Gestione dell'Informazione

Part A – Full-Text Information Management

Text Operations

Contents

- Document Processing
- Thesauri
- Word similarities
- Word Sense Disambiguation
- Hands-on with Python and NLTK

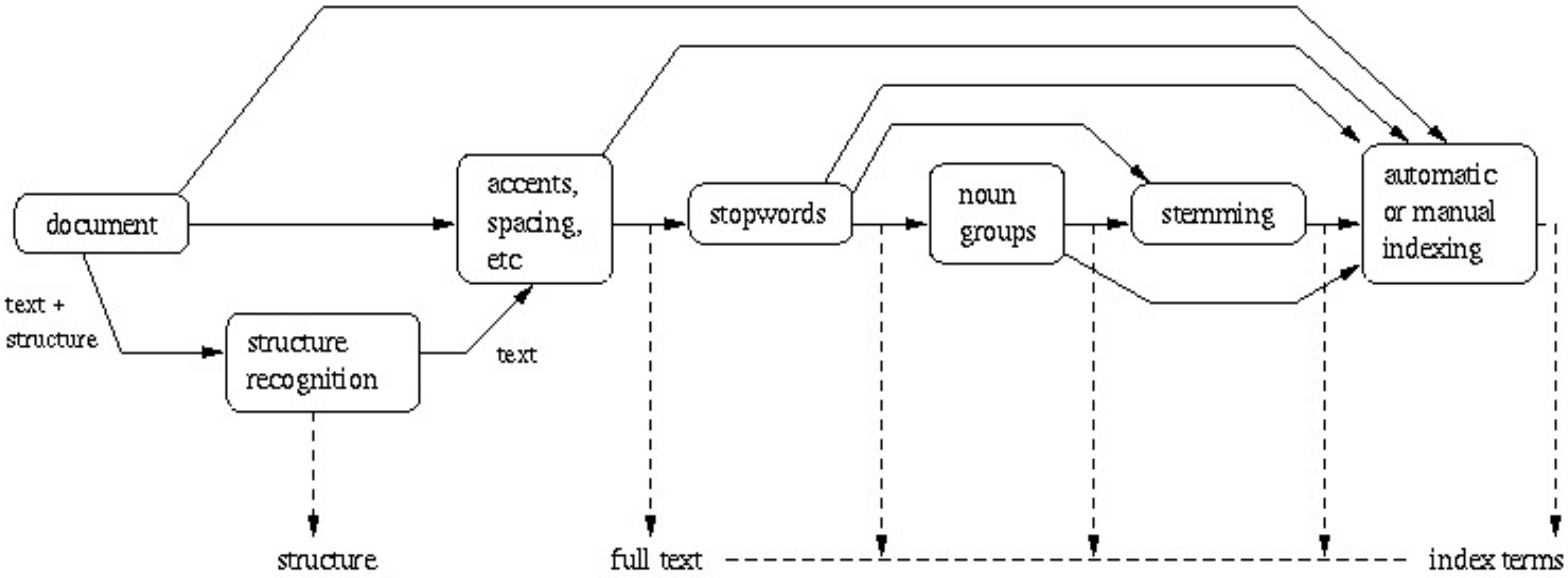
Document Preprocessing

Document preprocessing is a procedure which transforms a document into a set of index terms

Text Operations for Document Preprocessing

1. **Lexical Analysis of the text**
2. **Elimination of stopwords**
   - Filtering out the useless words for retrieval purposes
3. **Stemming of the remaining words**
   - Dealing with the syntactic variations of query terms
4. **Selection of index terms**
   - Determining the terms to be used as index terms
5. **Construction of term categorization structures**
   - Allowing the expansion of the original query with related term

Document Preprocessing



Lexical Analysis of the Text

- Process of converting a stream of characters into a stream of **tokens**
  - **Token**: group of characters with collective significance

- Produces candidate index terms

- Ways to implement a lexical analyser:
  - Use a lexical analyzer generator (e.g. UNIX tool lex)
  - Write a lexical analyser by hand ad hoc
  - Write a lexical analyser by hand as a finite state machine

- Ref. Example:

"He said that the chairs were enough"

He	said	that	the	chairs	were	enough

What counts as a token? Four particular cases

Digits Usually not good index terms because of its vagueness

- However digits as B6 (vitamin) are significant terms
- It needs some advanced lexical analysis procedure

Ex) *510B.C. , 4105-1201-2310-2213, 2000/2/12, ….*

Hyphens Breaking up hyphenated words might be useful

Ex) state-of-the-art → state of the art (Good!)

But, MS-DOS → MS DOS (???)

- It needs to adopt a general rule and to specify the exception on a case by case basis

Punctuation Marks Are removed entirely

Ex) *510B.C → 510BC*

- If the query contains '*510B.C*', remove of the dot both in query term and in the documents will not affect retrieval performance

- Require the preparation of a list of exceptions

Ex) *val.id → valid* (???)

The Case of Letters Converts all the text to either lower or upper case

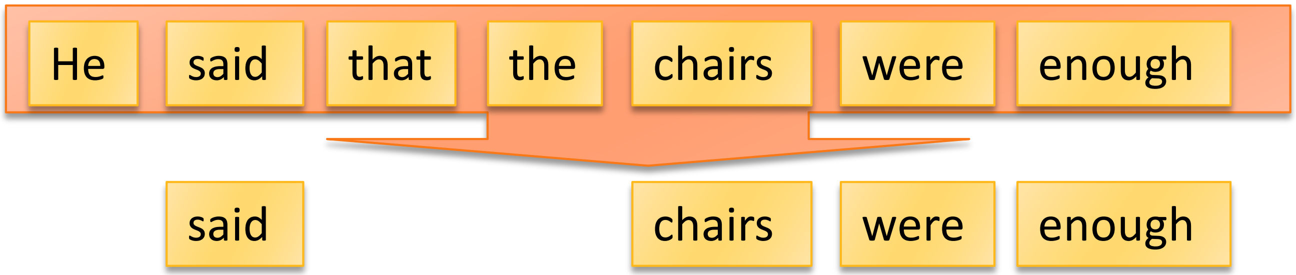
- Part of the semantics might be lost

Ex) *Korea University → korea university* (???)

Elimination of Stopwords

- Basic Concept
  - Filtering out words with very low discrimination values

    Ex) *a, the, this, that, where, when, ….*
- Advantage
  - A very frequent word is useless for purposes of retrieval
  - Reducing the size of the indexing structure considerably
- Ways to filter stoplist words from an input token stream
  - Examine lexical analyzer output and remove any stopwords
    - Standard list searching problem
    - Search trees, bynary search on an ordered array and hashing can be used
  - Remove stopwords as part of lexical analyzer
- Ref. Example:



Stemming & Lemmatization

- Basic Idea: "Provide searchers with ways of finding morphological variants of search terms"

  Ex) query '*stemming*' also search for '*stemmed*' and '*stem*'

- What is the **"stem"** ? The portion of a word which is left after the removal of its affixes (i.e., prefixes and suffixes)

  Ex) '*connect*' is the stem for the variants '*connected*', '*connecting*', '*connection*', '*connections*'

