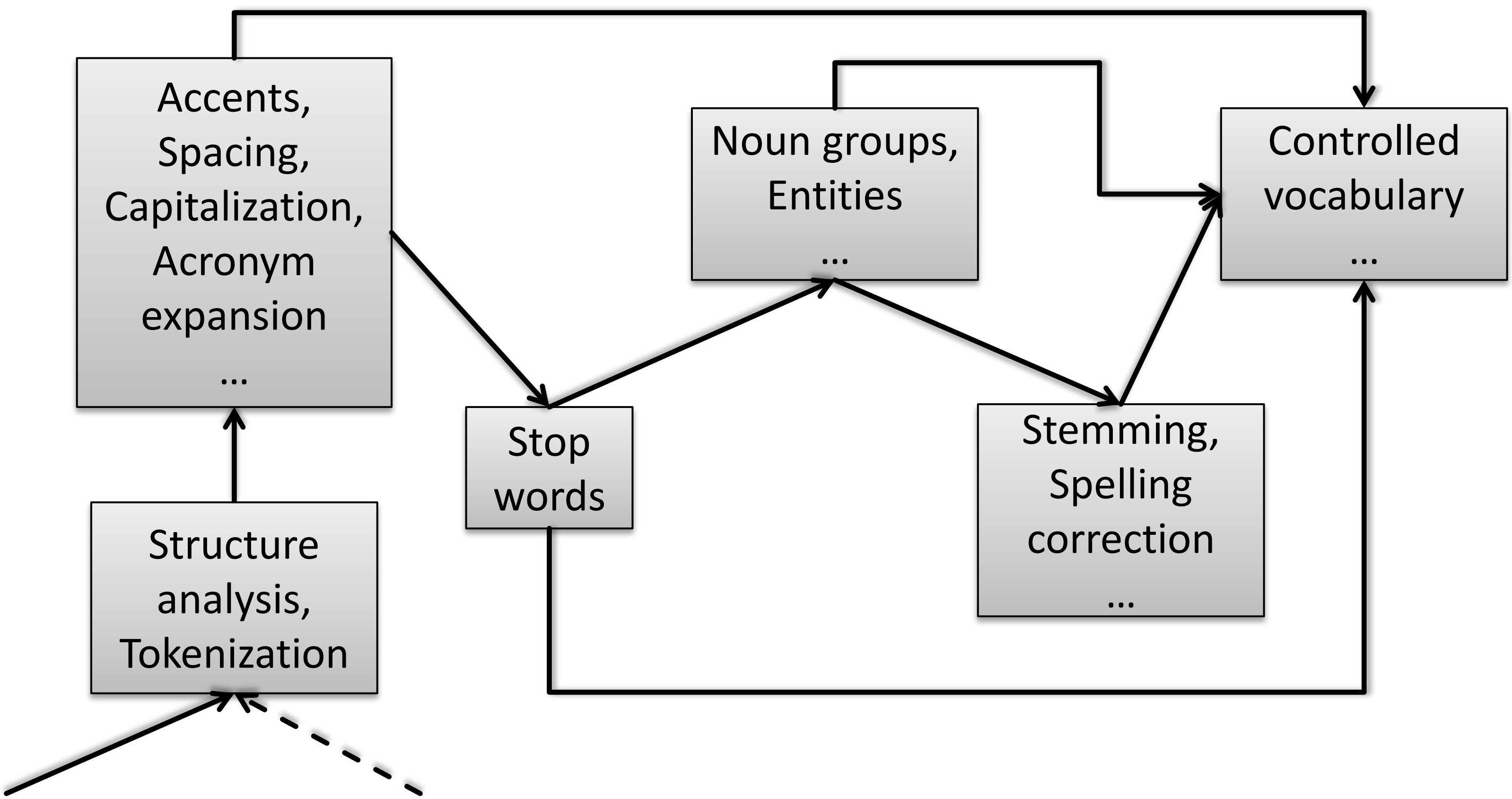
FROM QUERIES TO TOP-K RESULTS

Outline

- Intro
- Basics of probability and information theory
- Retrieval models
- Retrieval evaluation
- Link analysis
- From queries to top-k results
  - Query processing
  - Indexing
  - Top-k search
- Social search

