NLP ASSIGNMENT 1: THE CONCEPTS AND THE PROCEDURES EXPLAINED

NLTK library is a special NLP- based python library that provides various intuitive and easy-to-use classes for functionalities such as tokenization, stop- symbol/ stopword operations,stemming/ lemmatization, etc.

Major processes explained:

1. **Tokenization:**

   Tokenization is the process by which a large quantity of text is divided into smaller parts called tokens. These tokens are very useful for finding patterns and are considered as a base step for stemming and lemmatization.

   Here, we deal with sentence tokenization (division of the given text data into the constituent sentences) and word tokenization (division of the given text data into the constituent words and symbols).

   Each type bears a specific significance on the assignment that I have performed. Sentence tokenization can be used to know about the total number of sentences which gives the idea about the size of textual information. And, word tokenization can be used to know about the words and symbols present in the text. Because of the fact that word tokenization doesn’t just signify the word-tokens, in a sense , the term, “word tokenization” is a misnomer.

2. **Stop- symbol/ Stopword operations:**

   I have mentioned above that because of the fact that word tokenization doesn’t just signify the word-tokens, in a sense , the term, “word tokenization” is a misnomer. The conceptual analysis following that statement relates to stop- symbol/ stopword operations.

   In order to obtain words only, we need to perform an operation known as stop- symbol (or punctuation-symbol) cleaning. With the use of a regular expression library (known as “re”), here in our assignment, I have replaced all the non-alphabetical characters (coded as [^a-zA-Z]) by a blank space by using the substitute() method.

   Yet, there’s another issue we might have to deal with post the stop- symbol cleaning. In some cases where the semantic significance of the constituent words matters, we need to get rid of semantically-low-significance words known as “stopwords”. The list of stopwords in the domain of the english language contains the following words:

["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now"]

Stopword removal has two benefits:

A.Getting rid of semantically-low-significance words greatly increases the overall efficiency of problems constituting semantic analysis to a considerable extent.

B.Getting rid of semantically-low-significance words gives us a filtered-down version of the original corpus which is smaller in size and hence relatively much more easy to deal with. Filtered-down version of the original corpus finds great use in Vectorization processes such as Count- Vectorization, TFIDF-Vectorization, etc.

3.Stemming/ Lemmatization:

Stemming and Lemmatization, both of these operations, are very similar in their purpose of operation. Both of these processes attempt to map various semantically identical versions of a word to a common word to increase computational efficiency. But, in the way they operate, they differ.

