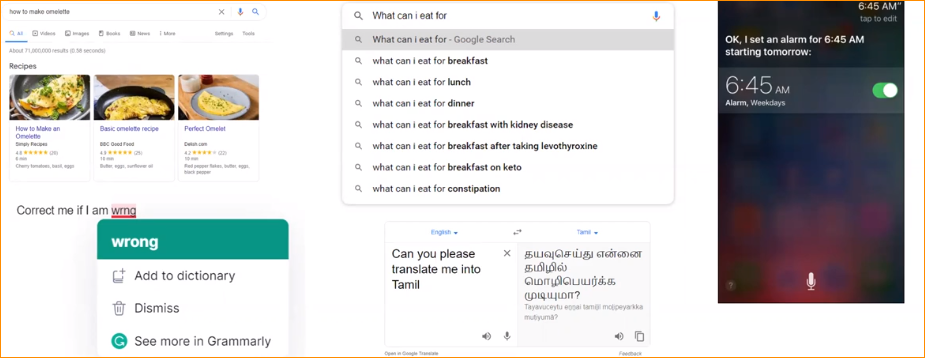
1. Natural language processing

NLP in day Today life –

Ex-Google Search (Recommendation, grammar Collection tools, translation Tool, Chatbot application (Siri, Alexa) these are tools that NLP unit. )

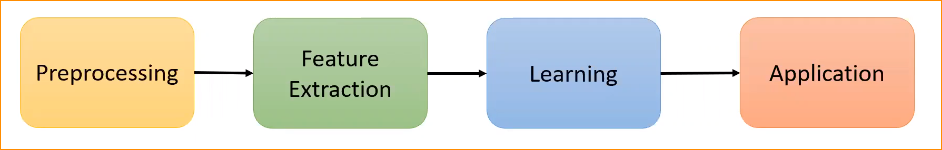


1. What is NLP ?

* Subset of AI
* Aim – Create a model to interact humans like a human being.
* For that we are using natural language to communicate with machines (Writing- text or speaking – form of audio)

How to deal with text –

NLP gets and divides into layers –



Note –

1. Text Processing:

* **Tokenization**: Breaking down text into individual words or units (tokens).
* **Stopword Removal**: Eliminating common words (e.g., "the," "and") that do not contribute significant meaning.
* **Stemming and Lemmatization**: Reducing words to their base or root form to normalize variations.

1. Feature Extraction:

* Bag-of-Words (BoW): Representing a document as an unordered set of words, ignoring grammar and word order.
* TF-IDF (Term Frequency-Inverse Document Frequency): Assigning weights to words based on their frequency in a document relative to their frequency in the entire corpus.
* Word Embeddings (e.g., Word2Vec, GloVe): Representing words as dense vectors in a continuous vector space based on contextual relationships.

1. Learning:

* Supervised Learning: Training models on labeled datasets, where the input (text) is associated with predefined output categories (e.g., sentiment labels).
* Unsupervised Learning: Discovering patterns and structures in data without labeled examples, such as clustering similar documents.
* Deep Learning: Leveraging neural networks, especially transformer models, for tasks like language modeling, sentiment analysis, and machine translation.

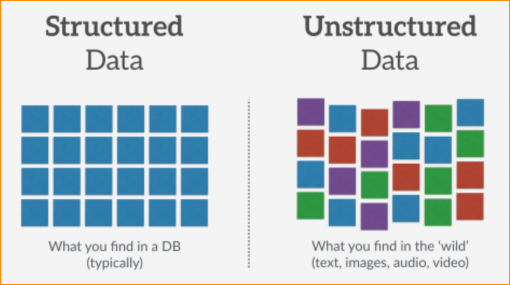
1. Applications:

* Sentiment Analysis: Determining the sentiment expressed in a piece of text (positive, negative, or neutral).
* Named Entity Recognition (NER): Identifying and classifying entities (e.g., names, organizations) in text.
* Text Classification: Categorizing documents into predefined classes or topics.
* Machine Translation: Automatically translating text from one language to another.
* Speech Recognition: Converting spoken language into text.
* Chatbots and Conversational Agents: Building systems that can engage in natural language conversations.
* Information Retrieval: Finding relevant documents or information in response to user queries.

