Problems with IR

- Often difficult to find what you want using a search engine
- Too many documents returned, results "swamped" by thousands of irrelevant hits
- Some index term are very ambiguous or general e.g. "football" "plants"
- Difficult to formulate a query that describes exactly what you want from your search
- How to increase precision without compromising recall? Much of modern IR research is devoted to solving this problem

Query Enhancement

- We are going to look at some of the more common query enhancement tactics
  - Mandatory v. Optional Keywords
  - Wild-card Matching, Phrase Matching
  - Filtering, Relevance Feedback
  - Automatic Query Expansion
  - Proximity Operators, Boolean Queries
  - NLP in IR, Multilingual IR
- We will see them at work using Altavista

Mandatory v. Optional Keywords

- Sometimes searches can be too general e.g. "Pacino" retrieves Al Pacino fan pages, references to Pacino's Italian ristorante. . .
- Solution: make the most important terms *mandatory*: A retrieved document must contain *all* mandatory terms.
- The other terms are *optional*
- In Altavista, a "+" symbol before a term indicates that it is mandatory
  e.g. "+Pacino Italian +Actor American"

Wild-card Matching

- Search engines make minimal use of stemmers and lemmatisers
- So a query on "gangster films" will not retrieve: "gangster film", "films about gangsters", "gangsterism on film". . .
- One solution: Enter all such forms as optional keywords - this is cumbersome for the user.
  OR
- Use *wild-card matching* to match on a specified stem plus any word ending.
- In Altavista add an * to symbolise the word ending.
  e.g. "film* gangster*" will retrieve all of the forms listed above.

Phrase Matching

- Phrases are one of the most important units in IR
- Phrase: A sequence of two or more terms occurring near one another more often than is due to chance.
- The meaning of a phrase is often different from that of its constituents:
  e.g "gangster film", "toy soldier", "hair dryer"
- A statistical analysis of the document collection can pick out phrases at indexing time (co-occurrence analysis)
- To denote a phrase in Altavista, use quotation marks
  e.g. ' "water pistol" '

Filtering

- Query results can be accurate, but one set of results can "swamp" another
  e.g. "Al Pacino gangster films" -
  references to "Scarface" far more numerous than to "Carlito's Way"
- *Filtering* is the Removal from the result list all documents containing certain terms (exclusion terms)
- Specify an exclusion term in Altavista by preceding it with "-":
  e.g. "Al Pacino films Carlito -Scarface"
- In this way a subset of the retrieved documents can be obtained and the rest discarded

Relevance Feedback

- Usually, a single retrieval does not retrieve many relevant documents
- Can we refine the query using the returned list to better reflect what the users wants?
- Solution - *Relevance Feedback*:
  - Get the user to pick some relevant documents from the retrieved list
  - Select extra terms from these relevant documents and add them to the query, re-weight the new query and re-run
  - Repeat these steps as often as necessary
- Many different term selection, weighting and ranking formulae exist

Automatic Query Expansion

- The problem of *polysemy* still remains
  e.g. "Holland" will not retrieve documents mentioning "Netherlands", "Utrecht", "BeNeLux" . . .
- Solve this by having the system automatically add extra terms at search time e.g. "Holland Netherlands Utrecht Amsterdam Groningen"
- This requires a hand-built terminology database or thesaurus such as the Library of Congress subject headings
- Thesauri have to be hand-compiled and maintained - an expensive business
- Research on automatic thesaurus construction ongoing for several decades

Boolean Queries

- Standard Altavista query uses list-of-terms query formulation
- Unless all terms are mandatory, a lot of irrelevant documents matching only one or two query terms are returned
- *Boolean querying* (advanced search) allows precise specification at expense of recall
- Boolean queries use the Boolean connectives AND, OR and NOT: e.g. "Pacino AND (films OR movies) AND (gangster OR mafia) AND NOT Scorsese AND NOT (Scarface or Godfather) AND Carlito".
- INSPEC and MEDLINE use Boolean searching

Proximity Operators

- A term may only be relevant if it occurs near another, "anchor" term
  e.g. "football" only relevant near "Netherlands"
- *Proximity Operators* allow the user to specify that both terms must occur near one another (e.g. within N words)
- Altavista - use NEAR e.g. "B.V. Veendam AND (football NEAR Friesland)"
- So extra information can be specified without adding thousands of irrelevant documents on "football"

NLP in IR

- Using NLP in IR seems like an intuitive thing to do
- One would expect a performance improvement over purely statistical
  methods
- Not the case: Experiments using NLP for parsing, phrase extraction and fancy indexing show NLP is not more effective but requires more effort.
- Main current use is in natural language query interfaces
  e.g. "what have you got on Pacino's early movies?" $\Rightarrow$ "Pacino early movies"
- Altavista has a limited question-answer facility

Multilingual IR

Multilingual IR seeks documents in several langauges.

- *Language Selection*: Altavista records the language of each page, allowing searches to be restricted to a single language
- *Document Translation*: Using traditional Machine Translation methods to translate retrieved documents
  - Altavista uses SYSTRAN (the EC's Machine Translation package) to translate pages when requested by the user
- *Query Translation*: Cross-Language IR
  - Translate the query into each of the collection languages
  - Combine the results into a single, list of retrieved documents in several languages

Tips when using Search Engines

- Investigate advanced options
- Use directories and catalogues
- Are you sure you know what you want?
- Use phrases, scientific terms and proper names: these tend to be less ambiguous
- Be prepared to refine your query several times

THINK before you TYPE
Happy Surfing!