكن الانفجار احدث فجوة كبيرة في الحافلة وتشاهد برك من الدم في المكان.

لوس انجليس = Los Angeles

Information Retrieval for Other Languages

In addition, two other issues can be considered in the area of multilingual search:

- Users may wish to access information in languages with which they are not familiar.

  We can use machine translation to cross the language barrier by translating queries or documents (which one is best? why?) in what is referred to as *cross-language information retrieval*.

- Document collections may contain information in more than one language. Retrieving from collections in more than one language is sometimes referred to as *multilingual information retrieval*.

Automatic Indexing

Since document archives are often large the process of document *indexing* is usually completely automatic.

- The original documents are preserved for retrieval.
- Stop word removal and conflation are merely designed to form effective and efficient representations for retrieval purposes.

Search File Structure

An important feature of an IR system is often the time taken for it to respond to a user's request.

An important factor in determining the speed of system response is the structure of the document representations.

The simplest way to compute the matching score between the query and each document is to compare each query term against each herms in each document - but this will be hugely inefficient.

So, we want a data structure that will minimise the computational cost of the query-document matching operation when a new request is entered.

Search File Structure

The speed of computing the query-document matching score will be maximised by considering only the terms that have been specified in the request.

Constructing a suitable data structure will take more time in preprocessing and indexing than simply tokenising and applying conflation to the content, and takes more space then the original documents.

This is a classic data processing situation where there is a trade off between space requirements, offline processing (indexing) and online processing (searching).

Search File Structure

In this case using more space and more complex offline processing enables faster and computationally cheaper online processing.

This is particularly useful in the case of search systems where the indexing only needs to be performed once[a] to enable rapid processing of the very large number of queries that may be received.

For many years the principal file structure for IR systems has been the *inverted file*.

---

[a] but of course needs to be incrementally updated if new documents are added or existing documents are revised or deleted.

Inverted File Example

Consider a very simple example. Suppose that a document archive consists of just 3 documents represented only by their titles.

D1. Information retrieval systems
D2. Database management systems
D3. Retrieval of information from computer systems

After stop word removal the following unique words remain:

Word	Word No
computer	T1
database	T2
information	T3
management	T4
retrieval	T5
systems	T6

Inverted File Example

Representing the documents in their original form produces:

Document	Contents
D1	T3,T5,T6
D2	T2,T4,T6
D3	T1,T3,T5,T6

Representing the documents in an inverted file organisation produces:

Term	Document
T1	D3
T2	D2
T3	D1,D3
T4	D2
T5	D1,D3
T6	D1,D2,D3

Matching documents are identified using the list of terms.

Inverted File Example

Storing Term Locations

Many complex or technical concepts and many names are multiword compounds or phrases.

It is often useful to be able to search for these as a phrase rather than just look for documents containing the individual words.

Additionally, terms appearing in close proximity are likely to be semantically related (from Luhn's analysis of text documents again in the 1950s.).

Thus if query terms appear close together in one document, they are more likely to be related (in the sense of referring to the same thing) than in another document where they are far apart.

A document where they appear close together is more likely to be relevant to the user than one where they are far apart.

Storing Term Locations

The term data in the inverted file can be extended to include the location of each term within each document.

This can be used to incorporate term proximity into retrieval matching functions.

Storing proximity information can also allow for phrasal searching.

Storing this proximity information means that we can give a higher score to documents where the query terms are closer together than those where they are further apart.

Storing Term Locations

There are various ways of storing location information. Two options are:

Biword indexes

- Consider every word pair in a document as a phrase.
- Treat each word pair as a vocabulary term.
- Build an inverted file which stores details of which documents contain which biwords.
- Process longer phrases by breaking them down into pairs.

Storing Term Locations

- Extended biwords can be used for longer phrases which include stop words,

  e.g. “cost overruns on a power plant” can be parsed to form the biwords “cost overruns” AND “overruns power” AND “power plant”

  This works because words in a phrase will typically always contain the same stop words, so removing them will give the same matching pair.

- Biwords have space implications!

Storing Term Locations

Positional Indexes:

Storing Term Locations

- Rather than store phrases in an inverted file, in this case locate phrases when the query is entered.

  Why is this a good idea?

  Why might it be a bad idea?

Hybrid Schemes

- Some phrases can be efficiently looked up at search time. This will be the case if the individual terms are relatively rare, e.g. “Britney Spears”.
- Other phrases consisting of common words, e.g. the name “John Smith” are better stored in a biword index.

Efficiency of Term Lookup

The efficiency of term lookup up is vitally important for use of inverted index files in IR.

Sequential string comparison of all words in the order in which they first occurred when processing the documents, i.e. the order in which they were added to the inverted file, will be very inefficient.

Some improvement can be gained by sorting the vocabulary alphabetically, and using an array to indicate the first location of a word with each leading letter.

Efficiency of Term Lookup

A much more efficient method is to compute term locations using a hashing algorithm or/and hash table.

The hashing algorithm should fulfil the following conditions:

- it must be repeatable;
- ideally have an even distribution;
- and minimise synonyms. Where synonyms occur string matching may be required in the inverted file to identify the correct search term.

Efficiency of Term Lookup

Preprocessing Summary

In summary,

- the search archive consists of document representations,
- (stop words removed and terms conflated)
- in an inverted file structure.

The inversion process is time consuming and computationally expensive, as well as expensive to maintain when the document collection changes.

- Consider what would be required to add or delete documents from an inverted file.

Retrieval Models for IR

Information retrieval researchers have developed a number of distinct models which seek to enable searchers to find relevant documents.

Operationally the main distinction is between Boolean and best-match searching models.

Early IR systems used a Boolean model, which was found to be effective for expert searchers such as librarians.

More recent work has focused on best match search models which, as we shall see, are better suited to non-experts such as web searchers.

Retrieval Models for IR

- A number of distinct approaches to best-match search have been developed.
- The best known are: the Vector-Space Model, the Probabilistic Model, Language Modelling and the Divergence from Randomness model.
- We will look at the Vector-Space Model and the Probabilistic Model.
- Many current IR systems are actually hybrids of Boolean and Best-Match searching.

Boolean Text-Retrieval Systems

Early IR systems were based on a Boolean retrieval model using AND, OR and NOT logical operators.

For example, retrieve documents which satisfy the following conditions: They contain: “information” AND “retrieval”, or “information” OR “retrieval”

Traditionally Boolean systems were almost always used by trained intermediaries (librarians) who chose the database to be searched, identified terms for inclusion in the query, formulated and submitted queries, refined them based on initial results and actually printed retrieved items for users or gave them details of a reference.

Boolean systems generally require the user to interactively build complex queries by incremental refinement.

Boolean Text-Retrieval Systems

An essential piece of information for effective operation of a Boolean IR system is the *postings* information.

The postings value of a term is the number of entries that it has in the inverted file, i.e. how many documents within a collection it occurs in.

Recall (see earlier slides in this section) that for IR we are generally interested in terms which occur in a medium number of documents - not too many, and not too few - which have maximum resolving power.

A trained search intermediary can use the postings information to decide which terms might be useful to add to a query to identify documents potentially relevant to the information need.

Boolean Text-Retrieval Systems

(As we will see later, information of the number of documents that a term occurs in is also vitally important for effective retrieval of with the vector-space and probabilistic IR models.)

By choosing an appropriate set of terms based on their understanding of the user's information need and the postings information, the search intermediary attempts to construct an effective query.

Boolean Text-Retrieval Systems

Boolean systems have a number of limitations:

- High complexity of query formulation for multi-concept subjects.
- Lack of control over the size of the output set that is produced.
- No mechanism for ranking the output in order of decreasing likelihood of relevance.
- Once selected for inclusion in the query, all terms are taken to be equally useful in determining relevant documents.

Boolean Text-Retrieval Systems

Let U represent the set of names of all documents stored.

Let D_1 be the set of documents containing a pattern P_1.

Let D_2 be the set of documents containing a pattern P_2.