Prepressing is the master place –

* Because text is unstructured

Structure data – simply can put data in to simple table



Why prepressing –

* text are unstructured
* Machine can understand only numbers (0 and 1)
* Natural language is highly ambiguous (different meanings)

Example – **ambiguity at world level .**



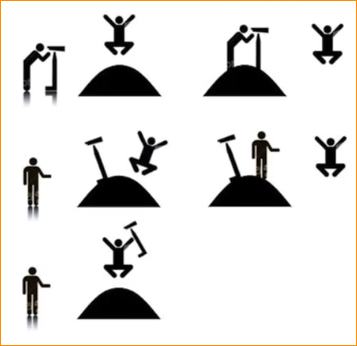
* 1, 2 are nouns and 3rd one is Work.

So we use NLP Context- So base on the context the word can get different meaning.

Example – **ambiguity at sentence level .**

**I show the man on the hill with the telescope**

In this sentence has several meaning-



05

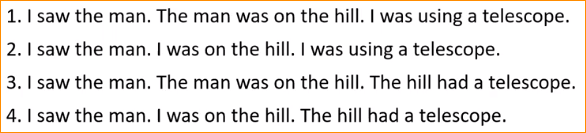
04

03

02

01

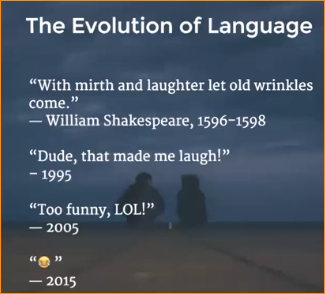
So there are so may possibilities can come –





So we should clean the text meaning. So we should put more effect to preprocessing the data.

* Language evolve with the time.

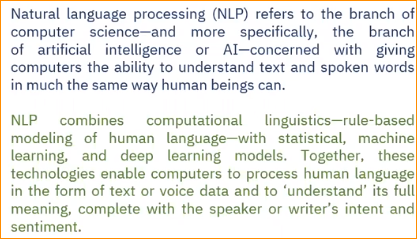


Like this - Using short terms (LOL)

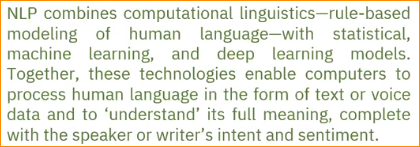
Emojes

Is there Any Structure to Text –

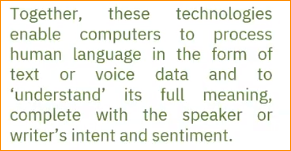
Document



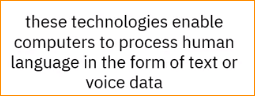
**Composed to Paragraphs**



**Composed to Sentences**



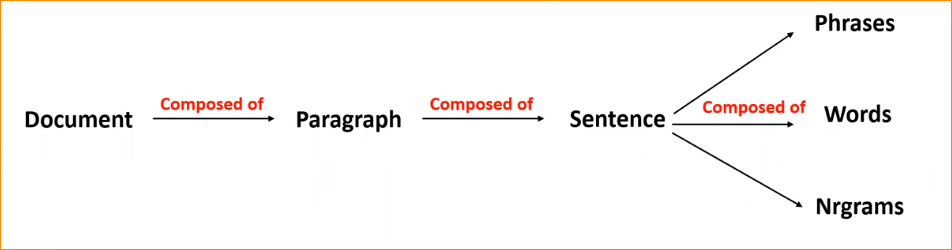
**Composed to phrases**



Phrases – are number of sentences, that convince some meaning full information.

**Composed to words**





**N-grams:**

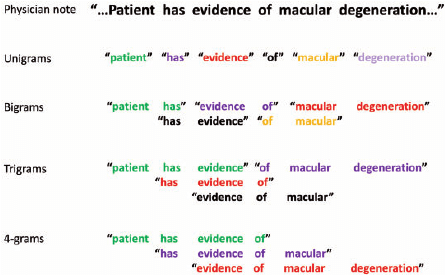
An n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words, or base pairs according to the application. The "n" in n-grams represents the number of items in each sequence.

