Text Analysis in Python

Introduction

* Python is considered a good tool for text analysis
* Natural Language Tool Kit (NLTK) is mostly used library in social science and is alos user-friendly
* TextBlob- new library that has a streamlined interface. This library uses NLTK and another library (pattern) in the background. So this library’s functionality is similar to NLTK.
* Stanford coreNLP- powerful library but not as user-friendly. Written in Java not Python. Python interfaces like wrappers and APIs can be used for integration. This library can be used if we meet limitations with NLTK.

NLTK

* Platform for building Python programs to work with human language data.
* Easu to use interfaces like WordNet, along with text processing libraries for classification, tokenization, stemming, tagging, parsing and sematic reasoning, wrappers for industrial-strength NLP libraries
* Suitable for linguists, engineers, students, educators, researchers, and similar industry users.
* Available for windows, MAC OS, Linux

Python program to process a text file:

>>>for line in open(“file.txt”):

>>>for word in line.split():

>>>if word.endswith(‘ing’):

>>>print(word)

NLTK defines an infrastructure that can be used to build NLP programs in Python. It provides basic classes for representing data relevant to natural language processing-standard interfaces for performing tasks such as part-of-speech tagging, syntactic parsing, text classification, standard implementation for each task which can be combined to solve complex problems.

>>>import nltk

>>>nltk.download()

>>>from nltk import \*

>>>text1

<Text:--- >

>>>text2

<Text:--- >

Searching:

>>>text1.concordance(“payment”)

A concordance view shows every occurrence of the word with source content

>>>text1.similar(“payment”)

>>>text4.dispersion-plot([“city”,”demo”])

NumPy and MatPlotlib packages are required to produce graphical plots

>>>text3.generate()

Generates a random text

>>>len(text2)

Gives the length of the text which contains words, punctuations and tokens

>>>sorted(set(text3))

sorted list of vocabulary items, beginning with various punctuation symbols and continuing with words starting with *A*.

>>>len(set(text3))

Gives only distinct words

>>>text3.count(“make”)

Occurrence of that word

Lexical\_diversity=len(set(text))/len(text)

A **collocation** is a sequence of words that occur together unusually often

>>>list(bigrams([“more”,”is”,”said”,”than”,”done”]))

[(“more”,”is”),(“is”,”said”),(“said”,”than”),(“than”,”done”)]

collocations are essentially just frequent bigrams

| **Example** | **Description** |
| --- | --- |
| fdist = FreqDist(samples) | create a frequency distribution containing the given samples |
| fdist[sample] += 1 | increment the count for this sample |
| fdist['monstrous'] | count of the number of times a given sample occurred |
| fdist.freq('monstrous') | frequency of a given sample |
| fdist.N() | total number of samples |
| fdist.most\_common(n) | the n most common samples and their frequencies |
| for sample in fdist: | iterate over the samples |
| fdist.max() | sample with the greatest count |
| fdist.tabulate() | tabulate the frequency distribution |
| fdist.plot() | graphical plot of the frequency distribution |
| fdist.plot(cumulative=True) | cumulative plot of the frequency distribution |
| fdist1 |= fdist2 | update fdist1 with counts from fdist2 |
| fdist1 < fdist2 | test if samples in fdist1 occur less frequently than in fdist2 |

Comparison operators

| **Function** | **Meaning** |
| --- | --- |
| s.startswith(t) | test if s starts with t |
| s.endswith(t) | test if s ends with t |
| t in s | test if t is a substring of s |
| s.islower() | test if s contains cased characters and all are lowercase |
| s.isupper() | test if s contains cased characters and all are uppercase |
| s.isalpha() | test if s is non-empty and all characters in s are alphabetic |
| s.isalnum() | test if s is non-empty and all characters in s are alphanumeric |
| s.isdigit() | test if s is non-empty and all characters in s are digits |
| s.istitle() | test if s contains cased characters and is titlecased (i.e. all words in s have initial capitals) |

