Introduction to Natural Language Processing

Natural Language and Dialogue Systems
http://nlds.soe.ucsc.edu

Center for Games and Playable Media
http://games.soe.ucsc.edu

Course Information

- Course webpage:
  - https://cmps143-spring16-01.courses.soe.ucsc.edu/home
- Check out
  - Syllabus, Announcements, Assignments, Lecture slides
- eCommons will be ready by the end of the week (hopefully!)
- Python 3 and NLTK 3 in the lab section. For interactive mode use the following commands:

```
> python3
> import nltk
```

GROUND RULES:

Do your homework (not at the last minute)!

Go to Section whenever you need!

Ask questions, talk in class!

Slow me down when I go too fast!

Today's Outline

- Working with data in NLTK
- Tokenization and Sentence Detection
- Some normalization: Stemming and Lemmatization
- Collocations
- Part-of-speech (POS) Tagging
- Introduction to Lexical Resources
  - WordNet

HW1: Working with Corpora
Due Friday, April 8

Simple comparison of two corpora: Fables and Blogs.
This is an easy homework, and it's a chance to play around a lot and try different things from Ch. 1 to 4.

If you are not comfortable with Python, I suggest you use this week to go through all the sections in the book that are trying to introduce you to Python.

Intro to Python in Chap 4: not in class

4 Writing Structured Programs

By now you will have a sense of the capabilities of the Python programming language for processing natural language. However, if you're new to Python or to programming, you may still be wrestling with Python and not feel like you are in full control yet. In this chapter we'll address the following questions:

1. How can you write well-structured, readable programs that you and others will be able to re-use easily?
2. How do the fundamental building blocks work, such as loops, functions and assignment?
3. What are some of the pitfalls with Python programming and how can you avoid them?

Along the way, you will consolidate your knowledge of fundamental programming constructs, learn more about using features of the Python language in a natural and concise way, and learn some useful techniques in visualizing natural language data. As before, this chapter contains many examples and exercises (and as before, some exercises introduce new material). Readers new to programming should work through them carefully and consult other introductions to programming if necessary; experienced programmers can quickly skim this chapter.

In the other chapters of this book, we have organized the programming concepts as dictated by the needs of NLP. Here we revert to a more conventional approach where the material is more closely tied to the structure of the programming language. There's not room for a complete presentation of the language, so we'll just focus on the language constructs and idioms that are most important for NLP.

4.1 Back to the Basics

Assignment

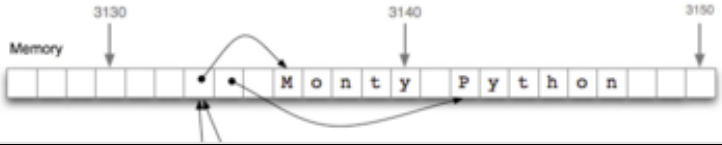
Assignment would seem to be the most elementary programming concept, not deserving a separate discussion. However, there are some surprising subtleties here. Consider the following code fragment:

```
>>> foo = 'Monty'
>>> bar = foo ❶
>>> foo = 'Python' ❷
>>> bar
'Monty'
```

This behaves exactly as expected. When we write `bar = foo` in the above code ❶, the value of `foo` (the string `'Monty'`) is assigned to `bar`. That is, `bar` is a **copy** of `foo`, so when we overwrite `foo` with a new string `'Python'` on line ❷, the value of `bar` is not affected.

However, assignment statements do not always involve making copies in this way. Assignment always copies the value of an expression, but a value is not always what you might expect it to be. In particular, the "value" of a structured object such as a list is actually just a *reference* to the object. In the following example, ❶ assigns the reference of `foo` to the new variable `bar`. Now when we modify something inside `foo` on line ❷, we can see that the contents of `bar` have also been changed.

```
>>> foo = ['Monty', 'Python']
>>> bar = foo ❶
>>> foo[1] = 'Bodkin' ❷
>>> bar
['Monty', 'Bodkin']
```



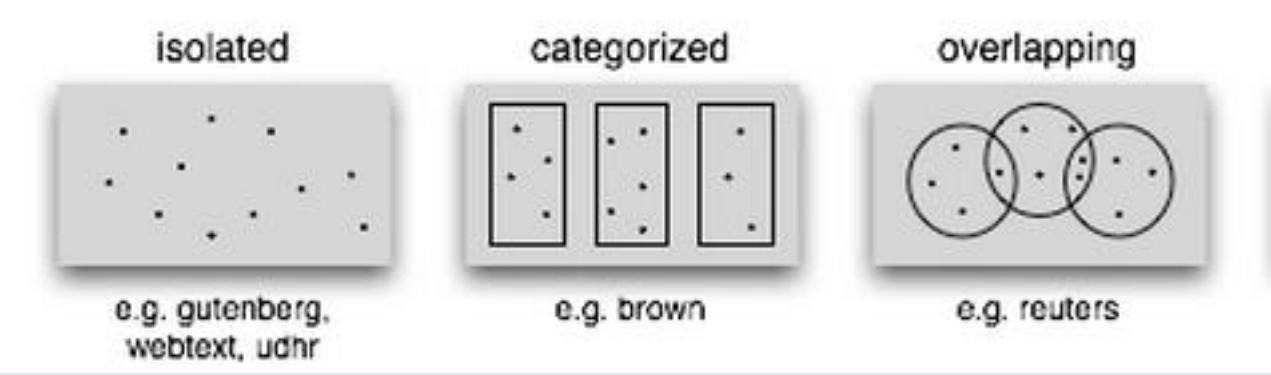
NLTK Corpora (Ch. 2)

- NLTK comes with lots of corpora
- Corpora may have structure & annotations
  - More info at http://www.nltk.org/howto/corpus.html

Table 1.2:

Some of the Corpora and Corpus Samples Distributed with NLTK: For information about downloading and using them, please consult the NLTK website.

Corpus	Compiler	Contents
Brown Corpus	Francis, Kucera	15 genres, 1.15M words, tagged, categorized
CESS Treebanks	CLiC-UB	1M words, tagged and parsed (Catalan, Spanish)
Chat-80 Data Files	Pereira & Warren	World Geographic Database
CMU Pronouncing Dictionary	CMU	127k entries
CoNLL 2000 Chunking Data	CoNLL	270k words, tagged and chunked
CoNLL 2002 Named Entity	CoNLL	700k words, pos- and named-entity-tagged (Dutch, Spanish)
CoNLL 2007 Dependency Treebanks (sel)	CoNLL	150k words, dependency parsed (Basque, Catalan)
Dependency Treebank	Narad	Dependency parsed version of Penn Treebank sample
FrameNet	Fillmore, Baker et al	10k word senses, 170k manually annotated sentences
Floresta Treebank	Diana Santos et al	9k sentences, tagged and parsed (Portuguese)
Gazetteer Lists	Various	Lists of cities and countries
Genesis Corpus	Misc web sources	6 texts, 200k words, 6 languages
Gutenberg (selections)	Hart, Newby, et al	18 texts, 2M words
Inaugural Address Corpus	CSpan	US Presidential Inaugural Addresses (1789-present)
Indian POS-Tagged Corpus	Kumaran et al	60k words, tagged (Bangla, Hindi, Marathi, Telugu)
MacMorpho Corpus	NILC, USP, Brazil	1M words, tagged (Brazilian Portuguese)
Movie Reviews	Pang, Lee	2k movie reviews with sentiment polarity classification
Names Corpus	Kantrowitz, Ross	8k male and female names
NIST 1999 Info Extr (selections)	Garofolo	63k words, newswire and named-entity SGML markup
Nombank	Meyers	115k propositions, 1400 noun frames
NPS Chat Corpus	Forsyth, Martell	10k IM chat posts, POS-tagged and dialogue-act tagged
Open Multilingual WordNet	Bond et al	15 languages, aligned to English WordNet
PP Attachment Corpus	Ratnaparkhi	28k prepositional phrases, tagged as noun or verb modifiers
Proposition Bank	Palmer	113k propositions, 3300 verb frames
Question Classification	Li, Roth	6k questions, categorized
Reuters Corpus	Reuters	1.3M words, 10k news documents, categorized
Roget's Thesaurus	Project Gutenberg	200k words, formatted text
RTE Textual Entailment	Dagan et al	8k sentence pairs, categorized



Review: Getting some data: NLPP Ch. 1

```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
>>> text1
<Text: Moby Dick by Herman Melville 1851>
>>> text1.concordance("monstrous")
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
```

Review: Tools for counting in NLTK

```
>>> len(text3)
44764
>>>
```

- Genesis has 44,764 words and punctuation symbols, or "tokens."

```
>>> len(set(text3))
2789
>>>
```

- Discover the size of the vocabulary indirectly, by asking for the number of items in the set, and use **len** to obtain this number

```
>>> len(set(text3)) / len(text3)
0.06230453042623537
>>>
```

- Calculate a measure of the lexical richness of the text: number of distinct words is just 6% of the total number of words

```
>>> text3.count("smote")
5
>>> 100 * text4.count('a') / len(text4)
1.4643016433938312
>>>
```

- Count how often a word occurs in a text, and compute what percentage of the text is taken up by a specific word.