1. $U - D_1$ is the set of documents not containing P_1. (NOT)
2. $D_1 \cap D_2$ is the set of documents containing both P_1 and P_2. (AND)
3. $D_1 \cup D_2$ is the set of documents containing either P_1 or P_2. (OR)
4. $D_1 \cup D_2 - D_1 \cap D_2$ is the set of documents containing either P_1 or P_2, but not both. (XOR)

Boolean Text-Retrieval Systems

D1. algorithm, information, retrieval

D2. retrieval, science

D3. algorithm, information, science

D4. pattern, retrieval, science

D5. science, algorithm

The Boolean expression “information” is the names of all documents containing the term “information”: $\{D_1, D_3\}$

The expression “information AND retrieval” is

$\{D_1, D_3\} \cap \{D_1, D_2, D_4\} =$

The expression “information OR retrieval” is

$\{D_1, D_3\} \cup \{D_1, D_2, D_4\} =$

Best-Match Searching

Best-match IR corresponds to the situation of a ranked output list pf a search engine with which we are familiar.

Ideally this list is ranked in decreasing likelihood of the document being relevant to the user's information need (as described by their query).

As stated earlier, there are a number of best-match IR ranking models. We look here at the *Vector-Space Mode*l and the *Probabilistic Model*.

The vector-space model and the probabilistic model have quite different derivations, but essentially do the same thing.

The similarity between a document and a request is a function of the number of search terms that they have in common.

Best-Matching Searching

Search *requests* in best-match search can be expressed in natural language which is then preprocessed to form a search *query* which is used for the actual matching.

Example of a search request and corresponding query.

"I am interested in novel matching algorithms for document-retrieval systems. Anything on best-match or nearest-neighbour retrieval, and partial-match searching might also be of interest."

After preprocessing this produces the following search query,

algorithm, best, docum, match (3), near, neighb, novel, part, retriev (2), search, system

Since in this simple form of best-match search we focus only on matching of single terms, the request can just be a list of search terms - as many people enter to web search engines.

Best-Match Searching

Let us consider a simple example based on the query “information science” in the previous example.

If we score the documents based only on the number of terms which match between the query and each document, we have what is referred to as “coordination” level matching, and the following ranked list is produced.

Document	Score
D3	2
D1	1
D2	1
D4	1
D5	1

What is the obvious weakness of this approach?

Best-Match Searching

An effective best-match IR system requires:

- A weighting scheme to reflect the importance of different indexing terms in measuring the similarity.

  Essentially this automates the role of the professional intermediary in determining effective terms for search, and incorporates this information into a ranking of the documents.

- A *matching score* between the query and each document.

  This provides a similarity coefficient to provide a numerical quantification of the degree of resemblance between a query and a document.

  This attempts to provide a meaningful ranking of documents in terms of their potential relevance.

Best-Match Searching

Advantages

- No need for complex Boolean queries - easier for novice users to use!
- Complete control over output size, since documents are ranked.
- Ability to differentiate between terms using weighting schemes (see later).

Limitations

- The absence of phrase and multi-word terms which would be enabled by inclusion of an AND operator.
- The absence of the OR operator precludes the use of synonyms.
- A query needs to contain several terms to generate meaningful rankings.
  - Consider the problem of ranking documents in a web IR system.

Boolean vs Best-Match Searching

- Detailed comparative experiments suggest that there are no significant differences in potential IR effectiveness.
- Experienced searchers may find that Boolean searching gives slightly better results.
- Inexperienced searchers will invariably find that best-match searching gives better results.
- The required computational resources are broadly comparable, but best-match searching requires far less human effort.
- Various hybrids between Boolean and Best-Match searching are possible.

Term Weighting

Simply summing the terms in common between the query and the document, as used in the earlier example, is a very basic approach.

Better retrieval performance can be achieved by the introduction of *term weighting*.

Terms weighting enables terms with high utility in differentiating relevant documents to be emphasised.

The idea behind term weighting is *selectivity*: what makes a term a good one is whether it can pick any of the relevant documents from the many non-relevant ones.

Term Weighting

There are three commonly used sources of weighting data:

- Collection Frequency
- Term Frequency
- Document Length

Collection Frequency

Key concept: Terms that occur in fewer documents are often more valuable than ones which occur in many documents.

Collection frequency weights (also known as *inverse document frequency weights*) are defined as follows for a search term $t(i)$.

Given

$n(i) =$ the number of documents term $t(i)$ occurs in,

$N =$ the total number of documents in the collection archive.

The cfw for term $t(i)$ is

$$cfw(i) = \log \frac{N}{n(i)}$$

The logarithm can be taken to any convenient base.

Term Frequency

This refers to the term's within-document frequency.

Key concept: The more often a term occurs in a document, the more likely it is to be important for that document.

Thus, while a term $t(i)$'s collection frequency is the same for any document, its document frequency varies.

The term frequency for term $t(i)$ in document $d(j)$ is:

$tf(i,j) =$ the number of occurrences of term $t(i)$ in document $d(j)$.

Document Length

Key concept: Document relevance is *independent* of document length.

A term that occurs the same number of times in a short document and in a long document, is likely to be more valuable for the former.

Without compensating for document length, longer documents will tend to have higher matching scores merely because they are long.

Document length of document $d(j)$ is

$dl(j) =$ the total number of term occurrences in document $d(j)$

One length compensation method requires a normalised average document length defined as follows:

$$ndl(j) = \frac{dl(j)}{\text{average } dl \text{ for all documents}}$$

"tf $\times$ idf"

One commonly used empirically derived weighting scheme is often referred to as "tf$\times$idf". (Note: $idf(i) = cfw(i)$.)

$$w(i,j) = f(tf(i,j)) * idf(i)$$

where $w(i,j)$ is the weight of term i in document j. Some $tf(i,j)$ functions,

$$f(tf(i,j)) = tf(i,j)$$

$$f(tf(i,j)) = 0.5 + 0.5\frac{tf}{maxtf}$$

$$f(tf(i,j) = \log(tf(i,j) + 1)$$

where $maxtf$ is the maximum term frequency in document j.

Vector-Space Model

Documents and queries are represented as vectors in a t-dimensional space. where t is the number of unique index terms in the collection.

The degree of similarity between document $d(j)$ and a query q is calculated as the cosine of the angle between the two vectors.

