Configuration Manual

Detecting Paraphrases in Hindi Language Using Machine Learning

Jyotsna Patel

Msc Data Analytics

National College of Ireland

x17108322@student.ncirl.ie

MSc Research Project in Data Analytics

1. Introduction

This Configuration Manual book of the research project "Paraphrasing in Hindi Language Using Machine Learning Techniques". The main concern was to improve the accuracy of paraphrase detection in Hindi language with the help of various feature extraction available for Hindi Language.

This configuration manual contains detail explanation Software Environment Specification, Knowledge of tools used for this research project, implementation work, comprehensive detail for this project.

2. Environment Specification

This research project was deployed on the Hp Laptop with "WINDOWS 10" as operating system. This research project result and output was deployed on following environment specification.

2.1 Hardware Specification

Device Specification:

Device Name:	HP Pavilion TS Sleekbook 14
Processor:	Intel® Core™ i5-3337U CPU @ 1.80GHz 1.80GHz
Installed RAM:	8.00 GB
Device ID:	F1A98F82-7F45-4096-8A98-8CECF7B77A4A
System type:	64-bit operating sytem, x64-based processor

Windows Specification:

Edition :	Windows 10 Home Single Language
Version:	1803
OS build:	17134.471

2.2 Software Specification

2.2.1 Windows 10 Installation Process

Step 1: Windows 10 installation file or flash drive must be inserted in to the laptop and can be downloaded from https://www.microsoft.com/en-us/software-download/windows10.

Step 2: Open the **Start** menu. Click the windows icon.

Step 3: Click the **Power** icon.

Step 4: Click the **Restart** icon (figure 1).

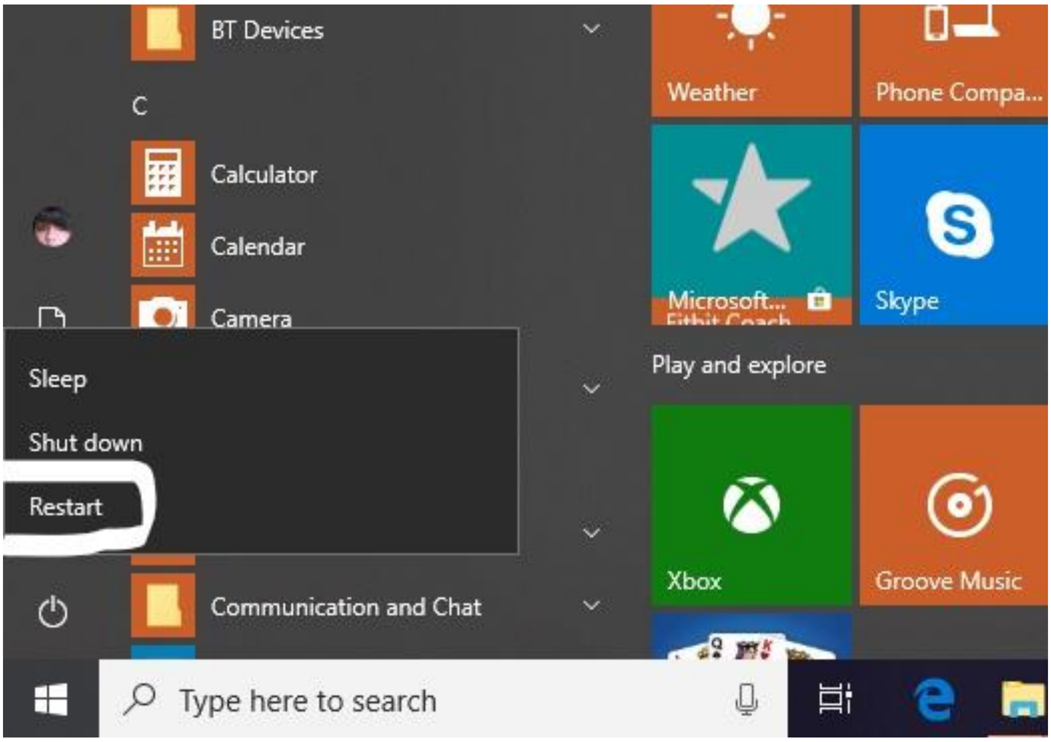


Figure 1: Step

Step 5: Click and hold Delete or F2to enter setup as shown in figure 2. It direct shows the boot environmnt display.