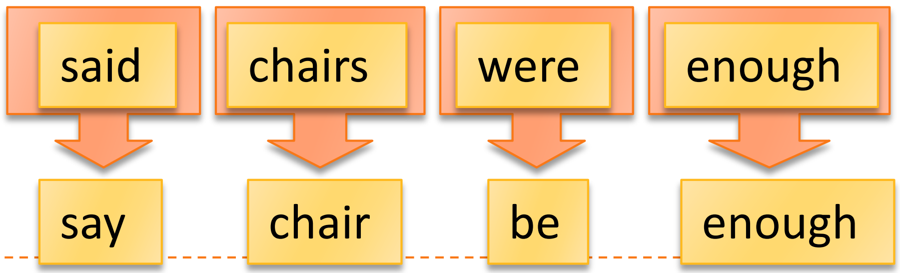
  Stemmer: Tool that performs stemmization

- What is the **"lemma"** ? The base or dictionary form of a word

  Ex) 'see' is the lemma for 'saw', 'seen'

  Lemmatizer: Tool that performs lemmatization

- Ref. Example:



Judging stemmers

- Correctness
  - Possible incorrectness
    - Overstemming: too much is removed
    - Understemming: too little is removed
- Retrieval effectiveness
  - Stemming Reduce variants of the same root word to a common concept
  - Stemming can affect retrieval performance (for the majority, positively)
  - The effect of stemming depends on the nature of the vocabulary
  - There are little differences between the retrieval effectiveness of different full stemmers
- Compression performance
  - Reduce the size of the indexing structure
  - 5 stemmers on 4 data sets: Compression from 26.1% to 47.5%

- There is controversy about the benefits of stemming
  - For this reason some Web search engines do not adopt any stemming algorithm (Google does)

Index Term Selection

- Not all words are equally significant for representing the semantics of a document

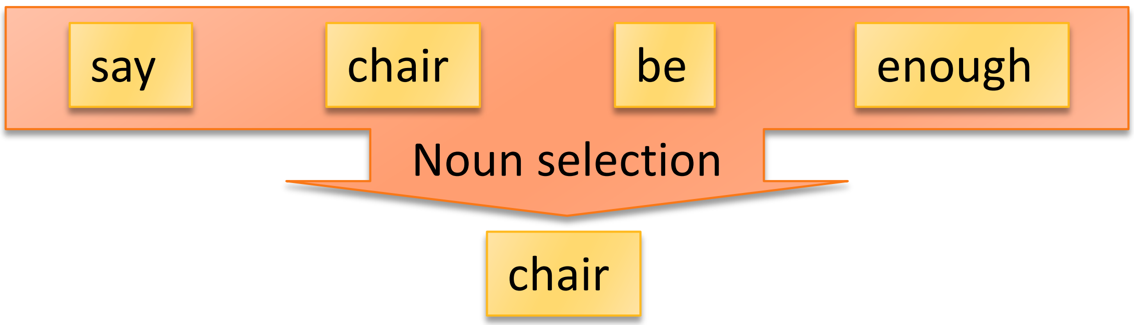
Manual Selection

- Selection of index terms is usually done by specialist

Automatic Selection of Index Terms

- Most of the semantics is carried by the noun words

- Ref. Example:



- How to automatically identify nouns? Parsers and Taggers

Other Text Operations

- In document preprocessing
  - Parsers
  - Taggers
  - Word Sense Disambiguation (see later)
- Improving efficiency
  - Text compression

Parsers and Taggers

- A **syntactic parser** is a tool that assigns a syntactic structure to any sentence in the language. It identifies:
  - Its parts, labeling each
  - The part of speech of every word
  - Semantic classes and functional classes, eventually
- It is based on a grammar
- The correctness of available general-purpose parsers do not exceed 90%
- The most promising approach is the statistical one, which is based on large bodies of readable parsed text (**treebank**)
  - Treebanks: Brown Corpus, LOB corpus, and Penn project
- Popular parsers: Link Grammar Parser, Apple Pie Parser, Cass

- **Taggers**: only assign to each word the part-of-speech (POS) it assumes in the context
  - Correctness 95-99% with reasonable efficiency
- Popular taggers: CLAWS, QTAG

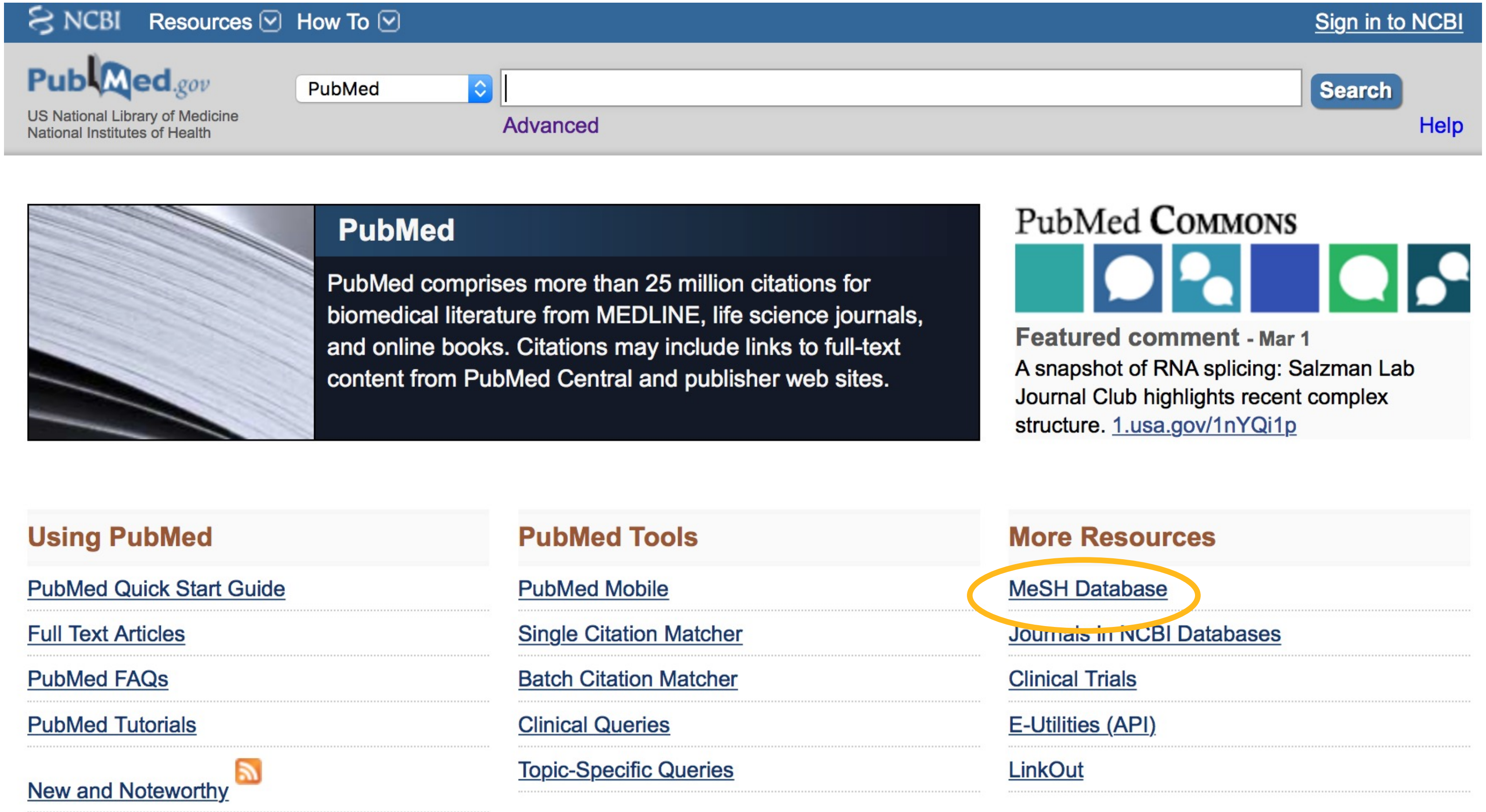
Contents

- Document Processing
- Thesauri
- Word similarities
- Word Sense Disambiguation
- Hands-on with Python and NLTK

What is a "Thesaurus"?

