NLP

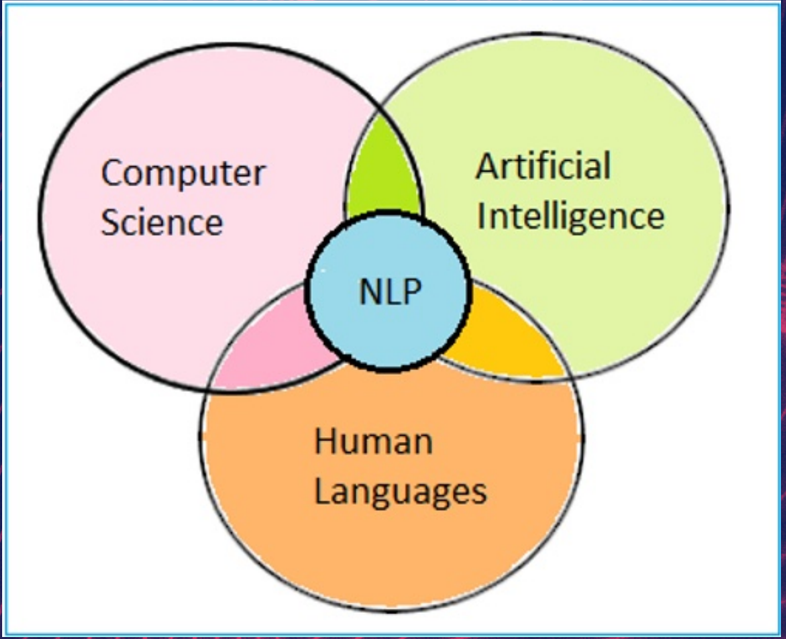
NATURAL LANGUAGE PROCESSING

Contents

- Intro to NLP
- Applications of NLP
- Text Preprocessing
- Word Embeddings
  - Bag of Words
  - TF-IDF
  - Word2Vec

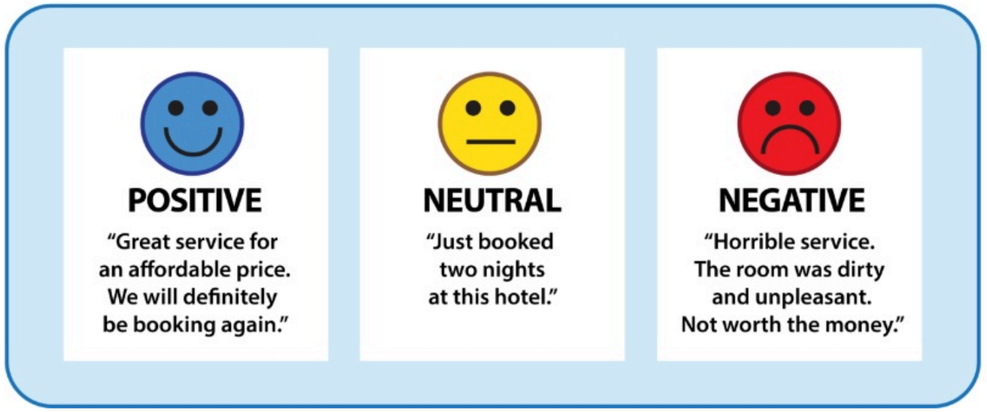
INTRO TO NLP

Computers and machines are great at working with tabular data or spreadsheets. However, as human beings generally communicate in words and sentences. Much information that humans speak or write is **unstructured**. So it is not very clear for computers to interpret such. In natural language processing (NLP), the goal is to make computers **understand the unstructured text and retrieve meaningful pieces of information from it**.



APPLICATIONS OF NLP

Sentiment Analysis



Sentiment analysis is the process of **classifying the emotional intent of text**. Generally, the input to a sentiment classification model is a piece of text, and the **output is the probability that the sentiment expressed is positive, negative, or neutral**.

Grammatical Error Correction

grammarly

Grammatical error correction models **encode grammatical rules** to correct the grammar within text. This is viewed mainly as a **sequence-to-sequence task**, where a model is trained on an ungrammatical sentence as input and a correct sentence as output. Online grammar checkers like Grammarly and word-processing systems like Microsoft Word use such systems to provide a better writing experience to their customers.

APPLICATIONS OF NLP

Spam detection

Spam detectors take as input an email text along with various other subtexts like title and sender's name. They aim to **output the probability that the mail is spam**. Email providers like Gmail use such models to provide a better user experience by detecting unsolicited and unwanted emails and moving them to a designated spam folder.