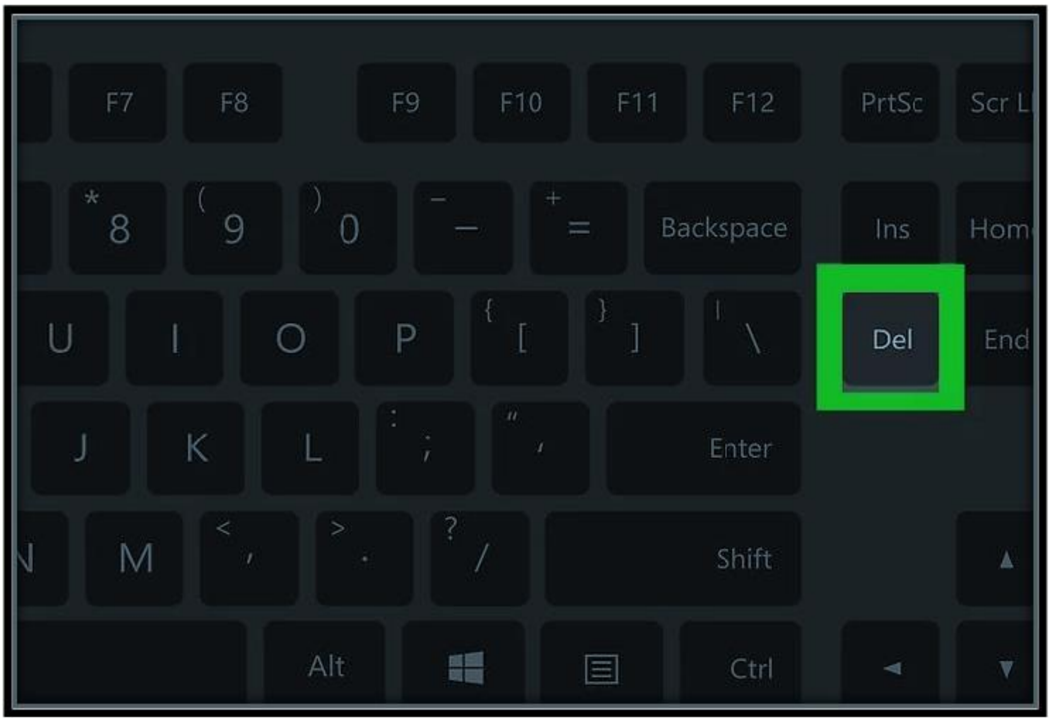
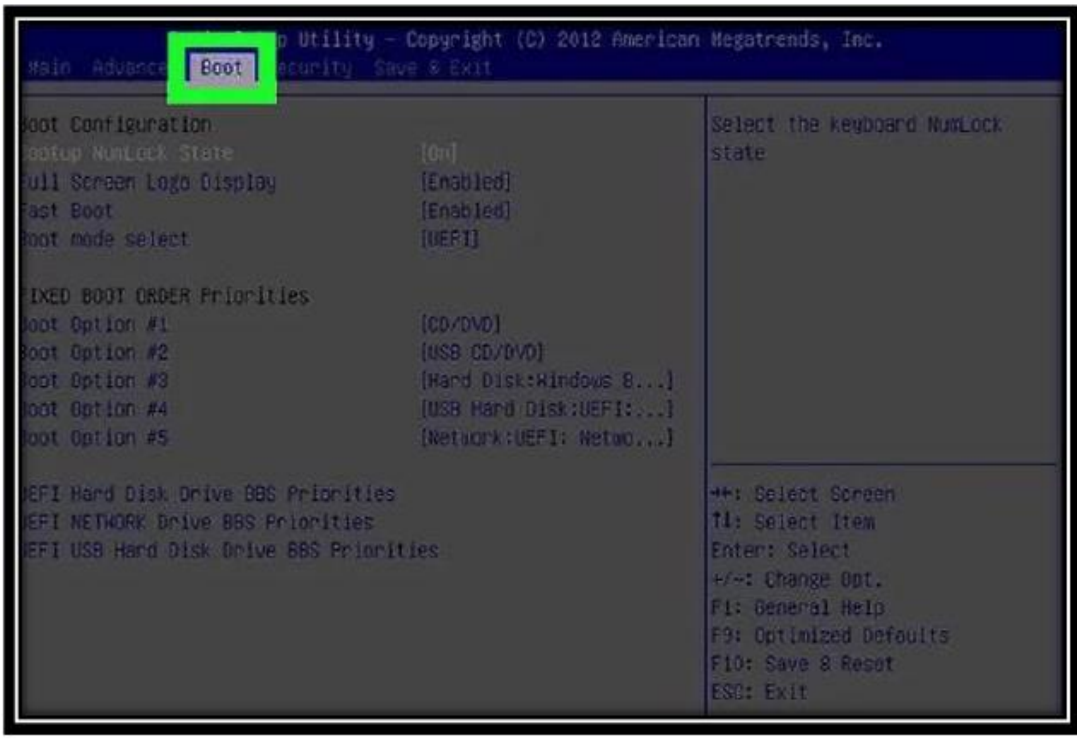