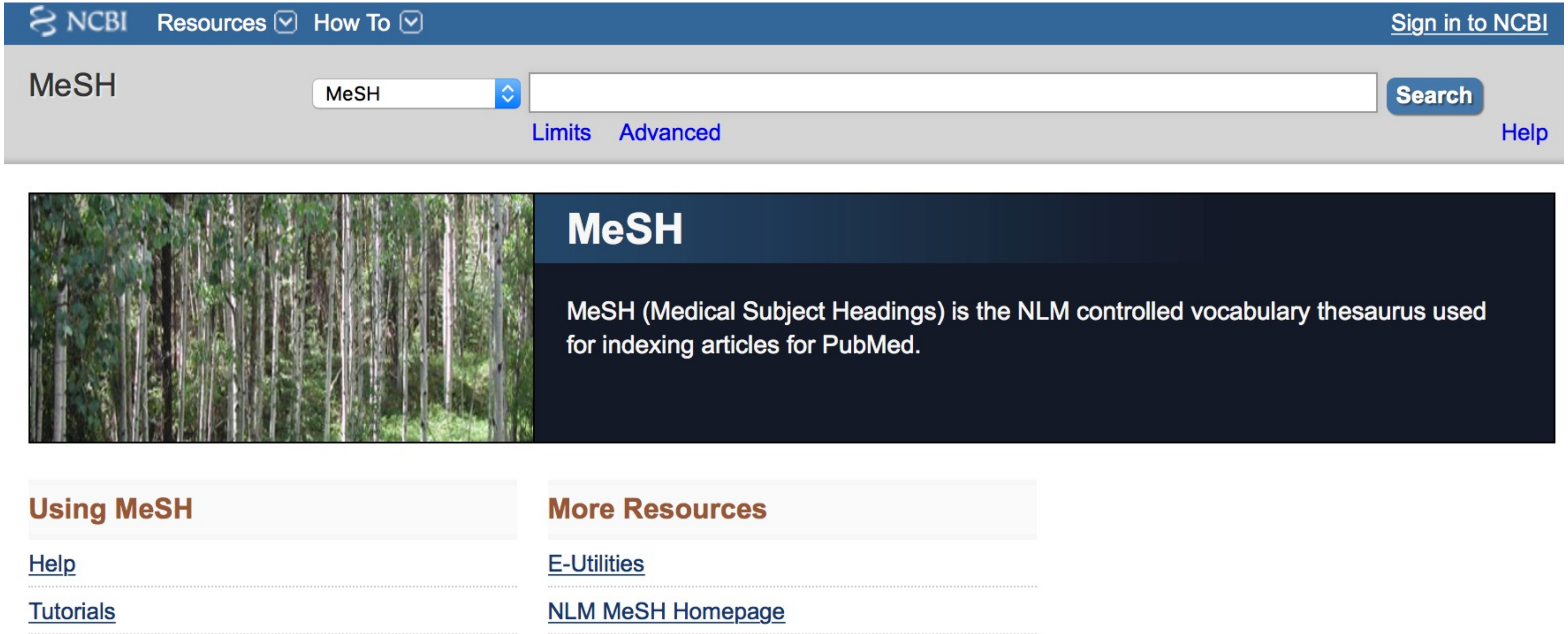
- A *list of* ***important words*** in a given domain of knowledge and for each word in this list, **a set of related words** such as common variation, derived from a **synonymity** relationship
- Examples of thesaurus
  - Roget (published in 1852), Wordnet, INSPEC thesaurus, MESH
- Thesauri can be constructed
  - Manually by domain experts
  - Automatically from a collection of documents or by merging existing thesauri
  - Semi-automatically with the help of domain experts

Example - Pubmed & MESH

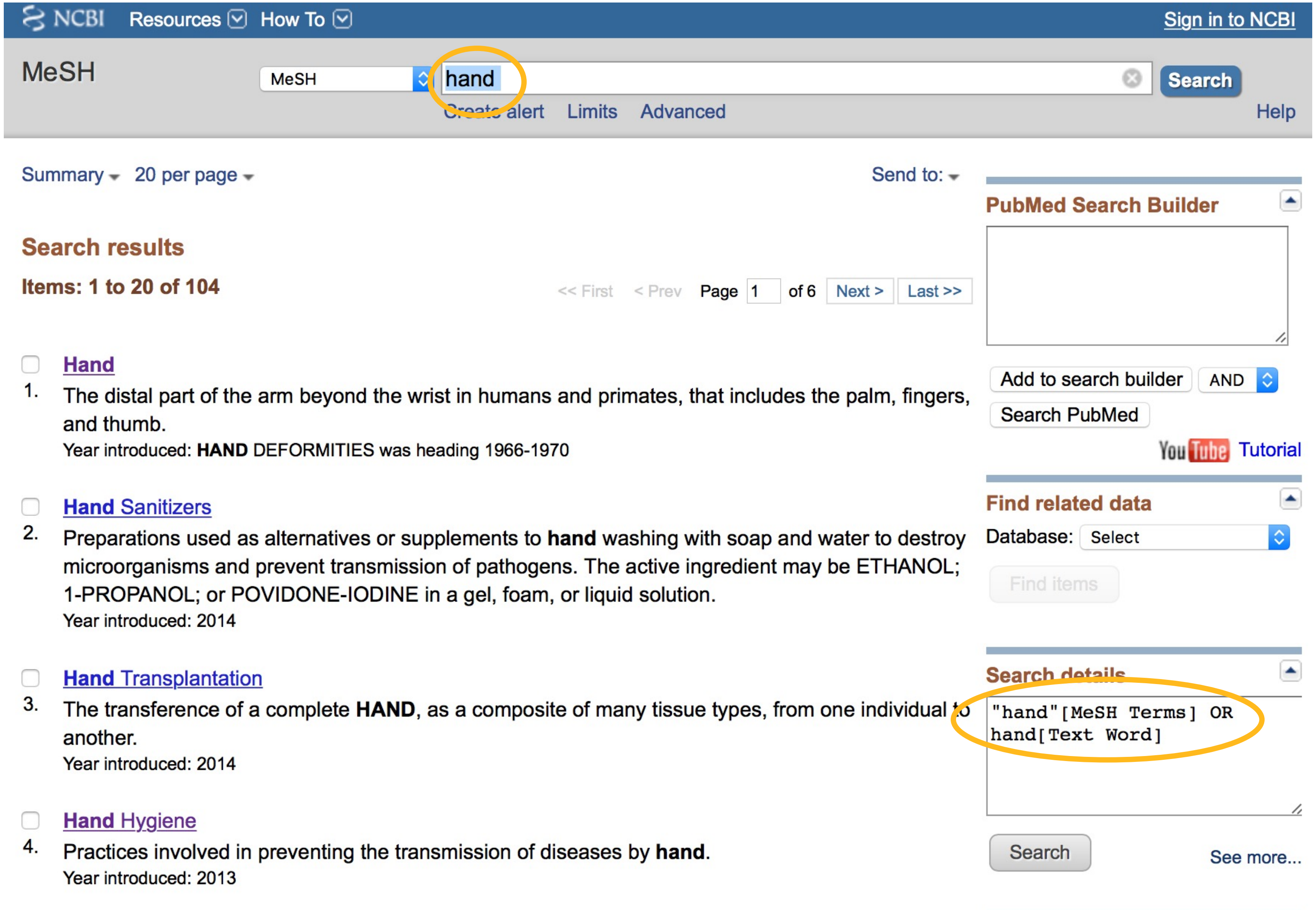


http://www.ncbi.nlm.nih.gov/pubmed

Example - Pubmed & MESH



Example - Pubmed & MESH



Example - Pubmed & MESH

Hand Hygiene

Practices involved in preventing the transmission of diseases by hand.
Year introduced: 2013

PubMed search builder options

Subheadings:

- [ ] economics
- [ ] history
- [ ] instrumentation
- [ ] methods
- [ ] organization and administration
- [ ] standards
- [ ] statistics and numerical data
- [ ] trends

- [ ] Restrict to MeSH Major Topic.
- [ ] Do not include MeSH terms found below this term in the MeSH hierarchy.

