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) |

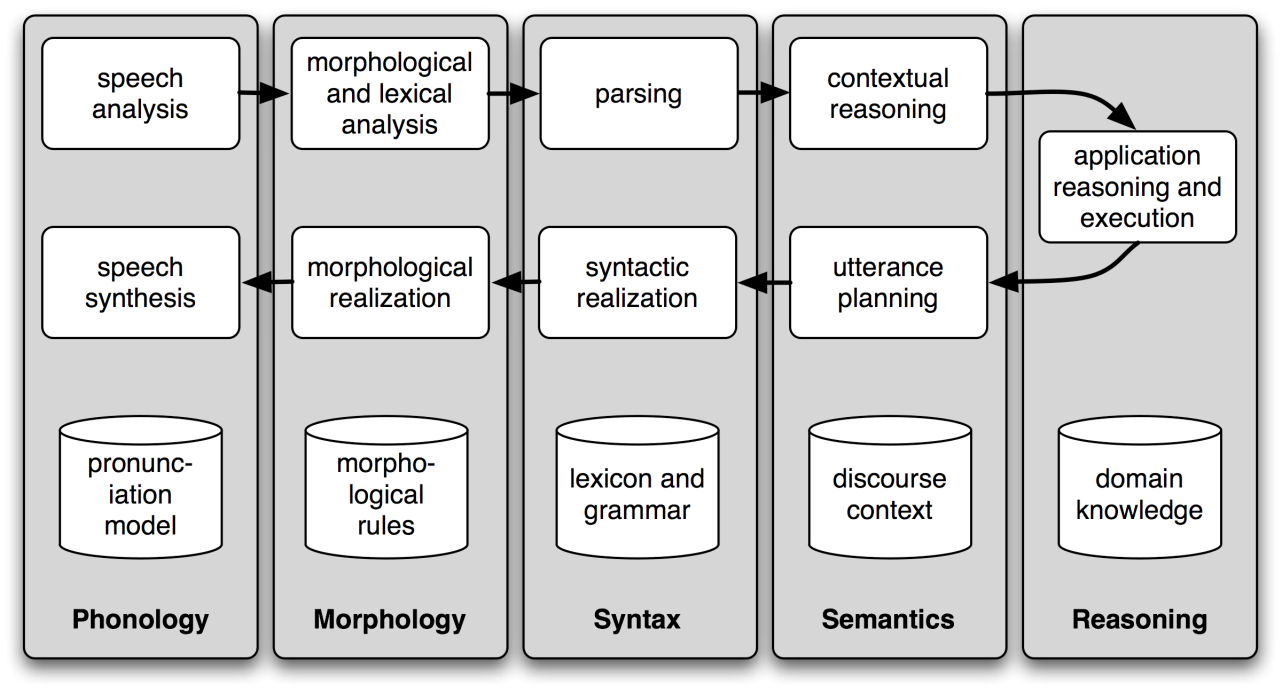
Automatic natural language understanding

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.

Machine learning in Python:

1. Import data:

To load data from scikit, run

# Import `datasets` from `sklearn`

from sklearn import datasets

# Load in the `digits` data

digits = datasets.load\_digits()

# Print the `digits` data

print(digits)

try extracting particular data set.

# Import the `pandas` library as `pd`

import pandas as pd

# Load in the data with `read\_csv()`

digits = pd.read\_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tra", header=None)

# Print out `digits`

print(digits)

the data is already split up in a training and a test set, indicated by the extensions .tra and .tes.

1. Explore data

Data description is not readily available with scikit-learn

1. Basic info on data

To see which keys you have available to already get to know your data, you can just run digits.keys().

# Get the keys of the `digits` data

print(digits.keys())

# Print out the data

print(digits.data)

# Print out the target values

print(digits.target)

# Print out the description of the `digits` data

print(digits.DESCR)

### What's required to create good machine learning systems?

* Data preparation capabilities.
* Algorithms – basic and advanced.
* Automation and iterative processes.
* Scalability.
* Ensemble modeling.