Working With Your Own Data

```
>>> fables_text = open('cmps143-data/fables/TheFoxAndTheCrow
.txt').read()
>>> sentences = nltk.sent_tokenize(fables_text)
>>> sentences
['A Crow was sitting on a branch of a tree with a piece of c
heese in her beak when a Fox observed her and set his wits t
o work to discover some way of getting the cheese.', 'Coming
 and standing under the tree he looked up and said, "What a
noble bird I see above me!', 'Her beauty is without equal, t
he hue of her plumage exquisite.', 'If only her voice is as
sweet as her looks are fair, she ought without doubt to be Q
ueen of the Birds."', 'The Crow was hugely flattered by this
, and just to show the Fox that she could sing she gave a lo
ud caw.', 'Down came the cheese,of course, and the Fox, snat
ching it up, said, "You have a voice, madam, I see: what you
 want is wits."']
```

Working With Your Own Data

- Reading from a file

```
>>> print (fables_text[:170])
A Crow was sitting on a branch of a tree with a piece of cheese in her beak when a Fox obs
erved her and set his wits to work to discover some way of getting the cheese.
```

2 3 4 5

A Crow was sitting on a branch of a tree

0 1 6 7 8 9 ...

```
print(fable_text[:5])  ⟶  A Cro
```

Tokenization

- **Tokenization** is the task of cutting a string into identifiable linguistic units that constitute a piece of language data.
  - Many corpora are already tokenized
  - NLTK includes some tokenizers
- Divide text into units called tokens (words, numbers, punctuations, …)

- **What is a word?**
- **Practical definition**:
  - an indivisible (!) sequence of characters
  - carries elementary meaning
  - is reusable in different contexts

Tokenization

- Whitespace does not always indicate a word break
- Punctuation
  - “You reminded **me,”** she remarked
  - **O’Hara** vs. **John’s**
- Compound words
  - San Francisco
  - The New York-New Heaven railroad
  - Wake up, work out
    - I couldn’t ***work*** the answer ***out***
- Contractions
  - can’t, won’t, etc.
- Merged words
  - Wanna, shoulda, etc.

Blogs01.txt

Today was a very eventful work day. Today was the start of the G20 summit. It happens every year and it is where 20 of the leaders of the world come together to talk about how to run their governments effectively and what not. Since there are so many leaders coming together their are going to be a lot of people who have different views on how to run the government they follow so they protest. There was a protest that happened along the street where I work and at first it looked peaceful until a bunch of people started rebelling and creating a riot. Police cars were burned and things were thrown at cops. Police were in full riot gear to alleviate the violence. As things got worse tear gas and bean bag bullets were fired at the rioters while they smash windows of stores. And this all happened right in front of my store which was kind of scary but it was kind of interesting since I've never seen a riot before.

Tokenization

- Simplest tokenization by whitespace

```
>>> for sent in sents:
...     print sent.split(),
...
['Saturday,', 'June', '26,', '2010', 'eventful', 'it', 'is', 'Today', 'was', 'a', 'very',
'eventful', 'work', 'day.']
['Today', 'was', 'the', 'start', 'of', 'the', 'G20', 'summit.']
['It', 'happens', 'every', 'year', 'and', 'it', 'is', 'where', '20', 'of', 'the', 'leaders',
'of', 'the', 'world', 'come', 'together', 'to', 'talk', 'about', 'how', 'to', 'run', 'their',
'governments', 'effectively', 'and', 'what', 'not.']
['Since', 'there', 'are', 'so', 'many', 'leaders', 'coming', 'together', 'their', 'are',
'going', 'to', 'be', 'a', 'lot', 'of', 'people', 'who', 'have', 'different', 'views', 'on',
'how', 'to', 'run', 'the', 'government', 'they', 'follow', 'so', 'they', 'protest.']
['There', 'was', 'a', 'protest', 'that', 'happened', 'along', 'the', 'street', 'where', 'I',
'work', 'and', 'at', 'first', 'it', 'looked', 'peaceful', 'until', 'a', 'bunch', 'of',
'people', 'started', 'rebelling', 'and', 'creating', 'a', 'riot.']
['Police', 'cars', 'were', 'burned', 'and', 'things', 'were', 'thrown', 'at', 'cops.']
['Police', 'were', 'in', 'full', 'riot', 'gear', 'to', 'alleviate', 'the', 'violence.']
['As', 'things', 'got', 'worse', 'tear', 'gas', 'and', 'bean', 'bag', 'bullets', 'were',
'fired', 'at', 'the', 'rioters', 'while', 'they', 'smash', 'windows', 'of', 'stores.']
['And', 'this', 'all', 'happened', 'right', 'in', 'front', 'of', 'my', 'store', 'which',
'was', 'kind', 'of', 'scary', 'but', 'it', 'was', 'kind', 'of', 'interesting', 'since',
"I've", 'never', 'seen', 'a', 'riot', 'before.']
['Since', 'it', 'all', 'happened', 'in', 'front', 'of', 'close', 'to', 'my', 'store,', 'my',
'coworkers', 'and', 'I', 'were', 'stuck', 'in', 'the', 'store', 'until', 'it', 'was', 'safe',
'to', 'come', 'out', 'so', 'we', 'saw', 'everything', 'happen', 'from', 'start', 'to',
'finish.']
['Sucks', 'that', 'everything', 'they', 'were', 'protesting', 'is', 'now', 'a', 'lost',
'cause', 'since', 'everyone', 'will', 'just', 'remember', 'the', 'riot', 'and', 'violence',
'they', 'created.']
['However,', 'I', 'bet', 'the', 'police', 'wanted', 'this', 'to', 'happen', 'just', 'to',
'justify', 'that', 'the', 'money', 'they', 'spent', 'for', 'the', 'extra', 'security', 'was',
'well', 'worth', 'it.']
['Posted', 'by']
```

Some Problems

- What if we tried to count the word frequencies?

26,	1	Sucks	1
And	1	There	1
As	1	about	1
G20	1	alleviate	1
However,	1	along	1
I've	1	bag	1
It	1	be	1
June	1	bean	1
Posted	1	before.	1
Saturday,	1		

Tokenization

- Tokenization turns out to be a far more difficult task than you might have expected.
  - No single solution works well across-the-board,
  - What counts as a token depending on the application
- When developing a tokenizer it helps to have access to raw text which has been manually tokenized
  - Compare the output of your tokenizer with high-quality **("gold-standard")** tokens.
  - Sample of **Penn Treebank** data, including the raw Wall Street Journal text (nltk.corpus.treebank_raw.raw()) and the tokenized version (nltk.corpus.treebank.words()).
- Contractions, such as *didn't*.
  - Normalize this form to two separate forms: *did*and *n't* (or *not*): a lookup table.

Tokenization

- Other (better) approaches
  - Define a set of rules to split punctuation
  - Use machine learning to identify word boundaries
  - Look for reoccurring patterns in large corpora
- Default tokenizer in NLTK uses option 3 (*Punkt,* (Kiss and Strunk, 2006))
  - Uses distributional definition of words to identify boundaries
  - Similar to the sentence delimiting module

Sentence Detection

- Sentence:
  - Something ending with a ., ?, ! (and sometime also :)
  - “You reminded me,” she remarked, “of your mother.”
    - Nested sentences
    - Note the .”
- Before tokenizing the text into words, we need to segment it into sentences.

Sentence Detection

- NLTK uses the *Punkt* sentence segmenter (Kiss and Strunk, 2006)
- Most ambiguities are from abbreviations
- Uses 3 simple rules to identify them
  - look for very tight collocation consisting of a truncated word and a final period
    - abbreviations are usually short
    - abbreviations can contain internal periods

```
>>> text = nltk.corpus.gutenberg.raw('chesterton-thursday.txt')
>>> sents = nltk.sent_tokenize(text)
>>> pprint.pprint(sents[79:89])
['"Nonsense!"',
 'said Gregory, who was very rational when anyone else\nattempted paradox.',
 '"Why do all the clerks and navvies in the\n'
 'railway trains look so sad and tired, so very sad and tired?',
 'I will\ntell you.',
 'It is because they know that the train is going right.',
 'It\n'
 'is because they know that whatever place they have taken a ticket\n'
 'for that place they will reach.',
 'It is because after they have\n'
 'passed Sloane Square they know that the next station must be\n'
 'Victoria, and nothing but Victoria.',
 'Oh, their wild rapture!',
 'oh,\n'
 'their eyes like stars and their souls again in Eden, if the next\n'
 'station were unaccountably Baker Street!"',
 '"It is you who are unpoetical," replied the poet Syme.']
```