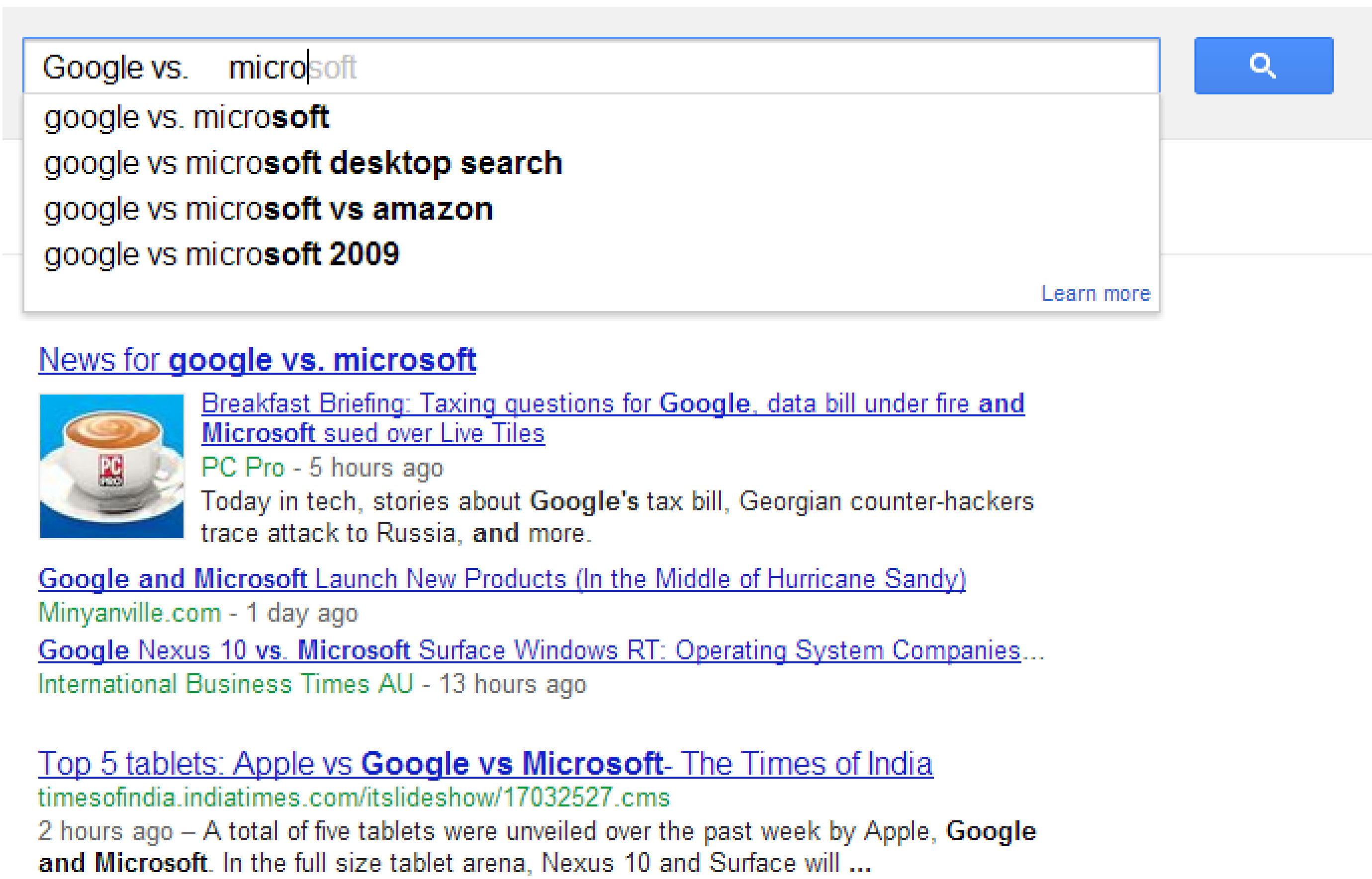
Query processing overview



query document (same process for document indexing)

Query normalization: clean query

- Remove punctuations, comas, semicolons, unnecessary spaces…
- Upper-case vs. lower-case spellings (language-dependent)
- Normalize and expand acronyms (e.g.: N.Y.→ NY → New York)
- Normalize language dependent characters (e.g.: ü → ue)