Figure 2: Step 5

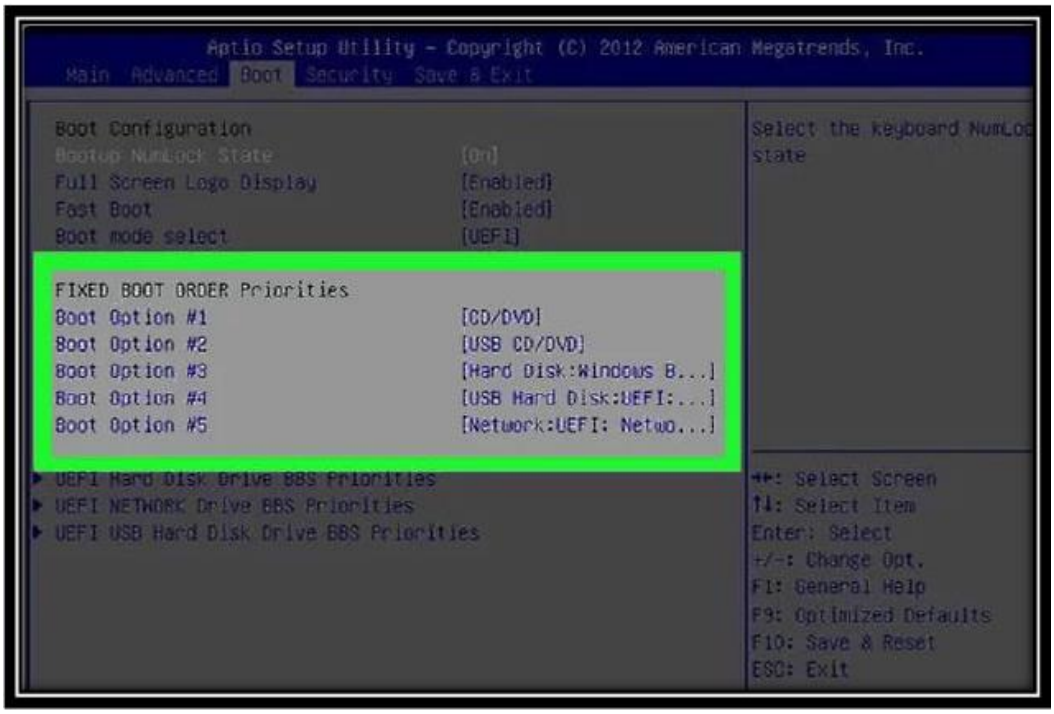
Step 6: Click on boot icon as shown in Figure 3.



```
             p Utility - Copyright (C) 2012 American Megatrends, Inc.
Main  Advance  Boot  ecurity  Save & Exit

oot Configuration                                   Select the keyboard NumLock
ootup NumLock State               [On]              state
ull Screen Logo Display           [Enabled]
ast Boot                          [Enabled]
oot mode select                   [UEFI]

IXED BOOT ORDER Priorities
oot Option #1                     [CD/DVD]
oot Option #2                     [USB CD/DVD]
oot Option #3                     [Hard Disk:Windows 8...]
oot Option #4                     [USB Hard Disk:UEFI:...]
oot Option #5                     [Network:UEFI: Netwo...]

EFI Hard Disk Drive BBS Priorities                  →←: Select Screen
EFI NETWORK Drive BBS Priorities                    ↑↓: Select Item
EFI USB Hard Disk Drive BBS Priorities              Enter: Select
                                                    +/-: Change Opt.
                                                    F1: General Help
                                                    F9: Optimized Defaults
                                                    F10: Save & Reset
                                                    ESC: Exit
```

