Low Level Design

Inner functioning of document

In this low level design document we deal with the theory (internal) part of the system.

Import necessary python libraries and data set files.

Pandas → data operations

Nltk → efficiency of the text

Sklearn → Algorithm

```
In [9]: import pandas as pd   #pandas library for data frame
        datatrain=pd.read_csv(r"C:\Users\welcome\Desktop\BBC News Train.csv")   #traindata
        datatest=pd.read_csv(r"C:\Users\welcome\Desktop\BBC News Test.csv")    #test data
```

1.Nltk

In the first step there is an urgent need to make the string or text free from Upercase,special charactes, number and stopwords (he,she,it..). Make a feature Extraction and encode it with vector. Make label for category columns.

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
corpus = []
corpust = []
for i in range(0, 1490):                                                    #making tr
  text = re.sub('[^a-zA-Z]', ' ', datatrain['Text'][i])
  text = text.lower()
  text = text.split()
  ps = PorterStemmer()
  all_stopwords = stopwords.words('english')
  all_stopwords.remove('not')
  text = [ps.stem(word) for word in text if not word in set(all_stopwords)]
  text = ' '.join(text)
  corpus.append(text)
for i in range(0, 735):                                                     #making tes
  textt = re.sub('[^a-zA-Z]', ' ', datatest['Text'][i])
  textt = textt.lower()
  textt = textt.split()
  ps = PorterStemmer()
  all_stopwords = stopwords.words('english')
  all_stopwords.remove('not')
  textt = [ps.stem(word) for word in textt if not word in set(all_stopwords)]
  textt = ' '.join(textt)
  corpust.append(textt)
```

Feature and label adding

```
from sklearn.feature_extraction.text import CountVectorizer    #feature extraction
cv = CountVectorizer()
X = cv.fit_transform(corpus).toarray()      #fit and transform features in train data
x = cv.transform(corpust).toarray()         # transform features of test data
y = datatrain.iloc[:,-1].values
from sklearn.preprocessing import LabelEncoder    # label for category
a=LabelEncoder()
Y=a.fit_transform(y)
```

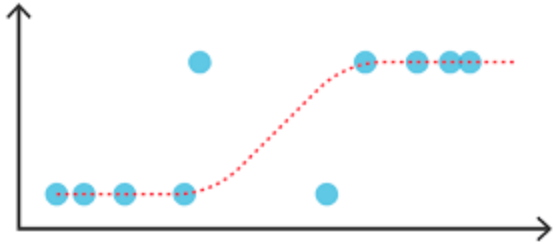
2. Apply ML Algorithim

Now , by keep the type of data in mind a best algorithm will be used (in this case logistic regression) .

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

```
from sklearn.linear_model import LogisticRegression

lg=LogisticRegression(random_state=0).fit(X,Y)
e=lg.predict(x)
print(a.inverse_transform(e))  #predicting
```



Logistic regression with 97% accuracy so selected

```
[19:25:39] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'm
ulti:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
[('Logistic Regresion', 0.9785522788203753), ('XGboost', 0.9624664879356568), ('SVlinear', 0.9490616621983914), ('Random Fores
t', 0.9115281501340483), ('naiveBayes', 0.900804289544236), ('Decision tree', 0.839142091152815), ('SVrbf', 0.739946380697051),
('KNN', 0.675603217158177)]
```

```
# datatest["Pr"]=a.inverse_transform(e)  # prediction of test dataset goes to PredictedTest.csv file
```

3. Predicting....

The direct filling of the test.csv file with pandas lib.

```
In [10]: datatest["Category"]=a.inverse_transform(e)  # prediction of test dataset goes to Finalt.csv file
         datatest.to_csv("Finalt.csv",index=False)
```

Direct category fill to csv file

That’s all for LLD

:-Vaibhav Chopra