Review: Tools for counting in NLTK

- What makes a text distinct?
- Automatically identify the words of a text that are most informative about the topic and genre of the text
- **Frequency Distribution:** the frequency of each vocabulary item in the text.

```
>>> fdist1 = FreqDist(text1) ❶
>>> print(fdist1) ❷
<FreqDist with 19317 samples and 260819 outcomes>
>>> fdist1.most_common(50) ❸
[(',', 18713), ('the', 13721), ('.', 6862), ('of', 6536), ('and', 6024),
('a', 4569), ('to', 4542), (';', 4072), ('in', 3916), ('that', 2982),
("'", 2684), ('-', 2552), ('his', 2459), ('it', 2209), ('I', 2124),
('s', 1739), ('is', 1695), ('he', 1661), ('with', 1659), ('was', 1632),
('as', 1620), ('"', 1478), ('all', 1462), ('for', 1414), ('this', 1280),
('!', 1269), ('at', 1231), ('by', 1137), ('but', 1113), ('not', 1103),
('--', 1070), ('him', 1058), ('from', 1052), ('be', 1030), ('on', 1005),
('so', 918), ('whale', 906), ('one', 889), ('you', 841), ('had', 767),
('have', 760), ('there', 715), ('But', 705), ('or', 697), ('were', 680),
('now', 646), ('which', 640), ('?', 637), ('me', 627), ('like', 624)]
>>> fdist1['whale']
906
>>>
```

Review: Ngrams & Counting Other Things

- Can basically count anything with `FreqDist`
- **N-grams:** sequences of *n* consecutive words e.g., "more is said than done"
  - Unigrams: "more", "is", "said", "than", "done"
  - Bigrams: "more is", "is said", "said than", "than done"
  - Trigrams: "more is said", "is said than", "said than done"
  - …
- Used a lot in NLP applications
  - Language models (next week)
  - Collocation (next)
  - Language Identification
  - Machine Translation

Review: Conditional Counts

- We can also get some more interesting information by using a `ConditionalFreqDist`
- How often have I seen $word_2$ given that $word_1$ immediately preceded it?
  - *fox* is seen exactly twice after having seen *the*

```
>>> import nltk
>>> fables_text = open('cmps143/fables/TheFoxAndTheCrow.txt').read()
>>> sentences = nltk.sent_tokenize(fables_text)
>>> words = [nltk.word_tokenize(sentence) for sentence in sentences]
>>> flat_words = [word.lower() for sentence in words for word in sentence]
>>> bigrams = nltk.bigrams(flat_words)
>>> bgcdist = nltk.ConditionalFreqDist(bigrams)
>>> bgcdist.tabulate(conditions=["the", "fox", "and", "the", "crow"])
        ,  birds  cheese  crow   fox   hue  just  observed  said   set  standing  that   the  tree   was
 the    0      1       2     1     2     1     0         0     0     0         0     0     1     0
 fox    1      0       0     0     0     0     0         1     0     0         0     1     0     0
 and    0      0       0     0     0     0     1         0     1     1         1     0     1     0
 the    0      1       2     1     2     1     0         0     0     0         0     0     1     0
crow    0      0       0     0     0     0     0         0     0     0         0     0     0     2
```

Collocations

Collocations: Definition

- "Collocations … are statements of the habitual or customary places of [a] word." (Firth 1957)
- Sequence of words that occur together unusually often
- Usually, we specify an **n-gram** window within which to analyse collocations:
  - bigram: *credit card, credit crunch*
  - trigram: *credit card fraud, credit card expiry*
  - …
- **The idea is to look at co-occurrence of words within a specific n-gram window**
- Characteristics/Expectations:
  - regular/frequently attested
  - occur within a narrow window (span of few words)
  - not fully compositional
  - non-substitutable
  - subject to category restrictions

A Motivating Example

- Consider phrases such as:
  - strong tea ? powerful tea
  - strong support ? powerful support
  - powerful drug ? strong drug
  - powerful computers ? strong computers

- Traditional semantic theories have difficulty accounting for these patterns.
  - *strong* and *powerful* seem near-synonyms
  - do we claim they have different senses?
  - what is the crucial difference?
  - why do *big* and *large* have different frequencies depending on the noun they modify?

Regularity / Frequency

- f(strong tea) > f(powerful tea)
- f(credit card) > f(credit bankruptcy)
- f(white wine) > f(yellow wine)
  - (even though white wine is actually yellowish)

Narrow Window (Textual Proximity)

- Usually collocates of a word occur close to that word.
  - may still occur across a span
- Examples:
  - bigram: *white wine, strong tea*
  - N > 2: *knock on the door; knock on X's door (vs. hit the door)*
- Can count n-grams with intervening words:
  - *federal **(.*)** subsidy*
  - matches: *federal subsidy, federal farm subsidy*, *federal manufacturing subsidy…*

Non-Compositionality

- *white wine*
  - not really “white”, meaning not fully predictable from component words + syntax
- Similarly:
  - *heavy rain* (not *big rain* or *powerful rain*)
  - *good practice guidelines*
- Extreme cases:
  - idioms such as *kick the bucket*
  - meaning is completely different

Non-Substitutability

- If a phrase is a collocation, we can't substitute a word in the phrase for a near-synonym, and still have the same overall meaning.

- E.g.:
  - *white wine* vs. *yellow wine*
  - *powerful tea vs. strong tea*
  - *Big rain vs. heavy rain*

Category Restrictions

- Frequency alone doesn't indicate collocational strength:
  - ***by the*** is a very frequent phrase in English
  - not a collocation
  - Also see http://en.wikipedia.org/wiki/Collocation

- Collocations tend to be formed from content words:
  - A+N: *strong tea*
  - N+N: *regression coefficient, mass demonstration*
  - N+PREP+N: *degrees of freedom*

Importance of Collocations

- Several applications need to “know” about collocations:
  - terminology extraction: technical or domain-specific phrases crop up frequently in text (*oil prices*)
  - document classification: specialist phrases are good indicators of the topic of a text
  - named entity recognition: names such as *New York* tend to occur together frequently; phrases like *new toy* don’t

Collocations in NLTK

```
>>> text4.collocations()
United States; fellow citizens; four years; years ago; Federal
Government; General Government; American people; Vice President; Old
World; Almighty God; Fellow citizens; Chief Magistrate; Chief Justice;
God bless; every citizen; Indian tribes; public debt; one another;
foreign nations; political parties
```

- ✓ Collocations are essentially just frequent bigrams
- ✓ **Except:** pay more attention to the cases that involve rare words
- ✓ Find bigrams that occur more often than we would expect based on the frequency of the individual words.

Collocations in NLTK

- Use the nltk.Text module

```
from nltk.corpus import gutenberg
words_per_sentence = [gutenberg.words(fileid) for fileid in gutenberg.fileids()]
words = [word.lower() for sublist in words_per_sentence for word in sublist]

gutext = nltk.Text(words)
gutext.collocations()
```

```
>>> gutext.collocations()
thou shalt; said unto; thou hast; thus saith; thou art; captain
wentworth; lord god; frank churchill; unto thee; every one; sperm
whale; burnt offering; jesus christ; lady russell; colonel brandon;
say unto; miss woodhouse; father brown; spake unto; buster bear
```

The Empiricist's View of Meaning

- Firth's view (1957):
  - "You shall know a word by the company it keeps"
  - This is a **contextual** view of meaning, akin to that espoused by Wittgenstein (1953).
    - i.e., **meaning is how it's used**
  - In the Firthian tradition, attention is paid to patterns that crop up with regularity in language.
  - Statistical work on collocations tends to follow this tradition.

Another View of Lexical Categories

- Word categories can be defined by the context in which they appear
- For a word ***w*** find all the contexts $w_1 w w_2$ in which ***w*** appears
- Find all words ***w'*** that share many frequent contexts

Finding Similar Words in NLTK

- Use the `nltk.Text` class
- Use the **`similar`** function
  - What other words appear in a similar range of contexts?

```
>>> gutext
<Text: emma by jane austen 1816>
>>> gutext.similar('woman')
man people time day thing men one king place lord house night land
word other earth way lady world gentleman
>>>
```

Stemming

Morphology

- Words can have compositional meaning from their parts
- Morphology is the study of the internal structure of words, of the way words are built up from smaller meaning units.
- Morpheme:
  - The smallest meaningful unit in the grammar of a language.
- Two classes of morphemes
  - Stems: “main” morpheme of the word, supplying the main meaning (i.e. *establish* in the example below)
  - Affixes: add additional meaning
    - Prefixes: Anti**dis**e**stablish**mentarianism
    - Suffixes: Antidis**establish**mentarianism
    - Infixes: hingi (*borrow*) – h**um**ingi (*borrower*) in Tagalog
    - Circumfixes: sagen (*say*) – **ge**sag**t** (*said*) in German

Stemming

- The removal of the inflectional part from words (strip off any affixes)
  - *Laughing, laugh, laughs, laughed → laugh*

- Problems
  - Can conflate semantically different words
    - *Gallery* and *gall* may both be stemmed to *gall*

- A further step is to make sure that the resulting form is a known word in a dictionary, a task known as ***Lemmatization***.