$$\mathbf{a}.\mathbf{b} = |a||b| \cos\theta$$

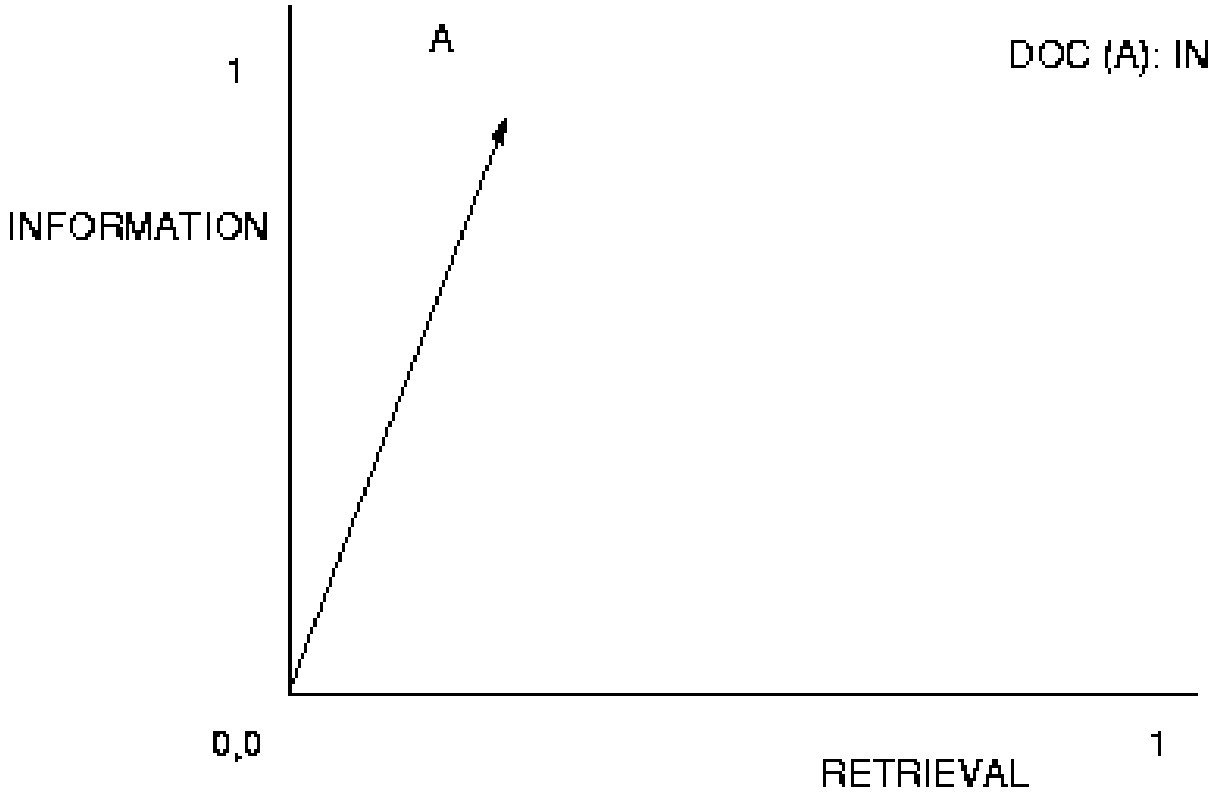
Vector-Space Model

The similarity between the query and each document $d(j)$ can be computed as,

$$Sim(q, d(j)) = \frac{\sum_{i=0}^{I-1} w_q(i).w_d(i,j)}{\sqrt{\sum_{i=0}^{I-1} w_q(i)^2}\sqrt{\sum_{i=0}^{I-1} w_d(i,j)^2}}$$

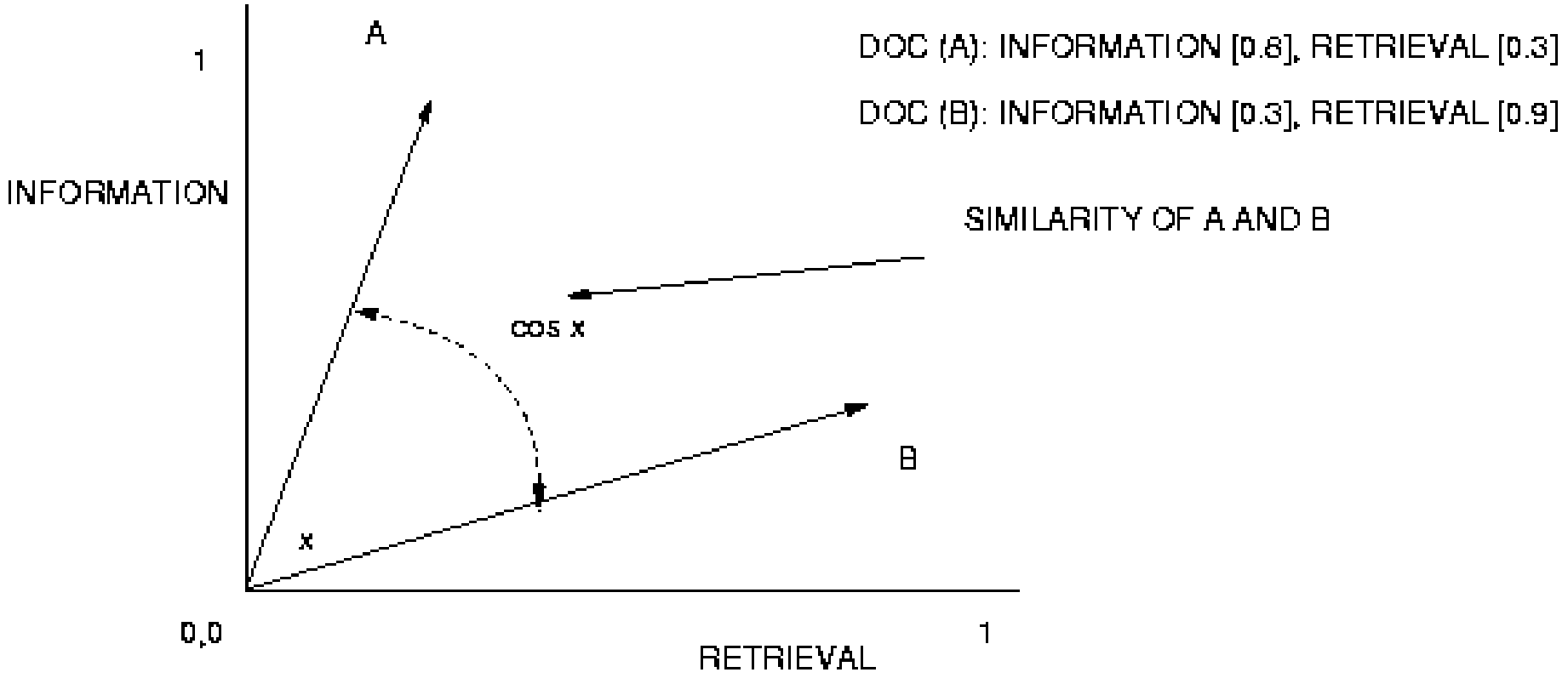
The documents are then ranked in decreasing order of similarity.

Vector-Space Model

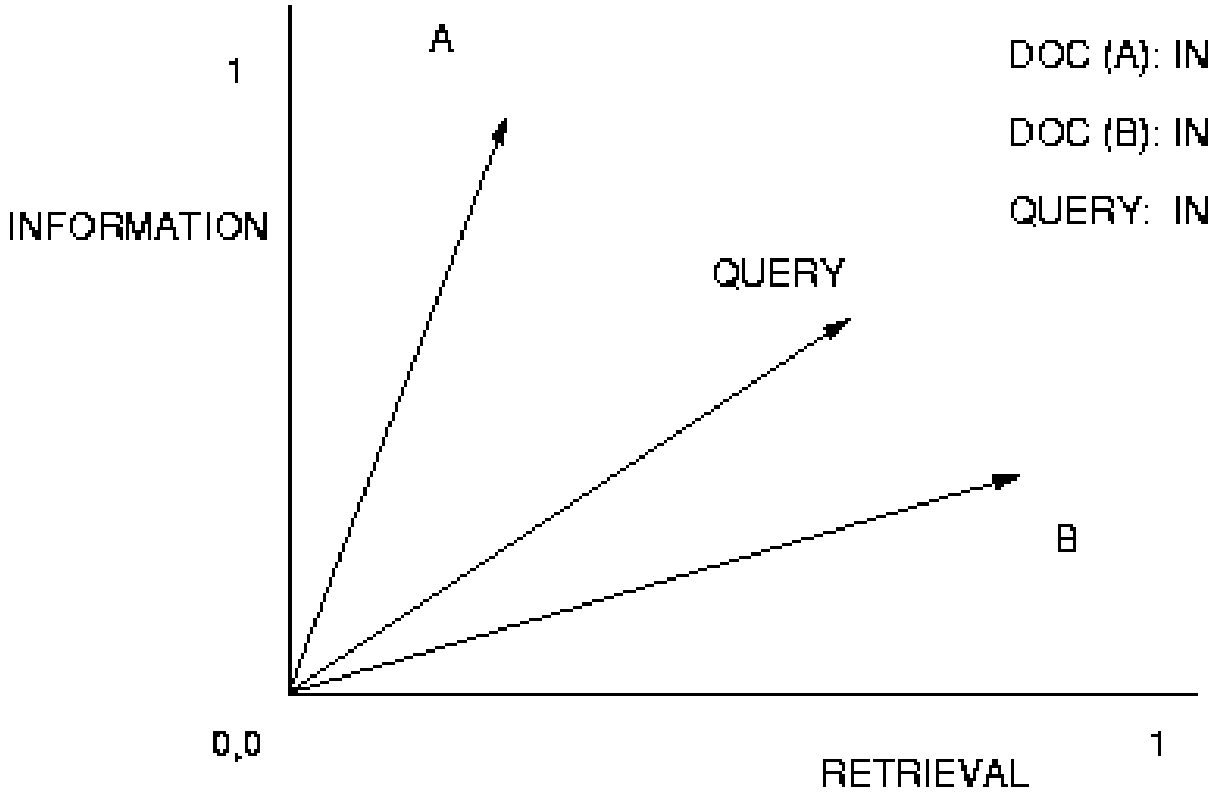


DOC (A): INFORMATION [0.6], RETRIEVAL [0.3]

