Techniques for Text Retrieval

- Text Indexing:
  - using the "laws" of statistical linguistics
  - Distributional characteristics of terms within docs and within the collection for estimating relevance
- Text pre-processing
  - Removing noise (including stop words, prefixes, and suffixes) for a better (compact) representation of th 'meaning' of queries and docs
- Lexical resources for polysemy, omonymy, and synonymy resolution
  - "Normalising the (natural) language used in docs and queries in order to alleviate the vocabulary mismatch problem

Text Indexing

- The choice of a representation for text means taking a stand on the issues of
  1. lexical semantics
  2. text semantics
- As in most of IR, issue 2 is usually neglected, and a doc d_i is represented as a sparse vector of weights $d_i = <w_{1i}, .. , w_{ni}>$ where $n=|T|$ and T is the dictionary of the doc collection.
- Weights w_{ki} may belong to
  - {0,1} (set of words approach SOW), representing the presence/absence of t_k in d_i
  - [0,1] (bag of words approach BOW), representing (quantitative) contribution of t_k to the semantics of d_i
- The BOW approach is to be preferred over the SOW approach (at least when the chosen model of matching contemplates it)

Automatic Indexing

- Late 1950 ideas
  - Index terms should be directly extracted from the text of a document
  - Weights can be assigned based on the frequency of index terms in documents and queries

1. (local considerations)
   a) A term t_k which occurs frequently in a doc d_i (i.e. high term frequency) is likely to be important for d_i
2. (global considerations)
   a) It is unlikely that a term occurring very frequently in the collection (high document/collection frequency) can, if used in a query, be discriminant
   b) It is unlikely that a term occurring very infrequently in the collection can, if used in a query, be very useful
   c) Then, the terms more useful for indexing purposes are those with an intermediate collection frequency

TF and IDF

A Popular way to implement the previous considerations

(1.a) is implemented by making w_{ki} grow with the term frequency of t_k in d_i

$$tf(t_k, d_i) = \begin{cases} 1 + \log \#(t_k, d_i) & \text{if } \#(t_k, d_i) > 0 \\ 0 & \text{otherwise} \end{cases}$$

(2.b) is implemented by removing from consideration all the terms which occurs in less than α docs ($1 \leq \alpha \leq 5$)

(2.a) and (2.c) are implemented by making w_{ki} grow with the inverse document frequency of t_k

$$idf(t_k) = \log \frac{|C|}{\#c(t_k)} \qquad (1)$$

TFIDF

The final weights are obtained by normalizing by cosine normalization, i.e.

$$w_{ki} = \frac{tfidf(t_k, d_i)}{\sqrt{\sum_{s=1}^{n} tfidf(t_s, d_i)^2}} = \frac{tf(t_k, d_i) \cdot idf(t_k)}{\sqrt{\sum_{s=1}^{n} (tf(t_s, d_i) \cdot idf(t_s))^2}}$$

Note that tf and tfidf equals 0 for terms t_k that not occur in d_i. This shows why the IREPs of documents and queries are sparse vectors

Several variants of the tfidf but they share (i) a tf-like component (ii) idf-like components and (iii) length normalization

TFIDF class of functions is the most popular class of weighting functions!

Text Pre-processing

- Before document indexing, a pass of document pre-processing is usually performed in order to remove noise from the document, thus transforming a text into a list of terms
- Text pre-processing consists in general of the following steps (among which steps 5,6,and 7 are optional)
  1. Reduction into ASCII format (e.g. removal of formatting/style characters, etc.)
  2. Conversion of uppercase into lowercase
  3. Identification of the 'words' (strings of contiguous characters delimited by blanks) within the text
  4. Removal of punctuation from 'words'
  5. Removal of numbers
  6. Removal of stop words
  7. Grouping words (conflation), typically those that share the same morphological root (stemming)
- Note however that what is 'noise' to a given task may be information to another!

Stop word removal

- In order to improve efficiency, words that occur too frequently in the collection (stop words) are removed , since they have almost null discrimination value:
  - E.g. articles, pronouns, prepositions, very frequent verbs (e.g. have) and adverbs (e.g. well)
  - This substantially reduces (up to 30%!) the size of data structures in secondary storage
- Pre-compiled lists of stop words (stop lists) are available in the public domain for every major language. Alternatively, they can be looked up in grammar books, or automatically extracted from text corpora by exploiting the fact that
  - They have very high DF
  - They have almost constant (i.e. document-independent) relative TF, i.e. for any stop word s_k and documents d_1 and d_2 it is the case that