NLTK Stemmers

- nltk.wordnet.morphy
  - A slightly more sophisticated approach
  - Use an understanding of inflectional morphology
    - Use an Exception List for irregulars
    - Handle collocations in a special way
  - Do the transformation, compare the result to the WordNet dictionary
    - If the transformation produces a real word, then keep it, else use the original word.
  - For more details, see
    - http://wordnet.princeton.edu/man/morphy.7WN.html

Is Stemming Useful

- For information retrieval, some improvement for smaller documents
  - Helps a lot for some queries, hurts a lot in other cases
- Mixed results for language modeling
- Problems
  - Word sense disambiguation on query terms: *business* may be stemmed to *busy, saw (*the tool*) to see*
  - A truncated stem can be unintelligible to users
- **However,** finding the root word (lemma) may be necessary to use lexical resources

Text Normalization

- Stemming
- Convert to lower case
- Identifying *non-standard words* including numbers, abbreviations, and dates, and mapping any such tokens to a special vocabulary.
  - For example, every decimal number could be mapped to a single token 0.0, and every acronym could be mapped to AAA. This keeps the vocabulary small and improves the accuracy of many tasks.
- Lemmatization
  - Make sure that the resulting form is a known word in a dictionary

Moving Beyond Words: Part-of-Speech

Part-of-Speech

- Sometimes lexical items are too specific
  - Don't generalize well
  - Sparse data problems
- Grouping words into a small set of word classes or lexical categories can be useful for many applications
- Part-of-speech is a common method of categorizing words
  - Grounded in linguistic theory

Useful Applications

- Word sense disambiguation
  - The rain **bank**ed the soil up behind the gate.
  - We put our money in the **bank**.
- Identifying writer styles or personality
- Used in downstream processes in the NLP pipeline
  - E.g., syntactic parsing
- Lot's of other applications

`pos_tag` and `help` with POS

```
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> text = nltk.word_tokenize("They refuse to permit us to obtain the refuse permit")
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'), (
 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
>>>
```

```
>>> nltk.help.upenn_tagset('RB')
RB: adverb
    occasionally unabatingly maddeningly adventurously professedly
    stirringly prominently technologically magisterially predominately
    swiftly fiscally pitilessly ...
>>>
```

Part-Of-Speech (POS)

- Hard to define exactly and no absolute agreement on definition, but it is broadly…
- "a linguistic category of words (or more precisely *lexical items*), which is generally defined by the syntactic or morphological behaviour of the lexical item in question" – Wikipedia
- 8 traditional POS tags in English
  - Noun, pronoun, adjective, verb, adverb, preposition, conjunction, interjection
- Can be further subcategorized by function

Part-Of-Speech (POS)

- 8 traditional POS tags in English
  - Noun, pronoun, adjective, verb, adverb, preposition, conjunction, interjection

  - I talk.
  - I am talking.
  - I talk slowly.
  - I talk to the cats.
  - I talk to the cats in the garden.
  - Oh! I love talking gleefully to the cats and the birds in the garden.

Tagsets

- The specific set of POS tags used for an application is called the tagset
- Most popular is the Penn Treebank Tagset
  - 36 tags
  - https://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html
  - Be wary of funny names for POS
    - *IN* is a preposition, *JJ* is an adjective, *NN* is a noun (kind of makes sense), *NNS* is a plural noun, etc.
- In contrast the Brown Tagset has 85 POS categories

Penn Tagset

1.	CC	Coordinating conjunction	25.	TO	*to*
2.	CD	Cardinal number	26.	UH	Interjection
3.	DT	Determiner	27.	VB	Verb, base form
4.	EX	Existential *there*	28.	VBD	Verb, past tense
5.	FW	Foreign word	29.	VBG	Verb, gerund/present participle
6.	IN	Preposition/subordinating conjunction	30.	VBN	Verb, past participle
7.	JJ	Adjective	31.	VBP	Verb, non-3rd ps. sing. present
8.	JJR	Adjective, comparative	32.	VBZ	Verb, 3rd ps. sing. present
9.	JJS	Adjective, superlative	33.	WDT	*wh*-determiner
10.	LS	List item marker	34.	WP	*wh*-pronoun
11.	MD	Modal	35.	WP$	Possessive *wh*-pronoun
12.	NN	Noun, singular or mass	36.	WRB	*wh*-adverb
13.	NNS	Noun, plural	37.	#	Pound sign
14.	NNP	Proper noun, singular	38.	$	Dollar sign
15.	NNPS	Proper noun, plural	39.	.	Sentence-final punctuation
16.	PDT	Predeterminer	40.	,	Comma
17.	POS	Possessive ending	41.	:	Colon, semi-colon
18.	PRP	Personal pronoun	42.	(	Left bracket character
19.	PP$	Possessive pronoun	43.	)	Right bracket character
20.	RB	Adverb	44.	"	Straight double quote
21.	RBR	Adverb, comparative	45.	‘	Left open single quote
22.	RBS	Adverb, superlative	46.	“	Left open double quote
23.	RP	Particle	47.	’	Right close single quote
24.	SYM	Symbol (mathematical or scientific)	48.	”	Right close double quote

Part-of-Speech Tagging

- Assign each word in continuous text a tag indicating its part of speech.
  - Essentially a classification problem (later in the course)

- Current state of the art:
  - taggers typically have 96-97% accuracy
  - evaluated on a per-word basis
  - in a corpus with sentences of average length 20 words, 96% accuracy can mean one tagging error per sentence

Sources of difficulty

- Mostly due to ambiguity when words have more than one possible tag.
  - need context to make a good guess about POS
  - context alone won't suffice

- A simple approach which assigns only the most common tag to each word performs with 90% accuracy!

Some Features for Automatic Tagging

- Syntagmatic information: the tags of other words in the context of a word
  - Not sufficient on its own.
- Lexical information ("dictionary"): most common tag(s) for a given word (will see more a bit later today)
  - e.g. in English, many nouns can be used as verbs (*flour the pan*, *wax the car…*)
  - however, their most likely tag remains NN
  - distribution of a word's usages across different POS tags is uneven: usually, one highly likely, other much less
- Other lexical features
  - Is it uppercase?
  - Prefix, suffix?

Example of USING pos_tag

```
words = [nltk.word_tokenize(sentence) for sentence in sentences]
flat_words = [word.lower() for sentence in words for word in sentence]
```

```
>>> nltk.pos_tag(flat_words)
[('a', 'DT'), ('crow', 'NN'), ('was', 'VBD'), ('sitting', 'VBG'), ('on', 'IN'), ('a', 'DT
'), ('branch', 'NN'), ('of', 'IN'), ('a', 'DT'), ('tree', 'NN'), ('with', 'IN'), ('a', 'DT
'), ('piece', 'NN'), ('of', 'IN'), ('cheese', 'JJ'), ('in', 'IN'), ('her', 'PRP$'), ('beak
', 'NN'), ('when', 'WRB'), ('a', 'DT'), ('fox', 'NN'), ('observed', 'VBN'), ('her', 'PRP$'
, ('and', 'CC'), ('set', 'NN'), ('his', 'PRP$'), ('wits', 'NNS'), ('to', 'TO'), ('work',
VB'), ('to', 'TO'), ('discover', 'VB'), ('some', 'DT'), ('way', 'NN'), ('of', 'IN'), ('ge
ting', 'VBG'), ('the', 'DT'), ('cheese', 'JJ'), ('.', '.'), ('coming', 'VBG'), ('and', 'C
'), ('standing', 'NN'), ('under', 'IN'), ('the', 'DT'), ('tree', 'NN'), ('he', 'PRP'), ('
ooked', 'VBD'), ('up', 'RP'), ('and', 'CC'), ('said', 'VBD'), (',', ','), ('``', '``'), (
what', 'WP'), ('a', 'DT'), ('noble', 'JJ'), ('bird', 'NN'), ('i', 'PRP'), ('see', 'VBP'),
('above', 'IN'), ('me', 'PRP'), ('!', '.'), ('her', 'PRP$'), ('beauty', 'NN'), ('is', 'VB
'), ('without', 'IN'), ('equal', 'JJ'), (',', ','), ('the', 'DT'), ('hue', 'NN'), ('of',
IN'), ('her', 'PRP$'), ('plumage', 'NN'), ('exquisite', 'NN'), ('.', '.'), ('if', 'IN'),
'only', 'RB'), ('her', 'PRP$'), ('voice', 'NN'), ('is', 'VBZ'), ('as', 'RB'), ('sweet', '
J'), ('as', 'IN'), ('her', 'PRP$'), ('looks', 'NNS'), ('are', 'VBP'), ('fair', 'JJ'), (',
, ','), ('she', 'PRP'), ('ought', 'MD'), ('without', 'VB'), ('doubt', 'NN'), ('to', 'TO')
 ('be', 'VB'), ('queen', 'VBN'), ('of', 'IN'), ('the', 'DT'), ('birds', 'NNS'), ('.', '.'
, ("''", "''"), ('the', 'DT'), ('crow', 'NN'), ('was', 'VBD'), ('hugely', 'RB'), ('flatte
ed', 'VBN'), ('by', 'IN'), ('this', 'DT'), (',', ','), ('and', 'CC'), ('just', 'RB'), ('t
', 'TO'), ('show', 'VB'), ('the', 'DT'), ('fox', 'NN'), ('that', 'IN'), ('she', 'PRP'), (
could', 'MD'), ('sing', 'VB'), ('she', 'PRP'), ('gave', 'VBD'), ('a', 'DT'), ('loud', 'NN
), ('caw', 'NN'), ('.', '.'), ('down', 'IN'), ('came', 'VBD'), ('the', 'DT'), ('cheese',
NN'), (',', ','), ('of', 'IN'), ('course', 'NN'), (',', ','), ('and', 'CC'), ('the', 'DT'
, ('fox', 'NN'), (',', ','), ('snatching', 'VBG'), ('it', 'PRP'), ('up', 'RP'), (',', ','
```

Covered HW 1! ☺

Homework 2 is going to have a lot more to do in it.

