**NLP 5**

It is the part 5 article of NLP series, if you are new kindly refer to the previous articles here.

In this article, we will learn the following topics.

* Part of speech Tagging
  + Nouns
  + Noun Phrases
  + Dependency Parsing
* Topic Modelling
  + Latent Dirchlet Allocation Model
* Amazon reviews Case Study

Libraries used: spacy, genism

**NLP 1**

****

### **Introduction to the Series:**

Welcome to the exciting series on Natural Language Processing (NLP)! In this journey, we will unravel the secrets behind the techniques and models that power language-based applications. Here’s what you can look forward to:

1. **Text Pre-processing**
2. **Text Representation**
3. **Word Embedding**
4. **Language Modeling**
5. **Topic Modeling**
6. **Recurrent Neural Network (RNNs) for NLP**
7. **LSTM for NLP**
8. **Transformer Models**
9. **BERT**

#### **Prerequisite:**

* Basic knowledge of Python.

### **In This Article:**

Today, we’ll dive into the fascinating world of real-time NLP problem-solving restricted to the following topics:

* **What is Natural Language Processing (NLP)?**
* **The Need for NLP**
* **Text Pre-processing**
* **Case Study: Tweet Sentiment Classification**

### **What is Natural Language Processing (NLP)?**

Imagine a world where computers can understand and respond to human language just like we do. Natural Language Processing (NLP) makes this possible. It’s a branch of computer science and artificial intelligence that deals with how computers can be programmed to process and analyze vast amounts of natural language data.

Every smart application you interact with—like voice assistants, chatbots, or translation tools—relies on the power of NLP to communicate effectively and understand human language.

### **The Need for NLP:**

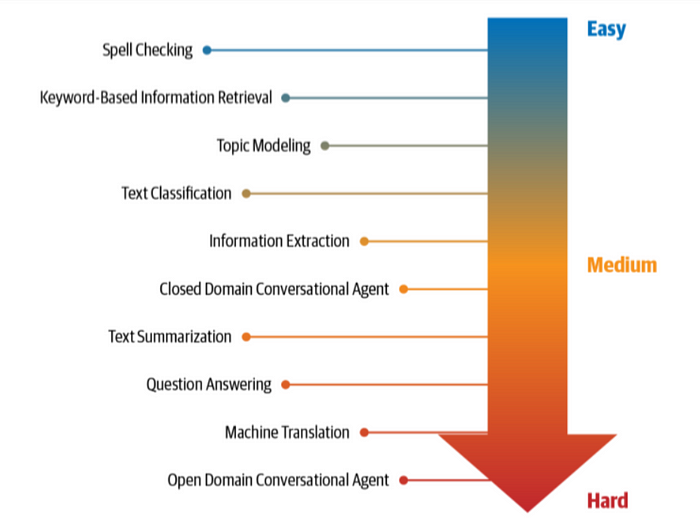
Human language is incredibly complex, filled with nuances, idioms, and varied expressions. NLP is crucial because it enables machines to:

* **Understand:** Extract meaningful insights from text or speech.
* **Generate:** Create coherent and contextually relevant text or speech.
* **Translate:** Seamlessly convert text or speech from one language to another.
* **Analyze Sentiment:** Determine the emotional tone behind a piece of text.

By harnessing NLP, we can bridge the gap between human communication and computer understanding, making technology more intuitive and user-friendly.

### **Applications of NLP:**

|  |  |  |
| --- | --- | --- |
| Category | Task | Application |
| Language Modeling | Predicts the next word in a sentence based on the previous words | Spelling Correction, Machine Translation |
| Text Classification | Categorize text into predefined categories based on its content | Spam identification, Sentiment Analysis |
| Information Extraction | Extracting relevant information from text | Gmail extracting travel details from tickets |
| **Information Retrieval** | Finding documents relevant to a user query from a large collection | Semantic search engines |
| **Question Answering** | Systems that can converse in human languages | Chatbots |
| **Text Summarization** | Creating concise summaries of longer documents while retaining key information | Automated abstracts |
| Topic Modeling | Uncovering the topical structure of a large collection of documents |  |



#### **Language Modeling:**

Language modeling involves predicting the next word in a sentence based on the previous words. This is essential for various applications such as:

* **Spelling Correction**
* **Speech Recognition**
* **Machine Translation**

#### **Text Classification:**

This involves categorizing text into predefined categories based on its content. Common applications include:

* **Email Spam Identification**
* **Sentiment Analysis**

#### **Information Extraction:**

Extracting relevant information from text, such as calendar events from emails or names from social media posts.

