Natural Language Processing

Tutorial 2: Text Normalization

Dr. Sun Aixin

Question 1

- Consider the following word segmentation algorithm in the lecture notes:

- Given a lexicon of Chinese, and a string
  - Start a pointer at the beginning of the string
  - Find the longest word in dictionary that matches the string starting at pointer
  - Move the pointer over the word in string
  - Goto2

- Following the algorithm, you perhaps end up with failing to segment a string, if you cannot find a matching.

Question 1 (cont)

- Example
  - String to segment: thetablesdownthere
  - lexicon: the table down there bled own.
- Discuss how to fix the above problem.

Answer Q1

- Start a pointer at the beginning of the string
- Find the longest word in dictionary that matches the string starting at pointer
  - If matched, move the pointer over the word in string
  - If no word is matched, skip to the next letter
- Goto2

  - String to segment: thetablesdownthere
  - lexicon: the table down there bled own.

Answer 1 (a bit more processing)

- Start a pointer at the beginning of the string
- Find the longest word in dictionary that matches the string starting at pointer
  - If matched,
    - Run morphological analysis from the beginning of the word
    - Move the pointer over the morphologically matched part of the string
  - If no word is matched, skip to the next letter
- Goto2

  - String to segment: thetablesdownthere
  - lexicon: the table down there bled own.

Question 2

- Try the tokenization demo on
  - http://text-processing.com/demo/tokenize/
  - (you may use other tokenizer APIs).
- Discuss your findings based on the output of different tokenizers.

- https://textanalysisonline.com/

Question 3

- Try the stemmer demo on
  - http://text-processing.com/demo/stem/
  - (or you may use other stemmer APIs).
- Discuss your findings based on the output of the stemmers.

- https://textanalysisonline.com/

Question 4

- Write a program to do the following tasks:
  - **Download** the Web page of a given link and **extract** the text content of the page
  - **Split** the text into sentences and **count** sentences
  - **Split** the text into tokens and **count** token types
  - **Find** lemmas (or stems) of the tokens and **count** lemma types
  - Do **stemming** on the tokens and **count** unique stemmed tokens

Answer 4

- Example URL:
  - https://en.wikipedia.org/wiki/Natural_language_processing
- urllib
  - https://docs.python.org/3/library/urllib.html
- NLTK
  - https://www.nltk.org/
- Beautiful Soup
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/#porting-code-to-bs4

Alternative libraries:
StanfordNLP
OpenNLP
spaCy
AllenNLP

Sample code

```
import urllib.request
import nltk
from bs4 import BeautifulSoup

with urllib.request.urlopen ('https://en.wikipedia.org/wiki/Natural_language_processing') as response:
    html=response.read()

text = BeautifulSoup(html, "lxml").get_text()
sentences = nltk.tokenize.sent_tokenize(text)
print ('Number of sentences: '+ str(len(sentences)))

tokens= nltk.tokenize.word_tokenize(text)

print ('Number of tokens: '+ str(len(tokens)))

token_types = list(set(tokens))

print ('Number of token types: '+ str(len(token_types)))

wnl=nltk.stem.WordNetLemmatizer()

stemmer = nltk.stem.porter.PorterStemmer()
lemma_types= set()
stemmed_types= set()

for token_type in token_types:
    lemma_types.add(wnl.lemmatize(token_type))
    stemmed_types.add(stemmer.stem(token_type))

print ('Number of lemma types: '+ str(len(lemma_types)))
print ('Number of stemmed types: '+ str(len(stemmed_types)))
```

Download the Web page of a given link and **extract** the text content of the page

Split the text into sentences and **count** sentences

Split the text into tokens and **count** token types

Find lemmas (or stems) of the tokens and **count** lemma types

Do **stemming** on the tokens and **count** unique stemmed tokens

Sample code

- text = BeautifulSoup(html, "lxml").get_text()

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers Beautiful Soup version 4.8.1. The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

https://beautiful-soup-4.readthedocs.io/en/latest/

Sample code

https://www.nltk.org/api/nltk.tokenize.html

- sentences = nltk.tokenize.sent_tokenize(text)
- tokens= nltk.tokenize.word_tokenize(text)