Lexicons & Lexical Semantics
Ch. 2 NLPP
Sec. 4, 5, 6

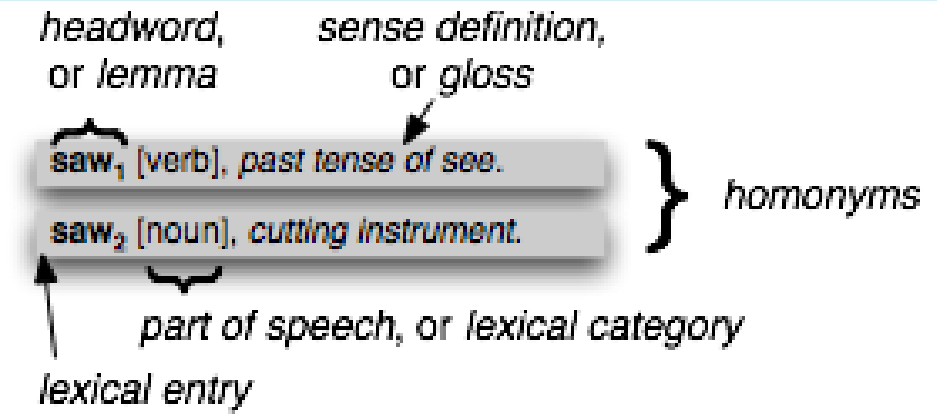
Lexical Resources

- A lexicon, or lexical resource, is a collection of words and/or phrases along with some associated information such as part of speech and sense definitions.

- A vocabulary (list of words in a text) is the simplest lexical resource

- Lexical entry

A **lexical entry** typically consists of a **headword** (also known as a **lemma**) along with additional information such as the part of speech and the sense definition.



Different types of Lexical Dictionaries

- **LIWC:** Linguistic Inquiry and Word Count: categorizes words into a hierarchical set of lexical categories (for sentiment classification as well as others)
- **Sentiment Lexicons:** Classify words into positive and negative polarity (and neutral)
- **Wordnet:** (see more later today!)
  - classifies words hierarchically according to an ontology of things in the world, e.g. cup **is-a** container
  - Tells you the different senses of a word and groups words with their synonyms
- **Verbnet:**
  - groups verbs by their meaning into an ontology
  - Tells you how verbs ‘subcategorize’ for their arguments

Wordnet in NLTK

Has a great API to do lots of stuff!

Incredibly useful!

WordNet

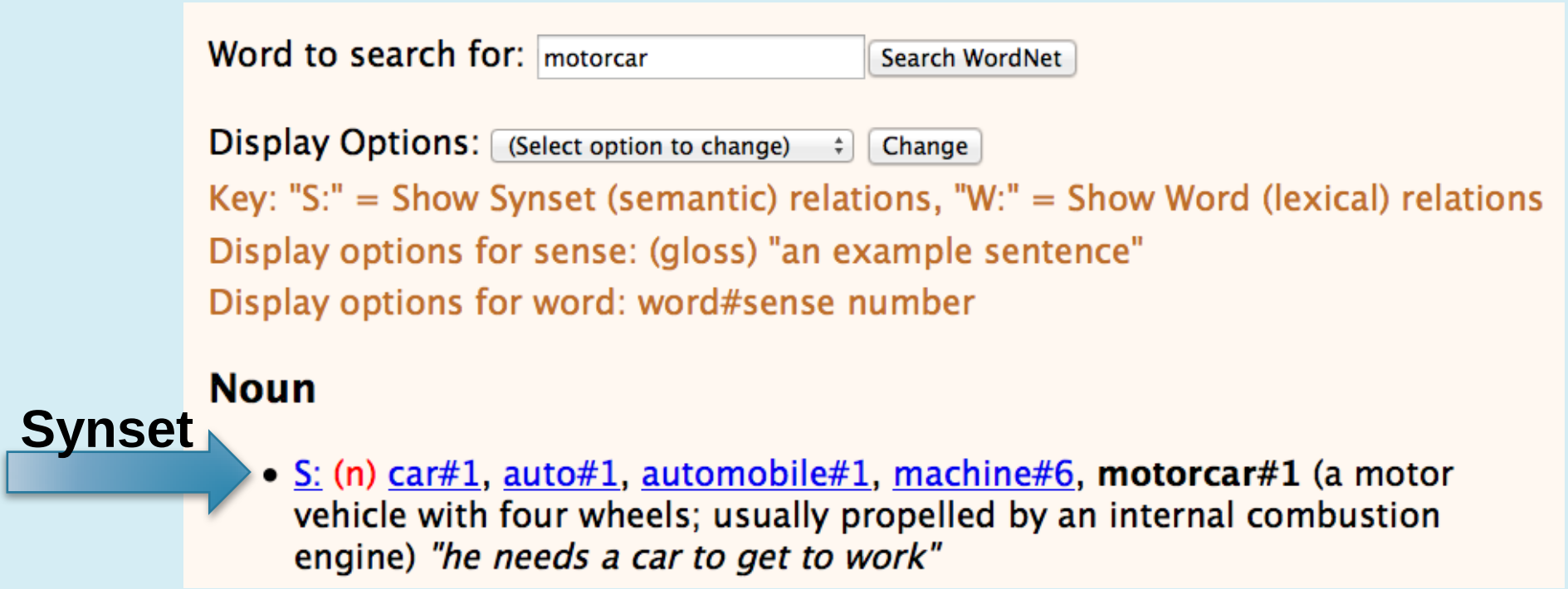
- **WordNet** s a large lexicon of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.
- **Synsets** are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser.
- NLTK includes the English WordNet, with 155,287 words and 117,659 synonym sets.
- **Senses and Synonyms**
  - Consider the 2 sentences:
    - *Benz is credited with the invention of the motorcar*
    - *Benz is credited with the invention of the automobile*.
  - *motorcar* and *automobile* have the same meaning, i.e. they are **synonyms**.

The WordNet Hierarchy

- Some WordNet synsets correspond to abstract concepts.
- These concepts are linked together in a hierarchy. Some concepts are very general, such as *Entity*, *State*, *Event* — these are called unique beginners or root synsets.

WordNet (Synsets)

- *motorcar* has just one possible meaning and it is identified as car.n.01, the first noun sense of *car*.
- The entity car.n.01 is called a **synset**, or "synonym set", a collection of synonymous words (or "lemmas"):



Words in Fables

- [04] S: (n) **graze#2 (graze%1:04:00::)**, grazing#1 (grazing%1:04:01::) (the act of grazing)