Figure 3: Step 6

Step 7: Select the Booting option. Removable devices for USB flash Drive and CD ROM Drive option for disc Installation Figure 4.



```
Aptio Setup Utility – Copyright (C) 2012 American Megatrends, Inc.
 Main  Advanced  Boot  Security  Save & Exit

  Boot Configuration                                      Select the keyboard NumLoc
  Bootup NumLock State                [On]                state
  Full Screen Logo Display            [Enabled]
  Fast Boot                           [Enabled]
  Boot mode select                    [UEFI]

  FIXED BOOT ORDER Priorities
  Boot Option #1                      [CD/DVD]
  Boot Option #2                      [USB CD/DVD]
  Boot Option #3                      [Hard Disk:Windows B...]
  Boot Option #4                      [USB Hard Disk:UEFI:...]
  Boot Option #5                      [Network:UEFI: Netwo...]
▶ UEFI Hard Disk Drive BBS Priorities                     →←: Select Screen
▶ UEFI NETWORK Drive BBS Priorities                       ↑↓: Select Item
▶ UEFI USB Hard Disk Drive BBS Priorities                 Enter: Select
                                                          +/-: Change Opt.
                                                          F1: General Help
                                                          F9: Optimized Defaults
                                                          F10: Save & Reset
                                                          ESC: Exit
```

Figure 4: Step 7

Step 8: Click "+" key from keyboard until boot option selection after Removable Device option or CD-ROM Drive appear on the screen and select as default boot option as shown in figure 5.

