Natural Language Processing With Python Notes

Sunday, October 14, 2018

11:48 AM

PREFACE NOTES:

* Natural Language: a langauge that is used for everyday communication by humans.
* Natural Language Processing (NLP) for computers means covers, in a wide sense, any kind of computer manipulation of natural language
* LIST OF THINGS THAT USE NLP:

phones and handheld computers support predictive text and handwriting recognition; web search engines give access to information locked up in unstructured text; machine translation allows us to retrieve texts written in Chinese and read them in Spanish; text analysis enables us to detect sentiment in tweets and blogs. By providing more natural human-machine interfaces, and more sophisticated access to stored information, language processing has come to play a central role in the multilingual information society.

*From <*[*https://www.nltk.org/book/ch00.html*](https://www.nltk.org/book/ch00.html)*>*

* Preface is an introduction of what NLP is and applications it is used for.
* Reasons to use/know NLP: used for people working in human-computer interaction, business information analysis, and web software development.
* The book is meant to teach someone to write programs that analyze written language, regardless of previous programming experience.
* Book is meant to establish a healthy divide between theory and applicational use of NLP
* This is what the book outlines that will be learned throughout the next chapters:

-How simple programs can help you manipulate and analyze language data, and how to write these programs

-How key concepts from NLP and linguistics are used to describe and analyse language

-How data structures and algorithms are used in NLP

-How language data is stored in standard formats, and how data can be used to evaluate the performance of NLP techniques

*From <*[*https://www.nltk.org/book/ch00.html*](https://www.nltk.org/book/ch00.html)*>*

* Python is object oriented; meaning each variable is an entity that has certain defined attributes and methods.
* To apply a method to an object: write the object name followed by a period, followed by the method name.
* Methods have arguments expressed inside parenthesis
* Book explains that Python is good for NLP and other things because it is easy to pick up regardless of programming experience.
* NLTK = Natural Language Tool Kit
* Preface details many changes in the Python 3 and NLTK 3 versions of python.
* Things the book thinks I should download:

<http://nltk.org/>.

|  |  |
| --- | --- |
| **Python:** | The material presented in this book assumes that you are using Python version 3.2 or later. (Note that NLTK 3.0 also works with Python 2.6 and 2.7.) |
| **NLTK:** | The code examples in this book use NLTK version 3.0. Subsequent releases of NLTK will be backward-compatible with NLTK 3.0. |
| **NLTK-Data:** | This contains the linguistic corpora that are analyzed and processed in the book. |
| **NumPy:** | (recommended) This is a scientific computing library with support for multidimensional arrays and linear algebra, required for certain probability, tagging, clustering, and classification tasks. |
| **Matplotlib:** | (recommended) This is a 2D plotting library for data visualization, and is used in some of the book's code samples that produce line graphs and bar charts. |
| **Stanford NLP Tools:** |  |
|  | (recommended) NLTK includes interfaces to the Stanford NLP Tools which are useful for large scale language processing (see <http://nlp.stanford.edu/software/>). |
| **NetworkX:** | (optional) This is a library for storing and manipulating network structures consisting of nodes and edges. For visualizing semantic networks, also install the *Graphviz* library. |
| **Prover9:** | (optional) This is an automated theorem prover for first-order and equational logic, used to support inference in language processing. |

*From <*[*https://www.nltk.org/book/ch00.html*](https://www.nltk.org/book/ch00.html)*>*

NLTK was designed with four primary goals in mind:

|  |  |
| --- | --- |
| **Simplicity:** | To provide an intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP without getting bogged down in the tedious house-keeping usually associated with processing annotated language data |
| **Consistency:** | To provide a uniform framework with consistent interfaces and data structures, and easily-guessable method names |
| **Extensibility:** | To provide a structure into which new software modules can be easily accommodated, including alternative implementations and competing approaches to the same task |
| **Modularity:** | To provide components that can be used independently without needing to understand the rest of the toolkit |

*From <*[*https://www.nltk.org/book/ch00.html*](https://www.nltk.org/book/ch00.html)*>*

NATURAL LANGUAGE PROCESSING AND PYTHON NOTES:

* Chapter addresses the following questions:

1. What can we achieve by combining simple programming techniques with large quantities of text?
2. How can we automatically extract key words and phrases that sum up the style and content of a text?
3. What tools and techniques does the Python programming language provide for such work?
4. What are some of the interesting challenges of natural language processing?

*From <*[*https://www.nltk.org/book/ch01.html*](https://www.nltk.org/book/ch01.html)*>*

* This chapter shows how to use and understand the Python IDLE using various examples
* We think of text as nothing more than a sequence of words and punctuation for NLP
* Indexes are a common way to access the words of a text or basically the elements of a list

we can identify the elements of a Python list by their order of occurrence in the list. The number that represents this position is the item's **index**.

*From <*[*https://www.nltk.org/book/ch01.html*](https://www.nltk.org/book/ch01.html)*>*

* Indexes are a common way to access a given word in a text
* Slicing: the ability to extract manageable pieces of language from large texts. Sub-lists.
* Frequency Distribution: Tells us the frequency of each vocabulary item in the text.
* Collocations: a sequence of words that occur together unusually often. EX: red wine.
* Bigrams: A list of word pairs

Functions Defined for NLTK's Frequency Distributions

|  |  |
| --- | --- |
| **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 |

*From <*[*https://www.nltk.org/book/ch01.html*](https://www.nltk.org/book/ch01.html)*>*

* Token: technical name for a sequence of characters
* Video on creating a virtual environment in python: <https://www.youtube.com/watch?v=Feu-qKBwMf4>
* Word sense Disambiguation: when we want to work out which sense of a word was intended in a given context.
* Text alignment: automatically pairing up sentences
* Summary:

* Texts are represented in Python using lists: ['Monty', 'Python']. We can use indexing, slicing, and the len() function on lists.
* A word "token" is a particular appearance of a given word in a text; a word "type" is the unique form of the word as a particular sequence of letters. We count word tokens using len(text) and word types using len(set(text)).
* We obtain the vocabulary of a text t using sorted(set(t)).
* We operate on each item of a text using [f(x) for x in text].
* To derive the vocabulary, collapsing case distinctions and ignoring punctuation, we can write set(w.lower() for w in text if w.isalpha()).
* We process each word in a text using a for statement, such as for w in t: or for word in text:. This must be followed by the colon character and an indented block of code, to be executed each time through the loop.
* We test a condition using an if statement: if len(word) < 5:. This must be followed by the colon character and an indented block of code, to be executed only if the condition is true.
* A frequency distribution is a collection of items along with their frequency counts (e.g., the words of a text and their frequency of appearance).
* A function is a block of code that has been assigned a name and can be reused. Functions are defined using the def keyword, as in def mult(x, y); x and y are parameters of the function, and act as placeholders for actual data values.
* A function is called by specifying its name followed by zero or more arguments inside parentheses, like this: texts(), mult(3, 4), len(text1).

*From <*[*https://www.nltk.org/book/ch01.html*](https://www.nltk.org/book/ch01.html)*>*