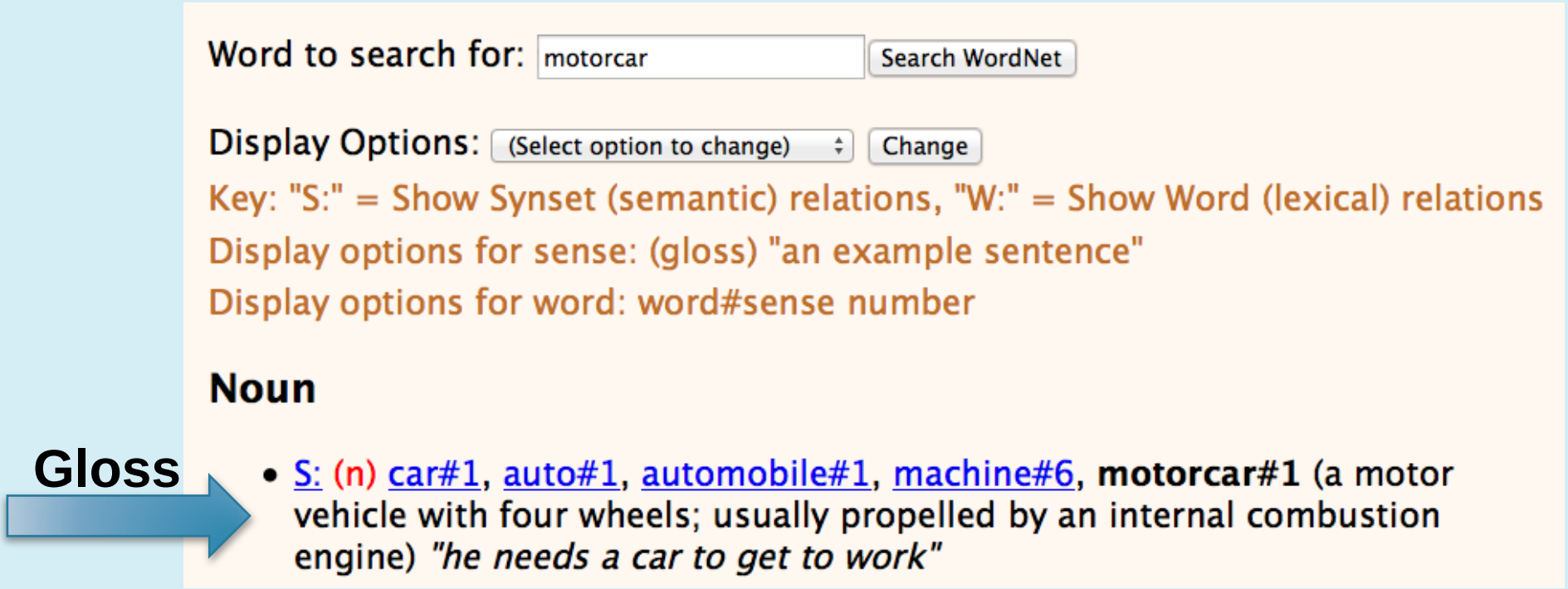
Verb

- [35] S: (v) crop#5 (crop%2:35:01::), browse#2 (browse%2:35:01::), **graze#1 (graze%2:35:01::)**, range#6 (range%2:35:02::), pasture#2 (pasture%2:35:00::) (feed as in a meadow or pasture) *"the herd was grazing"*
  - ***verb group***
    - [34] S: (v) range#7 (range%2:34:00::) (let eat) *"range the animals in the prairie"*
    - [35] S: (v) crop#4 (crop%2:35:10::), **graze#3 (graze%2:35:10::)**, pasture#1 (pasture%2:35:10::) (let feed in a field or pasture or meadow)
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - *derivationally related form*
  - *sentence frame*
- [35] S: (v) **graze#2 (graze%2:35:02::)** (break the skin (of a body part) by scraping) *"She was grazed by the stray bullet"*
- [35] S: (v) crop#4 (crop%2:35:10::), **graze#3 (graze%2:35:10::)**, pasture#1 (pasture%2:35:10::) (let feed in a field or pasture or meadow)
- [35] S: (v) **graze#4 (graze%2:35:00::)**, crease#3 (crease%2:35:02::), rake#6 (rake%2:35:02::) (scrape gently) *"graze the skin"*
- [34] S: (v) browse#4 (browse%2:34:00::), **graze#5 (graze%2:34:02::)** (eat

Synset

WordNet (Gloss)

- Each entry has a short definition or **gloss**
- E.g., "A motor vehicle with four wheels; usually propelled by an internal combustion engine"



Words in our Stories

- [04] S: (n) **graze#2 (graze%1:04:00::)**, grazing#1 (grazing%1:04:01::) (the act of grazing)

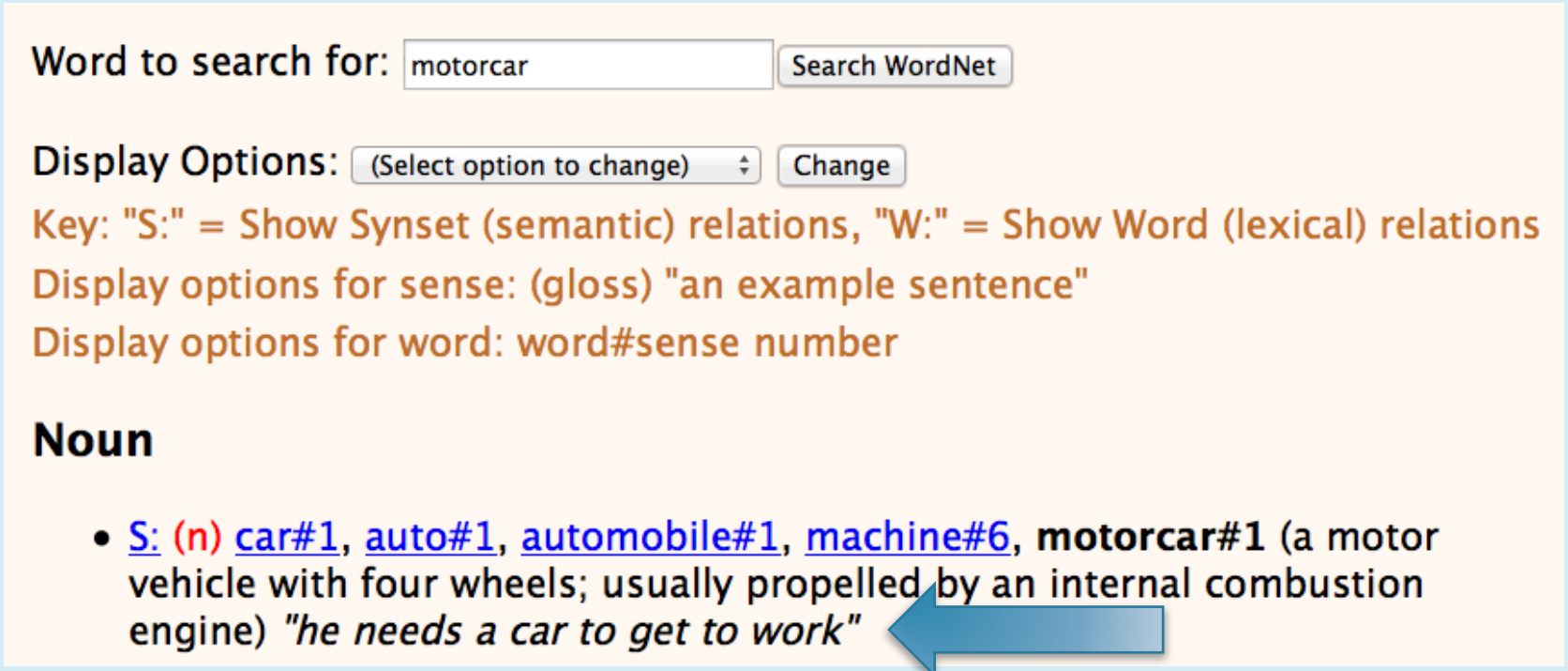
Verb

Gloss

- [35] S: (v) crop#5 (crop%2:35:01::), browse#2 (browse%2:35:01::), **graze#1 (graze%2:35:01::)**, range#6 (range%2:35:02::), pasture#2 (pasture%2:35:00::) (feed as in a meadow or pasture) *"the herd was grazing"*
  - ***verb group***
    - [34] S: (v) range#7 (range%2:34:00::) (let eat) *"range the animals in the prairie"*
    - [35] S: (v) crop#4 (crop%2:35:10::), **graze#3 (graze%2:35:10::)**, pasture#1 (pasture%2:35:10::) (let feed in a field or pasture or meadow)
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - *derivationally related form*
  - *sentence frame*
- [35] S: (v) **graze#2 (graze%2:35:02::)** (break the skin (of a body part) by scraping) *"She was grazed by the stray bullet"*
- [35] S: (v) crop#4 (crop%2:35:10::), **graze#3 (graze%2:35:10::)**, pasture#1 (pasture%2:35:10::) (let feed in a field or pasture or meadow)
- [35] S: (v) **graze#4 (graze%2:35:00::)**, crease#3 (crease%2:35:02::), rake#6 (rake%2:35:02::) (scrape gently) *"graze the skin"*
- [34] S: (v) browse#4 (browse%2:34:00::), **graze#5 (graze%2:34:02::)** (eat

WordNet

- Each entry usually has at least one example use from a corpus.
- E.g., "A motor vehicle with four wheels; usually propelled by an internal combustion engine"