* **Example:** Gmail extracting travel details from tickets.

#### **Information Retrieval:**

Finding documents relevant to a user query from a large collection.

* **Example:** Semantic search engines.

#### **Question Answering:**

Building systems that can converse in human languages.

* **Examples:** Chatbots, Voice Assistants

#### **Text Summarization:**

Creating concise summaries of longer documents while retaining key information.

* **Examples:** Inshorts, Automated abstracts

#### **Topic Modeling:**

Uncovering the topical structure of a large collection of documents, useful in many fields from literature to bioinformatics.

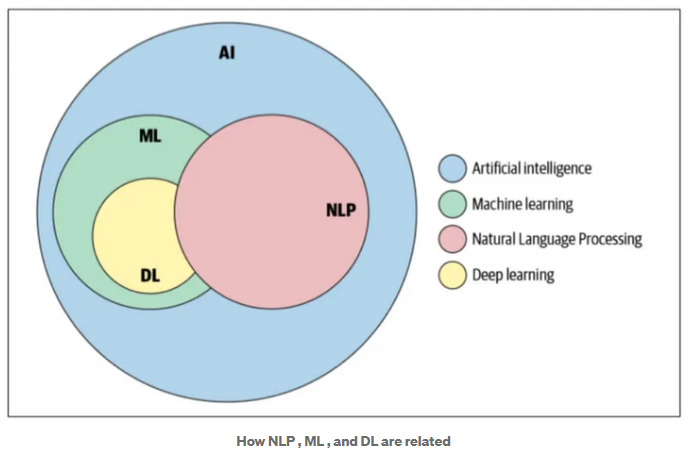
### **Why NLP is challenging?**

NLP is challenging due to the complexity and ambiguity of human language:

* **Ambiguity:** Words can have multiple meanings depending on context (e.g., "bank" can mean a financial institution or the edge of a river).
* **Complexity:** Language includes idioms, sarcasm, and metaphors that are difficult for machines to interpret.
* **Data Availability:** High-quality labelled data is expensive and time-consuming to obtain.
* **Ethical Considerations:** Issues such as bias, privacy, and accountability must be addressed.

Humans use intuition to understand language, but NLP models rely on pattern recognition in data, lacking human intuition.

NLP incorporates AI and its subsets ML and DL to solve a problem



### **Approaches to NLP:**

NLP problems can be addressed using various approaches:

#### **Heuristics-Based:**

Using predefined rules to analyze text based on patterns, syntax, and grammar.

* **Example:** Regular expressions (regex) for text analysis.

#### **Machine Learning-Based:**

Applying supervised and unsupervised machine learning techniques to textual data.

* **Examples:** Classification algorithms for categorizing news articles, clustering algorithms for grouping text documents.

#### **Deep Learning-Based:**

Training neural networks with multiple layers to process language, capturing complex relationships between words and their meanings.

* **Examples:** Neural networks for text generation or classification.

### **Text Pre-processing:**

#Text preprocessing is the first step in the NLP pipeline that involves transforming raw text into a clean and structured format suitable for analysis.

### **Tokenization:**

Tokenization involves breaking down text into tokens (words or phrases). It presents several challenges, especially with punctuation marks that serve multiple functions within a sentence.

* **Example Sentence:** "Clairson International Corp. said it expects to report a net loss for its second quarter ended March 26 and doesn’t expect to meet analysts’ profit estimates of $3.9 to $4 million, or 76 cents a share to 79 cents a share, for its year ending Sept. 24."

This sentence has several items of interest that are common for alphabetic and space-delimited languages.

* First, it uses periods (.) in three different ways :
  + Within numbers as a decimal point ($3.9)
  + To mark abbreviations (Corp. and Sept.)
  + To mark the end of the sentence, in which case the period following the number 24 is not a decimal point.
* The sentence uses apostrophes in two ways:
  + To mark the genitive case (analysts’ )
  + To show contractions (doesn’t)

The tokenizer must handle:

* **Periods:** Used as decimal points, in abbreviations, and to mark the end of sentences.
* **Apostrophes:** Used for contractions and genitive case.

### **Text cleaning:**

#The depth of this step varies for each problem statement.