$$\frac{\#(s_k,d_1)}{|d_1|} \approx \frac{\#(s_k,d_2)}{|d_2|}$$

Stemming

- In order to improve recall, words are reduced to their morphological root (stem) (e.g. comput*), of which they are inflected forms, in order to reduce the 'dispersion' effect due to
  - Variants of a morphological (e.g. lexicon/lexical) or ortographical (e.g. judgment/judgementm realize, realise) nature
  - Morphology-based antonyms (e.g. faithful, unfaithful)
  - Abbreviations (e.g. internat.)
- In stemming, two possible pitfalls must be avoided:
  - Overstemming: the stemmer returns too short a prefix, i.e. one that group also words that are not semantically related to each other (e.g. med*, which groups medical and media). Loss in precision!
  - Understemming: the stemmer returns too long a prefix, i.e. one that fails to group words that are semantically related to each other (e.g. computer*, which does not group computers and computationsl). Loss in recall!

Stemming

There are two types of stemmers:

- Algorithmic: incrementally reduce the word down to its stem
    - computationalizations
    - computationalization
    - computationaliz*
    - computational*
    - computation*
    - comput*
  - Apt for languages with simple morphological structure.
  - Non null error-rate

- Tabular: table of n pairs <word,stem>
  - Huge!
  - Apt for languages with complex morphological structure.
  - Better precision, smaller efficiency

Stemmers

There are various stemmers to English

- Different algorithms, same effectiveness
- Best known
  - Porter's stemmer (http://www.tartarus.org/~martin/PorterStemmer/index.html)
  - Lovins' stemmer
- Snowball project (http://snowball.tartarus.org/)
  - Public-domain algorithmic stemmers for a variety of languages
- Together with stop word removal, stemming is the only truly language dependent factor in IR systems

Stemmers

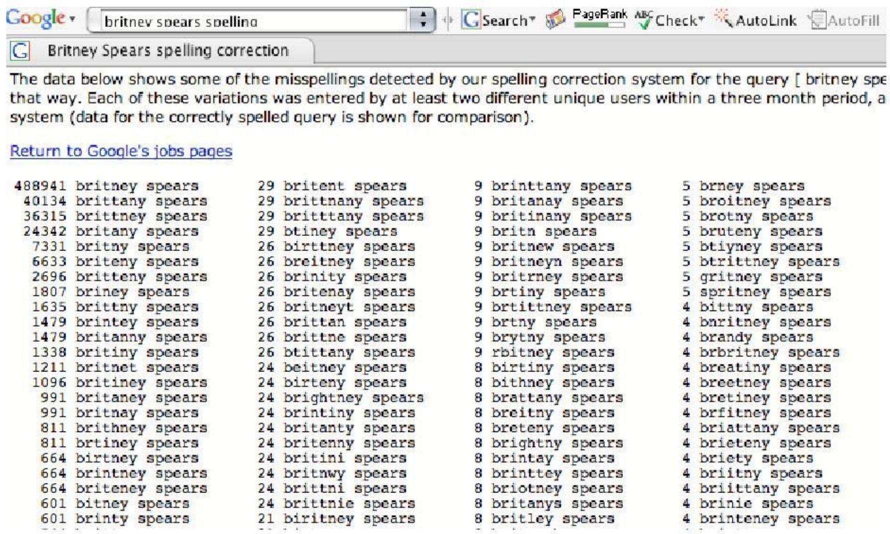
- Extracting stemmers automatically from text corpora

  [Bacchin+05] automatically identifies 'likely morphological roots' and 'likely suffixes' through a 'mutual reinforcement' process

- Decompouding – reducing a compound to its constituents
  - For word-compounding languages (German,Dutch, Finnish, etc.)

Spelling correction

- This would reduce the loss in recall due to typographical errors in queries and docs
- Spelling checking for an out-of-vocabulary (OOV) word w_1 can be done by picking w_2 such to minimize the edit distance. E.g. minimum number of chars replacements, insertion, deletions, switches. Non null error rate.
- Useful for OCR'ed documents
- Lower error rate for queries . We can use feedback from the user



Google britnev spears spellina Search PageRank Check AutoLink AutoFill

Britney Spears spelling correction

The data below shows some of the misspellings detected by our spelling correction system for the query [britney spe
that way. Each of these variations was entered by at least two different unique users within a three month period, a
system (data for the correctly spelled query is shown for comparison).

