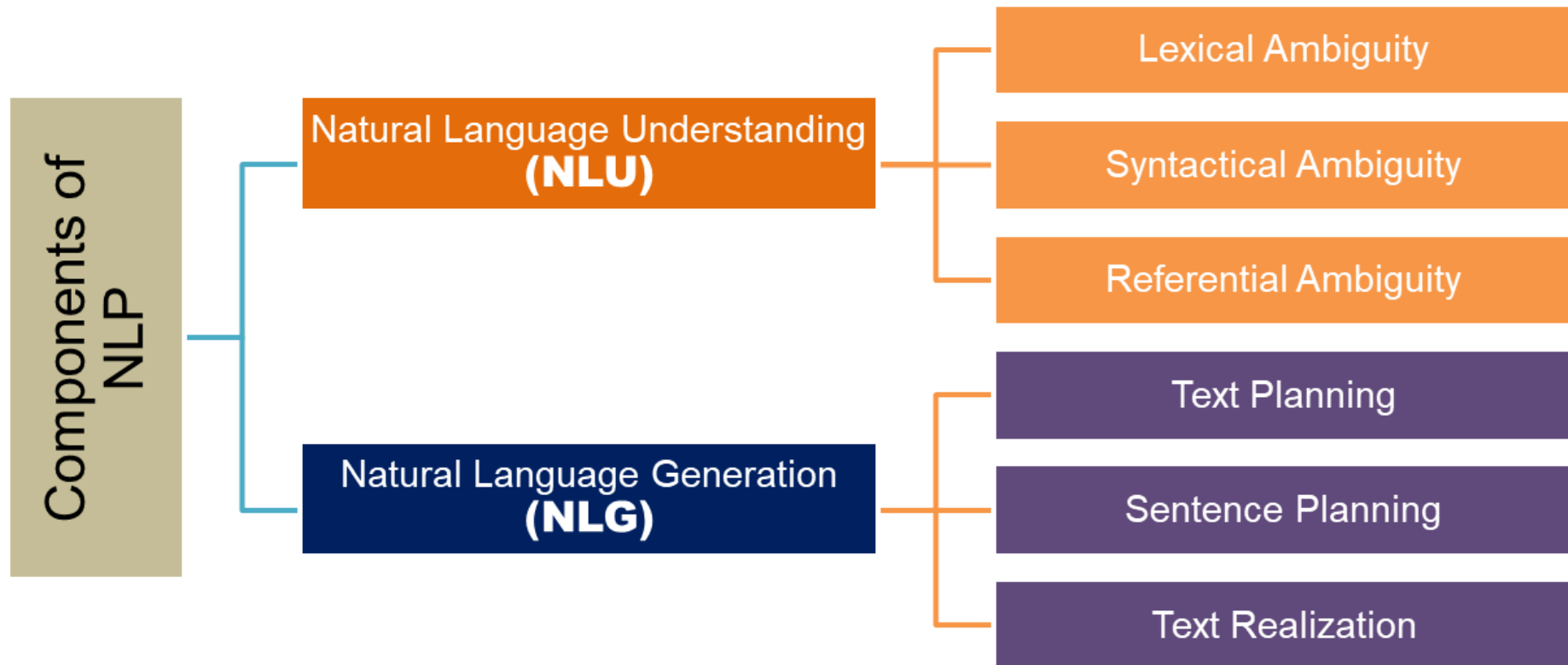


Components of NLP

NLP can be divided into two basic components.

- Natural Language Understanding
- Natural Language Generation



Natural Language Understanding (NLU)

NLU is naturally harder than NLG tasks. Really? Let's see what are all challenges faced by a machine while understanding.

There are lot of ambiguity while learning or trying to interpret a language.

He is looking for a match

What do you understand by 'match'?

Partner

Or **Cricket/Football Match**

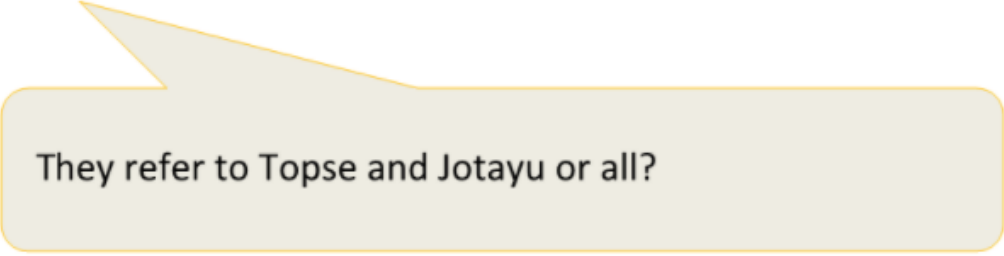
Lexical Ambiguity can occur when a word carries different sense, i.e. having more than one meaning and the sentence in which it is contained can be interpreted differently depending on its correct sense. Lexical ambiguity can be resolved to some extent using parts-of-speech tagging techniques.