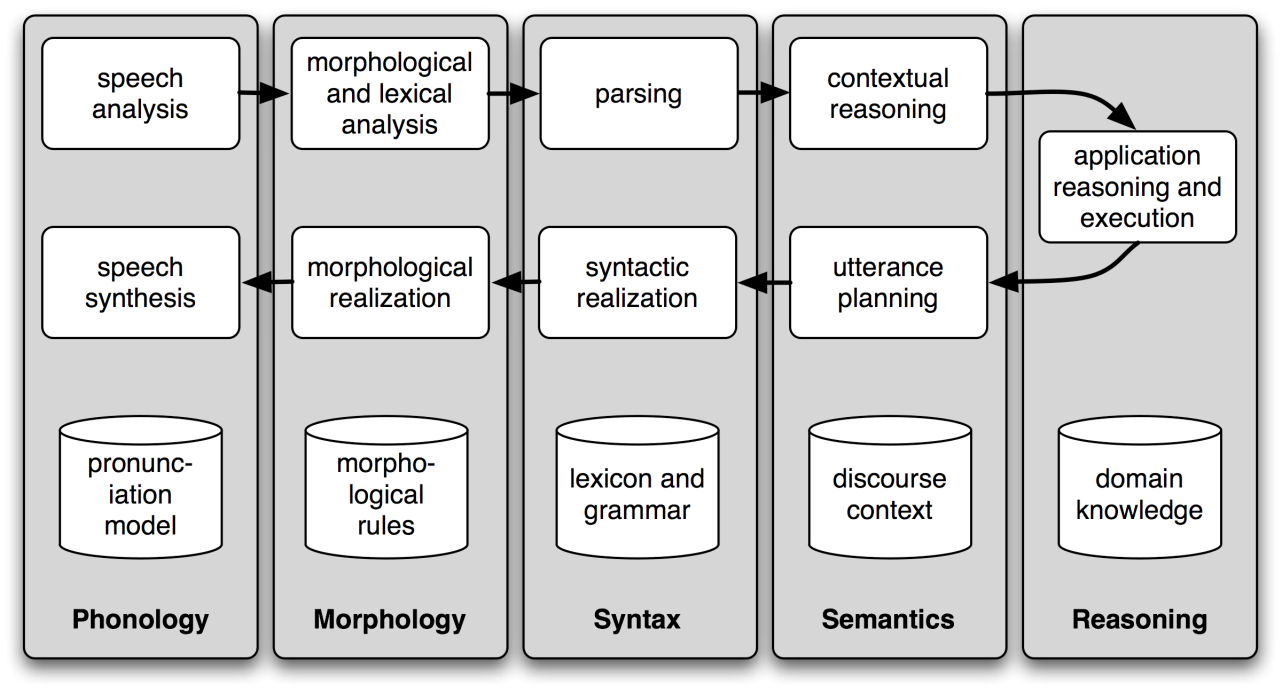
Automatic natural language understanding

Word sense disambiguation:

|  |  |  |
| --- | --- | --- |
| a. |  | The lost children were found by the *searchers* (searchers found the lost children) |

|  |  |  |
| --- | --- | --- |
| b. |  | The lost children were found by the *mountain* (the mountain is where the lost children are) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| c. |  | The lost children were found by the *afternoon* (afternoon is the time when lost children were found)   1. Pronoun resolution-defines “who did what to whom”  |  |  |  | | --- | --- | --- | | a. |  | The thieves stole the paintings. They were subsequently *sold*. |   (the paintings were sold)   |  |  |  | | --- | --- | --- | | b. |  | The thieves stole the paintings. They were subsequently *caught*.(the thieves were caught) |  |  |  |  | | --- | --- | --- | | c. |  | The thieves stole the paintings. They were subsequently *found*. |   (the paintings were found)  Computational techniques for tackling this problem include **anaphora resolution** — identifying what a pronoun or noun phrase refers to — and **semantic role labeling** — identifying how a noun phrase relates to the verb.   1. Generating language output-question answering and machine translation  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | a. |  | *Text:* ... The thieves stole the paintings. They were subsequently sold. ... |  |  |  |  | | --- | --- | --- | | b. |  | *Human:* Who or what was sold? |  |  |  |  | | --- | --- | --- | | c. |  | *Machine:* The paintings. | |  1. Machine transaltion: the machine should be able to translate text into another language correctly. But as the words in a sentence reframe, the machine fails to translate it correctly. Since it does a word-to-word mapping. 2. Spoken dialog systems:   S: How may I help you?  U: When is Saving Private Ryan playing?  S: For what theater?  U: The Paramount theater.  S: Saving Private Ryan is not playing at the Paramount theater, but  it's playing at the Madison theater at 3:00, 5:30, 8:00, and 10:30.  Architecture of a spoken dialog system:  speech analysis🡪 morphological or lexical analysis🡪 parsing🡪 contextual reasoning🡪 application reasoning and execution🡪 utterance planning🡪 syntactic realization🡪 morphological realization🡪 speech synthesis🡪  Spoken input is analyzed, words are recognized, sentences are parsed and interpreted in context, application-specific actions take place, a response is planned, realized as a syntactic structure, then to suitably inflected words, and finally to spoken output; different types of linguistic knowledge inform each stage of the process. |
|  |  |  |



Try: run nltk.chat.chatbots() to see available chatbots

1. Textual entailment: in natural language processing is a directional relation between text fragments. The relation holds whenever the truth of one text fragment follows from another text. In the TE framework, the entailing and entailed texts are termed *text (t)* and *hypothesis (h)*, respectively. Textual entailment is not the same as pure logical entailment- it has a more relaxed definition: "*t* entails *h*" (*t* ⇒ *h*) if, typically, a human reading *t* would infer that *h* is most likely true. The relation is directional because even if "*t* entails *h*", the reverse "*h* entails *t*" is much less certain.

Example:

|  |  |  |
| --- | --- | --- |
| a. |  | Text: David Golinkin is the editor or author of eighteen books, and over 150 responsa, articles, sermons and books |

|  |  |  |
| --- | --- | --- |
| b. |  | Hypothesis: Golinkin has written eighteen books |

In order to determine whether the hypothesis is supported by the text, the system needs the following background knowledge: (i) if someone is an author of a book, then he/she has written that book; (ii) if someone is an editor of a book, then he/she has not written (all of) that book; (iii) if someone is editor or author of eighteen books, then one cannot conclude that he/she is author of eighteen books.

1. Limitation of NLP

One of the biggest limitation now you may apparently notice is machine translation(MT). Even google translate cannot guarantee a good translation without any modifications. The alignment and language modeling have been a challenging issue for researchers to improve MT.

It is also unpredictable.