- nltk.tokenize
  - nltk.tokenize.api
  - nltk.tokenize.casual
  - nltk.tokenize.destructive
  - nltk.tokenize.legality_principle
  - nltk.tokenize.mwe
  - nltk.tokenize.nist
  - nltk.tokenize.punkt
  - nltk.tokenize.regexp
  - nltk.tokenize.repp
  - nltk.tokenize.sexpr
  - nltk.tokenize.simple
  - nltk.tokenize.sonority_sequencing
  - nltk.tokenize.stanford
  - nltk.tokenize.stanford_segmenter
  - nltk.tokenize.texttiling
  - nltk.tokenize.toktok
  - nltk.tokenize.treebank
  - nltk.tokenize.util

nltk.tokenize package

NLTK Tokenizer Package

Tokenizers divide strings into lists of substrings. For example, tokenizers can be used to find the words and punctuation in a string:

```
>>> from nltk.tokenize import word_tokenize
>>> s = '''Good muffins cost $3.88\nin New York.  Please buy me
... two of them.\n\nThanks.'''
>>> word_tokenize(s)
['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York', '.',
'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
```

This particular tokenizer requires the Punkt sentence tokenization models to be installed. NLTK also provides a simpler, regular-expression based tokenizer, which splits text on whitespace and punctuation:

```
>>> from nltk.tokenize import wordpunct_tokenize
>>> wordpunct_tokenize(s)
['Good', 'muffins', 'cost', '$', '3', '.', '88', 'in', 'New', 'York', '.',
'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
```

We can also operate at the level of sentences, using the sentence tokenizer directly as follows:

```
>>> from nltk.tokenize import sent_tokenize, word_tokenize
>>> sent_tokenize(s)
['Good muffins cost $3.88\nin New York.', 'Please buy me\ntwo of them.', 'Thanks.']
>>> [word_tokenize(t) for t in sent_tokenize(s)]
[['Good', 'muffins', 'cost', '$', '3.88', 'in', 'New', 'York', '.'],
['Please', 'buy', 'me', 'two', 'of', 'them', '.'], ['Thanks', '.']]
```

Sample code

https://www.nltk.org/api/nltk.stem.html
https://www.nltk.org/api/nltk.stem.wordnet.html
https://web.stanford.edu/~jurafsky/slp3/

- wnl=nltk.stem.WordNetLemmatizer()
- stemmer = nltk.stem.porter.PorterStemmer()

nltk.stem package

NLTK Stemmers

Interfaces used to remove morphological affixes from words, leaving only the word stem. Stemming algorithms aim to remove those affixes required for eg. grammatical role, tense, derivational morphology leaving only the stem of the word. This is a difficult problem due to irregular words (eg. common verbs in English), complicated morphological rules, and part-of-speech and sense ambiguities (eg. `ceil-` is not the stem of `ceiling`).

StemmerI defines a standard interface for stemmers.

Submodules

- nltk.stem.api module
- nltk.stem.arlstem module
- nltk.stem.arlstem2 module
- nltk.stem.cistem module
- nltk.stem.isri module
- nltk.stem.lancaster module
- nltk.stem.porter module
- nltk.stem.regexp module
- nltk.stem.rslp module
- nltk.stem.snowball module
- nltk.stem.util module
- nltk.stem.wordnet module

nltk.stem.wordnet module

class nltk.stem.wordnet.WordNetLemmatizer [source]

Bases: `object`

WordNet Lemmatizer

Lemmatize using WordNet's built-in morphy function. Returns the input word unchanged if it cannot be found in WordNet.

```
>>> from nltk.stem import WordNetLemmatizer
>>> wnl = WordNetLemmatizer()
>>> print(wnl.lemmatize('dogs'))
dog
>>> print(wnl.lemmatize('churches'))
church
>>> print(wnl.lemmatize('aardwolves'))
aardwolf
>>> print(wnl.lemmatize('abaci'))
abacus
>>> print(wnl.lemmatize('hardrock'))
hardrock
```

WordNet

For your information (in part 1)