The chicken is ready to eat.

Is the chicken ready to eat his food or the chicken is ready for someone else to it? You never know.

Syntactical Ambiguity means when we see more than one meaning in a sequence of words. It is also termed as grammatical ambiguity.

Feluda met Topse and Jotayu. **They** went to restaurant



They refer to Topse and Jotayu or all?

Referential Ambiguity: Very often a text mentions as entity (something/someone), and then refers to it again, possibly in a different sentence, using another word. Pronoun causing ambiguity when it is not clear which noun it is referring to

Natural Language Generation (NLG)

It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.

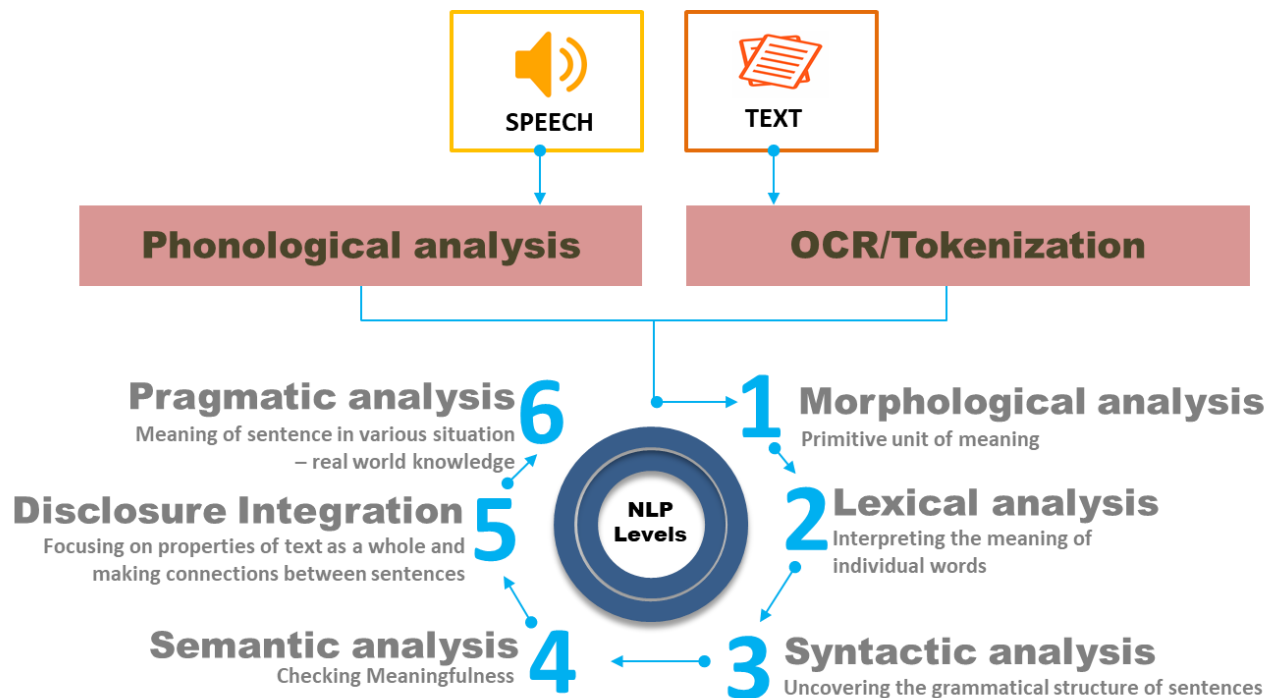
It involves –

- **Text planning** – It includes retrieving the relevant content from knowledge base.
- **Sentence planning** – It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
- **Text Realization** – It is mapping sentence plan into sentence structure.

Levels of NLP

In the previous sections we have discussed different problem associated to NLP. Now let us see what are all typical steps involved while performing NLP tasks. We should keep in mind that the below section describes some standard workflow, it may however differ drastically as we do real life implementations basis on our problem statement or requirements.

The source of Natural Language could be speech (sound) or Text.



Phonological Analysis: This level is applied only if the text origin is a speech. It deals with the interpretation of speech sounds within and across words. Speech sound might give a big hint about the meaning of a word or a sentence.

It is study of organizing sound systematically. This requires a broad discussion and is out of scope of our current note.

Morphological Analysis: Deals with understanding distinct words according to their morphemes (the smallest units of meanings) . Taking, for example, the word: “**unhappiness** ”. It can be broken down into three morphemes (prefix, stem, and suffix), with each conveying some form of meaning: the prefix un- refers to “not being”, while the suffix -ness refers to “a state of being”. The stem happy is considered as a free morpheme since it is a “word” in its own right. Bound morphemes (prefixes and suffixes) require a free morpheme to which it can be attached to, and can therefore not appear as a “word” on their own.