Vector-Space Model



Vector-Space Model

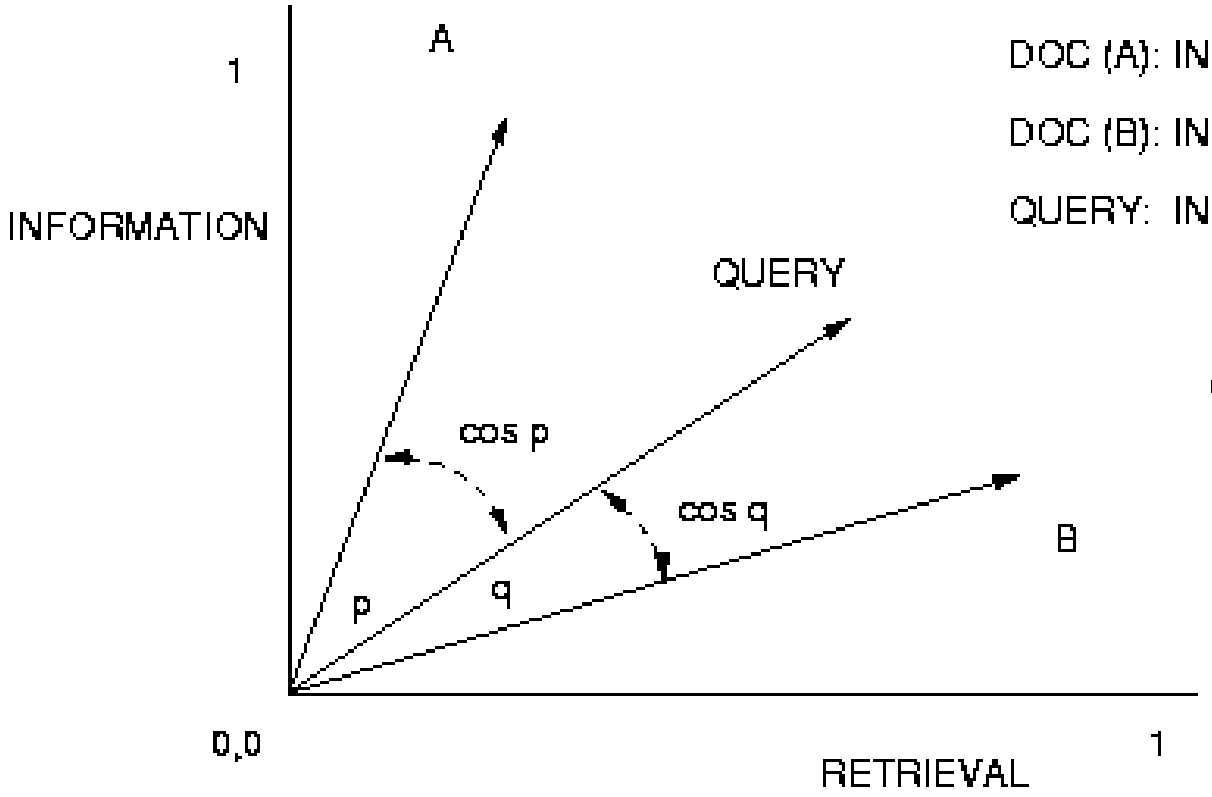


DOC (A): INFORMATION [0.6], RETRIEVAL [0.3]

DOC (B): INFORMATION [0.3], RETRIEVAL [0.9]

QUERY: INFORMATION [0.6], RETRIEVAL [0.7]

Vector-Space Model



DOC (A): INFORMATION [0.6], RETRIEVAL [0.3]

DOC (B): INFORMATION [0.3], RETRIEVAL [0.9]

QUERY: INFORMATION [0.6], RETRIEVAL [0.7]

cos p, q = [0 ... 1]

Probabilistic Model

Although the vector-space model employs frequency counts, its underlying mathematics as a retrieval model are fairly ad hoc.

- The matching score assigned to a document in a result set is not a probability of relevance, but rather a measure that attempts to estimate how much evidence there is in favour of a document being relevant

The probabilistic model of retrieval is an attempt to formalize the idea behind ranked retrieval in terms of probability theory.

It attempts *to compute the probability that a document is relevant to a query*, given that it possesses certain attributes or features, i.e. that it contains certain search terms.

Probabilistic Model

It is generally assumed that:

- each document is either relevant or irrelevant to the query.
- judging one document to be relevant or irrelevant tells us nothing about the relevance of another document.

The *Probability Ranking Principle* states that ranking documents by decreasing order of probability of relevance to a query will yield “optimal performance”, i.e. the best ordering based on the available data.

Probabilistic Model

The probabilistic model query-document matching score $ms(j)$ for a document j can be calculated in a similar manner to the other ranked retrieval schemes that we have already seen.

$$ms(j) = \sum_{i=0}^{I-1} w(i,j)$$

where $w(i,j)$ indicates a probabilistic term weighting scheme, and I is the set of all search terms.

This can be shown as part of the derivation of the probabilistic model to be a valid means of calculating the probability of the relevance of a document to a particular query.

Probabilistic Model

Derivation of a probabilistic model is beyond the scope of this module, but we can still examine and use a derived probabilistic model.

One effective probabilistic model is the *Okapi* BM25 *combined weighting* $cw(i,j)$ scheme developed at City University, London:

$$cw(i,j) = cfw(i) \times \frac{tf(i,j) \times (k_1 + 1)}{k_1 \times ((1-b) + (b \times ndl(j))) + tf(i,j)}$$

where $cw(i,j)$ indicates the combined weighting scheme, $ndl(j)$ is the normalised length of document j, and k_1 and b are experimentally determined constants that control the effect of $tf(i,j)$ and the degree of length normalisation respectively.

Okapi BM25

Okapi BM25

Example

Simple example of document scoring for a search query.

Consider a query consisting of 3 terms: $t(1)$, $t(2)$ and $t(3)$.

Suppose the document content representations are stored in an inverted file accessed via a hash table.

Each of the search terms must first be transformed to the corresponding hash table entry using the selected hash algorithm.

