## **What is NLP?**

NLP stands for **Natural Language Processing**, which is a part of **Computer Science, Human language,** and **Artificial Intelligence**. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as **translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction,** and **topic segmentation**.

**“Natural Language processing is that toolkit which gives machines the ability to read, understand and interpret human language.”**

**Why is natural language processing important?**

Businesses use massive quantities of unstructured, text-heavy data and need a way to efficiently process it. A lot of the information created online and stored in databases is natural human language, and until recently, businesses could not effectively analyse this data. This is where natural language processing is useful to analyse those data.

**Problem Statement:**

Every news that we consume is not real. If you listen to fake news, it means you are collecting the wrong information from the world which can affect society because a person’s views or thoughts can change after consuming fake news which the user perceives to be true.

Fake news detection is a hot topic in the field of natural language processing. We consume news through several mediums throughout the day in our daily routine, but sometimes it becomes difficult to decide which one is fake and which one is authentic. Our job is to create a model which predicts whether a given news is real or fake.

**Data Imbalanced Techniques:**

Imbalanced dataset means where the target class has an uneven distribution of observations, i.e one class label has a very high number of observations and the other has a very low number of observations.

1. Under sampling 🡪 Majority class observation will be removed in this process
2. Oversampling 🡪 Minority class observation will be added in this process
3. SMOTE 🡪 Synthetic Minority Oversampling Technique 🡪 KNN algorithm used and try to produce synthetic sample from existing dataset

**Handling Missing Values:** If we have numeric featured/column in that case replace missing value with mean/median/mode. But If We have categorical feature/column in that case replace with mode value

**Data Pre-processing:**

Data pre-processing involves preparing and "cleaning" text data for machines to be able to analyse it. pre-processing puts data in workable form and highlights features in the text that an algorithm can work with.

**1. Tokenization:**

Word Tokenizer is used to break the sentence into separate words or tokens. Or we can say is the process of splitting a string, text into a list of tokens.

**Why is Tokenization required?**

1. Tokens help in understanding the context.
2. Also, tokens help in interpreting the meaning of the text by analysing the sequence of words

**Types of Tokenization:**

1. **Word Tokenization:**

If the text is split into the words by using some separation techniques it is called as WT

1. **Sentence Tokenization:**

Same separation done for sentence is called as ST.

**3. Stop Words**

In English, there are a lot of words that appear very frequently like "is", “and” "the", and "a". NLP pipelines will flag these words as stop words.  **Stop words are those words in the text which does not add any meaning to the sentence and their removal will not affect the pre-processing of the text.**

**Why removed the stop words?**

**They are removed from the vocabulary to reduce noise to reduce dimension of the feature set.**

**4. Stemming**

Stemming is used to normalize words into its base form or root form. The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

**5. Lemmatization**

Lemmatization is quite similar to the Stemming. It is used to group different inflected forms of the word, called Lemma. The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

But if we compare speed in that case stemming is faster than lemmatization becoz in the lemmatization scan a corpus which consumed time and pre-processing.

Most of the time in the interview asked these questions what is the difference between stemming and lemmatization, and which one is better?

* We can answer like based on application we should decide.
* If speed is focused, then stemming should be used and if you are building language application in which language is important you should use lemmatization.

# Sentiment Analysis on TripAdvisor Hotel Reviews with Python and NLP

### [Building Your First NLP Application to Detect SPAM](https://blog.paperspace.com/nlp-spam-detection-application-with-scikitlearn-xgboost/)

Dataset <https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>

## **Email Spam Detection using Natural Language Processing with Python**

To get started, we should first import all the library.

Specifying the names of the columns while reading the tsv file. (Tab Separated Value).

our data contains a collection of 5574 SMS and also we have only 2 label: ham and spam.

As you may have noticed, our data is of type string. Therefore, we should transform it into a numeric vector to be able to perform the classification task. To do this, we use[**bag-of-words**](https://en.wikipedia.org/wiki/Bag-of-words_model#:~:text=The%20bag%2Dof%2Dwords%20model,word%20order%20but%20keeping%20multiplicity.)where each unique word in a text will be represented by a number.

First all of, we import CountVectorizer from sklearn learn:

Having the features represented as vectors, we can finally train our spam/ham classifier. You can use any classification algorithms . Here we use[*Support Vector Classification (SVC)*](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html)algortihm.

# **Approach :-**

# **1. Reading the Data**

To read the data, We already imported the Pandas library and read the data using pd.read\_csv() .

If you see here the data in the file is Tab (\t) separated, so we must provide the “sep” (separate) parameter. Also, the file does not contain any column names, so we should provide column names using “names” parameter.