Lexical Analysis: It involves identifying and analysing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of txt into paragraphs, sentences, and words. In order to deal with lexical analysis, we often need to perform **Lexicon Normalization**.

The most common lexicon normalization practices are Stemming:

- **Stemming:** Stemming is a rudimentary rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.
- **Lemmatization:** Lemmatization, on the other hand, is an organized & step by step procedure of obtaining the root form of the word, it makes use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations).

Syntactic Analysis: Deals with analysing the words of a sentence so as to uncover the grammatical structure of the sentence. E.g.. "Colourless green idea." This would be rejected by the Symantec analysis as colourless here; green doesn't make any sense.

Syntactical parsing involves the analysis of words in the sentence for grammar and their arrangement in a manner that shows the relationships among the words. Dependency Grammar and Part of Speech tags are the important attributes of text syntactics.

Semantic Analysis: Determines the possible meanings of a sentence by focusing on the interactions among word-level meanings in the sentence. Some people may think it's the level which determines the meaning, but actually all the level do. The semantic analyser disregards sentence such as "hot ice-cream".

Discourse Integration: Focuses on the properties of the text as a whole that convey meaning by making connections between component sentences. It means a sense of the context. The meaning of any single sentence which depends upon that sentences. It also considers the meaning of the following sentence. For example, the word "that" in the sentence "He wanted that" depends upon the prior discourse context.

Pragmatic Analysis: Explains how extra meaning is read into texts without actually being encoded in them. This requires much world knowledge, including the understanding of intentions, plans, and goals. Consider the following two sentences:





- The city police refused the demonstrators a permit because they feared violence.
- The city police refused the demonstrators a permit because they advocated revolution.

The meaning of "they" in the 2 sentences is different. In order to figure out the difference, world knowledge in knowledge bases and inference modules should be utilized.

Pragmatic analysis helps users to discover this intended effect by applying a set of rules that characterize cooperative dialogues. E.g., "close the window?" should be interpreted as a request instead of an order.

Widely used NLP Libraries

There are many libraries, packages, tools available in market. Each of them has its own pros and cons. As a market trend Python is the language which has most compatible libraries. Below table will gives a summarised view of features of some of the widely used libraries. Most of them provide the basic NLP features which we discussed earlier. Each NLP libraries were built with certain objectives, hence it is quite obvious that a single library might not provide solutions for everything, it is the developer who need to use those and that is where experience and knowledge matters when and where to use what.

Tools	Features
 NLTK	<ul style="list-style-type: none"> ▪ The most well-known and full NLP library ▪ Plenty of approaches to each NLP task ▪ Supports large number of languages ▪ No integrated Word Vectors
 spaCy	<ul style="list-style-type: none"> ▪ Fastest NLP framework ▪ Easy to learn as it has one single highly optimized tool for each task ▪ Supports neural networks for training some models ▪ Lesser Language support
 learn NLP toolkit	<ul style="list-style-type: none"> ▪ Most effective for Machine Learning implementation ▪ Good documentation available ▪ No neural network support for text processing
 gensim	<ul style="list-style-type: none"> ▪ Works with large datasets and processes data streams ▪ Supports Deep Learning ▪ Designed primarily of unsupervised text modeling

NLP Hands on Using Python NLTK (Simple Examples)

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources.

Latest version: NLTK 3.5 release: April 2020, add support for Python 3.8, drop support for Python 2.

NLTK comes with many corpora, toy grammars, trained models, etc. A complete list is posted at: http://nltk.org/nltk_data/.

Before we start doing experiments on some of the techniques which are widely used during Natural Language Processing task, let's first get hands on into the installation.

NLTK Installation

If you are using Windows or Linux or Mac, you can install NLTK using pip:

```
$ pip install nltk
```