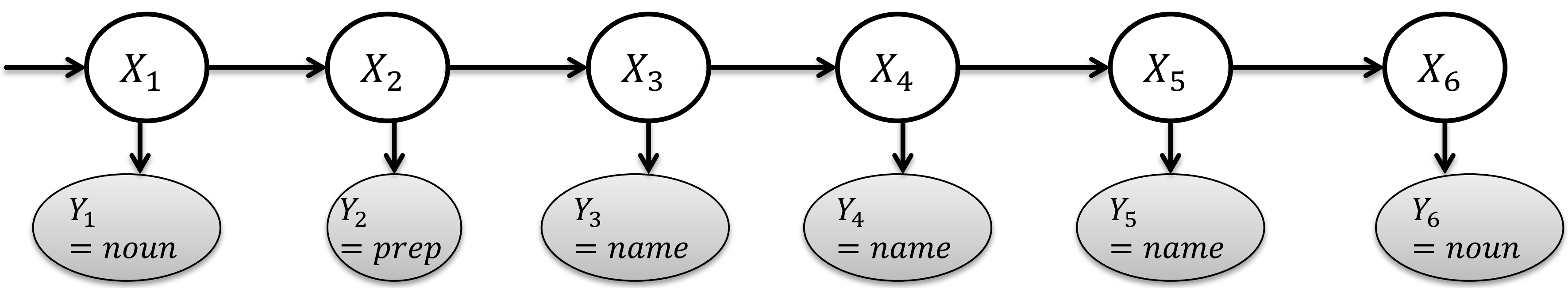
Query normalization: remove stop words

- Typically, maintained in so-called stop word lists, e.g.:

a	but	him	most	since	when
able	by	his	my	so	where
about	can	how	neither	some	which
across	cannot	however	no	than	while
after	do	i	nor	that	who
all	does	if	not	the	whom
almost	either	in	of	their	why
also	else	into	off	them	will
am	ever	is	often	then	with
among	every	it	on	there	would
an	for	its	only	these	yet
and	from	just	or	they	you
any	get	least	other	this	your
are	has	let	our	to	…
as	have	like	own	too	
at	he	likely	rather	us	
be	her	may	she	we	
because	hers	me	should	what	

Named-entity recognition in query

- Task
  - Identify named entities such as persons, locations, organizations, dates, etc. in query
  - Example: “flights to John F. Kennedy airport”
- Solutions
  - Look-up in dictionary or knowledge base (mapping is still difficult)
  - **Shallow parsing** (exploit internal structure of names and local context in which they appear)
  - Shallow parsing + probabilistic graphical models for sequential tagging (e.g.: **Hidden Markov Models** (HMMs), **Conditional Random Fields** (CRFs))
  - Example: HMM with 2 states {entity, non-entity}, find $\max_{\mathbf{X}} P(\mathbf{X}, \mathbf{Y})$



Query normalization: stemming

- **Morphological reduction** to the **stem of a term** (i.e., the ground form)
  - **Lemmatization** algorithms determine the **part of speech** (e.g., noun, verb, adjective, etc.) of a term and apply stemming rules to map terms to their **grammatical lemma** (≡ ground form)
  - There are different stemming rules for adjectives, verbs, nouns, …
  - Simple rules for stemming

[.]{3, }+ies → y	(countries → country, ladies → lady, …)
[.]{4, }+ing →	(fishing → fish, sing → sing, applying → apply, …)
[.]{2, }+ss|sses → ss	(press → press, guesses → guess, less → less, chess → chess, …)
[.]{3, }[^s]+s →	(books → book, symbols → symbol, …)

- Notes
  - Order in which stemming rules are applied is important
  - Indexed documents undergo the same stemming process as the query

The Porter Stemmer

- Stemming algorithm proposed by Martin Porter in 1980
- Standard algorithm for English stemming
- Uses different steps for
  - Mapping plural to singular form, e.g.:
    - sses → ss
    - ies → i
    - ss → ss
    - s → ϵ
  - Mapping past and progressive tense to simple present tense, e.g.:
    - eed → ee
    - ed → ϵ
    - ing → ϵ
  - Clean-up and handle endings (e.g.: y →i)
  - Derivational morphology, e.g.:
    - ational → ate
    - ization → ize
    - biliti → ble

Query reformulation on Google

Computer scientists with Master's degree in music

About 24,300,000 results (0.15 seconds)

The Best And Worst **Master's Degrees** For Jobs - Forbes
www.forbes.com/.../the-best-and-worst-**masters-degrees**-for...

by Jacquelyn Smith - in 177 Google+ circles
6 Jun 2011 – In Pictures: The Best **Master's Degrees** For Jobs ... By our count, **computer science** ties physician assistant studies for the No. .... pictures of the worst Masters have women in them and only have men for **Music** and the Clergy.