- A free, online, lexical resource
  - http://wordnet.princeton.edu
- Sub-databases
  - Nouns, Verbs, Adjectives, Adverbs
- Each database consists of Synsets
  - Synset is a synonym set
  - A group of words that are roughly synonymous in a given context

WordNet: Example (ignore the superscripts)

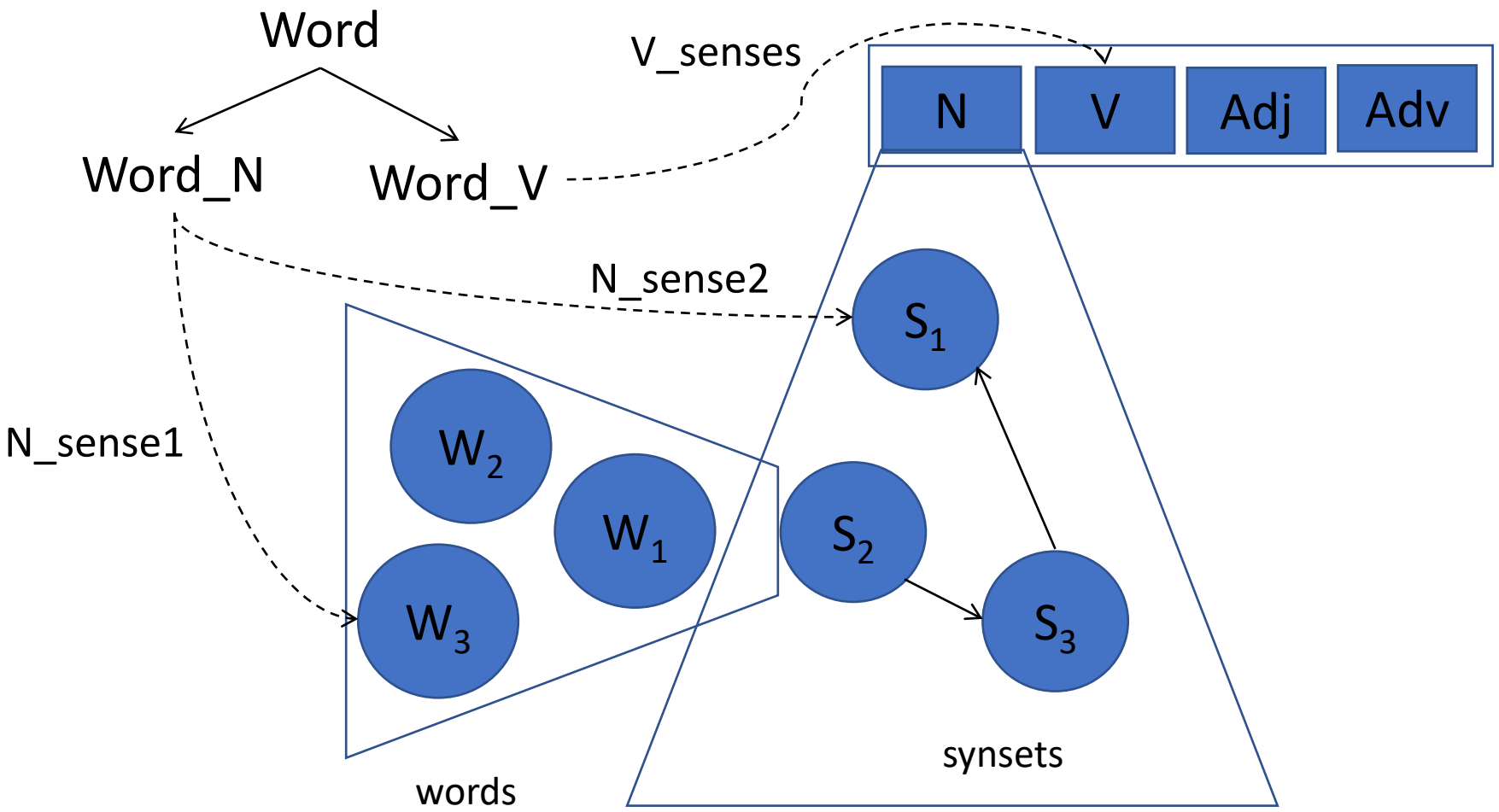
The noun "bass" has 8 senses in WordNet.