Chatbots

- Summarization
- Text generation
- Database query
- Conversation generation

Search engines

NLP-enabled search engines are designed to understand a searcher's natural language query and the context around it.

TEXT PREPROCESSING

It breaks down text into a form that machine learning algorithms can digest

NOISE REMOVAL

Noise removal is about removing digits, characters, and pieces of text that interfere with the process of text analysis.

The various ways to remove noise include:

- punctuation removal
- special character removal
- numbers removal
- HTML formatting removal

LOWERCASING

It consists of converting all text to lowercase.

But why should we do this?

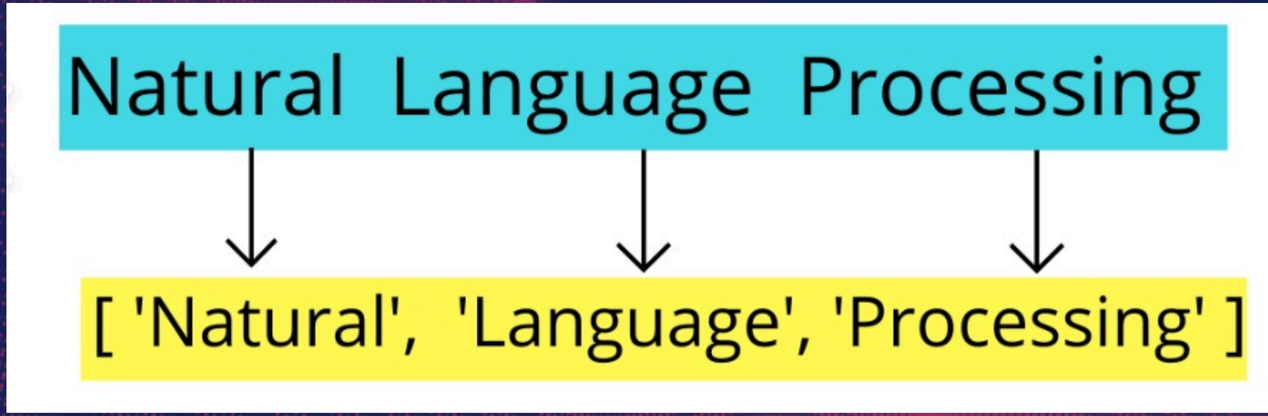
For example, a text is having mixed-case occurrences of the token ‘Canada’, i.e., at some places token ‘canada’ is and in other ‘Canada’ is used. To eliminate this variation, so that it does not cause further problems, we use the lowercasing technique to eliminate the sparsity issue and **reduce the vocabulary size**.

TOKENIZATION

It is the process of breaking a sentence or text into individual words or subwords, known as tokens.

Each token (words, phrases or symbols) represents a meaningful unit and it plays a crucial role in understanding the structure and meaning of the text.

The most widely used tokenization process is **white space tokenization**. In this process, the entire text is split into tokens by splitting them based on whitespace between two words.



REMOVAL OF STOP WORDS

Stop words are words that you want to ignore, so you filter them out of your text when you're processing it. Very common words like '**in**', '**is**', and '**an**' are often used as stop words since they don't add a lot of meaning to a text in and of themselves.

If we search 'what is text preprocessing', we want to focus more on 'text preprocessing' rather than 'what is'.

Warning!
Stop words can mean different things for different applications. In some applications, removing all stop words from determiners to preposition is appropriate. But in some applications, like sentimental analysis, removal of tokens like not, good, etc. can throw algorithms off their tracks.