#### **Stop Words:**

Common words that frequently appear in a language but carry little meaningful information, such as "the", "and", "in".

* **Importance:** Removing stop words helps reduce the dimensionality of text data.

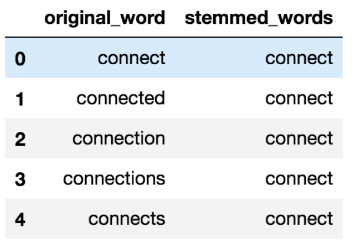
#### **Contractions:**

Shortened forms of words or phrases created by omitting certain letters and sounds, often replaced with an apostrophe.

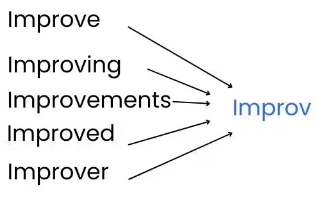
* **Examples:** "I'm" for "I am," "don't" for "do not."

#### **Stemming:**

Reducing words to their root form to treat different morphological variants as a single entity.



Stemming can sometimes be too aggressive, reducing words to stems that are not meaningful.



#### **Lemmatization:**

Reducing words to their base or dictionary form, known as a lemma. It considers the context and meaning of the word, ensuring the base form is valid.

#### 

#### **Context-Specific Preprocessing:**

Sometimes, text preprocessing requires domain expertise to remove certain strings or characters that are irrelevant to the NLP model.

* **Examples:** Handling hyperlinks, alphanumeric words, domain specific stop words

**Feature Extraction:**

Feature extraction in NLP involves converting textual data into numerical vectors for machine processing. Simple methods, like the frequency dictionary, calculate word frequencies for each class, resulting in a 3-dimensional vector:

[Bias, sum of positive frequency, sum of negative frequency].

Effective text representation techniques that capture both semantic and syntactic meanings will be discussed in the upcoming articles.

**Python Packages for NLP:**

Python provides built-in modules for text data collection, handling, and processing, making tasks like tokenization, stemming, and part-of-speech tagging straightforward. In our case study, we primarily use NLTK and spaCy, as these widely used packages offer robust tools to execute the NLP tasks.

**With this introduction, let’s delve into a case study on sentiment analysis of tweets.**

**Case Study: Sentiment Analysis of Tweets**

The tweets have been pulled from Twitter during Covid-19 breakout and the tweets are manually labelled. The government wants to learn the public sentiment on the new strain of COVID-19.

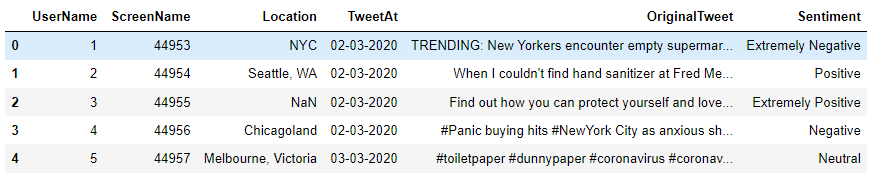
The task is to classify the sentiment of a tweet to either Positive or Negative.

**Understanding the Dataset:**

1. Dataset source: kaggle covid tweets [link](https://www.kaggle.com/datatattle/covid-19-nlp-text-classification)
2. Total 11,663 tweets with positive and negative labels

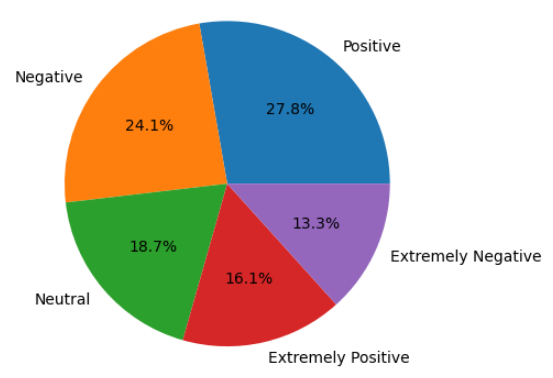
**Data Exploration:**





**From the dataset, our feature vector is “Original Tweet” and the Target is “Sentiment” and other columns can be dropped.**

**Distribution of Labels:**



Tweets are categorized into five types, but we'll simplify the problem by focusing on binary classification with "Positive" and "Negative" labels. To prevent model bias due to class imbalance, we'll balance the dataset by extracting an equal number of positive and negative reviews.