Optionally you can also use Anaconda prompt.

```
$ conda install nltk
```



```
Anaconda Prompt - conda install nltk
(base) C:\Users\Dibyendu>conda install nltk
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: E:\Anaconda3

added / updated specs:
- nltk

The following packages will be downloaded:

package | build
-----|-----
certifi-2020.4.5.1 | py37_0 159 KB
conda-4.8.3 | py37_0 3.0 MB
openssl-1.1.1f | he774522_0 5.8 MB
-----|-----
Total: 9.0 MB

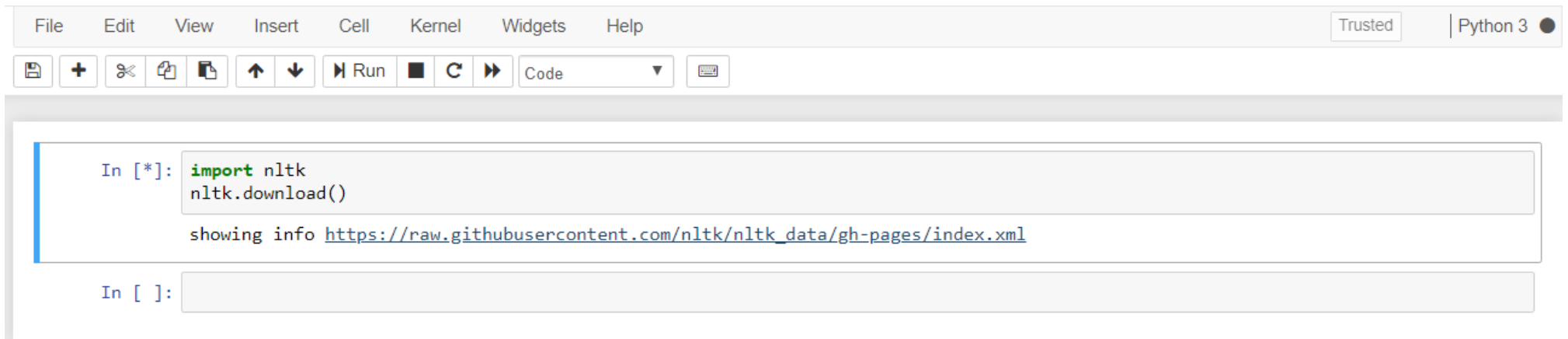
The following packages will be UPDATED:

certifi 2019.11.28-py37_0 --> 2020.4.5.1-py37_0
conda 4.8.2-py37_0 --> 4.8.3-py37_0
openssl 1.1.1d-he774522_4 --> 1.1.1f-he774522_0

Proceed ([y]/n)?
```

If everything goes fine, that means you've successfully installed NLTK library. Once you've installed NLTK, you should install the NLTK packages by running the following code:

Open your Jupyter Notebook and run the below commands.



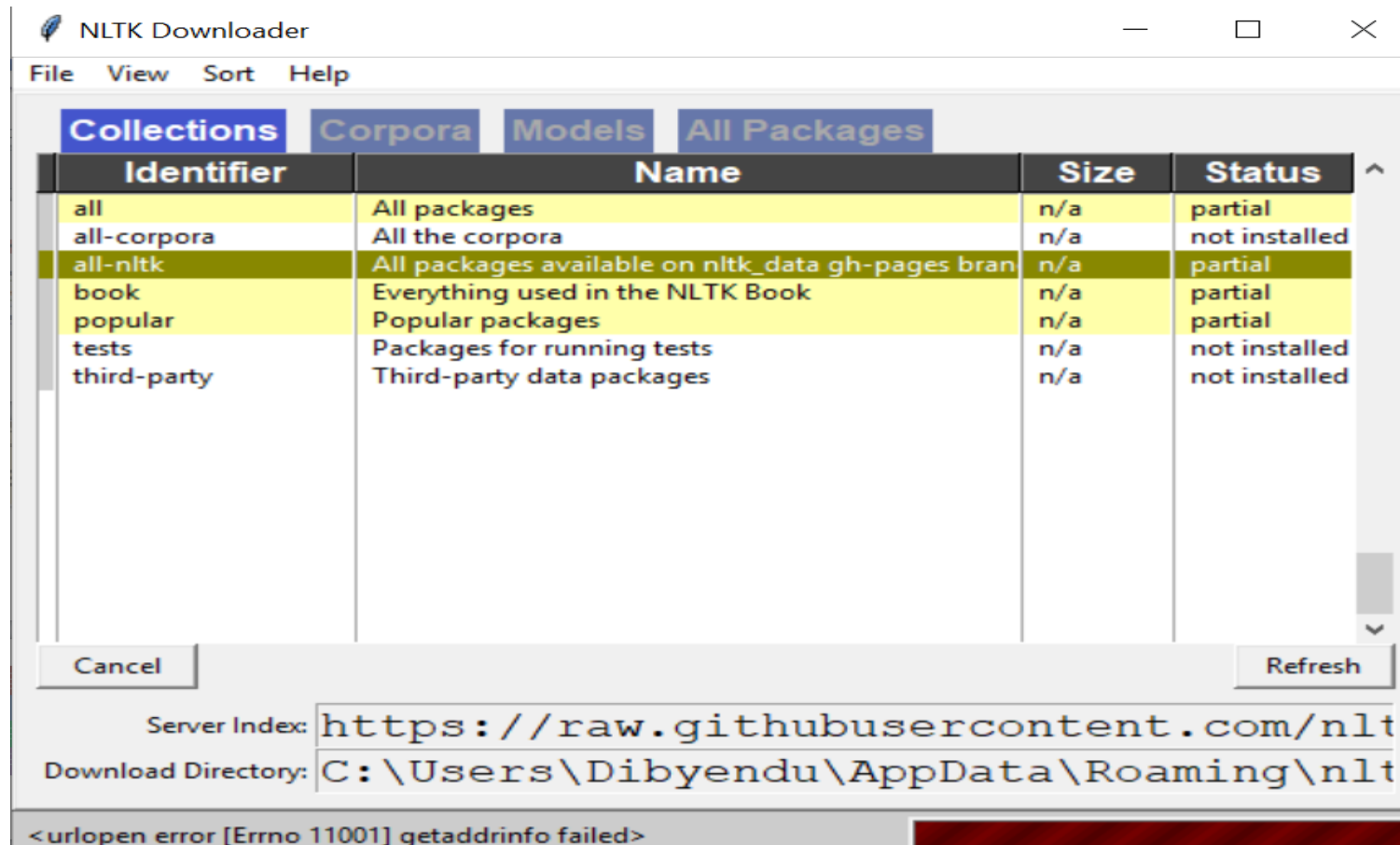
The image shows a Jupyter Notebook interface. At the top is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. To the right of the menu bar is a 'Trusted' status indicator and a 'Python 3' environment selector. Below the menu bar is a toolbar with icons for saving, adding a new cell, undo, redo, copy, paste, and navigation arrows. There are also buttons for 'Run', a 'Code' dropdown menu, and a terminal icon. The main area contains two code cells. The first cell, labeled 'In [*]:', contains the code `import nltk` and `nltk.download()`. Below the code, it says 'showing info' followed by a URL: https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml. The second cell, labeled 'In []:', is empty.

```
In [*]: import nltk
        nltk.download()

        showing info https://raw.githubusercontent.com/nltk/nltk\_data/gh-pages/index.xml

In [ ]:
```