Return to Google's jobs pages

488941	britney spears	29	britent spears	9	brinttany spears	5	brney spears
40134	brittany spears	29	brittnany spears	9	britanay spears	5	broitney spears
36315	brittney spears	29	britttany spears	9	britinany spears	5	brotny spears
24342	britany spears	29	btiney spears	9	britn spears	5	bruteny spears
7331	britny spears	26	birttney spears	9	britnew spears	5	btiyney spears
6633	briteny spears	26	breitney spears	9	britneyn spears	5	btrittney spears
2696	britteny spears	26	brinity spears	9	britrney spears	5	gritney spears
1807	briney spears	26	britenay spears	9	brtiny spears	5	spritney spears
1635	brittny spears	26	britneyt spears	9	brtittney spears	4	bittny spears
1479	brintey spears	26	brittan spears	9	brtny spears	4	bnritney spears
1479	britanny spears	26	brittne spears	9	brytny spears	4	brandy spears
1338	britiny spears	26	btittany spears	9	rbitney spears	4	brbritney spears
1211	britnet spears	24	beitney spears	8	birtiny spears	4	breatiny spears
1096	britiney spears	24	birteny spears	8	bithney spears	4	breetney spears
991	britaney spears	24	brightney spears	8	brattany spears	4	bretiney spears
991	britnay spears	24	brintiny spears	8	breitny spears	4	brfitney spears
811	brithney spears	24	britanty spears	8	breteny spears	4	briattany spears
811	brtiney spears	24	britenny spears	8	brightny spears	4	brieteny spears
664	birtney spears	24	britini spears	8	brintay spears	4	briety spears
664	brintney spears	24	britnwy spears	8	brinttey spears	4	briitny spears
664	briteney spears	24	brittni spears	8	briotney spears	4	briittany spears
601	bitney spears	24	brittnie spears	8	britanys spears	4	brinie spears
601	brinty spears	21	biritney spears	8	britley spears	4	brinteney spears

Word sense disambiguation for polysemy and omonymy resolution

- Different sense of the same word (e.g. bank), cause of insufficient precision
- Word sense disambiguation (WSD) may be performed
  - Word sense instead of words as index terms for IREPs
- The main WSD algorithm (Lesk) is based on the (normalized) number of terms shared between
  - The context of the word occurrence
  - The definition of the word sense in an on-line dictionary

E.g. Sense 1 is chosen in
'I got some money on loan from a bank this morning'

After consulting the dictionary

1. An establishment for the custody, loan, exchange, or issue, of money, and for facilitating the transmission of funds by drafts or bills of exchange; an institution incorporated...

2. The margin of a watercourse; the rising ground bordering a lake, river, or sea, or forming the edge of a cutting, or other hollow

- Based on recent research
  - Only extremely short query (2/3 words) benefit of a pass of WSD, unless extremely accurate WSD
  - However, for extremely short query WSD is difficult
  - Very small improvement in effectiveness obtained in TC running all the docs through a WSD system

Using thesauri for synonymy resolution

- The mismatch between terminology used by users/intermediaries within queries and by authors/indexers within docs (vocabulary mismatch) is the major cause of insufficient recall
- The query can be expanded trough the use of a thesaurus

Using thesauri for synonymy resolution

- A thesaurus is a set of words on which a topology is defined, which highlights the semantic interrelations between these words.
- Query expansion can be achieved by attributing a non-null weight to the terms 'semantically related' to the terms occurring in the query

Using thesauri for synonymy resolution

- A thesaurus may be
  - Thematic: a 'map' of the lexicon specific of a given discipline in a given language (the most frequent case)
  - General-purpose: a 'map' of an entire language
- For IR applications, various types of thesauri
  - Hierarchical, clustered and associative

Hierarchical thesauri

- The controlled vocabularies of Boolean systems are typically hierarchical thesauri, with the following binary relations
  - NT(t_1,t_2) 'Narrower term'
  - BT(t_1,t_2) 'Broader term'
  - UF(t_1,t_2) 'Used for'
  - RT(t_1,t_2) 'Related term'
- Ambiguous terms are often 'qualified' as
  - UF(hg,mercury(metal)),
  - UF(planet Mercury,mercury(planet))

Hierarchical thesauri

- Also used for (manual) indexing
  - The problem of polysemy is avoided as terms are 'qualified'
  - The problem of 'synonymy' is avoided by recurring to the UF relation
- Main drawbacks
  - They must be manually built (difficult task)
  - They require constant (manual) updating

Clustered thesauri

- Graph of groups (or clusters) of synonymous words
- Best known (and manually built) is WordNet (http://wordnet.princeton.edu/). Two synsets (clusters) may be related by one of the following relations
  - Antonymy ('opposite name', heavy/light)
  - Hyponymy ('sub-name', elephant/animal)
  - Meronymy ('part-name', ear/head)
  - Troponomy ('manner-name', run/walk)
  - Entailment, aka Presupposition (divorce/marry)
- WordNet is currently the de facto standard (generic) clustered thesaurus.
- Extensions to other languages are underway