**Text Pre-Processing:**

The steps in text pre-processing are context-dependent and in our case, we will follow these steps in order:

* **Removing hyperlinks and hash tags from the tweets**
* **Apply word tokenization, breaking down the text into individual words or tokens**
* **Clean the tokens by removing punctuation and stop words**
* **Apply stemming to reduce words to their base or root forms**





**Original tweet:**

**Consumers have increased their online shopping due to coronavirus.** [**https://t.co/5mYfz3RAD0**](https://t.co/5mYfz3RAD0) **#retail #ecommerce #study #coronavirus** [**https://t.co/Dz3H6zrWUT**](https://t.co/Dz3H6zrWUT)

**Pre-processed tweet:**

**['consum', 'increas', 'onlin', 'shop', 'due', 'coronaviru', 'retail', 'ecommerc', 'studi', 'coronaviru']**

**Apply the pre-processing to all the tweets and store the processed words in the dataframe.**



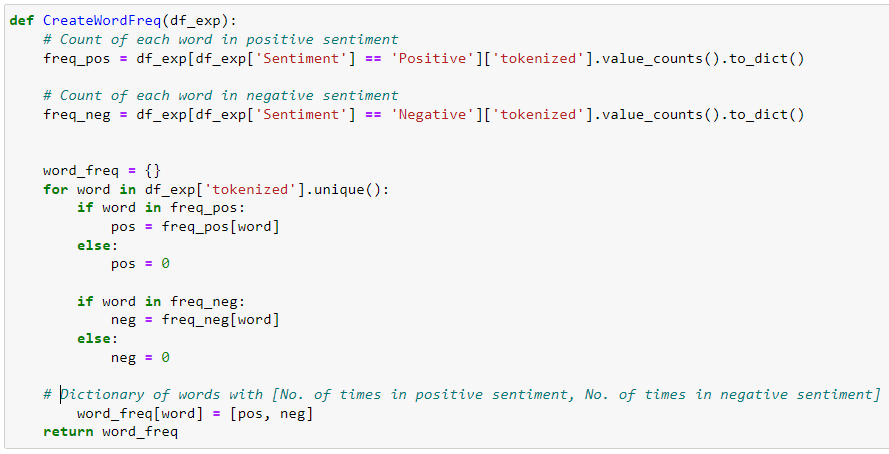
**Feature Extraction:**

**Since we consider it as binary classification problem, we formulate the frequency dictionary of 3 dimensional vector.**

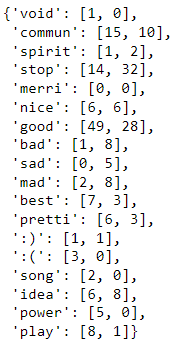
**Convert the tokenized tweets into separate rows, each accompanied by the sentiment label.**

In the transformed data frame, calculate the frequency of each word based on sentiment. Build a word frequency dictionary in the form:

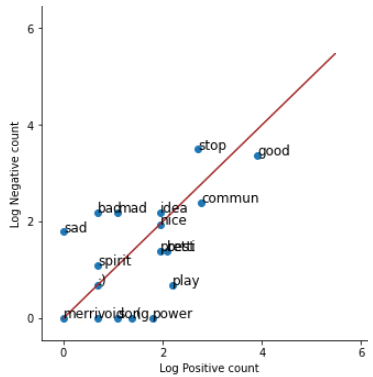
**{Word: [No. of occurrences in positive sentiment, No. of occurrences in negative sentiment]}**



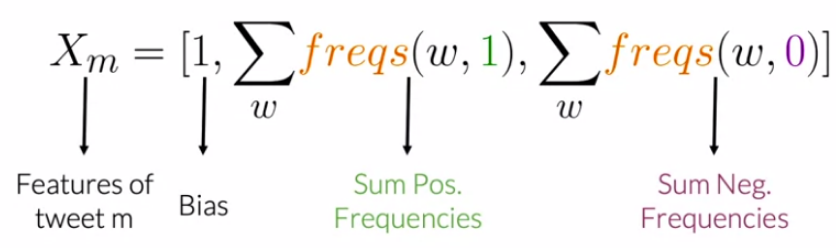
**outputs,**



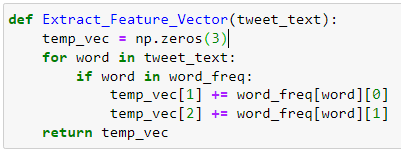
**The word frequency dictionary built for the tweets reveals that the words "bad," "sad," "mad," and "stop" are predominantly associated with negative sentiment tweets. Conversely, the words "good," "best," and "pretty" are predominantly used in positive sentiment tweets.**