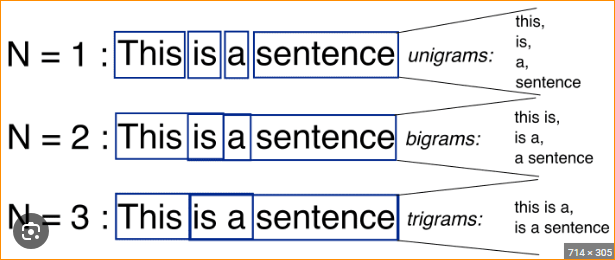
Here are some common types of n-grams:

* Unigrams (1-grams): Single words considered in isolation. For example, in the sentence "I love natural language processing," the unigrams are "I," "love," "natural," "language," and "processing."
* Bigrams (2-grams): Pairs of consecutive words. In the same sentence, examples of bigrams include "I love," "love natural," "natural language," "language processing."
* Trigrams (3-grams): Triplets of consecutive words. Examples include "I love natural," "love natural language," "natural language processing."
* 4-grams, 5-grams, etc.: Sequences of four, five, or more consecutive words.

N-grams are commonly used in natural language processing tasks, such as:

* Language Modeling: Predicting the probability of a word given the previous n-1 words.
* Text Generation: Creating coherent and contextually relevant text based on the patterns observed in n-grams.
* Information Retrieval: Matching and ranking documents based on the occurrence and frequency of n-grams.
* Spell Checking: Identifying and correcting misspelled words based on the context of surrounding words.





Why we need Nrgrams,

Unigram – dealing with one word

Bigram – that mean dealing with more words – that mean more information

1. **Basic Pre-Processing**
2. Tokenization

Process of splitting input sequence into tokens.

This input sequence should be document, paragraph, ngrams, words etc.

Input a Larger component and you break into small Peases that is tokenization. So what tool, library or person call tokenizer.



Using white space Tokenizer – It will break the sentence in to tokens base on the white spaces.



So that separate values we call tokens.

Note – Tokenization

Tokenization is the process of breaking down a text into individual units, called tokens. These tokens are often words, but they can also be subwords or characters, depending on the level of granularity desired. Tokenization is a fundamental step in natural language processing (NLP) and text analysis. The resulting tokens serve as the basic building blocks for various NLP tasks.

Here are some key points about tokenization:

1. Word Tokenization:

* In word tokenization, the text is split into individual words. It assumes that words are separated by spaces or punctuation marks.
* Example: The sentence **"Tokenization is important in NLP"** would be tokenized into the following words: **["Tokenization", "is", "important", "in", "NLP"].**

1. Sentence Tokenization:

* Sentence tokenization involves breaking down a text into individual sentences.
* Example: The paragraph **"Tokenization is important. It helps in various NLP tasks."** would be tokenized into two sentences: **["Tokenization is important.", "It helps in various NLP tasks."]**

1. Subword Tokenization:

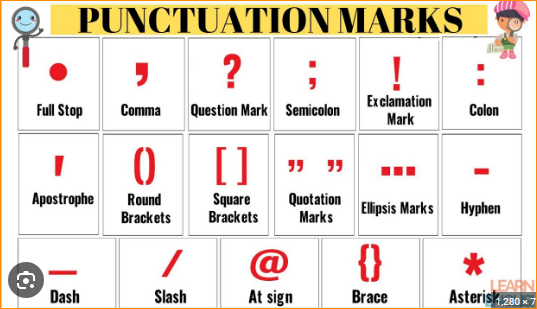
* Subword tokenization breaks words into smaller units, such as subword pieces or characters. This can be useful for handling rare or out-of-vocabulary words.
* Example: The word "**tokenization**" might be broken into subword pieces like **["to", "ken", "iza", "tion"].**
* Tokenization Challenges:
* Ambiguities: Some words can have multiple meanings, and tokenization might be challenging in such cases.
* Contractions: Words like "can't" or "don't" may be split into two tokens ("can" and "t", "do" and "n't").
* Special Cases: Tokenization needs to handle special characters, punctuation, and other linguistic nuances.
* Importance in NLP:
* Tokenization is a crucial preprocessing step for many NLP tasks, including text classification, sentiment analysis, machine translation, and information retrieval.
* It helps in analyzing the structure of sentences and understanding the meaning of individual words.
* Tokenization Libraries:

Various programming languages and NLP libraries provide built-in tokenization functions. For example, in Python, the **Natural Language Toolkit (NLTK), spaCy, and the tokenization functions in the nltk.tokenize** module are commonly used.

Note – end

Token base Punctuation





Note –

In natural language processing (NLP), tokenization is the process of breaking down a text into smaller units called tokens. Tokens are the building blocks used for further analysis in NLP tasks. Punctuation handling during tokenization is an important aspect of text processing, and different tokenization approaches may handle punctuation in various ways. Here are a few common approaches:

* Retaining Punctuation as Tokens:

In some cases, punctuation marks are treated as separate tokens. For example, the sentence **"Hello, world!"** might be tokenized **into ["Hello", ",", "world", "!"].**

* Removing Punctuation:

Another approach is to exclude punctuation marks from the list of tokens. In this case, the example sentence "**Hello, world!**" might be tokenized into **["Hello", "world"].**

* Incorporating Punctuation into Tokens:

Some tokenization methods include punctuation within word tokens. For instance, "Hello, world!" might be tokenized into ["Hello,", "world!"].

1. Token Normalization

We will be bringing the token as it **base** form.

Two forms -

1. **Stemming** – Rule base process of removal of the inflectional forms from the tokens

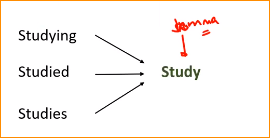
Output the **stem** of the word, in here **removed postfix** (ing)

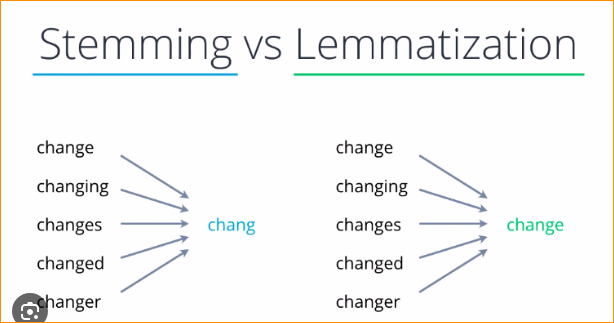
Studying **Study** -------------------- this is **stem**