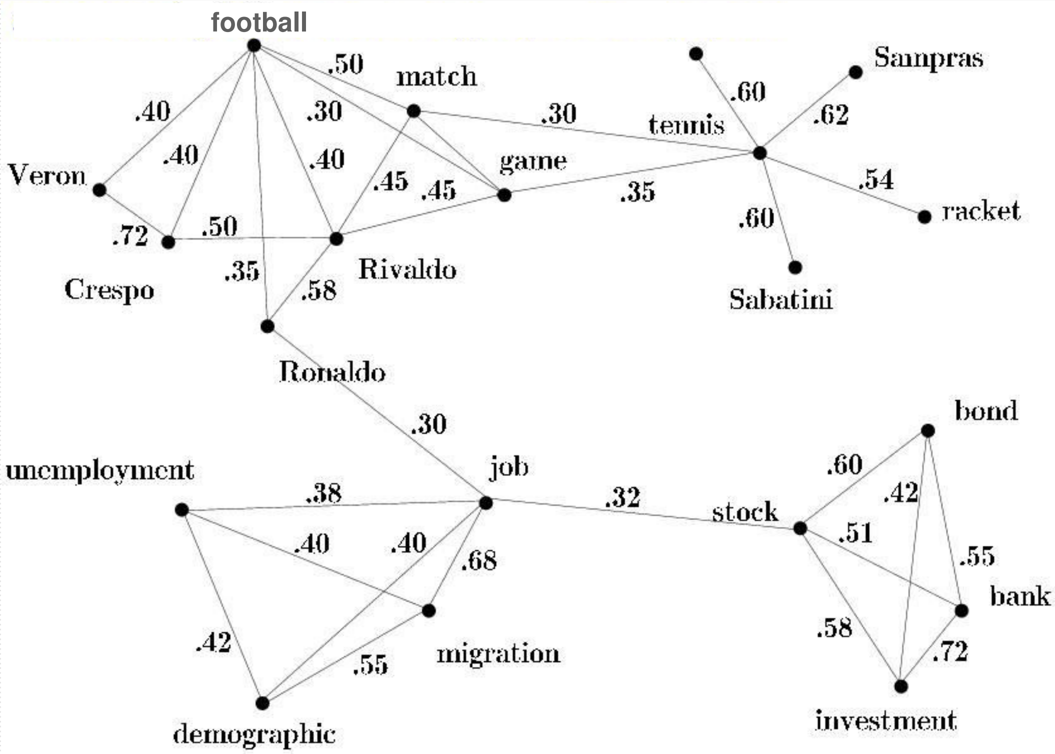
Associative thesauri

- Clustered thesauri can be buit automatically in the case in which no distinction is made between different types of semantic relation; in this case, clustered thesauri are a special case of associative thesauri
- Associative thesauri are graphs of words, where nodes represent words and edges represent a (generic) relationship of semantic similarity between two words
  - Edges may be oriented or not
  - Edges have an associated numerical weight w_{ij} (the strength of the semantic association)
- The advantage is that they may be built in a completely automatic way, starting from a collection of docs. The semantic relation between t_i and t_j mirrors the characteristics of the collection, usually based on co-occurrence (or co-absence) between t_i and t_j

Associative thesauri

- The general construction process of an associative thesaurus is the following
  1. Generate a matrix of term-term similarity values s_{ij}, using an appropriate function
  2. Apply a threshold value z to this matrix, in such a way that s_{ij} is put to 0 when $s_{ij} \leq z$
  3. In the resulting graph, individuate possible clusters (e.g. cliques)
  4. With the same method as step 1, generate the cluster-cluster similarity values, and go trough a step analogous to step 2
- Step 3 and 4 are optional. Step 1 is critical.

ASSOCIATIVE THESAURUS



Associative thesauri

- Step 1 accomplished by using the standard indexing theory in a 'dual' way (I.e. using documents as index terms of words)

Abstract Indexing Theory

Let $f_r \in F$ (features), $i_s \in I$ (items)

Let $\#(f_r,i_s)$ the number of times f_r occurs in i_s

Feature frequency is given by

$$ff(f_r, i_s) = \begin{cases} 1 + \log \#(f_r, i_s) & \text{if } \#(f_r, i_s) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Let $\#_I(f_r)$ the number of items in I in which f_r occurs at least once

Inverse item frequency is given by

$$iif(f_r) = \log \frac{|I|}{\#_I(f_r)}$$

Abstract Indexing Theory

The weighting function $ffiif(f_r,i_s)$ can now be defined and weights can be further normalized by $w_{rs} = ffiif(f_r,i_s)/||ffiif(f_r,i_s)||^2$

Moreover $SIM(i_s,i_t) = w_r w_t$

If F is the set of terms and I is the set of docs we obtain document-document similarity for document search

If F is the set of docs and I is the set of terms we obtain term-term similarity for associative thesauri