****

**Transform the tweet text into 3 features as follows,**



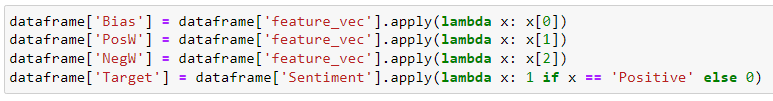
**Based on the word frequency dictionary calculate bias, sum of positive frequencies and sum of negative frequencies for each word in the tweet.**



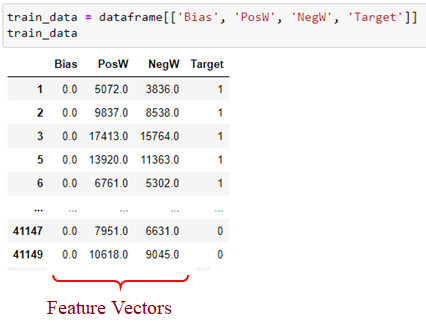
**Apply the transformation on the complete data.**



**Split the list into separate columns and convert the target feature to binary representation.**

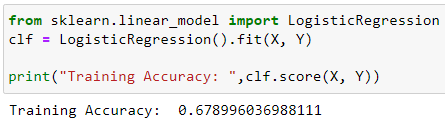


**Extract the feature vectors and target vector for training the model.**



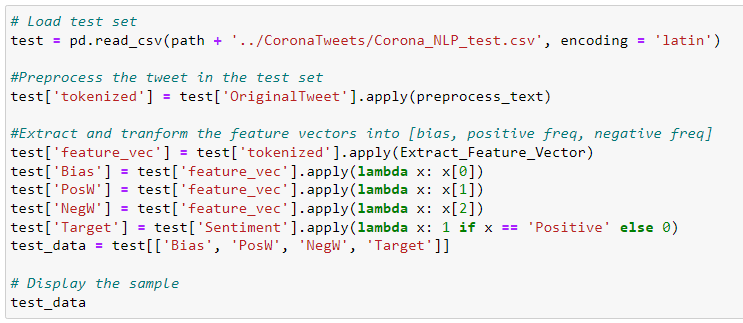
**Apply logistic regression on the transformed vector,**

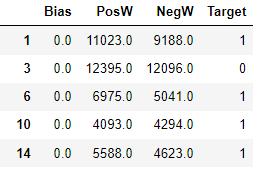


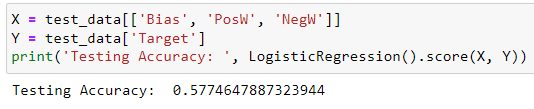


The training accuracy achieved is 67.8%. While this may not be highly convincing, it is a commendable result for a simple model.

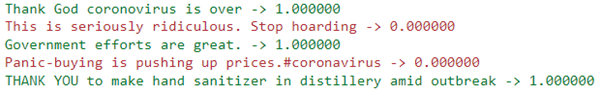
Loading the test data set to check the model performance on the test data.





After running the model on the test set, we achieved a testing accuracy of 57.7%. This result indicates that the model is able to classify sentiment with a moderate level of accuracy, demonstrating its effectiveness with the given approach.

A sample of correctly predicted results from the model are as follows



Case Study Summary:

* Text Pre-processing
  + *Remove Stop words, Hyperlinks, punctuations*
  + *Word Tokenization*
* Feature Extraction
  + *Build Word Frequency dictionary*
  + *Transform tweet to [Bias, Sum of positive frequencies, Sum of negative frequencies]*
* Model
  + *Build Logistic Regression Model*
  + *Performance on the training and Test set*

In the next article, we will delve into various text representation techniques:

* **Sparse Representation and Its Drawbacks:** Explore how sparse representations work and discuss their limitations.
* **Bag of Words (BoW):** Learn about this foundational technique for text representation and its application.
* **Term Frequency–Inverse Document Frequency (TF-IDF):** Understand how TF-IDF improves upon BoW by weighing word importance.
* **Cosine Similarity:** Discover how cosine similarity measures the similarity between text documents.
* **Case study:** Building a Medium article recommendation system