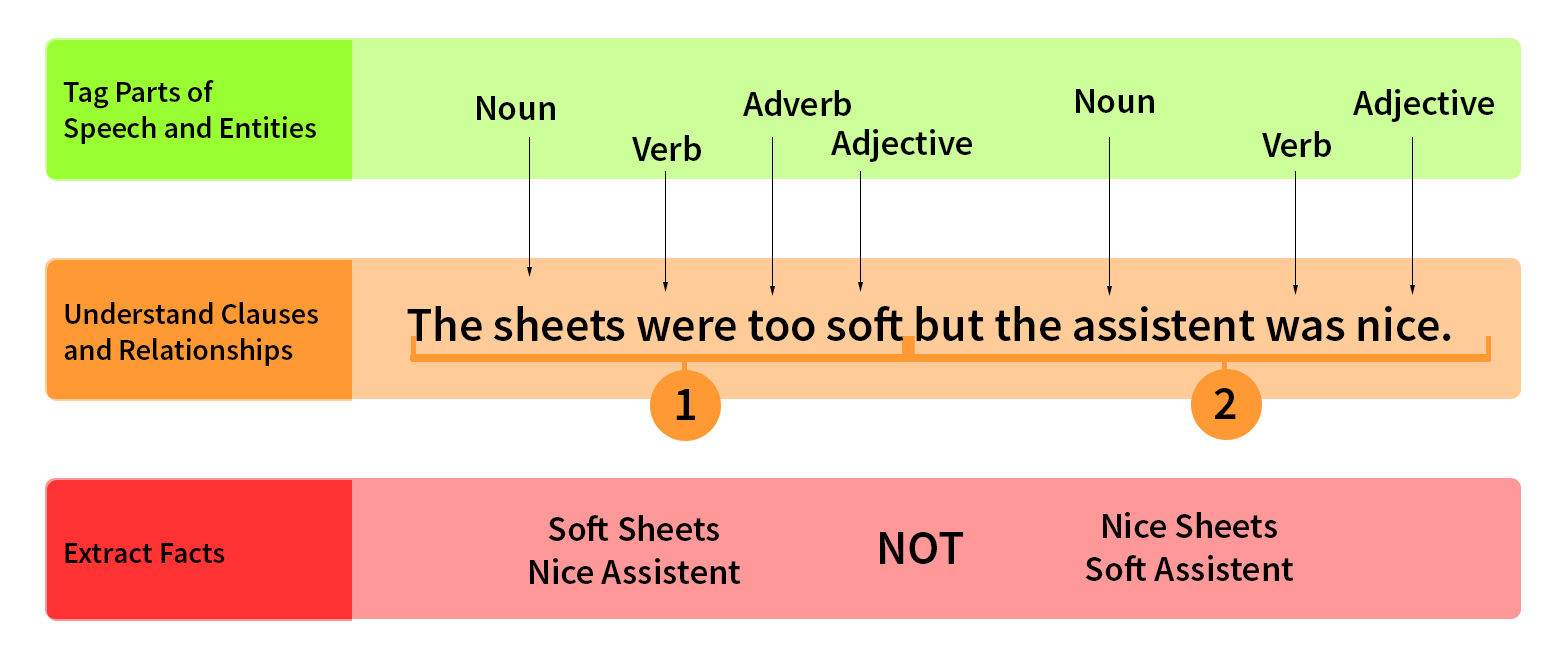
### Did you know?

* In machine learning, a target is called a label.
* In statistics, a target is called a dependent variable.
* A variable in statistics is called a feature in machine learning.
* A transformation in statistics is called feature creation in machine learning.

**Extracting data from unstructured documents**

* 1. **Content Types:**
* Paper but Not just paper
* Email
* Websites
* Electronic Documents
  1. **Documents Types:**
* Contracts
* Mortgage Documents
* Claims
* Customer Correspondence
* Healthcare EOBs
* Proposals
* Social Media
  1. **Business Processes:**
* Simple Search/Locate
* Analytics/Business Intelligence
* Customer Service/Sentiment Analysis
* Case Management
* Legal Discovery
* Report Generation
  1. **Technologies:**
* Data Entry
* [Text Analytics](http://en.wikipedia.org/wiki/Text_mining)/Word Bags (Simplest form: Search Engine, Word counting, content tagging (People, places, objects, addresses, dates, etc.)
* [Natural Language Processing](http://en.wikipedia.org/wiki/Natural_language_processing) (Clustering | JacardDistance | nGram | Lexicon |Grammar POS | Dictionary Lookup)
* [Machine Learning](http://en.wikipedia.org/wiki/Machine_learning) (That explores the construction and study of [algorithms](http://en.wikipedia.org/wiki/Algorithm) that can [learn](http://en.wikipedia.org/wiki/Learning) from data)

NLP difficulties:



**Common mining tasks:**

***Pre-processing****:* This involves all the preliminary tasks that can help in getting started with any of the actual mining tasks. Pre-processing could be removing anomalies and noise from the data that’s about to be mined, filling in missing values, normalising the data or compressing data using techniques like generalisation and aggregation.

***Clustering****:* This is partitioning a huge set of data into related sub-classes.  
***Classification****:* This is tagging or classifying data items into different user-defined categories.  
*Outlier analysis* helps in identifying those data elements which are deviant or distant from the rest of the elements in a dataset. This can help in anomaly detection.

***Associative analysis*** helps in bringing out hidden relationships among data items in a large data set. This can help in predicting the occurrence of a particular item in a transaction or an event whenever some other item is present. You can think of this as a conditional probability.  
*Regression* is used to predict values of a dependent variable by constructing a model or a mathematical function out of independent variables.

***Summarization*** helps in coming up with a compact description for the whole data set.  
Data mining is a combination of various techniques like pattern recognition, statistics, machine learning, etc. While there is a good amount of intersection between machine learning and data mining, as both go hand in hand and machine learning algorithms are used for mining data, we will restrict ourselves in this article to only those tools specialized for data mining.