This will show the NLTK downloader to choose what packages need to be installed. You can install all packages since they have small sizes, so no problem. Now let's start the show.



NLTK Downloader

FileViewSortHelp

CollectionsCorporaModelsAll Packages

Identifier	Name	Size	Status
twitter_samples	Twitter Samples	15.3 MB	not installed
udhr	Universal Declaration of Human Rights Corpus	1.1 MB	not installed
udhr2	Universal Declaration of Human Rights Corpus (Ur	1.6 MB	not installed
unicode_samples	Unicode Samples	1.2 KB	not installed
universal_tagset	Mappings to the Universal Part-of-Speech Tagset	18.6 KB	not installed
universal_treebanks_v	Universal Treebanks Version 2.0	24.7 MB	not installed
vader_lexicon	VADER Sentiment Lexicon	88.4 KB	not installed
verbnet	VerbNet Lexicon, Version 2.1	316.1 KB	not installed
verbnet3	VerbNet Lexicon, Version 3.3	470.7 KB	not installed
webtext	Web Text Corpus	631.1 KB	not installed
wmt15_eval	Evaluation data from WMT15	374.1 KB	not installed
word2vec_sample	Word2Vec Sample	47.1 MB	not installed
wordnet	WordNet	10.3 MB	not installed
wordnet_ic	WordNet-InfoContent	11.5 MB	not installed
words	Word Lists	740.0 KB	not installed
ycoe	York-Toronto-Helsinki Parsed Corpus of Old Engli	0.5 KB	not installed

CancelRefresh

Server Index:
Download Directory:

https://raw.githubusercontent.com/nlt

C:\Users\Dibyendu\AppData\Roaming\nlt

Downloading package 'punkt'

Basic NLP Operations: Do Yourself

Tokenize Text

Tokenization is the first step in NLP. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

A word (Token) is the minimal unit that a machine can understand and process. So any text string cannot be further processed without going through tokenization. Tokenization is the process of splitting the raw string into meaningful tokens. The complexity of tokenization varies according to the need of the NLP application, and the complexity of the language itself. For example, in English it can be as simple as choosing only words and numbers through a regular expression. But for Chinese and Japanese, it will be a very complex task.

Sentence Tokenization

Sentence tokenizer breaks text paragraph into sentences.

```
: from nltk.tokenize import sent_tokenize
text="Hello friends!. Good Morning! Today we will learn Natural Language Processing. It is very interesting"
tokenized_text=sent_tokenize(text)
print(tokenized_text)

['Hello friends!.', 'Good Morning!', 'Today we will learn Natural Language Processing.', 'It is very interesting']
```

Word Tokenization

Word tokenizer breaks text paragraph into words.

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Hello', 'friends', '!', '.', 'Good', 'Morning', '!', 'Today', 'we', 'will', 'learn', 'Natural', 'Language', 'Processing',  
'.', 'It', 'is', 'very', 'interesting']
```

Stopwords Removal

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK(Natural Language Toolkit) in python has a list of stopwords stored in 16 different languages.