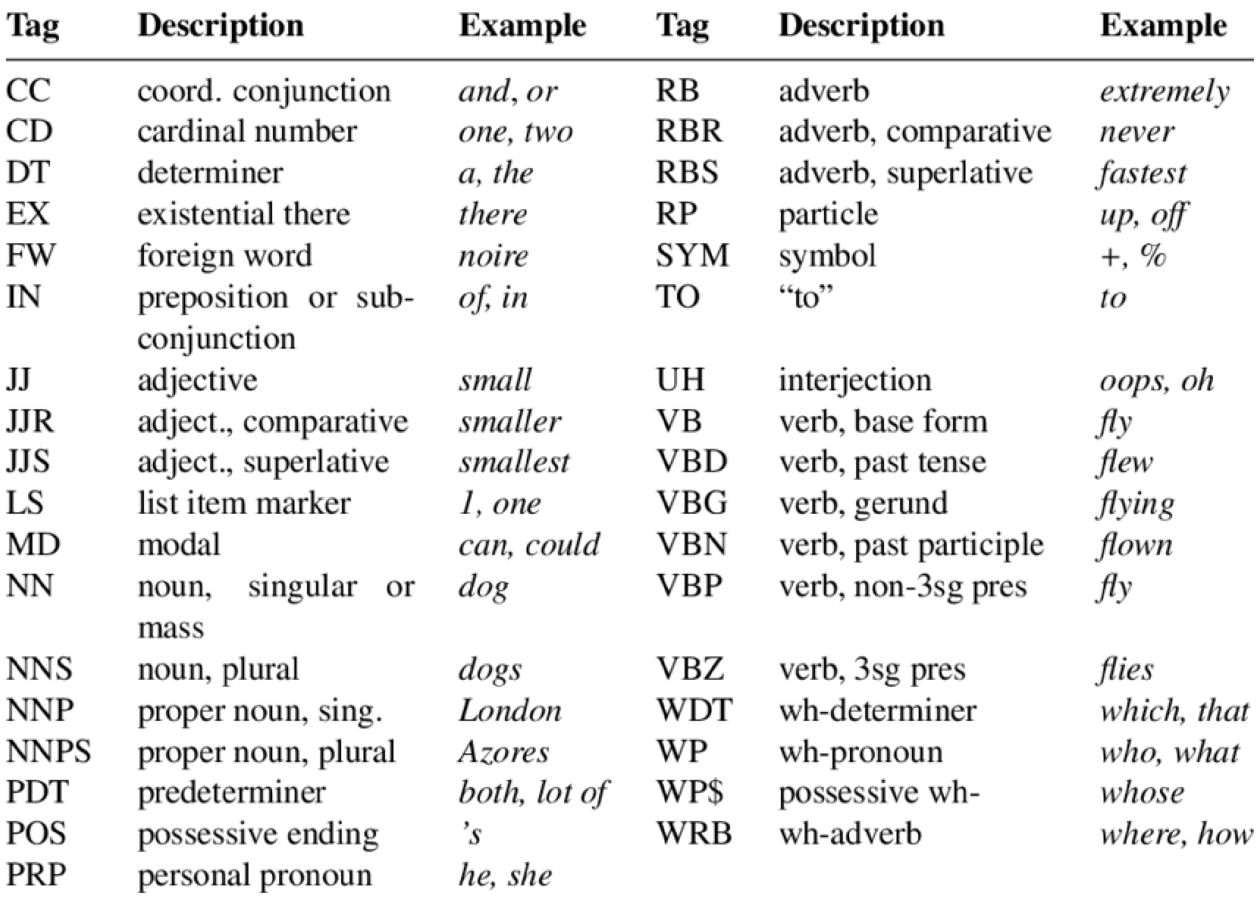
Stemming identifies the common root form of a word by removing or replacing word suffixes (e.g. “singing” is stemmed as “sing”), while lemmatization identifies the inflected forms of a word and returns its base form (e.g. “better” is lemmatized as “good”).

As we can observe, in terms of accuracy and range of functionality, stemming is relatively more limited than lemmatization, e.g, “better” is lemmatized as “good” but, can’t be stemmed to “good” since “”better” is not constructed by adding affixes to “good”. But, in terms of ease of operations, stemming is relatively easier than lemmatization.

Libraries used are PorterStemmer (for stemming) and WordNetLemmatizer(for lemmatization).

4. POS- Tagging

POS Tagging, acronym for Part-Of-Speech Tagging, in NLTK is a process to mark up the words in text format for a particular part of a speech based on its definition and context. Some NLTK POS tagging examples are: CC, CD, EX, JJ, MD, NNP, PDT, PRP$, TO, etc. POS tagger is used to assign grammatical information of each word of the sentence. The following tags, respective descriptions and examples can be used as a basic guide to understand POS-Tagging with Nltk.



Tag	Description	Example	Tag	Description	Example
CC	coord. conjunction	*and, or*	RB	adverb	*extremely*
CD	cardinal number	*one, two*	RBR	adverb, comparative	*never*
DT	determiner	*a, the*	RBS	adverb, superlative	*fastest*
EX	existential there	*there*	RP	particle	*up, off*
FW	foreign word	*noire*	SYM	symbol	+, %
IN	preposition or sub-conjunction	*of, in*	TO	"to"	*to*
JJ	adjective	*small*	UH	interjection	*oops, oh*
JJR	adject., comparative	*smaller*	VB	verb, base form	*fly*
JJS	adject., superlative	*smallest*	VBD	verb, past tense	*flew*
LS	list item marker	*1, one*	VBG	verb, gerund	*flying*
MD	modal	*can, could*	VBN	verb, past participle	*flown*
NN	noun, singular or mass	*dog*	VBP	verb, non-3sg pres	*fly*
NNS	noun, plural	*dogs*	VBZ	verb, 3sg pres	*flies*
NNP	proper noun, sing.	*London*	WDT	wh-determiner	*which, that*
NNPS	proper noun, plural	*Azores*	WP	wh-pronoun	*who, what*
PDT	predeterminer	*both, lot of*	WP$	possessive wh-	*whose*
POS	possessive ending	*'s*	WRB	wh-adverb	*where, how*
PRP	personal pronoun	*he, she*			