* 1. **Weka**: open source java based software. It comprises a collection of machine learning algorithms for data mining. It packages tools for data pre-processing, classification, regression, clustering, association rules and visualisation. The various ways of accessing it are – Weka Knowledge Explorer, Experimenter, Knowledge Flow and a simple CL. Explorer is a user-friendly graphical interface for two-dimensional visualisation of mined data. It lets you import the raw data from various file formats, and supports well known algorithms for different mining actions like filtering, clustering, classification and attribute selection. However, when dealing with large data sets, it is best to use a CL based approach as Explorer tries to load the whole data set into the main memory, causing performance issues. This software also provides a Java Appetiser for use in applications and can connect to databases using CJD. Weka has proved to be an ideal choice for educational and research purposes, as well as for rapid prototyping.
  2. **Rapid miner**: Besides the standard data mining features like data cleansing, filtering, clustering, etc, the software also features built-in templates, repeatable work flows, a professional visualisation environment, and seamless integration with languages like Python and R into work flows that aid in rapid prototyping. The tool is also compatible with weak scripts. Rapid Miner is used for business/commercial applications, research and education.
  3. **Orange**: It is a Python library that powers Python scripts with its rich compilation of mining and machine learning algorithms for data pre-processing, classification, modelling, regression, clustering and other miscellaneous functions. Orange also comes with a visual programming environment and its workbench consists of tools for importing data, and dragging and dropping widgets and links to connect different widgets for completing the workflow. The visual programming comes with an easy-to-use UI, with plenty of online tutorials for assistance. Due to the ease of programming and integration in Python, Orange can be a great take off point for novices and experts to plunge into data mining.
  4. **Knime:** Knime is one of the leading open source analytic, integration and reporting platforms that comes as free software and as well as a commercial version. Written in Java and built upon Eclipse, its access is through a GUI that provides options to create the data flow and conduct data pre-processing, collection, analysis, modelling and reporting.
  5. **DataMelt:** DataMelt or DMelt does much more than just data mining. It is a computational platform, offering statistics, numeric and symbolic computations, scientific visualisation, etc. DMelt provides data mining features like linear regression, curve fitting, cluster analysis, neural networks, fuzzy algorithms, analytic calculations and interactive visualisations using 2D/3D plots and histograms. One can play around with its IDE (integrated development kit) or its functions can be called from applications using its Java API. Both community and commercial editions of DMelt are available on Linux, Mac OS, Windows and Android platforms. DMelt is a successor to the jHepWork and SCaVis programs, which some people working in data analysis might be familiar with. This software is well suited for students, engineers and scientists.
  6. **Apache Mohout:** Mahout is primarily a library of machine learning algorithms that can help in clustering, classification and frequent pattern mining. It can be used in a distributed mode that helps easy integration with Hadoop. Mahout is currently being used by some of the giants in the tech industry like Adobe, AOL, Drupal and Twitter, and it has also made an impact in research and academics. It can be a great choice for anyone looking for easy integration with Hadoop and to mine huge volumes of data.
  7. **ELKI:** ELKI is open source software written in Java and licensed under AGPLv3. This software focuses especially on cluster analysis and outlier detection with a compilation of numerous algorithms from both these domains. The software is accessed through a GUI that displays the results once the selected algorithm is run. ELKI’s design goals are performance, scalability, completeness, extensibility and a modular design to welcome contributions. ELKI currently doesn’t offer professional support and the software is optimised for use in science and research. Hence, this option works best for those in research.
  8. **MOA:** Massive Online Analysis (MOA), as the name suggests, is primarily data stream mining software that is well suited for applications that need to handle volumes of real-time data streams at a high speed. MOA is distributed under GNU GPL, and can be used via the command line, GUI or Java API. It is a rich compilation of machine learning algorithms and has proved to be a great choice during the design of real-time applications. Stream mining algorithms typically require faster computations without storing all of the datasets in the memory and have to get the work done within a limited time. MOA is well suited for these requirements. Weka and MOA can be closely linked to each other and either of the classifiers can be called from the other one. For those looking to analyse and mine information from real-time data, MOA can be the best choice.
  9. **KEEL:** KEEL (Knowledge Extraction for Evolutionary Learning) is a Java based open source tool distributed under GPLv3. It is powered by a well-organised GUI that lets you manage (import, export, edit and visualise) data with different file formats, and to experiment with the data (through its data pre-processing, statistical libraries and some standard data mining and evolutionary learning algorithms). Since KEEL is based on Java, JVM has to be installed on the system to run its GUI and do data mining experiments. You may visit http://keel.es/ for the complete list of supported algorithms. KEEL is ideal for research and educational purposes. It serves as a useful aid for teachers.
  10. **RATTLE:** Rattle, expanded to ‘R Analytical Tool To Learn Easily’, has been developed using the R statistical programming language. The software can run on Linux, Mac OS and Windows, and features statistics, clustering, modelling and visualisation with the computing power of R. Rattle is currently being used in business, commercial enterprises and for teaching purposes in Australian and American universities.