```
: from nltk.corpus import stopwords
  from nltk.tokenize import word_tokenize

text = "India is my country"
text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in stopwords.words()]

print(tokens_without_sw)

['India', 'country']
```

You can see that the words is, my have been removed from the sentence.

Part of speech tagging

In your childhood, you may have heard the term Part of Speech (POS). It can really take good amount of time to get the hang of what adjectives and adverbs actually are. What exactly is the difference? Think about building a system where we can encode all this knowledge. It may look very easy, but for many decades, coding this knowledge into a machine learning model was a very hard NLP problem. POS tagging algorithms can predict the POS of the given word with a higher degree of precision. You can get the POS of individual words as a tuple

```
: from nltk import word_tokenize, pos_tag
  s = "The name of our country is India"
  print(pos_tag(word_tokenize(s)))

[('The', 'DT'), ('name', 'NN'), ('of', 'IN'), ('our', 'PRP$'), ('country', 'NN'), ('is', 'VBZ'), ('India', 'NNP')]
```

If you want to know the details of the POS, here is the way. Note we might need to download the 'tagset'. Below example shows NN is noun.

```
: nltk.download('tagsets')
nltk.help.upenn_tagset('NN')

[nltk_data] Downloading package tagsets to
[nltk_data] C:\Users\Dibyendu\AppData\Roaming\nltk_data...

NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed thermostat
    investment slide humour falloff slick wind hyena override subhumanity
    machinist ...

[nltk_data] Unzipping help\tagsets.zip.
```

For better understanding below is the other POS that we found in our example.


```
nltk.help.upenn_tagset('DT')
nltk.help.upenn_tagset('IN')
nltk.help.upenn_tagset('VBZ')
nltk.help.upenn_tagset('PRP$')
nltk.help.upenn_tagset('NNP')
```

DT: determiner
all an another any both del each either every half la many much nary
neither no some such that the them these this those

IN: preposition or conjunction, subordinating
astride among uppon whether out inside pro despite on by throughout
below within for towards near behind atop around if like until below
next into if beside ...

VBZ: verb, present tense, 3rd person singular
bases reconstructs marks mixes displeases seals carps weaves snatches
slumps stretches authorizes smolders pictures emerges stockpiles
seduces fizzes uses bolsters slaps speaks pleads ...

PRP\$: pronoun, possessive
her his mine my our ours their thy your

NNP: noun, proper, singular
Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos
Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin ODI Darryl CTCA
Shannon A.K.C. Meltex Liverpool ...

The meanings of all available POS codes are given below for your reference.

CC	Coordinating conjunction	NNS	Noun, plural	UH	Interjection
CD	Cardinal number	NNP	Proper noun, singular	VB	Verb, base form
DT	Determiner	NNPS	Proper noun, plural	VBD	Verb, past tense
EX	Existential there	PDT	Predeterminer	VBG	Verb, gerund or present participle
FW	Foreign word	POS	Possessive ending	VCN	Verb, past participle
IN	Preposition or subordinating conjunction	PRP	Personal pronoun	VBP	Verb, non-3rd person singular present
JJ	Adjective	PRP\$	Possessive pronoun	VBZ	Verb, 3rd person singular present
JJR	Adjective, comparative	RB	Adverb	WDT	Wh-determiner
JJS	Adjective, superlative	RBR	Adverb, comparative	WP	Wh-pronoun
LS	List item marker	RBS	Adverb, superlative	WP\$	Possessive wh-pronoun
MD	Modal	RP	Particle	WRB	Wh-adverb
NN	Noun, singular or mass	SYM	Symbol		
		TO	to		

Now look into an interesting though of information retrieval using POS tagging. I got an article about Cricket, trying to see what countries are mentioned in the document. Country names are proper noun, so using POS I can easily filter and get only the proper nouns. Apart from countries it may retrieve more words which are proper noun, but it make our job easy as none of the country name will missed out.

```

import nltk
from nltk.tokenize import PunktSentenceTokenizer

document = """Domestic cricket seasons in Australia, New Zealand, South Africa, India, Pakistan, Sri Lanka,Bangladesh,
Zimbabwe and the West Indies may therefore span two calendar years, A cricket season in England is described as a single
year. e.g. "2009". An international ODI series or tournament may be for a much shorter duration, and Cricinfo treats
this issue by stating any series or matches. In the record tables, a two-year span generally indicates that the record
was set within a domestic season in one of the above named countries."""
sentences = nltk.sent_tokenize(document)

data = []
for sent in sentences:
    data = data + nltk.pos_tag(nltk.word_tokenize(sent))

for word in data:
    if 'NNP' in word[1]:
        print(word)

```

```

('Australia', 'NNP')
('New', 'NNP')
('Zealand', 'NNP')
('South', 'NNP')
('Africa', 'NNP')
('India', 'NNP')
('Pakistan', 'NNP')
('Sri', 'NNP')
('Lanka', 'NNP')
('Bangladesh', 'NNP')
('Zimbabwe', 'NNP')
('West', 'NNP')
('Indies', 'NNPS')
('A', 'NNP')
('England', 'NNP')
('ODI', 'NNP')
('Cricinfo', 'NNP')

```

Stemming and Lemmatization