Tree Number(s): N06.850.780.200.412
MeSH Unique ID: D063373
Entry Terms:

- Hygiene, Hand

Previous Indexing:

- Hand Disinfection (1981-2012)

All MeSH Categories
Health Care Category
Environment and Public Health
Public Health
Public Health Practice
Communicable Disease Control
Hand Hygiene
Hand Disinfection

PubMed Search Builder

```
( "Hand Hygiene/methods"[Mesh] OR
"Hand Hygiene/trends"[Mesh] )
```

Add to search builder | AND
Search PubMed

Motivation for using a thesaurus

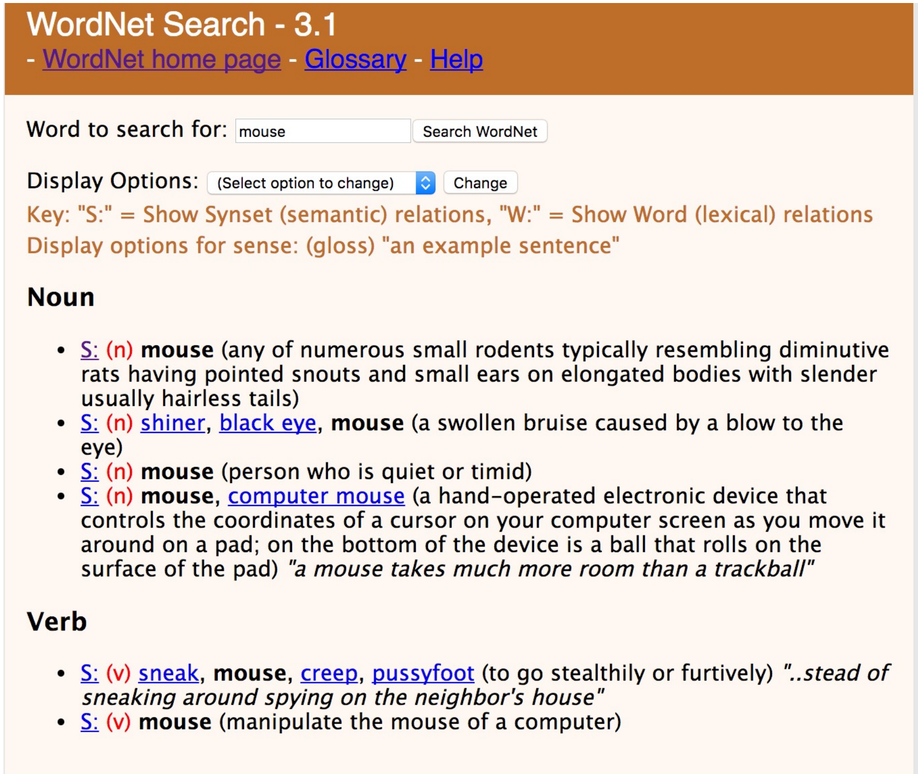
- Using *a controlled vocabulary* for
  - Indexing
    - Normalization of indexing concepts
    - Reduction of noise
    - Identification of indexing terms with a clear semantic
  - Searching
    - To assist users with proper query formulation
    - To provide classified hierarchies that allow the broadening and narrowing of the current query request
- Particularly important for specific domain (e.g. medicine)
- For general domain the usefulness is not clear

Thesaurus components

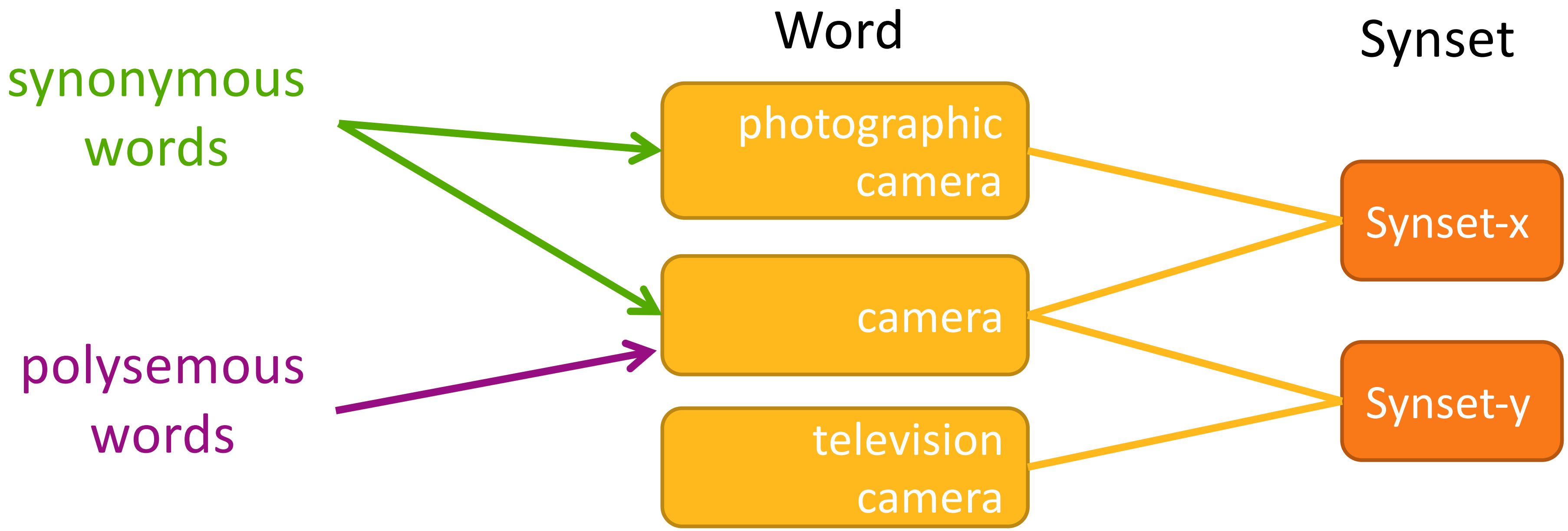
- Thesaurus Index Term
  - Used to denote a **concept** which is the basic semantic unit
  - Can be individual words, groups of words, or phrases
    E.g.) *Building, Teaching, Ballistic Missiles, Body Temperature*
  - Frequently, it is necessary to complement a thesaurus entry with a **definition** or an explanation
    - E.g.) *Seal (marine animals), Seal (documents)*
- Thesaurus Index Term Relationships
  - Mostly composed of synonyms and near-synonyms
  - BT(Broader Term), NT(Narrower Term), RT(Related Term)

