COM3110/4115/6115: Text Processing

Information Retrieval:

Term Manipulation
Web Search Ranking

Mark Hepple

Department of Computer Science
University of Sheffield

Overview

- Definition of the information retrieval problem
- Approaches to document indexing
  - manual approaches
  - automatic approaches
- Automated retrieval models
  - boolean model
  - ranked retrieval methods (e.g. vector space model)
- **Term manipulation:**
  - **stemming, stopwords, term weighting**
- **Web Search Ranking**
- Evaluation

What counts as a term?

Common to just use the **words**, but pre-process them for generalisation

- **Tokenisation**: split words from punctuation (get rid of punctuation)

  e.g. word-based. → word based   three issues: → three issues

- **Capitalisation**: normalise all words to lower (or upper) case

  e.g. Cat and cat should be seen as the same term, but should we conflate Turkey and turkey?

- **Lemmatisation**: conflate different inflected forms of a word to their basic form (singular, present tense, 1st person):

  e.g. cats, cat → cat   have, has, had → have   worried, worries → worry

What counts as a term? (ctd)

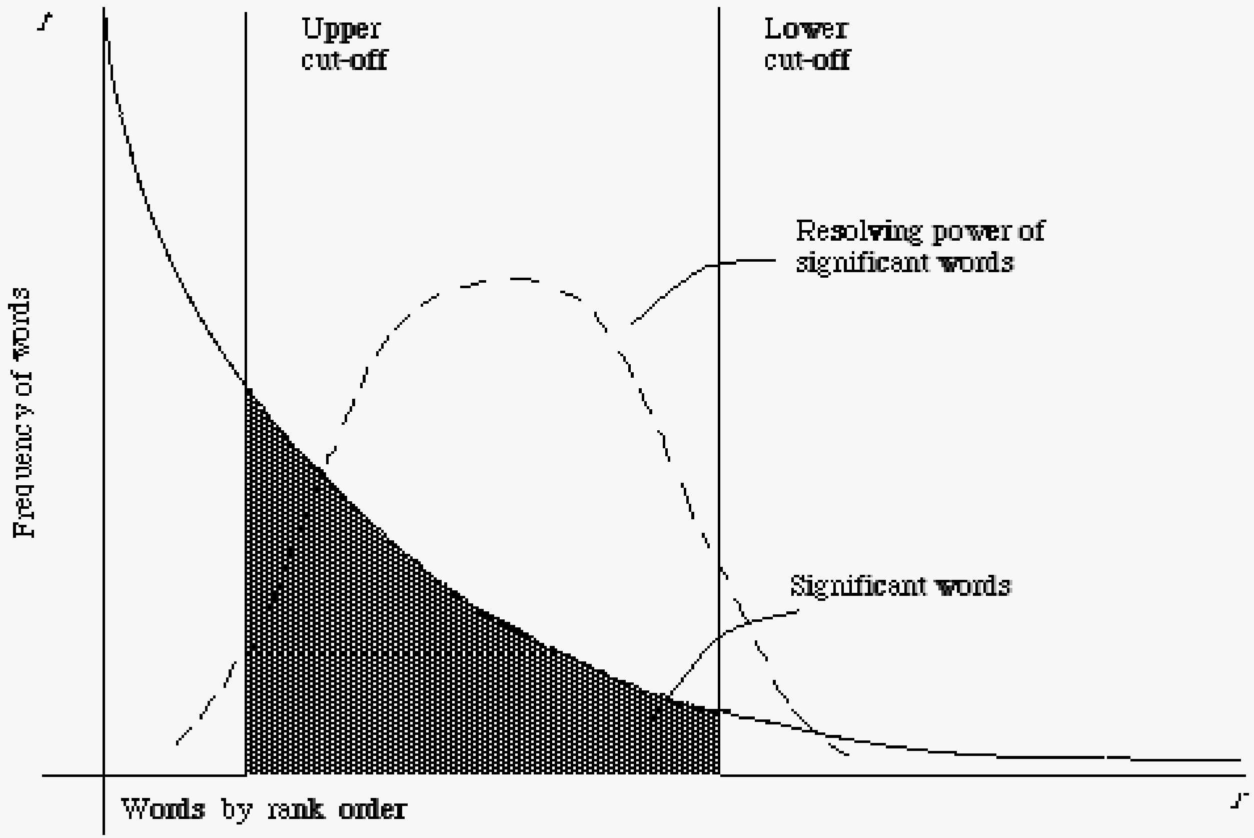
- **Stemming**: conflate morphological variants by chopping their affix:

CONNECT	WORRY	GALL
CONNECTED	WORRIED	GALLING
CONNECTING	WORRIES	GALLED
CONNECTION	WORRYING	GALLEY
CONNECTIONS	WORRYINGLY	GALLERY

- **Normalisation**: heuristics to conflate variants due to spelling, hyphenation, spaces, etc.

  e.g. USA and U.S.A. and U S A $\rightarrow$ USA
  e.g. chequebook and cheque book $\rightarrow$ cheque book
  e.g. word-sense and word sense $\rightarrow$ word-sense

Word Frequency and Term Usefulness



- The most and least frequent terms are not the most useful for retrieval
  - ◇ (Figure from van Rijsbergen (1979) *Information Retrieval* `http://www.dcs.gla.ac.uk/Keith/Preface.html`)

Stop words

- Use **Stop list** removal to exclude "non-content" words
- Usually most frequent (and least useful for retrieval)

a	always	both
about	am	being
above	among	co
across	amongst	could

◇ greatly reduces the size of the inverted index

◇ but what if we want to search for *phrases* that include these terms?

- Kings of Leon
- Let it be
- To be or not to be
- Flights to London

Single vs. Multi-word Terms

- To aid recognition of phrases, might allow *multi-word terms*

  e.g. Sheffield University

- Possible approach — allow *multi-word indexing*

  e.g. bigram indexing: store each bigram as a term in index

  For pease porridge in the pot get:

pease porridge
porridge in
in the
the pot

  ◇ Problem: number of bigrams is v.large c.f. number of words
    - leads to a huge increase in size of the index

- Alternative: identify multi-word phrases during retrieval

  ◇ Positional indexes, storing position terms in documents, can help
    - use to compute if occurrences of search terms in document are adjacent / close / far apart

Single vs. Multi-word Terms (ctd)

- Positional indexes:

Doc	*Text*
1	Pease porridge hot, pease porridge cold
2	Pease porridge in the pot
3	Nine days old
4	Some like it hot, some like it cold
5	Some like it in the pot
6	Nine days old

$\Longrightarrow$

Num	*Token*	*Docs*
1	cold	1:(6), 4:(8)
2	days	3:(2), 6:(2)
3	hot	1:(3), 4:(4)
4	in	2:(3), 5:(4)
5	it	4:(3, 7), 5:(3)
6	like	4:(2, 6), 5:(2)
7	nine	3:(1), 6:(1)
8	old	3:(3), 6:(3)
9	pease	1:(1, 4), 2:(1)
10	porridge	1:(2, 5), 2:(2)
11	pot	2:(5), 5:(6)
12	some	4:(1, 5), 5:(1)
13	the	2:(4), 5:(5)

Term Weighting

What do we use for the inverted index?

- binary weights - 0/1: whether or not term is present in document
  - But documents with multiple occurrences of query keyword may be more relevant
- Frequency of term in document: like the examples we have seen
  - But what if the term is also frequent in collection?
  - Common terms: not very useful for discriminating relevant documents
- Frequency in document vs in collection: weight terms highly if
  - They are **frequent** in relevant documents . . . *but*
  - They are **infrequent** in collection as a whole

Term Weighting (ctd)

- Key concepts:

document collection	D	collection (set) of documents
size of collection	$\|D\|$	total number of documents in collection
term freq	$tf_{w,d}$	number of times w occurs in document d
collection freq	cf_w	number of times w occurs in collection
document freq	df_w	number of documents containing w

Term Weighting (ctd)

The informativeness of terms

- Idea that *less common* terms are *more useful* to finding relevant docs:

  i.e. these terms are more *informative*

- Is this idea best addressed using *document frequency* or *collection frequency*?