```
from nltk.stem import PorterStemmer
e_words= ["Played", "Playing", "Play"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord)
```

```
play
play
play
```

Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

Based on the applicability you can choose any of the below lemmatizer

- Wordnet Lemmatizer
- Spacy Lemmatizer
- TextBlob
- CLiPS Pattern
- Stanford CoreNLP
- Gensim Lemmatizer
- TreeTagger

Here is one quick example using Wordnet lemmatizer.

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("plays"))
```

play

How to get Word Meanings, Synonyms and Antonyms

WordNet is a large lexical database of English. It is a widely used NLTK corpus. 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.

WordNet's structure makes it a useful tool for computational linguistics and natural language processing. You can simply import using

```
from nltk.corpus import wordnet
```

In the below simple example, let's try to see how easily we can get the synonym and antonym of the word 'love'. It's really cool!

```
from nltk.corpus import wordnet
syn = wordnet.synsets("love")
print(syn[0].definition())
print(syn[0].examples())
```

```
a strong positive emotion of regard and affection
['his love for his work', 'children need a lot of love']
```

```

synonyms = []
antonyms = []
for syn in wordnet.synsets('love'):
    for lemma in syn.lemmas():
        synonyms.append(lemma.name())
        if lemma.antonyms():
            antonyms.append(lemma.antonyms()[0].name())
print("***Synonyms of 'love' :",synonyms)
print("***Antonyms of 'love' ",antonyms)

```

```

***Synonyms of 'love' : ['love', 'love', 'passion', 'beloved', 'dear', 'dearest', 'honey', 'love', 'love', 'sexual_love', 'erotic_love', 'love', 'sexual_love', 'lovemaking', 'making_love', 'love', 'love_life', 'love', 'love', 'enjoy', 'love', 'sleep_together', 'roll_in_the_hay', 'love', 'make_out', 'make_love', 'sleep_with', 'get_laid', 'have_sex', 'know', 'do_it', 'be_intimate', 'have_intercourse', 'have_it_away', 'have_it_off', 'screw', 'fuck', 'jazz', 'eff', 'hump', 'lie_with', 'bed', 'have_a_go_at_it', 'bang', 'get_it_on', 'bonk']
***Antonyms of 'love' ['hate', 'hate']

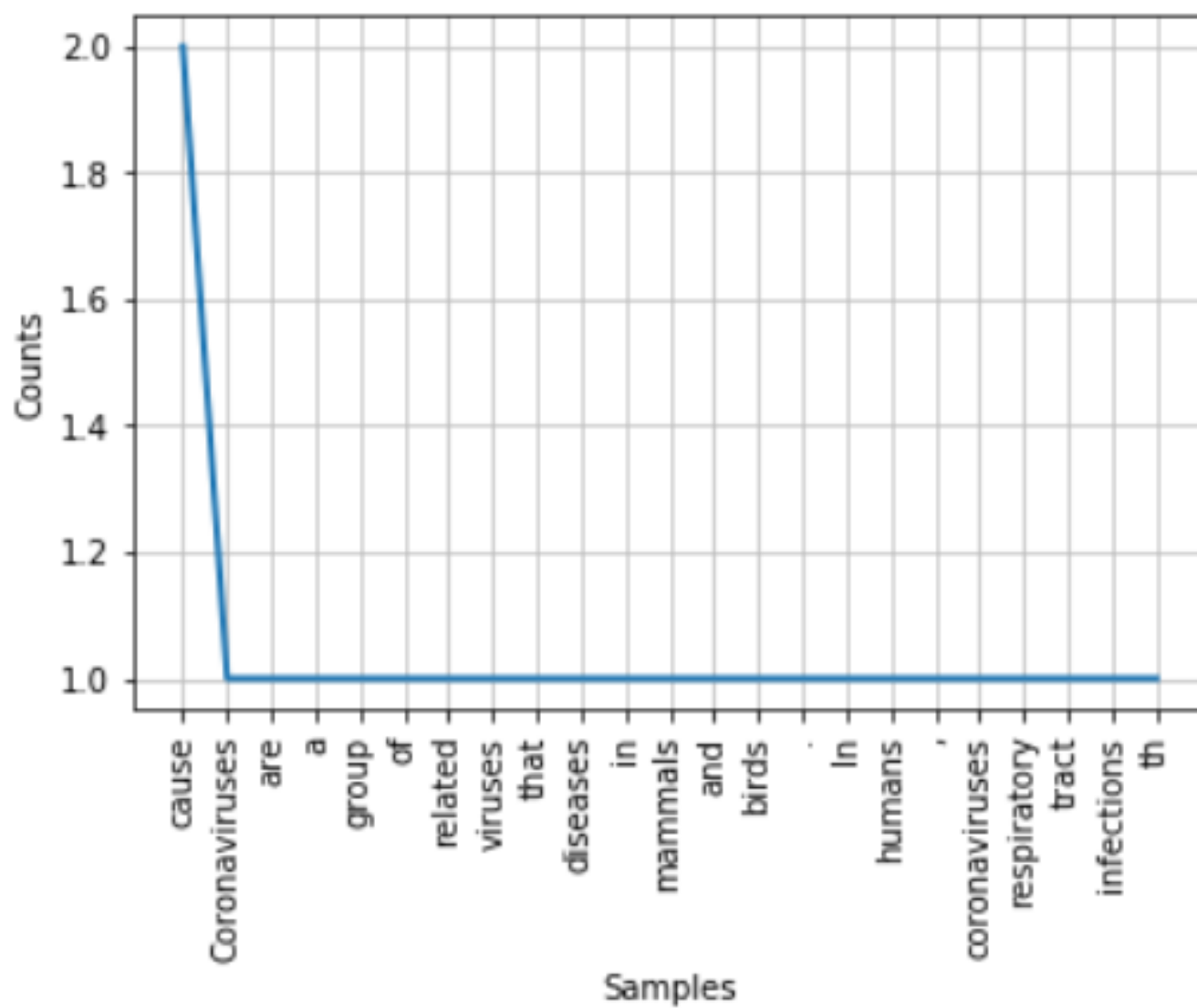
```