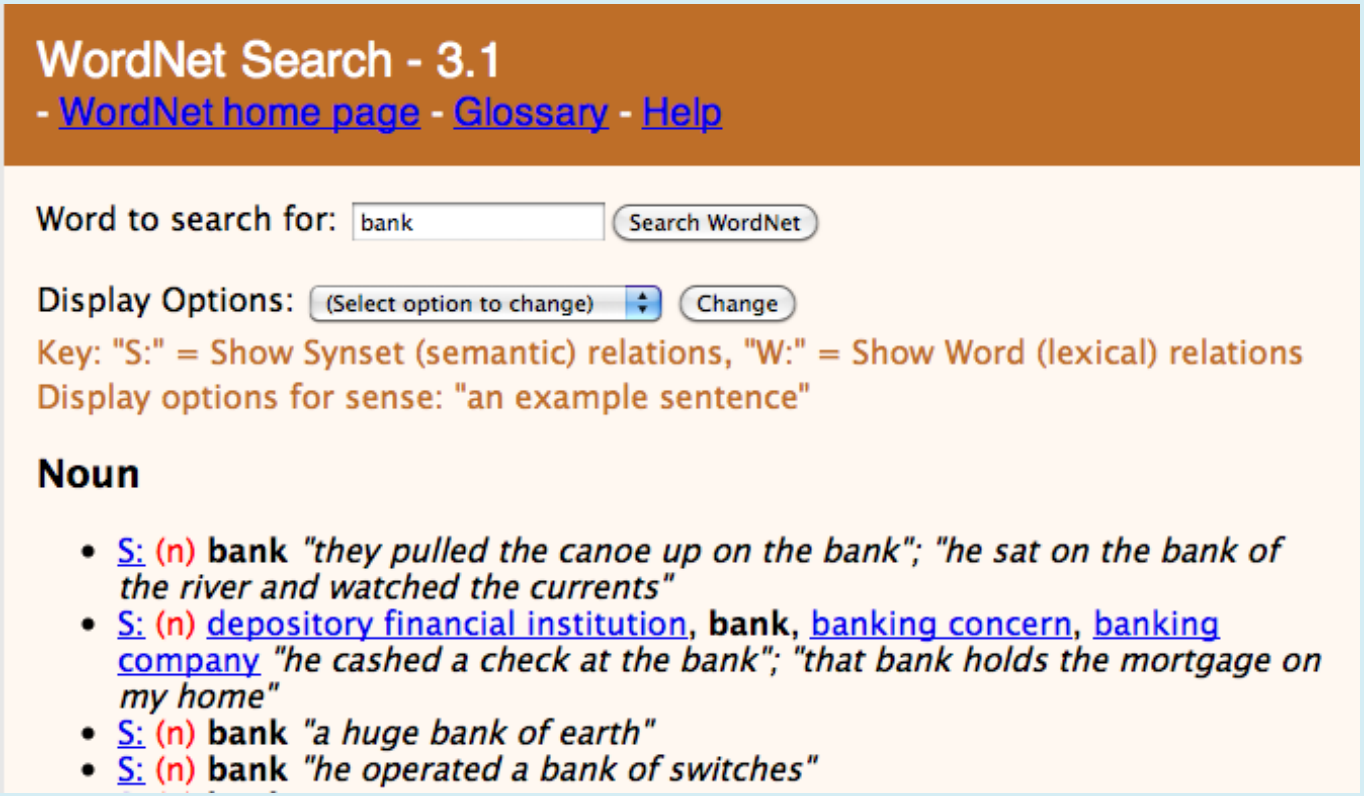
Word Sense Disambiguation

- An area of NLP which focuses on automatically finding the word senses for words in text or dialogue
- Not yet a solved problem
- Classic Example:
  - I took the money to the bank.
  - I went fishing at the bank.
- “Words go in Herds”: Money would rarely co-occur with fishing

Word Sense Disambiguation

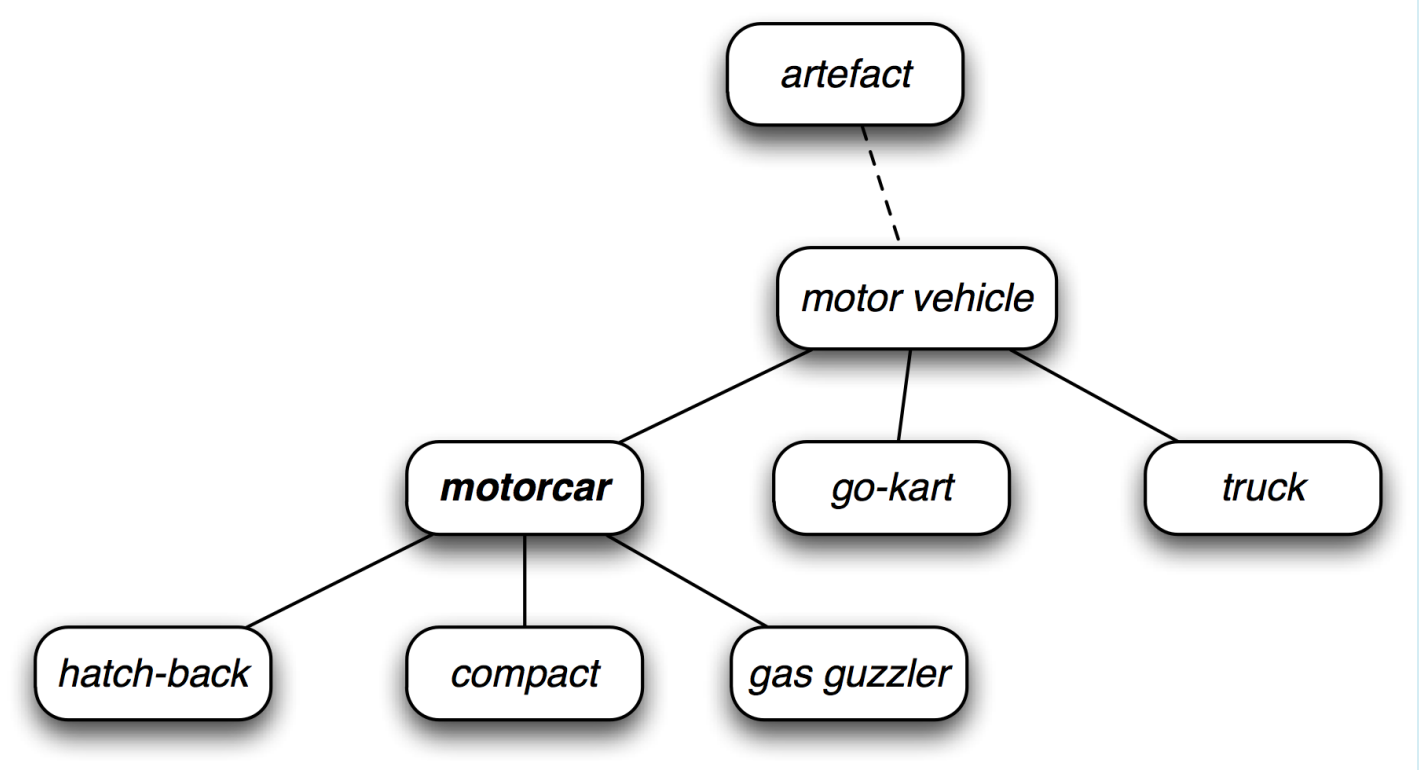
Classic Example:

- I took the money to the bank.
- I went fishing at the bank.



The WordNet Hierarchy in NLTK

- It's very easy to navigate between concepts. For example, given a concept like *motorcar*, we can look at the concepts that are more specific:
  - A **hyponym** is a word or phrase whose semantic field is more specific than its hypernym.



The WordNet Hierarchy in NLTK

- Hypernyms and hyponyms are called **lexical relations** because they relate one synset to another. These two relations navigate up and down the "*is-a*" hierarchy.

- S: (n) car#1, auto#1, automobile#1, machine#6, **motorcar#1** (a motor vehicle with four wheels; usually propelled by an internal combustion engine) *"he needs a car to get to work"*
  - *direct hyponym* / ***full hyponym***
    - S: (n) ambulance#1 (a vehicle that takes people to and from hospitals)
      - S: (n) funny wagon#1 (an ambulance used to transport patients to a mental hospital)
    - S: (n) beach wagon#1, station wagon#1, wagon#5, estate car#1, beach waggon#1, station waggon#1, waggon#2 (a car that has a long body and rear door with space behind rear seat)
      - S: (n) shooting brake#1 (another name for a station wagon)
    - S: (n) bus#4, jalopy#1, heap#3 (a car that is old and unreliable) *"the fenders had fallen off that old bus"*