WordNet thesaurus

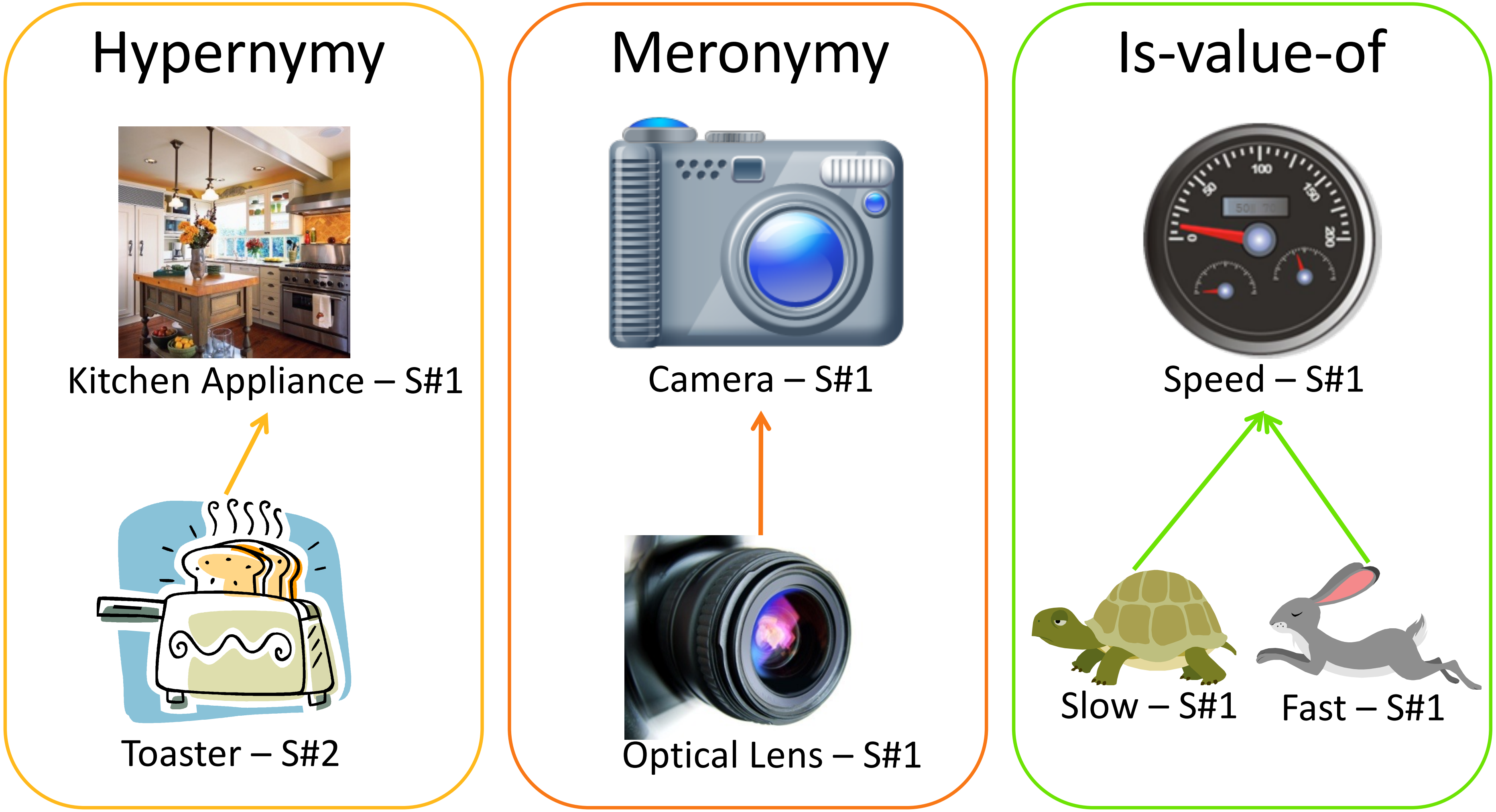
- WordNet Ontology - http://wordnet.princeton.edu/
  - It provides concepts from many domains
  - It can be easily extended to languages other than English
  - It presents relations between concepts which are easy to understand and use



WordNet: Lexical Database



WordNet: Semantic Relations

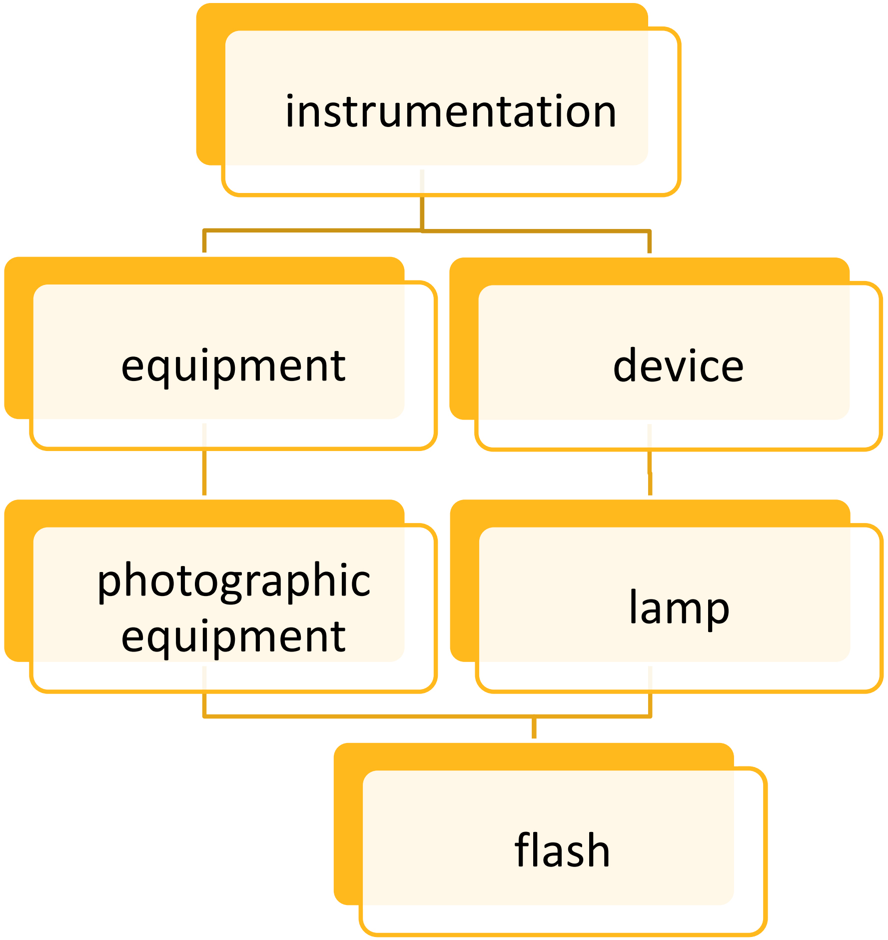


WordNet: Semantic Relations

Relation	Meaning	Examples
Synonymy (N, V, Adj, Adv)	Same sense	(camera, photographic camera) (mountain climbing, mountaineering) (fast, speedy)
Antonymy (Adj, Adv)	Opposite	(fast, slow) (buy, sell)
Hypernymy (N)	Is-A	(camera, photographic equipment) (mountain climbing, climb)
Meronymy (N)	Part	(camera, optical lens) (camera, view finder)
Troponymy (V)	Manner	(buy, subscribe) (sell, retail)
Entailment (V)	X must mean doing Y	(buy, pay) (sell, give)

WordNet: Hierarchy

Hypernymy Is-A relations



Use a thesaurus?