- Consider following counts (from New York Times data, $|D| = 10000$):

Word	cf_w	df_w
insurance	10440	3997
try	10422	8760

  ◇ term *insurance* semantically focussed, term *try* very general

  - document frequency reflects this difference
  - collection frequency fails to distinguish them (i.e. very similar counts)

Term Weighting (ctd)

- Informativeness is inversely related to (document) frequency

  i.e. *less common* terms are *more useful* to finding relevant documents

  more common terms are *less useful* to finding relevant documents

- Compute metric such as: $\frac{|D|}{df_w}$
  - ◇ Value reduces as df_w gets larger, tending to 1 as df_w approaches $|D|$

    e.g. $\frac{10000}{3997} = 2.5$ (insurance) $\frac{10000}{8760} = 1.14$ (try)

  - ◇ Value very large for small df_w — over-weights such cases

    e.g. $\frac{10000}{350} = \mathbf{28.6}$ (mischief)

- To moderate this, take *log*: **Inverse document frequency** (idf)

$$idf_{w,D} = log\,\frac{|D|}{df_w}$$

$log\,\frac{10000}{3997} = 0.398$ (insurance) $log\,\frac{10000}{8760} = 0.057$ (try) $log\,\frac{10000}{350} = 1.456$ (mischief)

Term Weighting (ctd)

- **BUT** Not all terms describe a document equally well
- Putting it all together: **tf.idf**
  - Terms which are frequent in a document are better:

$$tf_{w,d} = freq_{w,d}$$

  - Terms that are rare in the document collection are better:

$$idf_{w,D} = log \frac{|D|}{df_w}$$

  - Combine the two to give **tf.idf** term weighting:

$$tf.idf_{w,d,D} = tf_{w,d} \cdot idf_{w,D}$$

- Most commonly used method for term weighting.
  - Used in other fields too (e.g. summarisation)

Term Weighting (ctd)

tf.idf example:

Term	tf	df	$\|D\|$	idf	$tf.idf$
the	312	28,799	30,000	0.018	5.54
in	179	26,452	30,000	0.055	9.78
general	136	179	30,000	2.224	302.50
fact	131	231	30,000	2.114	276.87
explosives	63	98	30,000	2.486	156.61
nations	45	142	30,000	2.325	104.62
haven	37	227	30,000	2.121	78.48

For term the:

$$idf(the) = \log_{10}\left(\frac{30,000}{28,799}\right) = 0.018$$

$$tf.idf(the) = 312 \cdot 0.018 = 5.54$$

Putting things together

Example: Vector Space Model, tf.idf term weighting, cosine similarity

- tf.idf values for words in two documents D_1 and D_2, and in a query Q "hunter gatherer Scandinavia":

	Q	D_1	D_2
hunter	19.2	56.4	112.2
gatherer	34.5	122.4	0
Scandinavia	13.9	0	30.9
30,000	0	457.2	0
years	0	12.4	0
BC	0	200.2	0
prehistoric	0	45.3	0
deer	0	0	23.6
rifle	0	0	452.2
Mesolithic	0	344.2	0
$\sqrt{\sum_{i=1}^{n} x_i^2}$	41.9	622.9	467.5

(i.e. length of vector)

- $$sim(\vec{q},\vec{d}) = cos(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

Putting things together (ctd)

- $$sim(\vec{q},\vec{d}) = cos(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

$$\begin{aligned}
cos(Q, D_1) &= \frac{(19.2 * 56.4) + (34.5 * 122.4) + \cdots + (0 * 0) + (0 * 344.2)}{41.9 * 622.9} \\
&= \frac{5305.68}{26071.72} \\
&= 0.20
\end{aligned}$$

$$\begin{aligned}
cos(Q, D_2) &= \frac{(19.2 * 112.2) + (34.5 * 0) + \cdots + (0.0 * 452.2) + (0.0 * 0.0)}{41.9 * 467.5} \\
&= \frac{2583.8}{19570.0} \\
&= 0.13
\end{aligned}$$