Music & Tech **Degree** - Carnegie Mellon University
www.cmu.edu › Homepage Stories › Creativity and the Arts
Carnegie Mellon professors of **music**, engineering and **computer science** are coming together to offer two new **degrees in music** and ... Based in the School of **Music**, the Bachelor of Science in **Music** and Technology and the **Master** of Science ...

Masters Program | CCRMA
https://ccrma.stanford.edu/academics/**masters**
Music 223T **Computer Music** Improvisation and Algorithmic Performance **Music** 250A ... The **Master's degree in Music**, **Science**, and Technology. Description: ...

Query normalization: spelling correction

- Check spelling
  - E.g., by using similarity measures and occurrence frequencies on entries from dictionaries, query logs, web corpus

- Propose correction of misspelled words, e.g.:

  recieve → receive

  dissapoiting → disappointing

  acomodation → accommodation

  mulitplayer → multiplayer

  Playstaton → Playstation

  Schwarznegger → Schwarzenegger

Google's
instant autocorrection

Example: misspellings for Britney Spears on Google

488941	britney spears	29	britent spears	9	brinttany spears
40134	brittany spears	29	brittnany spears	9	britanay spears
36315	brittney spears	29	britttany spears	9	britinany spears
24342	britany spears	29	btiney spears	9	britn spears
7331	britny spears	26	birttney spears	9	britnew spears
6633	briteny spears	26	breitney spears	9	britneyn spears
2696	britteny spears	26	brinity spears	9	britrney spears
1807	briney spears	26	britenay spears	9	brtiny spears
1635	brittny spears	26	britneyt spears	9	brtittney spears
1479	brintey spears	26	brittan spears	9	brtny spears
1479	britanny spears	26	brittne spears	9	brytny spears
1338	britiny spears	26	btittany spears	9	rbitney spears

Source: http://www.cse.unl.edu/~lksoh/Classes/CSCE410_810_Spring06/sup1.html

- Observation: most used spelling is typically correct ("Wisdom-of-the-Crowds" effect)
- Can this observation be used for spellchecking and auto-correction?

Google's spelling correction approach (1)

- Preprocessing
  - Goal: generate triples of the form $(intended\_str, observed\_str, obs\_freq)$
    - Build a term list with frequent terms occurring on the web.
    - Remove non-words (e.g., too many punctuations, too short or too long).
    - For each term in the list, find all other terms in the list that are "**close**" to it and create pairs (str_1, str_2), where str_1, str_2 are "close enough" to each other.
    - From all pairs of the form (str_1, str_2) and maintain only those pairs (str', str), for which $freq(str') \geq 10 * freq(str)$.
    - Return list of remaining triples $(str', str, freq(str))$.
  - **Note**: computation can be done in parallel and is easy to distribute.

Source: Whitelaw et al. "*Using the Web for Language Independent Spellchecking and Autocorrection*". EMNLP 2009

Google's spelling correction approach (2)

- Input
  - Observed word w,
  - Candidate corrections $\{c \mid c$ "is close to" $w\}$
  - Data given by the set of triples $\{(c, w, freq(w)) \mid w \text{ is observed}\}$
- Output
  - Candidate corrections, ranked decreasingly by

$$P(c|w) \propto P(w|c)P(c)$$

Error model (word likelihood)

N-gram language model derived from the web

- Estimation of error model: for adjacent-substring partitions R of c and T of w estimate

$$P(w|c) \approx \max_{R,T:|R|=|T|} \prod_{i=1}^{|R|} P(T_i|R_i)$$

e.g., with $|R_i|, |T_i| \leq 3$

Google's spelling correction approach (3)