1. bass[1] - (the lowest part of the musical range)
2. bass[2], bass part[1] - (the lowest part in polyphonic music)
3. bass[3], basso[1] - (an adult male singer with the lowest voice)
4. sea bass[1], bass[4] - (the lean flesh of a saltwater fish of the family Serranidae)
5. freshwater bass[1], bass[5] - (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
6. bass[6], bass voice[1], basso[2] - (the lowest adult male singing voice)
7. bass[7] - (the member with the lowest range of a family of musical instruments)
8. bass[8] - (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

The adjective "bass" has 1 sense in WordNet.

1. bass[1], deep[6] - (having or denoting a low vocal or instrumental range)
   "a deep voice"; "a bass voice is lower than a baritone voice"; "a bass clarinet"

WordNet Structure



WordNet Hypernymy

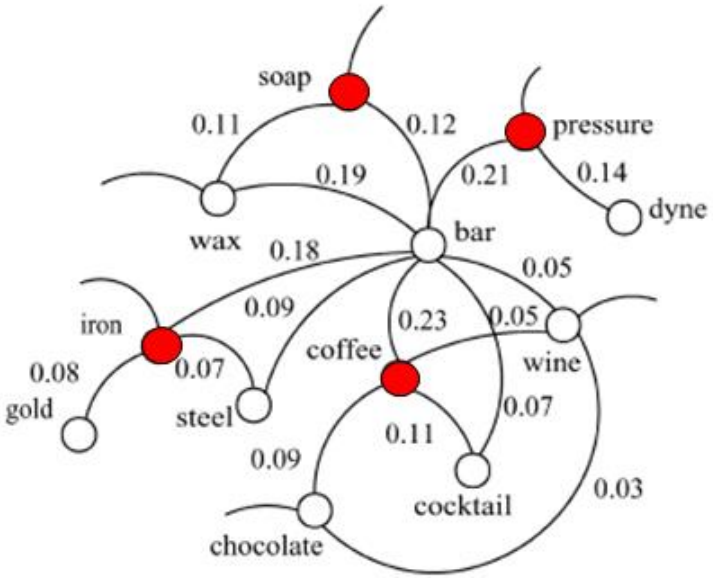
```
Sense 3
bass, basso --
(an adult male singer with the lowest voice)
=> singer, vocalist, vocalizer, vocaliser
   => musician, instrumentalist, player
      => performer, performing artist
         => entertainer
            => person, individual, someone...
               => organism, being
                  => living thing, animate thing,
                     => whole, unit
                        => object, physical object
                           => physical entity
                              => entity
               => causal agent, cause, causal agency
                  => physical entity
                     => entity

Sense 7
bass --
(the member with the lowest range of a family of
musical instruments)
=> musical instrument, instrument
   => device
      => instrumentality, instrumentation
         => artifact, artefact
            => whole, unit
               => object, physical object
                  => physical entity
                     => entity
```

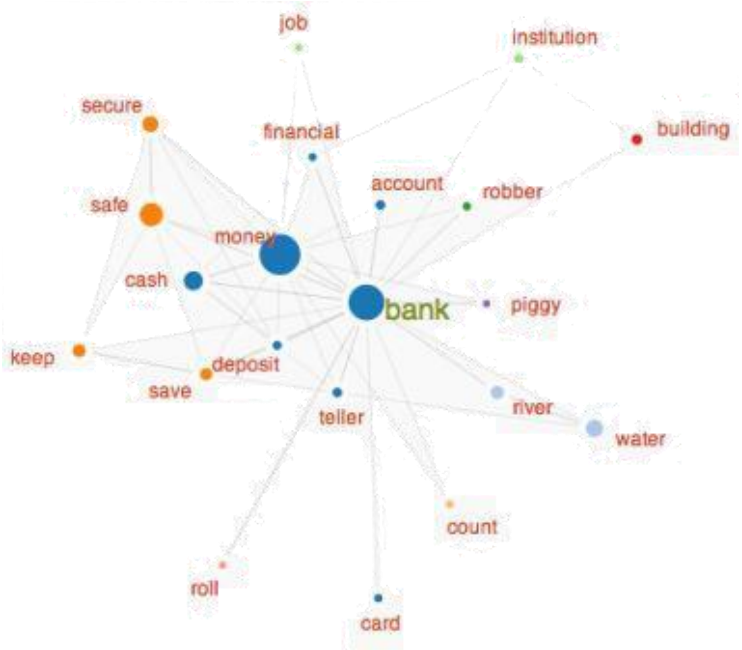
WordNet: semantic relations between nouns

