NATURAL LANGUAGE PROCESSING

* Subset of AI
* Enables machines to comprehend and analyse human language
* NLP pipeline refers to sequence of processes involved in analysing and understanding human language
* Here are the steps (usually) used in NLP
* Data acquisition- for building ml model, we need data related to our problem statement.
* Public dataset- publicly available data
* Web scrapping- technique to scrap data from websites, we can use Beautiful Soup to scrape text data from web pages
* Image to text- scrap data from images, thru Optical Character Recognition (OCR), library name Tesseract uses OCR
* Pdf to text- pdf data can be extracted via libraries like PyPDF2
* Data augmentation- if acquired data is not sufficient, we can generated fake data from existing data by synonym replacement, back translation, adding some noise.
* Text cleaning- sometimes the acquired data is not clean, might contain htlm tags, spelling errors, or special characters.
* Unicode normalisation- if text data contains symbols, emojis, graphic character or special character. We can either remove this text, or convert to machine readable text.
* Regex or regular expression- regular expression is the tool that is used for searching the string of specific patterns. Ex. Phone number, email address, url, and after that we can either keep or remove that particular pattern of text
* Spelling corrections- to overcome this problem, we create a corpus or dictionary of most common mistype words and replace these common mistakes with the correct word
* Text preprocessing- the clean texts may contain group of sentences and each sentence made of group of words. So we need to process it
* Tokenization- segmenting text into a list of tokens.

sentence tokenization- segmenting into different sentences, word tokenization- segmenting into different words.

* Lowercasing- all the text is converted into lowercase letters. This step is useful in various NLP tasks such as text classification, information retrieval and sentiment analysis.
* Stop word removal- commonly occurring words, like “the”, “a”, “and”, “for” etc. cause they do not carry much meaning.
* Stemming or lemmatization- these are used to reduce words to their base form, which help reduce vocabulary and simplify text. Stemming involves stripping the suffix from word to get their stem. Lemmatization involves reducing words to their base form based on their part of speech. ==
* Removing digit/punctuation- removes digits and punctuations from the text
* POS tagging- assigning a part of speech tag to each word in text. ==
* Named entity recognition (NER)- identifying and classifying named entities in text, such as people, organizations and locations.

import nltk

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import SnowballStemmer, WordNetLemmatizer

from nltk.tag import pos\_tag

from nltk.chunk import ne\_chunk

import string

# sample text to be preprocessed

text = 'GeeksforGeeks is a very famous edutech company in the IT industry.'

# tokenize the text

tokens = word\_tokenize(text)

# remove stop words

stop\_words = set(stopwords.words('english'))

filtered\_tokens = [token for token in tokens if token.lower() not in stop\_words]

# perform stemming and lemmatization

stemmer = SnowballStemmer('english')

lemmatizer = WordNetLemmatizer()

stemmed\_tokens = [stemmer.stem(token) for token in filtered\_tokens]

lemmatized\_tokens = [lemmatizer.lemmatize(token) for token in filtered\_tokens]

# remove digits and punctuation

cleaned\_tokens = [token for token in lemmatized\_tokens

if not token.isdigit() and not token in string.punctuation]

# convert all tokens to lowercase

lowercase\_tokens = [token.lower() for token in cleaned\_tokens]

# perform part-of-speech (POS) tagging

pos\_tags = pos\_tag(lowercase\_tokens)

# perform named entity recognition (NER)

named\_entities = ne\_chunk(pos\_tags)

# print the preprocessed text

print("Original text:", text)

print("Preprocessed tokens:", lowercase\_tokens)

print("POS tags:", pos\_tags)

print("Named entities:", named\_entities)

* Feature engineering- represents text in numeric vectors so that ML algorithms can understand

Two common approach

* Classical or traditional approach- we create a vocabulary of unique words assign a unique id (integer value) for each word. Each word is treated as feature. So when vocab is large, features become very large, tough for ML models.

One Hot Encoding- each token is represented as binary vector. First mapped each token to integer, then integer is represented as binary, all values 0, index of integer has value 1