fig: POS- tags, respective descriptions and examples

5. Chunking and Chinking with NLTK:

Chunking is defined as the process of natural language processing used to identify parts of speech and short phrases present in a given sentence. Chunking is used to get the required phrases from a given sentence. However, POS tagging can be used only to spot the parts of speech that every word of the sentence belongs to.

Chinking is the process of removing a sequence of tokens from a chunk. Chinking is a lot like chunking, it is basically a way for you to remove a chunk from a chunk. The chunk that you remove from your chunk is your chink. The code is very similar, you just denote the chink, after the chunk, with }{ instead of the chunk's {}.

5. Standard cases of Chunking :

There are many standard cases of chunking that are generally used in common sentences of the English language. One of them is the case of a noun-phrase. And the other is a verb phrase. Sometimes, they are also used in various combinations such as using the noun- phrase/ s and noun- phrase/s in the same sentence. The following are some common cases:

5.1. Noun Phrases (NP):

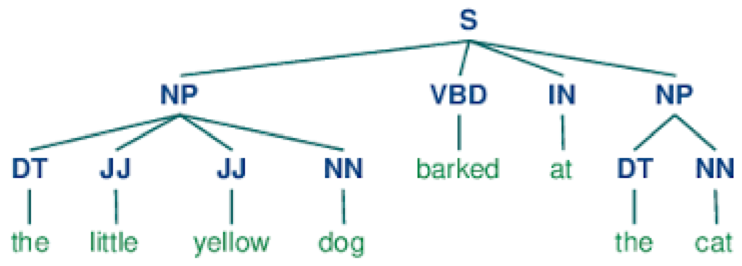


Fig: Noun phrase grammatically defined as the combination: "(At least one) Determiner Followed by (At least one) Adjective Followed by Singular Noun"

Though noun phrases are very commonly used phrases, there are several structures that the noun phrases can assume which can be a little confusing sometimes. To make it more understandable, the following various cases of noun phrase that can be commonly observed in sentences can be studied and analyzed:

Case 1: Noun followed by Noun

Grammar: 'NP: {<NN.?>+<NN.?>}'

Case 2: (At least one) Determiner Followed by (At least one) Adjective Followed by Singular Noun

Grammar: 'NP: {<DT>*<JJ>*<NN>}'
The above figure labeled ``Noun phrase grammatically defined as the combination: "(At least one) Determiner Followed by (At least one) Adjective Followed by Singular Noun"' represents this case.

5.2. Noun Phrases (NP) and Verb Phrases (VP):

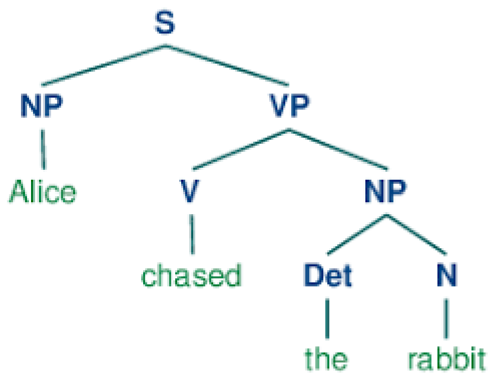


Fig: A sentence containing the combination of Noun Phrases and Verb Phrases

The case of the noun phrases and verb phrases is quite interesting because the chunk, “noun-phrase” can itself be a part of “verb- phrase”, so here, we might observe some sort of nesting of chunk- operations during the chunking of verb- phrases.

Grammar:

1. ‘V:{ <VB>* <VBD>* <VBG>* <VBN>* <VBP>* <VBZ>* }’ #Defining a verb.
2. 'VP: {<V><NP>*<RB>*}’ #Defining a verb phrase.

References:

1. https://www.guru99.com/pos-tagging-chunking-nltk.html
2. https://www.pythonstudio.us/language-processing/noun-phrase-chunking.html