- Context of a word can also be taken into account
  - Generate triples of the form $(w_l c w_r, w_l\, w\, w_r, freq(w))$
- Language model can be down-weighted (relatively to the error model), in case errors are common, e.g.:

$$P(c|w) \propto P(w|c)P(c)^{\lambda}$$

with $\lambda \geq 0$

- Reported error rates $< 4\%$ when error model trained on corpus size of ~10 Mio. terms
- How to find "close" strings?
  - Use similarity measures on strings (e.g., based on edit distance, Jaccard similarity on substring partitions, …)

String distance measures: edit distance (1)

- Minimal number of editing operations to turn a string s_1 into another string s_2
- **Levenshtein distance (edit distance)**
  - Uses replacement, deletion, insertion of a character as editing operations
  - Input: $s_1[1..i]$ and $s_2[1..j]$
  - Conditions:

$$edit(0,0) = diff(i,j),$$
$$edit(i,0) = i + edit(0,0),$$
$$edit(0,j) = j + edit(0,0).$$

  - Output: $edit(i,j) = \min\{ edit(i-1,j) + 1,$ // delete
    $edit(i,j-1) + 1,$ // insert
    $edit(i-1,j-1) + diff(i,j) \}$ // replace

e.g., with $diff(i,j) = \begin{cases} 1 & if\ s_1[i] \neq s_2[j] \\ 0 & otherwise \end{cases}$

→efficient computation by dynamic programming

String distance measures: edit distance (2)

- **Levenshtein distance**

	K	N	I	T	T	I	N	G
S	**1**	2	3	4	5	6	7	8
I	2	**2**	**2**	3	4	5	6	7
T	3	3	3	**2**	3	4	5	6
T	4	4	4	3	**2**	3	4	5
I	5	5	4	4	3	**2**	3	4
N	6	5	5	5	4	3	**2**	3
G	7	6	6	6	5	4	3	**2**

$$
\begin{aligned}
edit(i,j) = \min\{ & edit(i-1,j) + 1, && \text{// delete} \\
& edit(i,j-1) + 1, && \text{// insert} \\
& edit(i-1,j-1) + diff(i,j) \} && \text{// replace}
\end{aligned}
$$

Approximate string containment with edit distance

- Levenshtein distance for approximate string containment
  - Slightly different starting conditions
  - “colour” is contained in “kolorama” with 2 errors

		k	o	l	o	r	a	m	a
	0	0	0	0	0	0	0	0	0
c	1	1	1	1	1	1	1	1	1
o	2	2	1	2	1	2	2	2	2
l	3	3	2	1	2	2	3	3	3
o	4	4	3	2	1	2	3	4	4
u	5	5	4	3	2	2	3	4	5
r	6	6	5	4	3	**2**	3	4	5

Source: Modern Information Retrieval,
Baeza-Yates, Ribeiro-Neto

String distance measures: edit distance (3)

Demerau-Levenshtein distance

- Uses replacement, deletion, insertion, and transposition of character as editing operations
  - Input: $s_1[1..i]$ and $s_2[1..j]$
  - Conditions:

    $$edit(0,0) = diff(i,j),$$
    $$edit(i,0) = i + edit(0,0),$$
    $$edit(0,j) = j + edit(0,0).$$

  - Output: $edit(i,j) = \min\{ edit(i-1,j) + 1,$
    $edit(i,j-1) + 1,$
    $edit(i-1,j-1) + diff(i,j)$
    $edit(i-2,j-2) + 1\}$ // transpose

with $diff(i,j) = \begin{cases} 1 & if\ s_1[i] \neq s_2[j] \\ 0 & otherwise \end{cases}$

Other useful string distance measures

- Hamming distance

  $d_H(s_1, s_2) = \#\{i \mid s_1[i] \neq s_2[i]\}$, for $|s_1| = |s_2|$

- Jaccard distance

  $G_N(s) := \{\text{substrings of length } N\}$, i.e., subset of N-grams

  Example

  $G_3(\text{"schwarzenegger"}) := \{$sch, chw, hwa, war, arz, rze, zen, ene …$\}$

  $d_J(s_1, s_2) = 1 - \frac{|G_N(s_1) \cap G_N(s_2)|}{|G_N(s_1) \cup G_N(s_2)|}$

- Simple N-gram-based distance

  $d(s_1, s_2) = |G_N(s_1)| + |G_N(s_2)| - 2|G_N(s_1) \cap G_N(s_2)|$

- **Theorem 1**: for string s_1 and a target string s_2

  $|G_N(s_1) \cap G_N(s_2)| < |s_1| - (N-1) - dN \Rightarrow d_{edit}(s_1, s_2) > d$