- Consequence of manual selection
  - Time consuming
  - The person using the retrieval system has to be familiar with the thesaurus
  - Thesauri are sometimes incoherent
- Consequence of automatic selection
  - Computationally too expensive in real-world settings
  - Coverage
  - Language dependence
  - Need of **Word Sense Disambiguation (WSD)** techniques (see later)
  - The resulting representations may be too explicit to deal with the vagueness of a user's information need
- Alternative: a document is simply an unstructured set of words appearing in it: **bag of words**

Contents

Word Similarity

- **Synonymy**: a binary relation
  - Two words are either synonymous or not
- **Similarity** (or **distance**): a looser metric
  - Two words are more similar if they share more features of meaning

- We often distinguish **word similarity** from **word relatedness**
  - **Similar words**: near-synonyms
  - **Related words**: can be related any way
    - `car, bicycle`: **similar**
    - `car, gasoline`: **related**, not similar

Why word similarity?

- Information retrieval
- Question answering
- Machine translation
- Natural language generation
- Language modeling
- Automatic essay grading
- Plagiarism detection
- Document clustering
- Word sense disambiguation
- …

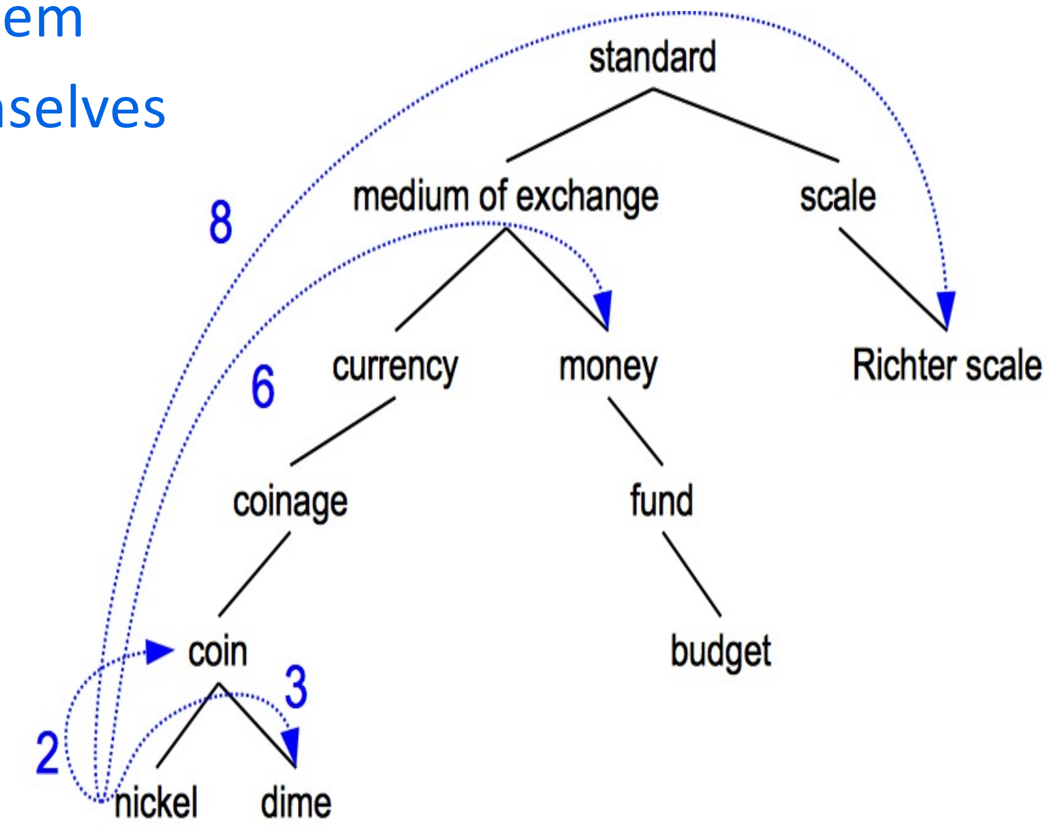
Two classes of similarity measures

- Path-based measures
  - E.g. Are words “nearby” in hypernym hierarchy?
- Information-content measures
  - Do words have similar distributional contexts?

Path-based similarities

Two concepts (senses/synsets) are similar if they are near each other in the thesaurus hierarchy

- have a short path between them
- concepts have path 1 to themselves



Path-based similarities

- **Path Distance Similarity**: based on the shortest path that connects the senses in the is-a (hypernym/hypnoym) taxonomy

$$sim_{path\text{-}distance}(c_1,c_2)=1/(shortest\text{-}path(c_1,c_2)+1)$$

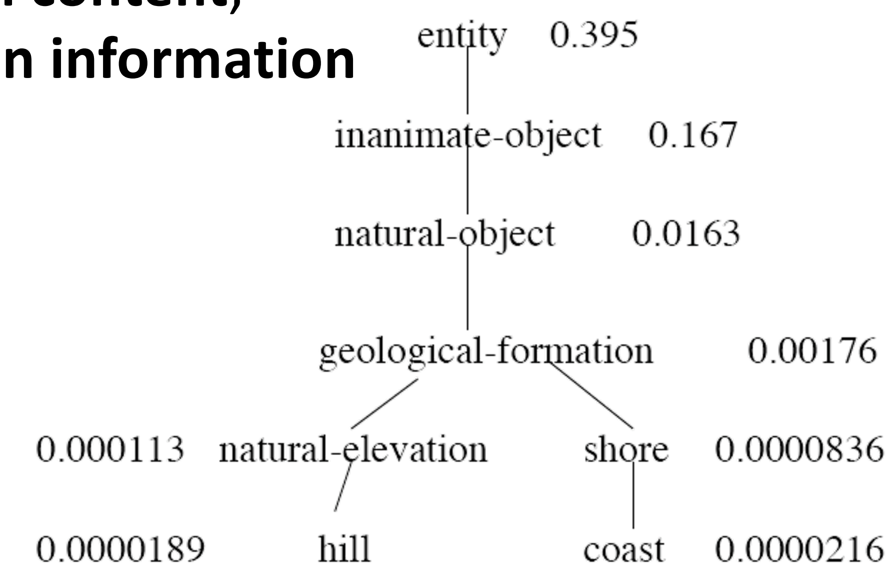
- **Wu-Palmer Similarity**: based on the depth of the two senses in the taxonomy and that of their Least Common Subsumer (most specific ancestor node)

$$simwu\text{-}palmer(c1,c2)= 2*depth(LCS(c1, c2)) / (depth(c1) + depth(c2))$$

- …

Information-content similarities

- The similarity between two senses is related to their common information
- The more two senses have in common, the more similar they are
- **P(c)**: is the probability that a randomly selected word in a corpus is an instance of concept c
- **Information-content:** $IC(c) = -\log P(c)$
- In information theory, the **information content**, **self-information**, **surprisal**, or **Shannon information** can be interpreted as quantifying the level of *surprise* of a particular sense
- IC(c) is high when c is a rare sense
- IC(c) is low when c is a frequent sense



Information-content similarities

- **Resnik similarity**: based on the Information Content (IC) of the Lowest Common Subsumer (most specific ancestor node)
- Measures common information as:
  - $sim_{resnik}(c_1,c_2) = -\log P(\ LCS(c_1,c_2)\ )$
- Other IC similarities:
  - Jiang-Conrath
  - Lin
  - …