Example

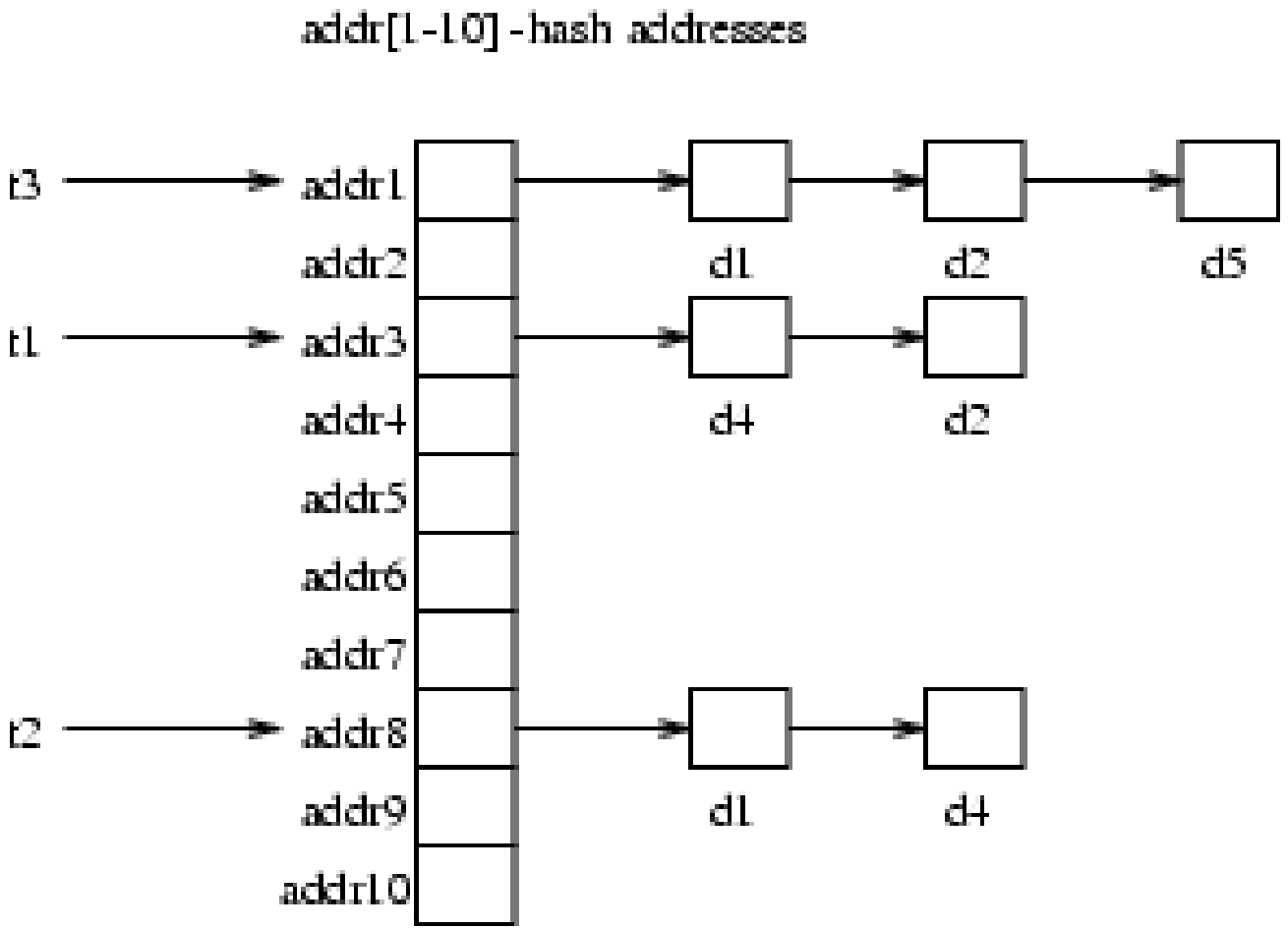
Thus,

$$t(1) \rightarrow x(1)$$
$$t(2) \rightarrow x(2)$$
$$t(3) \rightarrow x(3)$$

In the example,

$$t(1) \rightarrow addr(3)$$
$$t(2) \rightarrow addr(8)$$
$$t(3) \rightarrow addr(1)$$

Example


addr[1-10] - hash addresses
t3
t1
t2
addr1
addr2
addr3
addr4
addr5
addr6
addr7
addr8
addr9
addr10
d1
d2
d5
d4
d2
d1
d4

Example

Let the matching score between the query and document $d(j)$ be $ms(j)$. Initially $ms(j)$ is set to 0.0 for all documents j.

For each term $t(i)$ the hash table entry $x(i)$ is evaluated.

The hash table connects to the term entries in the inverted file structure.

Entries in the inverted file can simply be connected using a linked list where the entry for one document points to the address of the entry for the next document in the list.

Example

When each matching document $d(j)$ in the linked list for term $t(i)$ is reached the matching score $ms(j)$ is increased by $cw(i,j)$, the term weight for $t(i)$ in $d(j)$. Thus working sequentially through $t(1)$, $t(2)$, $t(3)$, the scores for this example are incrementally calculated as follows.

	init.	$t(1)$	$t(2)$	$t(3)$
$ms(1)$	0.0		$+cw(2,1)$	$+cw(3,1)$
$ms(2)$	0.0	$+cw(1,2)$		$+cw(3,2)$
$ms(3)$	0.0			
$ms(4)$	0.0	$+cw(1,4)$	$+cw(2,4)$	
$ms(5)$	0.0			$+cw(3,5)$

Example

The final values of $ms(j)$ will obviously depend on values of $cw(i, j)$ for this document set. But let us suppose that ranked in order of deceasing matching score the system output was as follows:

$$\left.\begin{array}{lll} 1 & d(4) & \rightarrow \\ 2 & d(2) & \rightarrow \\ 3 & d(5) & \rightarrow \\ 4 & d(1) & \rightarrow \end{array}\right\} \text{links to documents}$$

The user is now free to select any of the retrieved documents. A link can access the full original contents of the document.

Example

Note 1: document $d(4)$ does not appear in the list since no terms matched.

Note 2: score $ms(5) > ms(1)$ despite having fewer matching terms, rank order is determined by the term weights $cw(i, j)$.

Importantly, it does not matter here that all terms in the query do not match since the output matching score is merely the sum of the weights which do match.

Example

The likely effect of adding further terms to the query is either: to refine the ordering, add some additional documents in the lower part of the ranked list, or to have no effect if the term is not present in the documents.

This example illustrates retrieval in a simple static document set. In practice this is rarely the case and individual documents may be added or removed at any time.

This means that the inverted file structure must be regularly updated and term weights similarly updated. This process needs to be planned very carefully if management of the inverted file is to be efficient.

Example

The example also assumes a one-to-one mapping between $t(i)$ and $x(i)$. This is rarely (if ever) the case, usually it is many-to-one.

Thus once the $x(i)$ has been located, the system must ensure that it matches only with the correct $t(i)$ entries.

This is most easily accomplished by including the string $t(i)$ in the linked list entry and comparing $t(i)$ with the term string in each entry.

Clearly, this is inefficient and more complex strategies, involving using additional alternative hashing functions, are often used in practice.

Techniques for Improving Information Retrieval

In IR there is often mismatch between the terms that appear in a document and in a query. Essentially the ASK problem from earlier.

For example, the query

“reports on new Smartphones”

will fail to find a document containing only the following relevant fragment

“press releases for the latest mobile computing devices”

In these cases a query will either fail to retrieve all the relevant items, or they will be retrieved a low rank (probably often too low for the user to spend time looking for them!).

Techniques for Improving Information Retrieval

IR researchers have explored a number of methods to modify or reformulate the search query to attempt to overcome this problem.

These can broadly be categorised as *global* and *local* methods.

Global Query Reformulation Methods

Vocabulary Tools for Query Reformulation

The ASK problem means that users are often not able to describe their information need effectively.

After an initial attempt to address the information need, the IR system can provide the user with information which may help them form a more effective reformulated query. The user might be shown:

- Words from their query treated as stop words.
- Stems of the words as used by the IR system.
- No of hits of each word. (The “postings” information.)
- Details of use of words in a phrase.

Global Query Reformulation Methods

Query Expansion and Thesauri

A possible solution to the matching problem is to try to expand the query by adding terms that are related to the original query terms.

A thesaurus describes relationships between words and phrases.

A simple use of a thesaurus method is to add (or suggest synonyms) of the original query word, e.g. “internet” $\rightarrow$ “WWW” “world wide web” “net” “web”

Thesauri may add unhelpful terms, e.g. the inclusion of “web” and “net” may retrieve documents on “spiders” and “fishing”, or may fail to include useful terms, e.g. if we are generally interested in computing networking “WAN” or “LAN” might be useful or perhaps protocols such as “TCP/IP”

Using a thesaurus for query expansion means that no user input is required.

Global Query Reformulation Methods

It is this failure to take into account the context of the term, i.e the way in which it is used in the query, that leads to this being referred to as a *global* reformulation method.

Adding additional words in this way generally leads to improved recall, i.e. more of the relevant documents in the collection may be returned.

However, the failure to take context into account can also lead to the return of many additional non-relevant documents.

The unreliable nature of the expansion terms means that they can be downweighted relative to the original query terms.

Generally thesaurus based query expansion has not been found to be a reliable method for improving precision in IR.

Global Query Reformulation Methods

Query Expansion via Log Mining

The query can be compared to archives of logged queries from other users.

The IR system may suggest similar queries taken from the log archives to the user.

The user can accept the previous query in its entirely, or potential revise their own query by adding or removing words based on the suggestions from the log.

Local Query Reformulation Methods

Local query reformulation methods adjust the query based on the documents retrieved in the current or previous runs for this query.