Phonetic similarities

- **Soundex algorithm**

  Idea: map words onto 4-letter codes, such that words with similar pronunciation have the same code

  - First letter of the word becomes first code letter
  - Then map

    b, p, f, v → 1

    c, s, g, j, k, q, x, z → 2

    d, t → 3

    l → 4

    m, n → 5

    r → 6
  - For letters with the same soundex number that are immediately next to each other, only one is mapped
  - a, e, i, o, u, y, h, w are ignored (exept for the first character)
  - If code length > 4, keep only first four characters of the code

- Examples: Penny →P500, Ponny→P500, Powers→P620 , Perez →P620

Query reformulation (1)

- A specific search need can be expressed in different ways, some formulations lead to better results than others

- Already discussed some strategies for (implicit) query reformulation
  - Integration of relevance feedback (e.g., Rocchio algorithm), implicit feedback (using clicks and similar queries from query log), pseudo-relevance feedback (assuming top-k results are relevant)
  - Example: estimate the probability of w' given $w \in q$ from query log

$$P(w'|w) \approx \sum_d P(w'|d \in RelDocs(q)) \cdot P\big(d \in RelDocs(q)|d \in RelDocs(w)\big)$$

where $RelDocs(q)$ and $RelDocs(w)$ are the documents that were (implicitly) rated as relevant for the query and the keyword w respectively

Query reformulation (2)

- Linguistic techniques that use
  - stemming/lemmatization and spelling correction (through edit distance)
  - thesauri or dictionaries for term expansion or replacement by synonyms, hypernyms, hyponyms, meronyms, holonyms, …
  - Naive reformulation

    Substitute (or expand) keyword w with $S_w = \{w_1, \ldots, w_k\}$ such that for each $w_i \in S_w : sim(w, w_i) \geq \tau$ (with threshold τ)

$$Score(q, d, \cup_{w \in q} S_w) = \sum_{w \in q} \sum_{w_i \in S_w} sim(w, w_i) * Sc(w_i, d)$$

Query reformulation: example (1)

english thesauri

About 1,110,000 results (0.63 seconds)

Thesaurus.com | Find Synonyms and Antonyms of Words at ...
thesaurus.com/
Thesaurus.com - the largest and most trusted free online **thesaurus** brought to you by Dictionary.com. **...** Try to solve the classic tentacled question of **English** **...**
Search - Free Thesaurus Tools ... - Mobile - 'A' Synonyms

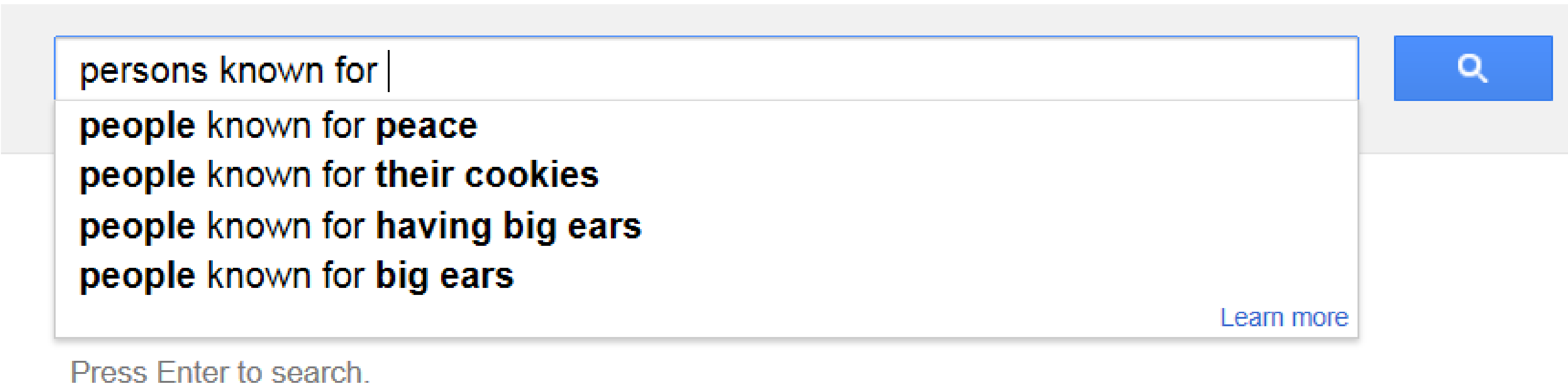
Thesaurus - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Thesaurus**
1 History; 2 **Thesauri** in IT; 3 Literary **thesauri**; 4 Specialized **thesauri** for information retrieval; 5 **Thesauri** formats **...** **Thesaurus** of **English** Words & Phrases (ed.