Data is getting stored in a DataFrame named “df”.

**To view the first 5 rows of data, we should use messages.head()**

We can see here that the first column contains the **Labels (dependent variable)**i.e., a message is Spam or Ham and the second column contains the actual **Messages (independent variable)**

# **2. Exploratory Data Analysis**

**Df.info()**

We can see here that there are 5572 rows and 2 columns. It means that there are 5572 messages and 2 columns named “Label” and “Message”.

**Df.isna().sum()**

There are no missing values in the data.

**Now we check target variable count**

**Df[‘msg’].value\_counts()**

.value\_counts() helps to return total counts for each Category i.e. ‘Ham’ and ‘Spam’ . We can see that the ham messages are more than the Spam messages.

We see that 4825 out of 5572 messages, or 86.6%, are ham.  
This means that any machine learning model we create has to perform **better than 86.6%** to beat random chance.

# **3. Data Preprocessing**

Messages contain text but with many punctuation, Stop-words special characters, and many forms of verb. Now we will clean messages by removing the unnecessary things.

So, We already import **re** (regex library) and from nltk library, we imported **stopwords** and **WordNet Lemmatizer** and create its object as well.

**Re.sub()**

Now we will iterate through every message and by using regex (substitute method), we will take everything from the message except small alphabets(a-z) and capital alphabets(A-Z) and substitute it with blank space.

**Review.lower()**

Next, we will lowercase the message (as abc is not the same as ABC) to make learning easy for the machine

and then split it into words or we generate word tokens.

**Stopwords**

So once tokenization is done then we will check each word, that whether it exists in the stopwords collection by nltk or not.

**Lemmatization:**

Now we will apply lemmatizer. So first we create WordNet Lemmatizer object.

**Join words**

once each word is lemmatized, then we will again join the words and form a sentence

now we append final preprocessing output into the corpus list.

So in this corpus list now cleaned messages are available

**Replacing the messages with the cleaned messages that is available in the corpus.**

**Df[‘msg’]=corpus**

**Df.head()**

# **4.Model Building**

# **4.1 Data Splitting**

Splitting into ‘X’ , which contain the **independent variables**i.e., messages and ‘y’ include **dependent variable** (target variable) i.e., labels (spam or ham)

***Train Test Split***

Using the scikit-learn library, we can split the data into train and test. Here I have split the data into 70% (training data) and 30% (testing data)

**4.2 Dealing with Text data**

We can’t directly pass the text to the machine learning model as the machine only understands data in the form of 0’s and 1’s.

To solve this problem, we will use the concept of [TF-IDF Vectorizer](https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a) (Term Frequency-Inverse Document Frequency). It is a standard algorithm to transform the text into a meaningful representation of numbers and is used to fit the machine algorithm for prediction.

**Practical**

We can use the TF-IDF vectorizer from the scikit-learn library. Next, create an object of TF-IDF vectorizer and fit\_transform to the data, which will convert into a matrix of words and sentences.

Here, 3733 are the sentences of the X\_train, and 5772 are the total words obtained from the sentences.

**4.3 Pipelining**

We are doing Pipelining as we need to perform the same procedures for the test data to get predictions

If you don’t know about the Pipeline, it takes a list of tuple where each tuple takes the name set by you and calls any method you want to perform.

However, what is convenient about this pipeline object is that it can perform all these steps for you in a single cell, which means you can directly provide the data. It will be both vectorized and run the classifier in a single step.

Note:- When we will predict custom text later, we can directly pass the custom text to Pipeline, and it will help to predict the label

Practical :

We will import the MultinomialNB model from the scikit-learn library. Next, we will create a model named “*text\_mnb*” using Pipeline, where we first provided *TfidfVectorizer()* object and then *MultinomialNB()* object. It should be provided in a sequence as we want that firstly TfidfVectorizer should be executed and the output of it will be provided to the model, and at last, we fit the model with X\_train and y\_train.

Now every internal functionality will be handled by Pipeline and will perform the steps accordingly.

To make a prediction, we need to pass the X\_test data, and the Pipeline object will handle it, i.e., automatically vectorize it and make predictions for us.

“y\_preds\_mnb” contains the predictions from the X\_test made by our model and reaching an accuracy of approx 97%, which is relatively better than random chance.

We must know that accuracy itself is not capable of justifying that the model is working fine. We will use scikit-learn library to get a report on confusion\_matrix and classification\_report

To make a prediction, we need to pass the X\_test data, and the Pipeline object will handle it, i.e., automatically vectorize it and make predictions for us.