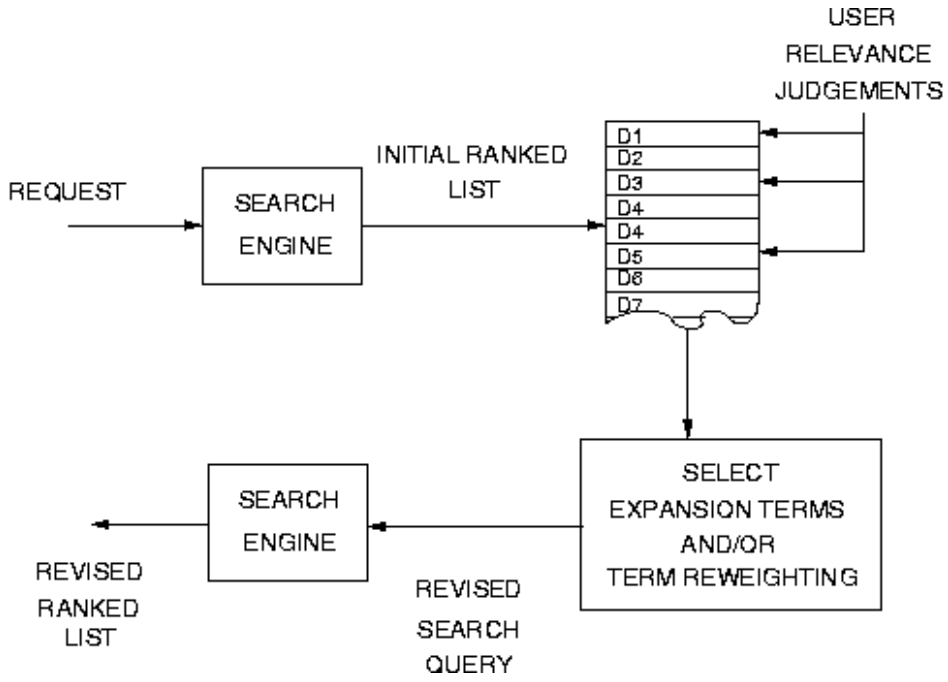
The most popular form of local query reformulation is *relevance feedback*.

Relevance Feedback

Relevance feedback uses relevance information from an initial retrieval run for the current query to:

- add significant terms from relevant documents to the current query.
- modify the weights of terms to improve the ranks of documents which are likely to be relevant.

Relevance Feedback



Relevance feedback can be applied in the case of both the vector-space model and the probabilistic model.

Relevance Feedback

Relevance information can come from three possible sources:

- Explicit feedback: The user can mark documents from the current run as relevant (or non-relevant).
- Implicit feedback: The system can assume that returned documents clicked on by the user are relevant.
  This assumes that the user clicks on documents because they are trying to locate relevant items, and that the snippet summaries (or other item surrogates) are reasonable indicators of relevance to the query.
- Blind or pseudo feedback: The system can simply assume the top ranked documents are all relevant, e.g. the top 5 or 10 returned documents.

Relevance Feedback

Relevance feedback makes several important assumptions:

- The user is assumed to know enough about the topic to make a reasonable initial attempt at a query to retrieve relevant items.
- Relevant documents are assumed to be similar to each other, and that additional unseen relevant documents will be similar to those that are already known.

  Relevance feedback methods assume that this similarity will reveal itself in the form of similar distributions of terms in all relevant documents.

  By implication this means that relevant documents will be dissimilar to non-relevant ones.

Sufficient documents must be judged, otherwise can be unstable.

Relevance Feedback in the Vector-Space Model

Query expansion can be seen as adjusting term weights in the query vector.

Adding a new term to the query corresponds to giving this term a non-zero weight in the vector.

Emphasizing or reducing the importance of a query term corresponds to increasing or decreasing its weight.

Given an initial query vector represented by non-zero weighted terms,

$$Q = (w_1, w_2, \ldots, w_t)$$

the relevance feedback process produces the new vector,

$$Q' = (w'_1, w'_2, \ldots, w'_t, w'_{t+1}, \ldots, w'_{t+k})$$

the weights have been updated and k new terms added.

Relevance Feedback in the Vector-Space Model

The standard relevance feedback for Vector-Space IR was developed by Rocchio. The standard Rocchio query modification is:

$$Q' = \alpha Q + \beta \frac{1}{R} \sum_{x=0}^{R-1} RD(x) - \gamma \frac{1}{NonR} \sum_{x=0}^{NonR-1} NonRD(x)$$

Q' = modified query vector
Q = original query vector
R = no of **known** relevant documents
$NonR$ = no of **known** non-relevant documents
$RD(x)$ = relevant document vector x
$NonRD(x)$ = non-relevant document vector x
α, β, γ = are eimpircally determined constants

Relevance Feedback in the Vector-Space Model

Q' is the vector sum of the original query plus vectors of the relevant and non-relevant documents.

Many variants on Rocchio have been developed.

e.g., investigations have explored:

- reweighting the original terms and not adding new terms.
- only adding the highest weighted new terms (how many?
- only relevant documents to complete Q', i.e. set $\gamma = 0$

What would be the effect of each of these?

Like the vector-space model itself, Rocchio is an empirical method without an underlying formal theory.

Relevance Feedback in the Probabilistic Model

The relevance feedback methods for the probabilistic model are again theoretically motivated.

Term reweighting and query expansion are treated separately.

Computing relevance weights seeks to answer the question:

“How much evidence does the presence of this term provide for the relevance of this document?”

Selecting new terms to add to a query should answer a different question:

“How much will adding this term to the request benefit the overall performance of the search fomulation?”

Relevance Feedback in the Probabilistic Model

Terms are reweighted by replacing $cfw(i)$ by $rw(i)$ (known as the Robertson/Sparck Jones weight).

$$rw(i) = \log \frac{(r(i) + 0.5)(N - n(i) - R + r(i) + 0.5)}{(n(i) - r(i) + 0.5)(R - r(i) + 0.5)}$$

$n(i)$ and N are defined as previously, R and $r(i)$ are new variables.

$n(i) =$ the number of documents term $t(i)$ occurs in,

$N =$ the total number of documents in the collection archive.

$r(i) =$ the number of **known relevant** documents term $t(i)$ occurs in,

$R =$ the total number of **known relevant** documents in the collection archive.

Relevance Feedback in the Probabilistic Model

$rw(i)$ reduces to something similar to $cfw(i)$ if $r(i) = 0$ and $R = 0$, i.e. $cfw(i)$ is essentially $rw(i)$ where is no relevance information available.

The 0.5 constants are added for smoothing, otherwise $rw(i)$ would be set to 0 for terms in the query that have not been observed in any relevant documents so far.

All terms in appearing relevant documents are potential query expansion terms.

These terms are first ranked using the Robertson offer weight $ow(i)$.

$$ow(i) = r(i) \times rw(i)$$

Relevance Feedback in the Probabilistic Model

A number of top ranked terms are then added to the original query.

How many should we add?

Note: Terms with high $ow(i)$ values, and which are thus potentially good expansion terms, generally need both high $r(i)$ and high $rw(i)$.

What does this tell us in about terms which are typically added to a query?

Relevance Feedback in the Probabilistic Model

Query expansion for the probabilistic model also often incorporates other empirical elements.

Original query terms are often upweighted using a scalar constant.

This is done since the original query terms were entered by the user, they can be regarded as more reliable than automatically selected expansion terms.

This is reflected in giving them more significance in computing the query-document matching score, i.e. the relevance feedback $cw(i, j)$ of the original query terms is calculated as $\alpha \times cw(i, j)$. This gives these terms a greater contribution to the overall matching score.
α is typically set between 1.5 and 3.5.