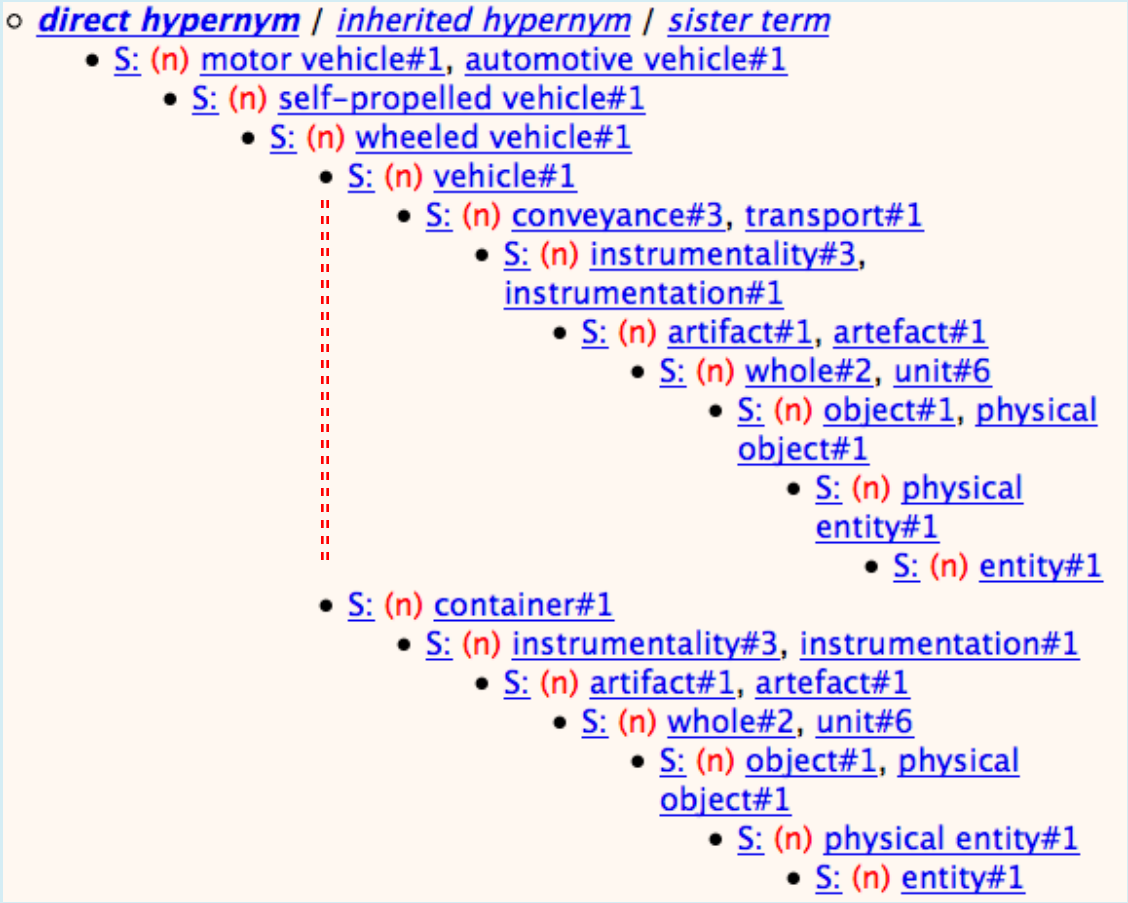
Uses the SENSE KEY

- It's very easy to navigate between concepts. For example, given a concept like *motorcar*, we can look at the concepts that are more specific; the (immediate) **hyponyms**.

```
>>> motorcar = wn.synset('car.n.01')
>>> types_of_motorcar = motorcar.hyponyms()
>>> types_of_motorcar[26]
Synset('ambulance.n.01')
>>> sorted([lemma.name for synset in types_of_motorcar for lemma in synset.lemmas])
['Model_T', 'S.U.V.', 'SUV', 'Stanley_Steamer', 'ambulance', 'beach_waggon',
'beach_wagon', 'bus', 'cab', 'compact', 'compact_car', 'convertible',
'coupe', 'cruiser', 'electric', 'electric_automobile', 'electric_car',
'estate_car', 'gas_guzzler', 'hack', 'hardtop', 'hatchback', 'heap',
'horseless_carriage', 'hot-rod', 'hot_rod', 'jalopy', 'jeep', 'landrover',
'limo', 'limousine', 'loaner', 'minicar', 'minivan', 'pace_car', 'patrol_car',
'phaeton', 'police_car', 'police_cruiser', 'prowl_car', 'race_car', 'racer',
'racing_car', 'roadster', 'runabout', 'saloon', 'secondhand_car', 'sedan',
'sport_car', 'sport_utility', 'sport_utility_vehicle', 'sports_car', 'squad_car',
'station_waggon', 'station_wagon', 'stock_car', 'subcompact', 'subcompact_car',
'taxi', 'taxicab', 'tourer', 'touring_car', 'two-seater', 'used-car', 'waggon',
'wagon']
```

The WordNet Hierarchy in NLTK

- We can also navigate up the hierarchy by visiting hypernyms. Some words have multiple paths, because they can be classified in more than one way. There are two paths between *car.n.01* and *entity.n.01* because *wheeled_vehicle.n.01* can be classified as both a vehicle and a container.



Words in Fables: The Wily Lion

- A Lion watched a fat Bull feeding in a meadow, and his mouth watered when he thought of the royal feast he would make, but he did not dare to attack him, for he was afraid of his sharp horns.

Words in our Stories

- [04] S: (n) **graze#2 (graze%1:04:00::)**, grazing#1 (grazing%1:04:01::) (the act of grazing)

Verb

- [35] S: (v) crop#5 (crop%2:35:01::), browse#2 (browse%2:35:01::), **graze#1 (graze%2:35:01::)**, range#6 (range%2:35:02::), pasture#2 (pasture%2:35:00::) (feed as in a meadow or pasture) *"the herd was grazing"*
  - ***verb group***
    - [34] S: (v) range#7 (range%2:34:00::) (let eat) *"range the animals in the prairie"*
    - [35] S: (v) crop#4 (crop%2:35:10::), **graze#3 (graze%2:35:10::)**, pasture#1 (pasture%2:35:10::) (let feed in a field or pasture or meadow)
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - *derivationally related form*
  - *sentence frame*
- [35] S: (v) **graze#2 (graze%2:35:02::)** (break the skin (of a body part) by scraping) *"She was grazed by the stray bullet"*
- [35] S: (v) crop#4 (crop%2:35:10::), **graze#3 (graze%2:35:10::)**, pasture#1 (pasture%2:35:10::) (let feed in a field or pasture or meadow)
- [35] S: (v) **graze#4 (graze%2:35:00::)**, crease#3 (crease%2:35:02::), rake#6 (rake%2:35:02::) (scrape gently) *"graze the skin"*
- [34] S: (v) browse#4 (browse%2:34:00::), **graze#5 (graze%2:34:02::)** (eat

Some highly ambiguous Words: Horn

Noun

- (7){03542265} <noun.artifact>[06] S: (n) **horn#1 (horn%1:06:06::)** (a noisemaker (as at parties or games) that makes a loud noise when you blow through it)
- (3){01328058} <noun.animal>[05] S: (n) **horn#2 (horn%1:05:01::)** (one of the bony outgrowths on the heads of certain ungulates)
- (1){07280214} <noun.communication>[10] S: (n) **horn#3 (horn%1:10:02::)** (a noise made by the driver of an automobile to give warning)
- (1){03542111} <noun.artifact>[06] S: (n) **horn#4 (horn%1:06:04::)**, saddle horn#1 (saddle_horn%1:06:00::) (a high pommel of a Western saddle (usually metal covered with leather))
- (1){03115320} <noun.artifact>[06] S: (n) cornet#1 (cornet%1:06:00::), **horn#5 (horn%1:06:01::)**, trumpet#1 (trumpet%1:06:00::), trump#3 (trump%1:06:01::) (a brass musical instrument with a brilliant tone; has a narrow tube and a flared bell and is played by means of valves)
- (1){01328494} <noun.animal>[05] S: (n) **horn#6 (horn%1:05:02::)** (any hard protuberance from the head of an organism that is similar to or suggestive of a horn)
- {14782206} <noun.substance>[27] S: (n) **horn#7 (horn%1:27:00::)** (the material (mostly keratin) that covers the horns of ungulates and forms hooves and claws and nails)
- {03542421} <noun.artifact>[06] S: (n) **horn#8 (horn%1:06:07::)** (a device

Hyponyms and Hypernyms from WordNet

A Lion watched a fat Bull feeding in a meadow, and his mouth watered…

Verb

- [35] S: (v) crop#5 (crop%2:35:01::), browse#2 (browse%2:35:01::), **graze#1 (graze%2:35:01::)**, range#6 (range%2:35:02::), pasture#2 (pasture%2:35:00::) (feed as in a meadow or pasture) *"the herd was grazing"*
  - *verb group*
    - [34] S: (v) range#7 (range%2:34:00::) (let eat) *"range the animals in the prairie"*
    - [35] S: (v) crop#4 (crop%2:35:10::), **graze#3 (graze%2:35:10::)**, pasture#1 (pasture%2:35:10::) (let feed in a field or pasture or meadow)
  - ***direct hypernym*** / *inherited hypernym* / *sister term*
    - [34] S: (v) feed#6 (feed%2:34:00::), eat#3 (eat%2:34:02::) (take in food; used of animals only) *"This dog doesn't eat certain kinds of meat"; "What do whales eat?"*

WordNet: More Lexical Relations

- Some lexical relationships hold between lemmas, e.g., **antonymy**:

Noun

- S: (n) **supply#1**
- S: (n) **supply#2**
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - ***antonym***
    - W: (n) demand#2 [Opposed to: supply]

- There are also relationships between verbs. For example, the act of *walking* involves the act of *stepping*, so walking **entails** stepping. Some verbs have multiple entailments:

Verb

- S: (v) walk#1 (use one's feet to advance; advance by steps)
  - *direct troponym* / *full troponym*
  - *verb group*
  - *direct hypernym* / *inherited hypernym* / *sister term*
  - *entailment*
    - S: (v) step#1 (shift or move by taking a step)

To do

- Turn in HW 0 if you haven't done yet!
- Read relevant sections from Chapters 2, 3 and 5
- Review Python: Chapter 4, if you think you need to!
- Start HW 1
  - Will be posted tonight
  - Due Friday, April 8
- Go to the lab section!
  - Soc Sci I Mac Lab – Room 135
- Reminder:
  - My office hours : Thursdays, 2:00-3:30 pm, E2-255

Next…

- More on WordNet API in NLTK and lexical relations
- Review of probability and conditional probability
  - Required for statistical modeling of language data
- Corpus-based statistical NLP
- Language Models
- Regular Expressions