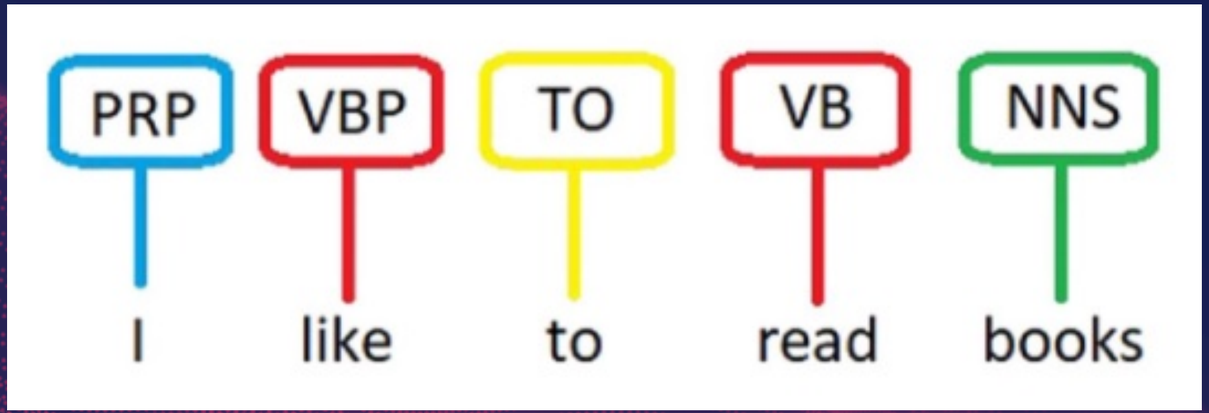
PARTS OF SPEECH TAGGING

Part of speech is a grammatical term that deals with the roles words play when you use them together in sentences. Tagging parts of speech, or POS tagging, is the task of labeling the words in your text according to their part of speech.

In NLTK:

Tags that start with	Deal with
JJ	Adjectives
NN	Nouns
RB	Adverbs
PRP	Pronouns
VB	Verbs

Example:



NORMALIZATION : STEMMING AND LEMMATIZATION

Normalization is the process of converting the token into its basic form.

t helps in reducing the number of unique tokens and **redundancy** in the data. It **reduces the data dimensionality** and removes the variation of a word from the text.

For example, change, changed, changing, changes, all these words have similar meaning but instead of having each of these words in our vacabulary we can find the ‘root’ of these words, i.e., ‘change’, and then add it to the vocabulary.

Two such techniques are STEMMING and LEMMATIZATION.

Stemming involves removing suffixes from words to obtain their base form while lemmatization involves converting words to their morphological base form.

Stemming is a **simpler** and **faster** technique compared to lemmatization. However, stemming can sometimes produce a base form that is not valid, in which case it can also lead to ambiguity.

Lemmatization is a **more complex** and **slower** technique that uses vocabulary and morphological analysis to determine the base form of a word.

Application:

Stemming is preferred when the meaning of the word is not important for analysis. For example: Spam Detection

Lemmatization would be recommended when the meaning of the word is important for analysis. for example: Question Answer

QUIZ TIME!!

WORD EMBEDDINGS

Word embeddings are numeric representation of words.

They are a method of extracting features out of text so that we can input those features into a machine learning model to work with text data.

It allows words with similar meanings to have a similar representation.

BAG OF WORDS

We can represent a sentence as a bag of words vector (a string of numbers).

This method involves converting text into a vector based on the frequency of words in the text, without considering the order or context of the words.

Let’s take an example:

Review 1: This movie is very scary and long
Review 2: This movie is not scary and is slow
Review 3: This movie is spooky and good

We will first build a vocabulary from all the unique words in the above three reviews. The vocabulary consists of these 11 words: ‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’.

We can now take each of these words and mark their occurrence in the three movie reviews above with 1s and 0s. This will give us 3 vectors for 3 reviews:

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review (in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	1	1	0	1	1	0	0	9
Review 3	1	1	1	0	0	1	0	0	0	1	1	6

Corresponding BoW matrix:
[[1 1 1 1 1 1 1 0 0 0 0],
[1 1 2 0 1 1 0 1 1 0 0],
[1 1 1 0 0 1 0 0 0 1 1]]

Limitations:

- **No Word Order** : It doesn’t care about the order of words, missing out on how words work together.
- **Ignores Context** : It doesn’t understand the meaning of words based on the words around them.
- **Computationally inefficient** : If the data includes a lot of sentences then the vocabulary becomes vast and the vectors will include a lot of zeroes and makes computation complex.

TF-IDF

(TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY)

Term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

Let’s understand each of these terms independently:

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

$$IDF(t) = log_e\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}\right)$$

$$TF - IDF score = TF * IDF$$

TF term is the measure of **how frequently** the word appears in a document. A higher TF means the term is more significant in the context of that specific document.

IDF measures the importance of a term or **how rare** a term is across a collection of documents. It reflects how rare or common a term is across the corpus.

SIgnificance of TF-IDF:

- TF-IDF increases the weight of terms that are **frequent in a document but rare across the corpus**, making them stand out as important features.
- The IDF component ensures that terms important within a specific context are given more weight. This helps in capturing the **contextual significance** of terms.

Let’s understand with an example

Suppose we have a small corpus of three documents:

1. Document 1: "Artificial intelligence is transforming technology."
2. Document 2: "Machine learning and artificial intelligence are subsets of technology."
3. Document 3: "The technology in sports is improving rapidly."