Contents

Word Sense Disambiguation

- WSD involves the association of a given word in a text with a definition or meaning
- Two steps:
  - Determine all the different senses
  - Assign each occurrence of a word to the appropriate sense
- First step: adoption of a dictionary or thesaurus
  - The results depend on the adopted solution
- Second step: Analysis of the context
  - Bag of words approach: the context is a window of words next to the term to disambiguate
  - Relational information approach: along with the bag of words other information such as their distance are also extracted

An approach for Word Sense Disambiguation

- WordNet is used as reference thesaurus

```
For each noun N, for each WordNet sense s_N of N
    Compute the confidence C_{s_N} in choosing s_N as
    sense of N
Select the sense with the highest confidence
```

- Noun sense disambiguation
  - The **context** for each noun is the set of the other nouns in the sentence
  - Intuition
    - If two polysemic words in the context are similar then their similar concepts provide information about the most suitable meanings

An approach for Word Sense Disambiguation

- The confidence C_{s_N} in choosing s_N as sense of N is influenced by
  - The similarity between s_N and all the senses of the nouns in the context
  - The frequency of that sense
- The similarity between two senses is influenced by
  - The distance in the hypernymy hierarchy of WordNet (see path-based similarities)
  - If a relational-based approach is adopted:
    The distance among the involved words

An approach for Word Sense Disambiguation

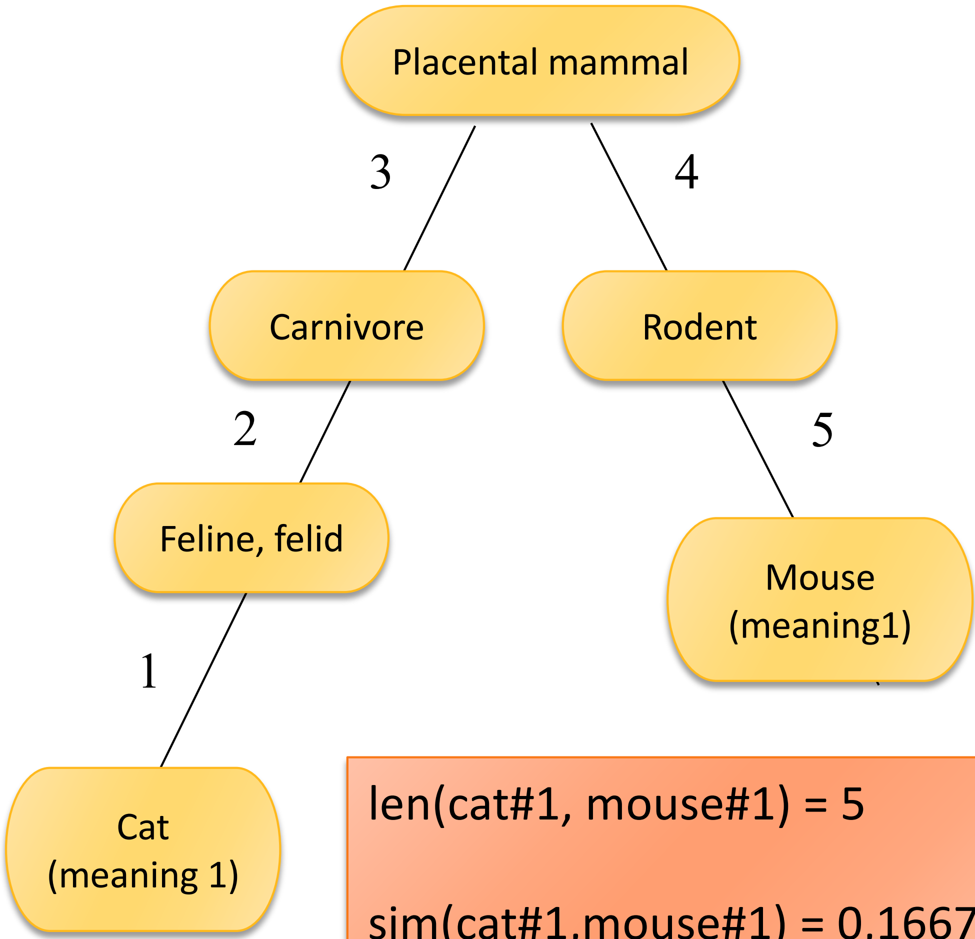
Example:

"The **cat** is hunting the **mouse**"

The highest confidence among the meanings of "cat" is the one in the hierarchy. The same for "mouse".

→Meaning of Cat: #1

→Meaning of Mouse: #1



len(cat#1, mouse#1) = 5

sim(cat#1,mouse#1) = 0,1667

Contents

- Document Processing
- Thesauri and Word Sense Disambiguation
- Hands-on with Python and NLTK

Introduction to NLTK

`https://www.nltk.org/`

- The Natural Language Toolkit (NLTK) provides:
  - Basic classes for representing data relevant to natural language processing
  - Several text processing utilities, corpora
    - Brown, Penn Treebank corpus…
  - Standard interfaces for performing tasks, such as tokenization, tagging, and parsing.
  - Standard implementations of each task, which can be combined to solve complex problems

`http://www.nltk.org/book/`

- Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit

Installing NLTK

- Download
  - http://nltk.org/
- Install
  - Windows:
    - Right click and run the three installers for Python, PyYAML, and NLTK
  - Mac OSX:
    - Similar, but some terminal work required for PyYAML
- Download NLTK data

```
>>> import nltk
>>> nltk.download()
```

Download the collection named book

NLTK

Ok, now let's try something!

NLTK – Tokenization

- **import nltk**

- **text = "This is a test"**
- **tokens = nltk.word_tokenize(text)**

- **print(tokens)**

  ['This', 'is', 'a', 'test’]

NLTK – Stopwords removal & Lemmatization

- **from nltk.corpus import stopwords**

- **wnl = nltk.WordNetLemmatizer()**

- **for t in tokens:**

```
for t in tokens:
    if not t in stopwords.words('english'):
        print(wnl.lemmatize(t))
```

This

test

NLTK –Stemmers

Porter & Lancaster: very popular stemmers

- **from nltk.stem.porter import PorterStemmer**
- **porter = PorterStemmer()**
- **print([porter.stem(t) for t in tokens])**

- **from nltk.stem.lancaster import LancasterStemmer**
- **lancaster = LancasterStemmer()**
- **print([lancaster.stem(t) for t in tokens])**

thi

is
a
test

NLTK – POS Tagging

- **print(nltk.pos_tag(tokens))**

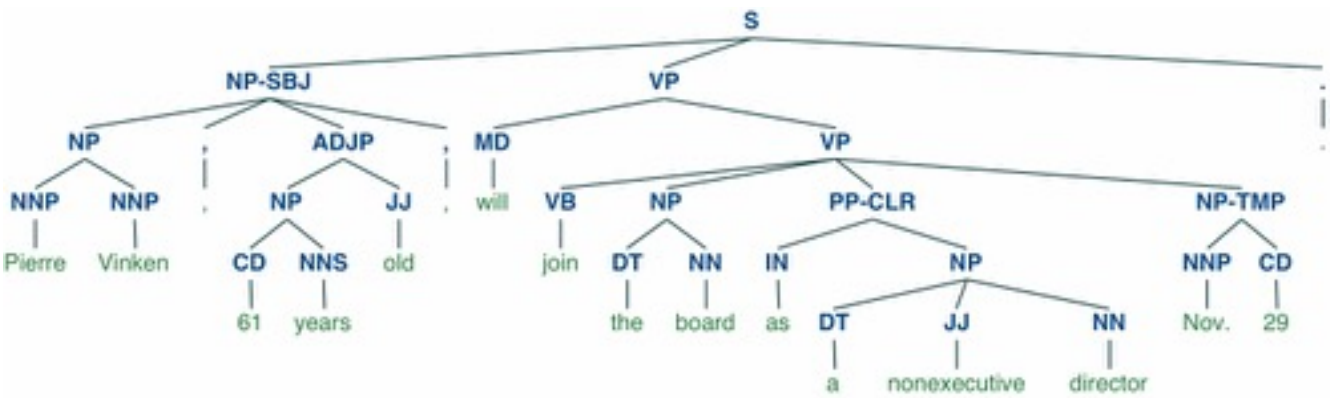
  [('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('test', 'NN')]