Data Fusion

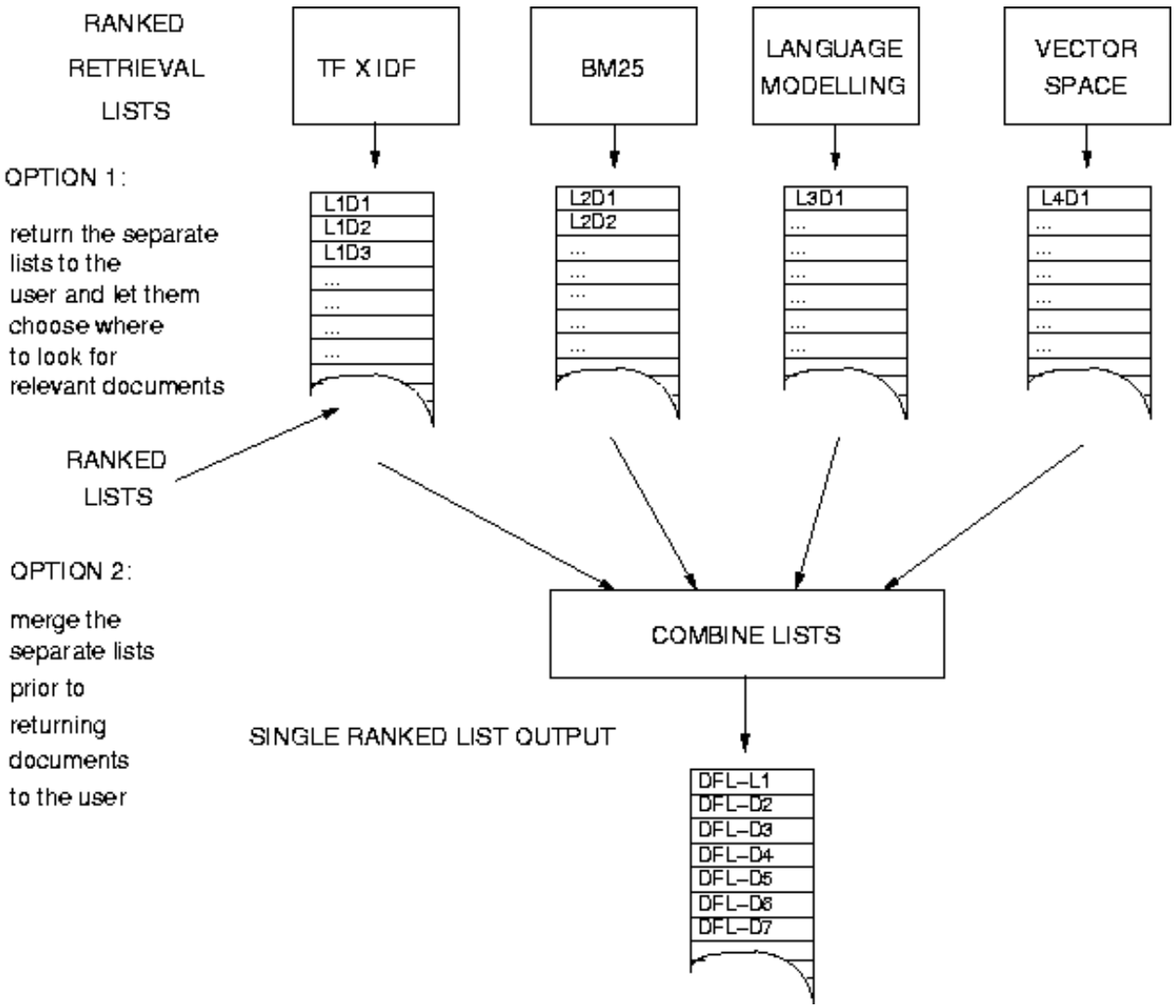
For single collections different term weighting schemes usually produce different ranking of documents.

Improved overall retrieval performance can often be achieved be merging ranked lists generated independently by information retrieval systems using the different term weighting schemes.

In this case the merging process involves adding the query-document matching scores assigned by the different weighting schemes.

For example, a document ranked 3rd by all the IR systems may end up ranked 2nd, if the alternative IR systems each rank a different document at rank 2.

Data Fusion


RANKED
RETRIEVAL
LISTS
TF X IDF
BM25
LANGUAGE
MODELLING
VECTOR
SPACE
OPTION 1:
return the separate
lists to the
user and let them
choose where
to look for
relevant documents
L1D1
L1D2
L1D3
L2D1
L2D2
L3D1
L4D1
RANKED
LISTS
OPTION 2:
merge the
separate lists
prior to
returning
documents
to the user
COMBINE LISTS
SINGLE RANKED LIST OUTPUT
DFL-D1
DFL-D2
DFL-D3
DFL-D4
DFL-D5
DFL-D6
DFL-D7

Data Fusion

The different weighting schemes are likely to produce matching scores which are incompatible in absolute value or possibly in range over the ranked list.

Merging methods:

Raw Score: ignore score incompatibility issues and merge documents based on query-document matching scores.

Normalised Score: for each list divide the query-document matching score of each document by the maximum score for the list (maximum score for each list is then 1.0). Then merge as for Raw Score method.

Many other more complex methods have been explored.

Evaluation of IR Systems

Some important questions in evaluation of an IR system:

1. Does it work correctly? - program testing - really a software engineering issue.
2. Is it useful?
3. Is it better than other systems which attempt to do the same thing?

Evaluation of 2 and 3 must adopt a planned strategy. Should not rely on anecdotal evidence from individual users with individual requests.

i.e., the system should be evaluated over a group of users and sets of queries.

Evaluation of IR Systems

In order to know the how effective an IR system is, i.e. how good it is at retrieving relevant documents and not retrieving non-relevant ones, how much users like it, and how much it enables them to address their information needs, the IR system must be evaluated.

There two categories of evaluation:

- Qualitative evaluation: from user feedback.
- Quantitative evaluation: from user tests and accuracy of document retrieval.

Evaluation of IR Systems

Qualitative evaluation: users impressions, e.g. using interviews, questionnaires, ...

Quantitative evaluation: testing user task completion time (use methods from standard users testing of interactive systems), laboratory-based experiments based on IR *test collections*.

Here we are concerned only with evaluation IR using test collections in a laboratory setting.

Evaluation of IR Systems

A document that satisfies a user's information need is said to be *relevant* to the information need.

The notion of relevance should not be confused with documents scored highly by the IR system.

Relevance can only be judged by humans. Human relevance judgements form the basis of quantitative IR evaluation.

Assumption: if an IR system performs well in a (valid) experimental setting then it will perform well in an operational setting.

Document Relevance

What is a “good” (or relevant) document?

There is a great deal about relevance which is hard (impossible!) to quantify:

1. does the document completely satisfy the initial information need?
2. does it answer part of the information need?
3. does it merely help the user to develop their understanding without really answering the problem, in which case it might be argued to be at least partially relevant to the information need?
4. Is it completely non-relevant to the information need?

Document Relevance

Look back to the ASK model of IR as a cognitive process in the Introduction section.

Documents of relevance class 3. can help with revising a request by informing the searcher about the subject, which might then enable documents of types 1. and 2. to be retrieved in a subsequent retrieval operation.

Document Relevance

In IR evaluation relevance is most easily treated as a binary decision:

- a document is deemed either relevant or non relevant to the request.
- human assessors actually generally prefer to give a graded judgment.
  - assessors judge each document as one of the four classes on the earlier slide.
- a set of graded relevance judgments can be reduced to a binary one by making a cutoff, e.g. only documents of class 1 are labelled relevant, and documents of class 2, 3 and 4 are labelled non-relevant.
  In terms of assessing the relative effectiveness of the IR system, it generally doesn't matter where the cutoff is taken, i.e. a good system is still a good system irrespective of the cut off position.

Questions about relevance

Relevance is a subjective notion. Different users may differ about the relevance or non-relevance of a document to a given search request.

Questions arise about relevance and assessment of relevance:

- The document is relevant to what, *exactly*?
- Judged by whom? When, how, on the basis of what data?
- Subjectively/objectively of relevance judgements?
- Degrees of relevance?
- Individual items/sets of items/sequencing of items?
  Sequencing effect can introduce bias.
- Summary Judgement? – which system is better overall?

Studies of Relevance

Studies of relevance have shown that:

- Even with very carefully defined statements of the information need expressed in a user request, different judges may disagree on the relevance of individual documents.
- On the whole, these disagreements are on the edges, i.e. documents for which it is difficult to decide relevance.
- On average, IR systems show the same relative effectiveness with the slightly different sets of relevance judgements created by alternative assessors, i.e., if a systems is better based on the judgments of one assessor, it will still be better based on the judgments of another assessor, even if the absolute values of the evaluation metrics are different.

Evaluation of IR Systems

For quantitative evaluation of document retrieval by an IR system we need:

- alternative IR systems to be evaluated – these might be completely different or closely related since *any* change in an IR system (e.g. changing a conflation algorithm or stop word list) gives us a different system which should be evaluated separately.
- a *test collection* consisting of:
  - a set of documents from which items are to be retrieved.
  - a set of requests expressing user information needs.
  - relevance data telling us which documents are relevant to each request.

Evaluation of IR Systems

To run the experiment:

- enter the documents into the IR system(s) via indexing;
- put each request to each system;
- calculate the matching score for each document for each request in each system;
- collect the output.