**Stemming is faster than Lemmatization .. but in here It give result that may not in dictionary.**

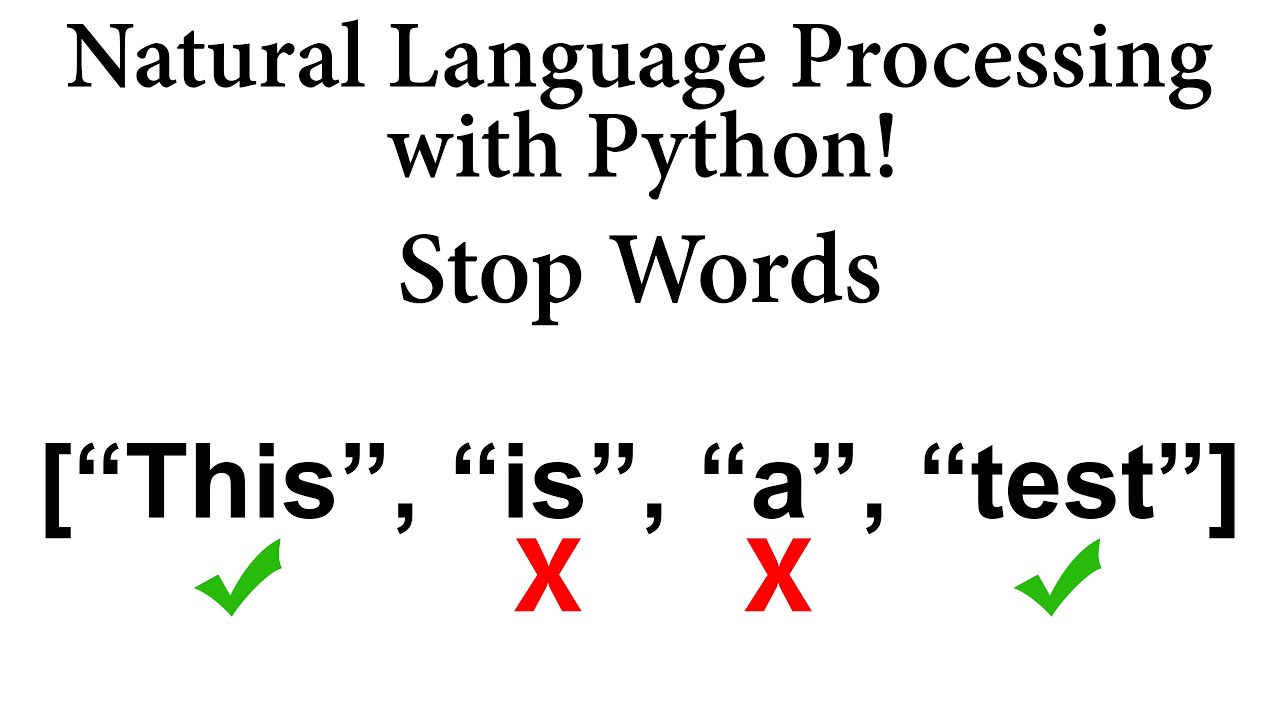
1. **Lemmatization**

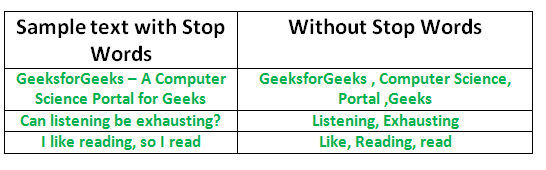
Systematic process of reducing a token to its **lemma**. Here Always searching what could be the base / root form of the word. (here – lemma is study). In here lemmatization finning the root words so it takes time. So have more words mean more time.





1. Stop words removal





Removing words which do not add much meaning

Pronouns, articles, prepositions and conjunctions are commonly considered as stop words.

Ex- the, is, a , what ,why, he, she , at



Can create our own stop word collection.

* Some time those words are very important our test processing.

Example – Question and Answering (We are giving a question abd paragraph – it should be able to find tha answer )

Simply – The Question that start where??????

Model search “at” pleases in the paragraph – so in here if you miss the Pronouns, articles, prepositions and conjunctions that may be a problem.

* **Domain specific stop word**

Note –

Domain-specific stop words refer to words that are commonly used in a specific field or domain but may not provide significant semantic value when analyzing text within that domain. In the context of natural language processing (NLP) and text analysis, stop words are typically words that are filtered out before or during the processing of text data.

For example, in general text processing, common stop words might include words like "and," "the," "is," "in," etc. However, in domain-specific contexts, different stop words may be more relevant.

Here are some examples of domain-specific stop words for different fields:

Medical Domain:

* disease
* patient
* treatment
* study
* results

Legal Domain:

* contract
* agreement
* party
* clause
* hereby

Finance and Banking:

* account
* transaction
* deposit
* balance
* credit

Academic Research:

* methodology
* hypothesis
* literature
* findings
* analysis

When analyzing text within a specific domain, it can be beneficial to identify and remove these domain-specific stop words to focus on the more meaningful terms that contribute to the understanding and interpretation of the content. This process can help improve the accuracy and relevance of text analysis tasks such as topic modeling, sentiment analysis, or information retrieval within that particular domain.

Note end-

If you dealing with large collection of move reviews , So that movie name appearing again and again so we know everting about the movie, so model should recognize movie word using again and again. It useless. So we call the Domain specific stop words.

With your Domain can customize the stop words.

1. Other Preprocessing steps

* Remove punctuation
* Removeing the numbers
* Making it to lower case letter