Relation	Also Called	Definition	Example
Hypernym	Superordinate	From concepts to superordinates	$breakfast^1 \rightarrow meal^1$
Hyponym	Subordinate	From concepts to subtypes	$meal^1 \rightarrow lunch^1$
Instance Hypernym	Instance	From instances to their concepts	$Austen^1 \rightarrow author^1$
Instance Hyponym	Has-Instance	From concepts to concept instances	$composer^1 \rightarrow Bach^1$
Member Meronym	Has-Member	From groups to their members	$faculty^2 \rightarrow professor^1$
Member Holonym	Member-Of	From members to their groups	$copilot^1 \rightarrow crew^1$
Part Meronym	Has-Part	From wholes to parts	$table^2 \rightarrow leg^3$
Part Holonym	Part-Of	From parts to wholes	$course^7 \rightarrow meal^1$
Substance Meronym		From substances to their subparts	$water^1 \rightarrow oxygen^1$
Substance Holonym		From parts of substances to wholes	$gin^1 \rightarrow martini^1$
Antonym		Semantic opposition between lemmas	$leader^1 \Longleftrightarrow follower^1$
Derivationally Related Form		Lemmas w/same morphological root	$destruction^1 \Longleftrightarrow destroy^1$

Word Senses

- One of the biggest challenges of NLP is the ambiguity of natural language, e.g., the same word can have many different meanings (senses) depending on the context





Word Sense

- Refers to a lexeme's meaning component

bank[2]

[bangk] Show IPA

–noun

1. an institution for receiving, lending, exchanging, and safeguarding money and, in some cases, issuing notes and transacting other financial business.
2. the office or quarters of such an institution.
3. *Games* .
   a. the stock or fund of pieces from which the players draw.
   b. the fund of the manager or the dealer.
4. a special storage place: *a blood bank; a sperm bank.*
5. a store or reserve.
6. *Obsolete* .
   a. a sum of money, esp. as a fund for use in business.
   b. a moneychanger's table, counter, or shop.

–verb (used without object)

7. to keep money in or have an account with a bank: *Do you bank at the Village Savings Bank?*
8. to exercise the functions of a bank or banker.
9. *Games* . to hold the bank.

bank[1]

[bangk] Show IPA

–noun

1. a long pile or heap; mass: *a bank of earth; a bank of clouds.*
2. a slope or acclivity.
3. *Physical Geography* . the slope immediately bordering a stream course along which the water normally runs.
4. a broad elevation of the sea floor around which the water is relatively shallow but not a hazard to surface navigation.
5. *Coal Mining* . the surface around the mouth of a shaft.
6. Also called **cant**, **superelevation.** the inclination of the bed of a banked road or railroad.
7. *Aeronautics* . the lateral inclination of an aircraft, esp. during a turn.
8. *Billiards, Pool* . the cushion of the table.

–verb (used with object)

9. to border with or like a bank; embank: *banking the river with sandbags at flood stage.*
10. to form into a bank or heap (usually fol. by *up*): *to bank up the snow.*
11. to build (a road or railroad track) with an upward slope from the inner edge to the outer edge at a curve.
12. *Aeronautics* . to tip or incline (an airplane) laterally.
13. *Billiards, Pool* .
    a. to drive (a ball) to the cushion.
    b. to pocket (the object ball) by driving it against the bank.
14. to cover (a fire) with ashes or fuel to make it burn long and

Word Sense Disambiguation (WSD)

- The task of determining which of various senses of a word is invoked in context

- Generally viewed as a categorization task
  - Similar to POS tagging
  - Refer to a particular existing sense repository (e.g. WordNet)

- Alternative view, dividing the usages of a word into different meanings without respect to sense repository
  - Involves unsupervised techniques