- so document D_1 is more similar to Q than D_2

Web Search Ranking

- Web docs contain info beyond their mere *"textual content"*
  - ◇ *state-of-the-art web search* engines, like Google, exploit this
  - ◇ achieve *much more effective* retrieval than could without it
- HTML contains clues that some terms are *more important*

  e.g. terms in regions marked as title or headings

  e.g. terms *emphasised by formatting*: bold / bigger / colour
  - ◇ can use clever term weighting schemes, that add weight to such terms
- Link text — commonly provide *description of target* doc
  - ◇ often a better description than doc provides of *itself*

  e.g. "Hey, here's a great intro to calculus for beginners – check it out!"
  - ◇ Google treats link text as *part of* target doc
- Link structure of web *more broadly*
  - ◇ if page A *points to* page B, implies B is worth looking at
  - ◇ can be used as a measure of *authority / quality*

Exploiting Link Structure: the PageRank Algorithm

- Key method to exploit link structure of web: PageRank algorithm
  - ◇ named after its inventor: Larry Page (co-founder of Google)
  - ◇ assigns a score to each page on web: its *PageRank score*
    - can be seen to represent the page's *authority* (or *quality*)
- PageRank algorithm — key idea:
  - ◇ link from page A to page B confers **authority** on B
  - ◇ *how much* authority is conferred depends on:
    - the authority (PageRank score) of A, and its number of *out-going links*

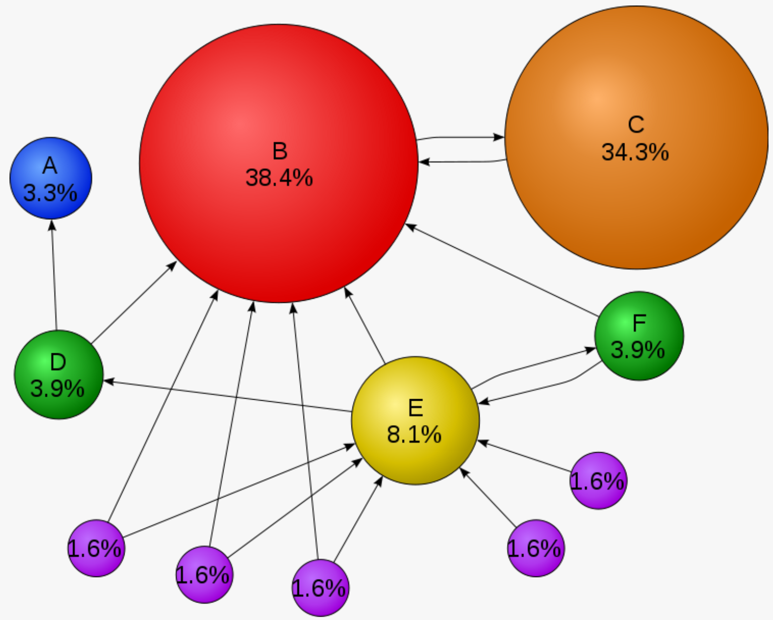
      i.e. A's authority is *shared out* amongst its out-going links
  - ◇ note that this measure is *recursively defined*

    i.e. score of any page depends on score of every other page
- PageRank scores have an alternative interpretation:
  - ◇ probability that a *random surfer* will visit that page

    i.e. one who starts at a random page, clicks randomly-chosen links forward, then (getting bored) jumps to a new random page, and so on . . .

Exploiting Link Structure: the PageRank algorithm (ctd)

- Graphical intuition:



- During retrieval, rank score of doc *d* is a *weighted combination* of:
  - ◇ its PageRank score: a measure of its authority
  - ◇ its IR-Score: how well *d* matches the query *q*, based on
    - Vector Space model, TF.IDF, *up-weighting* of important terms, etc