The Cambridge French-**English Thesaurus**: Marie-Noklle Lamy ...
www.amazon.com › ... › Dictionaries & Thesauruses › Thesauruses
Students of French learn the basics of grammar and vocabulary, but, unless they spend time in France or have other opportunities to immerse themselves in the **...**

National Monuments Record **Thesauri** - **English** Heritage
thesaurus.**english**-heritage.org.uk/
National Monuments Record **Thesauri**. New Users - Frequent Users. Setting the standards for recording the past. © **English** Heritage 1999. National Monuments **...**

Query reformulation: example (2)

- Instant query reformulation (thesaurus-based?)



Careful thesaurus-based query reformulation

- Find important **phrases** in query (or from best initial query results)

- Try to map found phrases onto synonyms, hyponyms, hypernyms from some thesaurus

- If a phrase is mapped to one concept expand it with synonyms and hyponyms

- Compute score as

$$Score(q, d, Th\colon thesaurus) = \sum_{w \in q} \max_{c \in Th} \{sim(w, c) * Sc(c, d)\}$$

→ avoids unfair expansion of terms that have many concepts

Thesaurus-based query reformulation

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: particle physics [Search WordNet]

Source: http://wordnetweb.princeton.edu/perl/webwn

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **particle physics**, high-energy physics, high energy physics (the branch of physics that studies subatomic particles and their interactions)
  - ***domain term category***
    - S: (n) flavor, flavour ((physics) the six kinds of quarks)
    - S: (n) charm ((physics) one of the six flavors of quark)
    - S: (n) strangeness ((physics) one of the six flavors of quark)
    - S: (n) color, colour ((physics) the characteristic of quarks that determines their role in the strong interaction) *"each flavor of quarks comes in three colors"*
    - S: (n) M-theory ((particle physics) a theory that involves an eleven-dimensional universe in which the weak and strong forces and gravity are unified and to which all the string theories belong)
    - S: (n) string theory ((particle physics) a theory that postulates that subatomic particles are one-dimensional strings)
  - *direct hypernym* | *inherited hypernym* | *sister term*
    - S: (n) physics, natural philosophy (the science of matter and energy and their interactions) *"his favorite subject was physics"*

Possible similarity measures

- Dice

$$sim(w_i, w_j) = \frac{2 \cdot |\{docs(w_i)\} \cap \{docs(w_j)\}|}{|\{docs(w_i)\}| + |\{docs(w_j)\}|}$$

- Jaccard

$$sim(w_i, w_j) = \frac{|\{docs(w_i)\} \cap \{docs(w_j)\}|}{|\{docs(w_i)\} \cup \{docs(w_j)\}|}$$

- Point-wise mutual information (PMI)

$$sim(w_i, w_j) = \frac{freq(w_i \; and \; w_j)}{freq(w_i) \cdot freq(w_j)}$$

- Similarity on ontology graphs

$$sim^*(w, w') = \max\left\{\prod_{(c_i, c_j) \in p} sim(c_i, c_j) \,\middle|\, p \; is \; path \; from \; w \; to \; w'\right\}$$

Summary

- Query analysis
  - Parsing, tokenisation
- Query cleaning (remove punctuations, comas, stop words)
- Named-entity/noun-group recognition (dictionaries, shallow parsing, HMMs)
- Stemming
- Spelling correction
  - Similarity/distance measures (edit distance, n-gram-based Dice, Jaccard, …)
- Query reformulation (linguistic, thesaurus-based)
  - Thesaurus-based similarity