Work Frequency: Quick Visualization

In the below example lets' try to read some text from live url and see the frequencies of words.

```
import bs4 as bs
import urllib.request
import nltk
URL='https://en.wikipedia.org/wiki/Coronavirus'
# Gettings the data source
source = urllib.request.urlopen(URL).read()
# Parsing the data/ creating BeautifulSoup object
soup = bs.BeautifulSoup(source,'xml')
# Fetching the data
text = ""
for paragraph in soup.find_all('p'):
    text += paragraph.text

text = text[:150]
words = nltk.tokenize.word_tokenize(text)
fd = nltk.FreqDist(words)
fd.plot()
```



NLP, what is the future?

As we have seen, NLP provides a wide set of techniques and tools which can be applied in all the areas of life. By learning them and using them in our everyday interactions, our life quality would highly improve, as well as we could also improve the lives of those who surround us.

NLP techniques help us improving our communications, our goal reaching and the outcomes we receive from every interaction. They also allow us overcome personal obstacles and psychological problems. NLP help us using tools and techniques we already have in us without being aware of it.

Everything is a lot faster and better because we can now communicate with machines, thanks to natural language processing technology. Natural language processing has afforded major companies the ability to be flexible with their decisions thanks to its insights of aspects such as customer sentiment and market shifts. Smart organizations now make decisions based not on data only, but on the intelligence derived from that data by NLP-powered machines.

As NLP becomes more mainstream in the future, there may be a massive shift toward this intelligence-driven way of decision making across global markets and industries.

If there is one thing we can guarantee will happen in the future, it is the integration of natural language processing in almost every aspect of life as we know it. The past five years have been a slow burn of what NLP can do, thanks to integration across all manner of devices, from computers and fridges to speakers and automobiles.

Humans, for one, have shown more enthusiasm than a dislike for the human-machine interaction process. NLP-powered tools have also proven their abilities in such a short time.

These factors are going to trigger increased integration of NLP: ever-growing amounts of data generated in business dealings worldwide, increasing smart device use and higher demand for elevated service by customers.

As regards natural language processing, the sky is the limit. The future is going to see some massive changes as the technology becomes more mainstream and more advancement in the ability are explored. As a major facet of artificial intelligence, natural language processing is also going to contribute to the proverbial invasion of robots in the workplace, so industries everywhere have to start preparing.

References:

Books

- Natural Language Processing with Python - Written by Steven Bird, Ewan Klein and Edward Loper. O'Reilly.
- Natural Language Processing: Python and NLTK Learning Path - Written by Nitin Hardeniya, Jacob Perkins, Deepti Chopra, Nisheeth Joshi, Iti Mathur. Packt
- Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data by Dipanjan Sarkar. Apress
- NLTK Essentials by Nitin Hardeniya. Packt
- Natural Language Processing Recipes: Unlocking Text Data with Machine Learning and Deep Learning using Python by Akshay Kulkarni, Adarsha Shivananda. Apress