- **nltk.help.upenn_tagset()**

  # pos tag list

NLTK - Parsing

- **from nltk.corpus import** treebank
- t = treebank.parsed_sents('wsj_0001.mrg')[0]
- t.draw()



- NLTK Parsing: https://www.nltk.org/book/ch08.html

Exercise 1

- Go to the NLTK book Web page
- Download the content of one of the online books of Project Gutenberg and convert it in utf8 by following the tips of Section 3 “Processing Raw Text”

Implement the following pre-processing phases:

- Tokenization
- Elimination of stopwords
- Stemming
- Selection of nouns

Given a text item, your program will therefore output the keywords that could be used to index it.

NLTK – WordNet

- **from nltk.corpus import wordnet as wn**
- **print(wn.synsets('dog’))**

  [Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'), Synset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01'), Synset('chase.v.01')]

- **print(wn.synsets('dog’,wn.VERB))**

  [Synset('chase.v.01')]

- **dog = wn.synset('dog.n.01’)**
- **print(dog.definition())**

  a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds

NLTK – WordNet

- **print(dog.examples())**

  ['the dog barked all night’]

- **print(dog.hypernyms())**

  [Synset('domestic_animal.n.01'), Synset('canine.n.02')]

NLTK – WordNet Morphy

- Look up forms not in WordNet, with the help of Morphy

- **print(wn.morphy('denied',wn.VERB))**
  deny

- **print(wn.morphy('abaci'))**
  abacus

NLTK – Path-Distance Similarity

- **cat = wn.synset('cat.n.01')**
- **computer = wn.synset('computer.n.01’)**
- **print(dog.path_similarity(cat))**

  0.2

- **print(dog.path_similarity(computer))**

  0.0909090909091

NLTK – Wu-Palmer Similarity

- **print(dog.wup_similarity(cat))**

  0.857142857143

- **print(dog.wup_similarity(computer))**

  0.444444444444

NLTK – Resnik Similarity

- **from nltk.corpus import wordnet_ic**
- **brown_ic = wordnet_ic.ic('ic-brown.dat')**
- **print(dog.res_similarity(cat,brown_ic))**

  7.91166650904

- **print(dog.res_similarity(computer,brown_ic))**

  1.53183374322

A (very) simple WSD algorithm with NLTK

- Let’s try a simple algorithm for the disambiguation of terms, exploiting one of the term similarity formulas
- Basic idea:
  - Given a list of terms {t_1,…,t_n}
    - for instance {t_1,…,t_n} can be the keywords of a document
  - Disambiguate each term t_i in the list by exploiting the context provided by the other terms {t_1,…,t_i-1,t_i+1,…,t_n}
  - For each sense s_ti of t_i compute a score (confidence) by:
    - Considering each term t_j in the context (t_i ≠ t_j)
    - Adding to the score the similarities between s_ti and the sense s_tj of t_j which is the most similar to s_ti
  - The sense s_ti with the highest score will be the most probable one

A (very) simple WSD algorithm with NLTK

```
def disambiguateTerms(terms):
	for t_i in terms:    # t_i is target term
		selSense = None
		selScore = 0.0
		for s_ti in wn.synsets(t_i, wn.NOUN):
			score_i = 0.0
			for t_j in terms:    # t_j term in t_i's context window
				if (t_i==t_j):
					continue
				bestScore = 0.0
				for s_tj in wn.synsets(t_j, wn.NOUN):
					tempScore = s_ti.wup_similarity(s_tj)
					if (tempScore>bestScore):
						bestScore=tempScore
				score_i = score_i + bestScore
			if (score_i>selScore):
				selScore = score_i
				selSense = s_ti
		if (selSense is not None):
			print(t_i,": ",selSense,", ",selSense.definition())
			print("Score: ",selScore)
		else:
			print(t_i,": --")
```

A (very) simple WSD algorithm with NLTK

- **from nltk.corpus import wordnet as wn**

  …

- **disambiguateTerms(["cat","mouse"])**

  cat : Synset('cat.n.01') , feline mammal usually having thick soft fur and no ability to roar: domestic cats; wildcats

  Score: 0.814814814815

  mouse : Synset('mouse.n.01') , any of numerous small rodents typically resembling diminutive rats having pointed snouts and small ears on elongated bodies with slender usually hairless tails

  Score: 0.814814814815

A (very) simple WSD algorithm with NLTK

- **disambiguateTerms(["computer","mouse"])**

  computer : Synset('computer.n.01') , a machine for performing calculations automatically

  Score: 0.777777777778

  mouse : Synset('mouse.n.04') , a hand-operated electronic device that controls the coordinates of a cursor on your computer screen as you move it around on a pad; on the bottom of the device is a ball that rolls on the surface of the pad

  Score: 0.777777777778

Exercise 3: Thesaurus-based query expansion

- **Query expansion** is the process of supplementing additional terms or phrases to the original query to improve the retrieval performance
- The central problem of query expansion is the selection of the expansion terms based on which user's original query is expanded
- One possibility is to exploit a thesaurus like Wordnet
- Write a Python algorithm that given a query expressed as a set of words, expands the input query by adding all (or some) word synonyms

Keyword extraction: interesting libraries and readings

- *spaCy: all in one python library for NLP tasks*
  - *spaCy home page https://spacy.io/*
- *RAKE: Rapid Automatic Keyword Extraction*
  - The algorithm is described here: https://www.researchgate.net/publication/227988510_Automatic_Keyword_Extraction_from_Individual_Documents
  - Python implementation of RAKE using NLTK is available here: https://pypi.org/project/rake-nltk
- YAKE Yet Another Keyword Extractor (Yake) library
  - light-weight unsupervised automatic keyword extraction method
  - selects the most important keywords using the text statistical features
  - Full source is available here https://github.com/LIAAD/yake
- Web pages:
  - https://textminingonline.com/getting-started-with-keyword-extraction
  - https://www.airpair.com/nlp/keyword-extraction-tutorial
  - https://towardsdatascience.com/keyword-extraction-process-in-python-with-natural-language-processing-nlp-d769a9069d5c

Exercise 4

- Extends the code of Exercise 1 by adding the packages for keyword extraction listed in slides 69-70
- Compare the results of the different approaches for the selection of the index terms

Bibliography

- R. Baeza-Yates, B. Ribeiro-Neto "Modern Information Retrieval" Addison Wesley
- W. B. Frakes, R. Baeza-Yates "Information Retrieval" Prentice Hall
- F. Mandreoli, R. Martoglia, P. Tiberio, "EXTRA: A System for Example-based Translation Assistance" In Machine Translation, Volume 20, Number 3, 2006
- F. Mandreoli, R. Martoglia, E. Ronchetti, "Versatile Structural Disambiguation for Semantic-aware applications", in proc. of ACM CIKM 2005
- Philip Resnik. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. IJCAI 1995.
- Philip Resnik. 1999. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. JAIR 11, 95-130.