Now, let's focus on two terms: "**artificial**" and "**technology**."

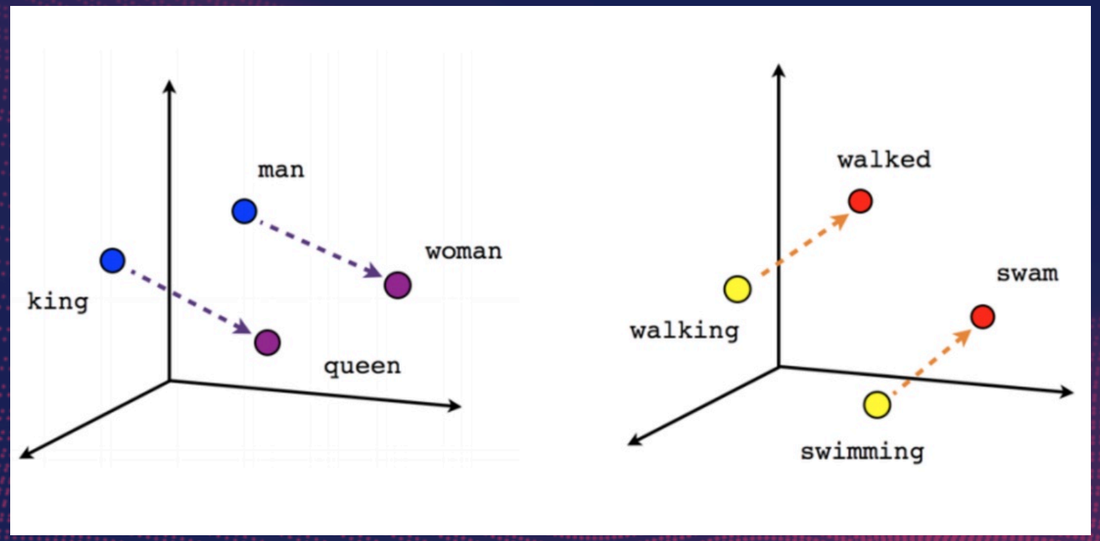
Term	Doc 1 TF	Doc 2 TF	Doc 3 TF	IDF	Doc 1 TF-IDF	Doc 2 TF-IDF	Doc 3 TF-IDF
artificial	0.167	0.111	0	0.176	0.029	0.020	0
technology	0.167	0.111	0.167	0	0	0	0

- "artificial" receives a positive TF-IDF score in Document 1 and Document 2, highlighting its **relative importance** in those documents.
- "technology" receives a TF-IDF score of 0 in all documents, indicating its **lack of discriminative power** since it is common across the entire corpus.

WORD2VEC

Word2Vec allows words to be represented as vectors in a continuous vector space to capture the relationships between words.

- **Semantic representation** : Words are represented in the vector space so that similar words are near to one another. This enables the model to interpret words according to their context within a particular corpus.
- **Vector arithmetic** : Vector arithmetic, for instance, can be used to record word relationships. One well-known example is that the vector representation of “queen” could resemble the vector representation of “king” less “man” plus “woman.”



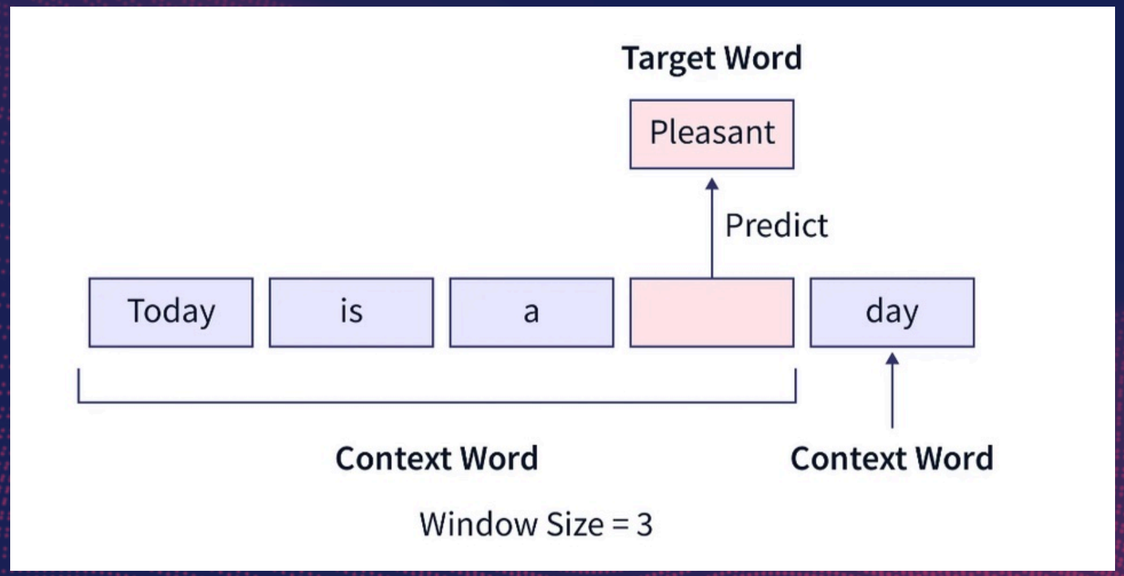
It utilizes two architectures to learn the embeddings:

1. CBOW (Continuous Bag of Words)
2. Skip Gram

Given a set of sentences the model loops on the words of each sentence and either tries to use the current word w in order to predict its neighbors (i.e., its context), this approach is called “Skip-Gram”, or it uses each of these contexts to predict the current word w, in that case the method is called “Continuous Bag Of Words” (CBOW)

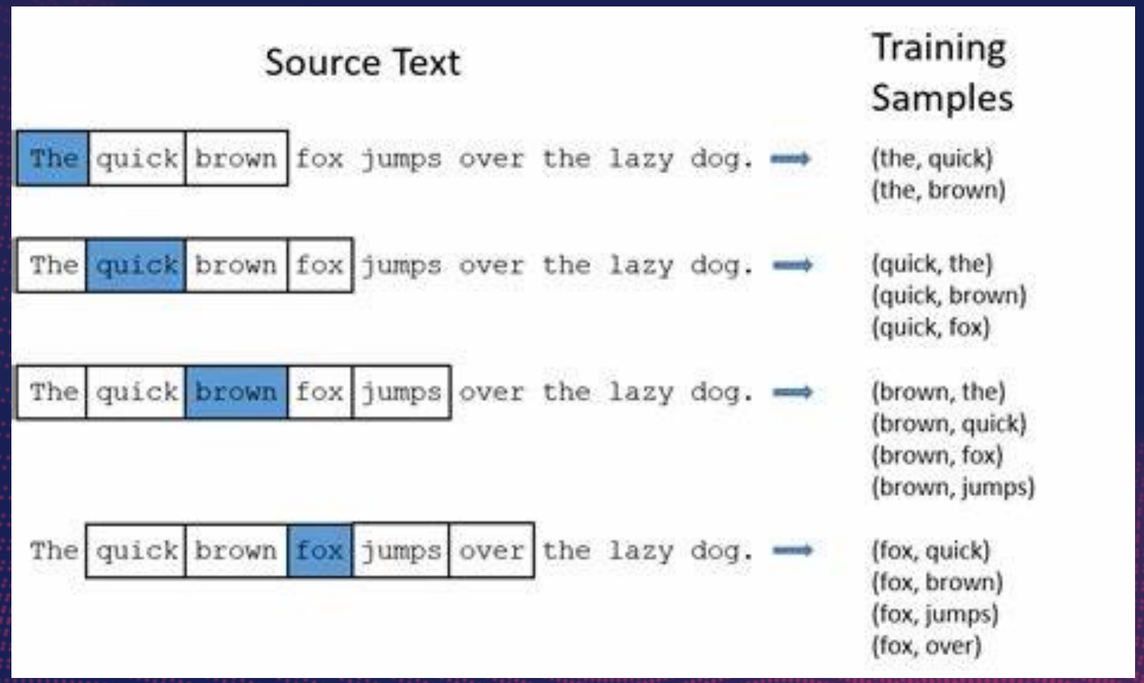
Continuous Bag of Words (CBOW)

- Objective: Predict the target word (center word) given its context words (surrounding words).
- Process:
  a. For a given context window (e.g., 2 words on either side of the target word), the CBOW model takes the context words as input.
  b. It averages the embeddings of these context words.
  c. The model then predicts the target word based on this average context vector.



Skip-gram

- Objective: Predict the context words given a target word.
- Process:
  a. For a given target word, the Skip-gram model tries to predict the context words within a certain window size.
  b. This model is typically more effective for small datasets and captures semantic relationships better than CBOW.



QUIZ TIME!!

CODE IMPLEMENTATION