Then,

- evaluate the output for each request, document by document using the relevance data for each request;
- analyse the results.

Measurement of IR Performance

(assuming simple binary relevance categorisation)

In essence, the goal of an IR system is:

1. to retrieve relevant documents;
2. not to retrieve non-relevant ones.

For any request a number of relevant documents may be present in the collection.

Measurement of IR Performance

A measure for 1. is:

$$\text{Recall} = \frac{\text{No of relevant docs retrieved}}{\text{Total relevant docs in collection}}$$

Within the retrieved documents, what proportion of the overall relevant documents have been retrieved?

For a Boolean retrieval system only a single Recall value is computed based on the retrieved documents.

For a Best-Match retrieval system Recall is usually calculated at specific cut off points in ranked order, e.g. what is the Recall after the top 5 scoring documents?

Measurement of IR Performance

A measure for 2. is:

$$\text{Precision} = \frac{\text{No of Relevant docs retrieved}}{\text{Total docs retrieved}}$$

What proportion of the retrieved documents are relevant?

For Boolean retrieval systems a single precision value is computed.

For Best-Match retrieval precision is calculated at selected cut off points in the ranked list. For example, at a cut off of 5 documents how many of the retrieved documents are relevant?

These are the two traditional measurements of IR system performance.

Measurement of IR Performance

A single measure of performance for best-match retrieval is the *Mean Average Precision (MAP)*, calculated as follows.

For each request first calculate the single figure for the *Average Precision for this request*,

$$\text{Average Precision for Request} = \frac{\sum_{\text{all rel. docs}} \text{Precision at rel doc cut off}}{\text{Total no of rel. docs}}$$

Then, calculate the Mean Average Precision over all the test requests.

For Boolean systems, a Mean Precision can be calculated by taking the average of the precision values over all the requests.

Measurement of IR Performance

$$\text{Mean Average Precision} = \frac{\sum_{\text{request set}} \text{Average Precision for Request}}{\text{No of Requests}}$$

Observations:

- In general, the higher the Recall and Precision values the better the IR system.
- It is often found that tuning a system to improve Recall will impair Precision and vice versa.
- Operational systems are usually developed to produce a compromise between Recall and Precision.

Example

Suppose that there is a small collection of 1000 documents.

For a particular request a trial Boolean system retrieves 16 documents of which 8 are relevant.

Suppose also that there are a further 12 documents which are relevant, but NOT retrieved. For this example,

If there are 4 other requests for which the precisions are respectively: 0.57, 0.41, 0.73, 0.48. the Average Precision is:

Example

For the same collection with a Best-Match, suppose the following retrieval behaviour is observed.

Rank	Rel	Precision
1	Rel	
2	Rel	
3	Not Rel	—
4	Rel	
5	Rel	
6	Not Rel	—
7	Rel	
8	Rel	
9	Not Rel	—
10	Rel	

Calculate the Recall and Precision at cut offs of 5 and 10 documents.

Example

At cut off 5 documents in the ranked list.

At cut off 10 documents in the ranked list.

We could equally well choose any other cut off values.

Comparing retrieval performance at defined cut off levels for different systems gives a means of examining variations in behaviour (what variations might you expect to see?), as well as the performance of individual systems.

Example

Mean Average precision for this request would be calculated as follows:

- Work down the ranked list adding the precision value each time a relevant document is found.
- After all retrieved documents have been examined divide the total by the number of relevant documents.

The overall Mean Average Precision for the 5 test requests would then be calculated in a similar manner to that for the Boolean system.

Test Collections

Some considerations in the design of IR test collections are:

- Scale – the number of documents in the collection and the size of the request set.
- Variety of documents - are they from just one source or a variety; would the contents be expected to change over time e.g. news material; is the collection typical of the operational environment of the system?
- Variety of representation of both documents and requests – how are they represented within the retrieval system? (e.g Boolean vs Best-Match)
- Scope of relevance judgements – how have they been gathered, and to what extent can they be trusted?

Test Collections

A key question for a test collection is :

Where do the relevance judgements come from?

Note: They must be gathered manually!

- – Ideally the relevance of each document should be judged for each request.
  - – This relevance judgement should be independent, that is the document is judged either relevant or not relevant irrespective of the relevance of other documents.
  - – For this reason documents are usually presented to the assessor in a randomised sequence.

Test Collections

- – In a large collection (or even a modest one) there are too many documents for an individual assessor to judge the relevance of each document to each request.
  – Since the relevance of all documents cannot be judged for each request, an approximation to the relevance set for each query is often obtained from the *pooled* output of a number of search engines.

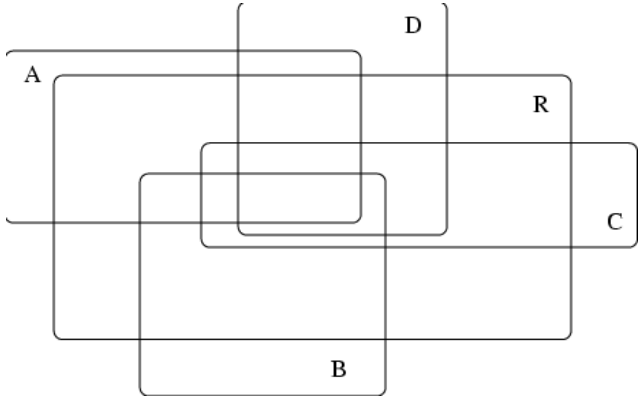
To use the pooling method for relevance set creation, the retrieval output is obtained first. This is then used to gather the judgments of relevance for the document in the pool.

This enables the evaluation metric values to be calculated for each system for each request.

Pooling

Since pooling only involves the assessment of the relevance of a subset of the document set, the relevance set developed using pooling is only an approximation of the true complete relevance set.

Pooling



R represents all the relevant documents for a particular request $r1$ contained within this document set.

A, B, C and D represent documents retrieved by four different IR systems for request $r1$. In each case this may correspond to all the documents retrieved or only those above a selected cutoff point in a ranked output list.

Thus different systems have retrieved different relevant (and non-relevant) documents.

Pooling

Pooling implies that many systems (of different kinds) must be used in the construction of the test collection, in order to generate a set of relevant items.

Of course, without manually judging the relevance of each document **we do not know the true composition of R**.

The approximate evaluation of R is carried out by "pooling" the documents retrieved by A, B, C, and D and **manually** judging which are relevant.

As shown, there may be some relevant documents not in the pooled relevance set because they have not been retrieved by any of the systems.

Thus, using this method **we can never be sure that all relevant documents have been found** and **R will always only be an approximation**, although hopefully a good one!

Interactive Evaluation of IR Systems

Can the methods of IR system evaluation that we have studied be adapted to a more user-inclusive perspective to evaluate operational interactive IR systems?

Major problem: If we are dealing with a user's ASK rather than a request:

- It is not directly visible to us ... and we cannot capture it.
- It exists only for a short space of time, and it is likely to be changed by interacting with the system and by examining documents.
- Hence, we cannot hope to put it to more than one system ... or to get extensive relevance judgements.

Interactive evaluation is hard! - and is not explored further in this module.

IR Evaluation Workshops

This evaluation methodology of test collection development using pooling, and precision and recall based evaluation metrics, is used within a number of international IR evaluation workshops.

The best known and longest running is the annual Text REtrieval Conference (TREC) workshop details at `trec.nist.gov`.

Registered participants are given an agreed set of documents and retrieval requests, they submit their retrieval output for this data collection by a strict deadline, and performance is then computed for agreed evaluation metrics after pooling the results from the participants.

TREC is ongoing with new tasks being developed each year, e.g. web retrieval, question-answering, video retrieval.

TREC

The aims of TREC:

- To encourage IR research based on standard large-scale collections.
- To increase communication between industry, academia and government creates an open forum for exchange of research ideas.
- To speed the transfer of knowledge from research laboratories into commercial products.
- To develop and make available new evaluation techniques for use in academia and industry to encourage the development of new and more effective IR technologies.

TREC

TREC began in 1992 and around 100 research groups (including DCU) now participate annually.

Other evaluation benchmarks include CLEF, NTCIR, FIRE, MediaEval.

Most of the leading international IR researchers are active participants in TREC or one or more of the other evaluation workshops.