NLTK summary:

**Spet-1: Data cleaning**

The most of today’s data is unstructured and is tidy. Data cleaning plays an important role.



1. **Noise entities removal:**

* General approach is to have a dictionary of all the noise words and iterate the text by tokens and eliminate the tokens from the dictionary.
* Use regular expressions to remove special characters, punctuations, URL’s and Hashtags etc.…

1. **Lexicon Normalization:** For ex: – “play”, “player”, “played”, “plays” and “playing” are the different variations of the word – “play”, Though they mean different but contextually all are similar.

Most common lexicon normalization practices are:

1. Stemming: stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word

*from nltk.stem.porter import PorterStemmer*

*stem = PorterStemmer()*

1. Lemmatization: it’s a step-by-step process which uses vocabulary (is the preferred way)

*from nltk.stem.wordnet import WordNetLemmatizer*

*lem = WordNetLemmatizer()*

1. **Object Standardization:**

Text data often contains words or phrases which are not present in any standard lexical dictionaries. These pieces are not recognized by search engines and models.

Eg: acronyms, hashtags with attached words, and colloquial slangs. With the help of regular expressions and manually prepared data dictionaries, this type of noise can be fixed.

There are many other steps in cleaning the data:

* **Apostrophe Lookup:** To avoid any word sense disambiguation in text, it is recommended to maintain proper structure in it and to abide by the rules of context free grammar. When apostrophes are used, chances of disambiguation increases.

APPOSTOPHES = {“'s" : " is", "'re" : " are", ...} ## Need a huge dictionary

* **Slang lookup, Grammar checking, Spelling corrections.**

**Text to Feature [feature engineering on text]**

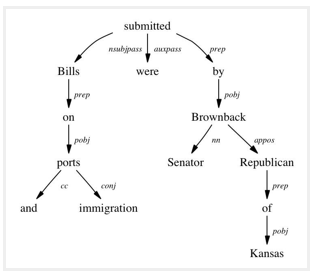
To analyze a preprocessed data, it needs to be converted into features. Depending upon the usage, text features can be constructed using assorted techniques – Syntactical Parsing, Entities / N-grams / word-based features, Statistical features, and word embedding’s.

1. Syntactical parsing

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 Parts of speech tagging are the important attributes of text syntactic.

1. Dependency tree:

Sentence are composed of some words and sewed together. The relationship among the words in a sentence is determined by the basic dependency grammar. Consider the sentence – “*Bills* *on ports and immigration were submitted by Senator Brownback, Republican of Kansas.”* The relationship among the words can be observed in the form of a tree representation as shown: 



The tree shows that “submitted” is the root word of this sentence, and is linked by two sub-trees (subject and object subtrees). Each subtree is itself a dependency tree with relations such as – (“Bills” <-> “ports” <by> “proposition” relation), (“ports” <-> “immigration” <by> “conjugation” relation).

1. Parts of speech tagging:

Every word in a sentence is associated with a parts of speech tag (noun, verb, adjective, and adverb). POS tag defines the usage and function of a word in the sentence

1. Entity extraction: [entities as features]

Entities are defined as the most important chunks of a sentence – noun phrases, verb phrases or both.

*Topic Modelling & Named Entity Recognition are the two key entity detection methods in NLP.*

* 1. **Named Entity Recognition:**

The process of detecting the named entities such as person names, location names, company names etc from the text is called as NER. For example:

Sentence –> Sergey Brin, the manager of Google Inc. is walking in the streets of New York.

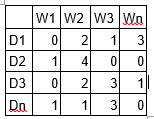
Named Entities –> (“person”: “Sergey Brin”), (“org”: “Google Inc.”), (“location”: “New York”)

* 1. **Topic modeling** is a process of automatically identifying the topics present in a text corpus (“a repeating pattern of co-occurring terms in a corpus”).

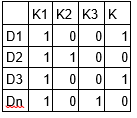
A good topic model results in –

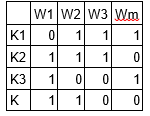
* “health”, “doctor”, “patient”, “hospital” for a topic – Healthcare, and
* “Farm”, “crops”, “wheat” for a topic – “Farming”.

LDA is a matrix factorization technique. In vector space, any corpus (collection of documents) can be represented as a document-term matrix. The following matrix shows a corpus of N documents D1, D2, D3 … Dn and vocabulary size of M words W1,W2 .. Wn. The value of i,j cell gives the frequency count of word Wj in Document Di.

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/08/Modeling2.png)

LDA converts this Document-Term Matrix into two lower dimensional matrices – M1 and M2.  
M1 is a document-topics matrix and M2 is a topic – terms matrix with dimensions (N,  K) and (K, M) respectively, where N is the number of documents, K is the number of topics and M is the vocabulary size.

* [](https://www.analyticsvidhya.com/wp-content/uploads/2016/08/modeling3.png)

[](https://www.analyticsvidhya.com/wp-content/uploads/2016/08/Modeling4.png)

Notice that these two matrices already provides topic word and document topic distributions, however, these distribution needs to be improved, which is the main aim of LDA. LDA makes use of sampling techniques in order to improve these matrices.

* 1. **N-gram:**

A combination of N words together are called N-Grams. N grams (N > 1) are generally more informative as compared to words (Unigrams) as features. Also, bigrams (N = 2) are considered as the most important features of all the others.

Statistical feature extraction:

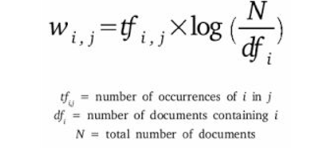
Text data can also be quantified directly into numbers using several techniques.

**Term frequency – Inverse document frequency:** used for information retrieval problems

It aims to convert the text documents into vector models on the basis of occurrence of words in the documents without taking considering the exact ordering

**Term Frequency (TF)** – TF for a term “t” is defined as the count of a term “t” in a document “D”

**Inverse Document Frequency (IDF)** – IDF for a term is defined as logarithm of ratio of total documents available in the corpus and number of documents containing the term T.



Word embedding:

Word embedding ix the modern way of representing words as vectors. The aim of word embedding is to redefine the high dimensional word features into low dimensional feature vectors by preserving the contextual similarity in the corpus. They are widely used in deep learning models such as Convolutional Neural Networks and Recurrent Neural Networks.

**Word2Vec** and **GloVe** are the two popular models to create word embedding of a text. These models takes a text corpus as input and produces the word vectors as output.