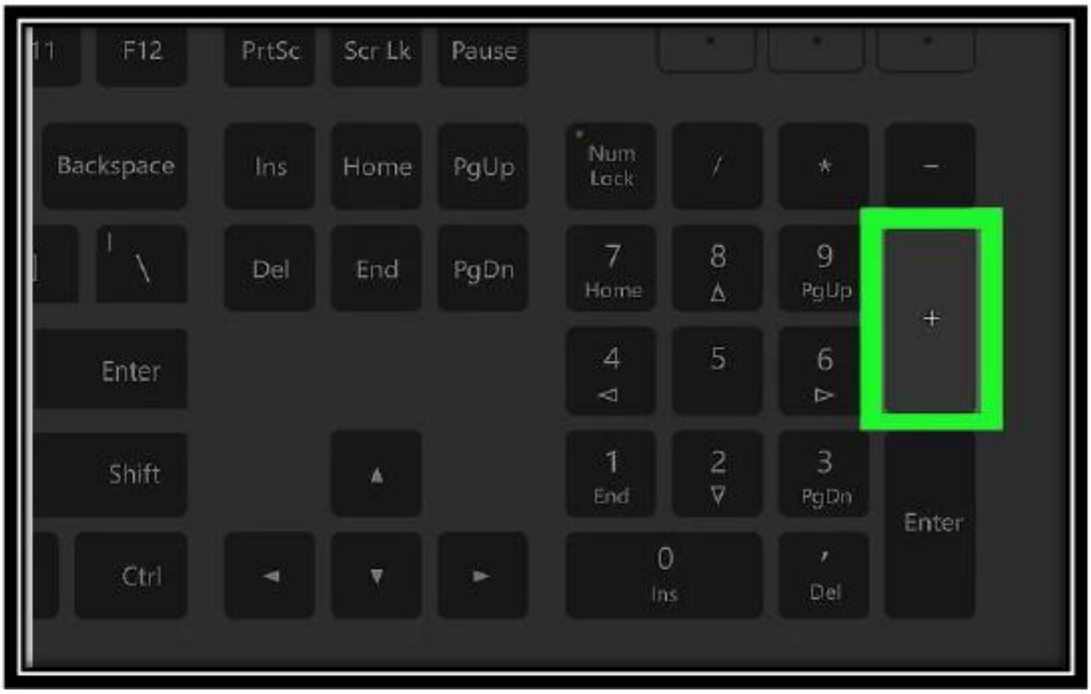
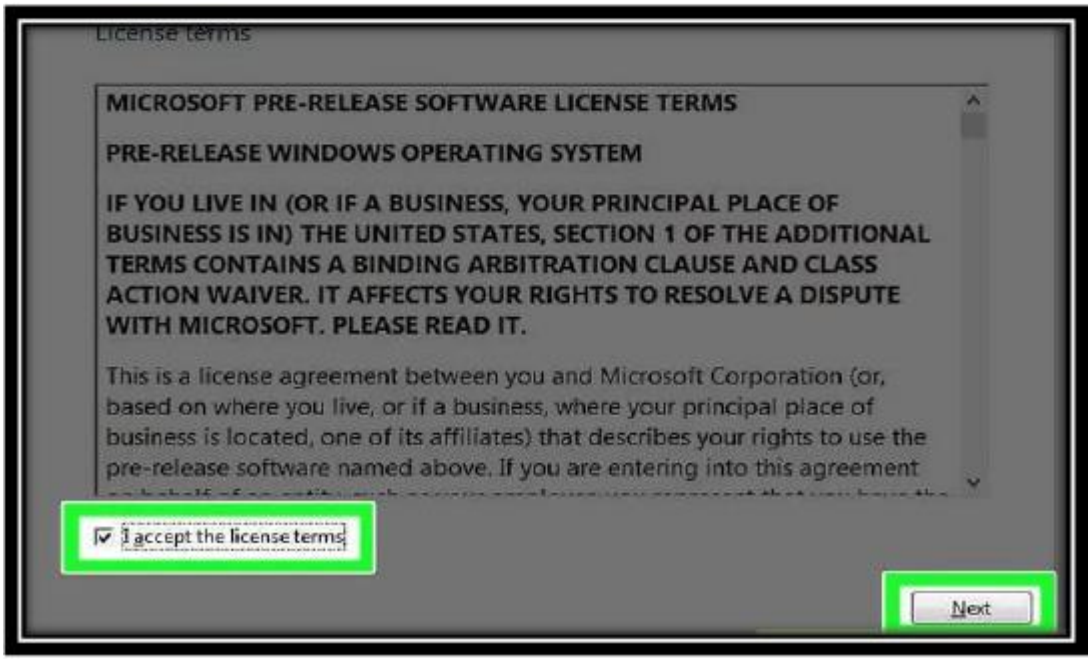


Figure 5: Step 8

Step 9: Save settings and exit (restart the laptop). Click enter to changes. Wait till the laptop restart and when its finish, windows icon shows on screen.

Step 10: Choose language, click Next. Press install now option: this process takes time, then click next. Accept the licence terms and click Next icon.



License terms

MICROSOFT PRE-RELEASE SOFTWARE LICENSE TERMS

PRE-RELEASE WINDOWS OPERATING SYSTEM

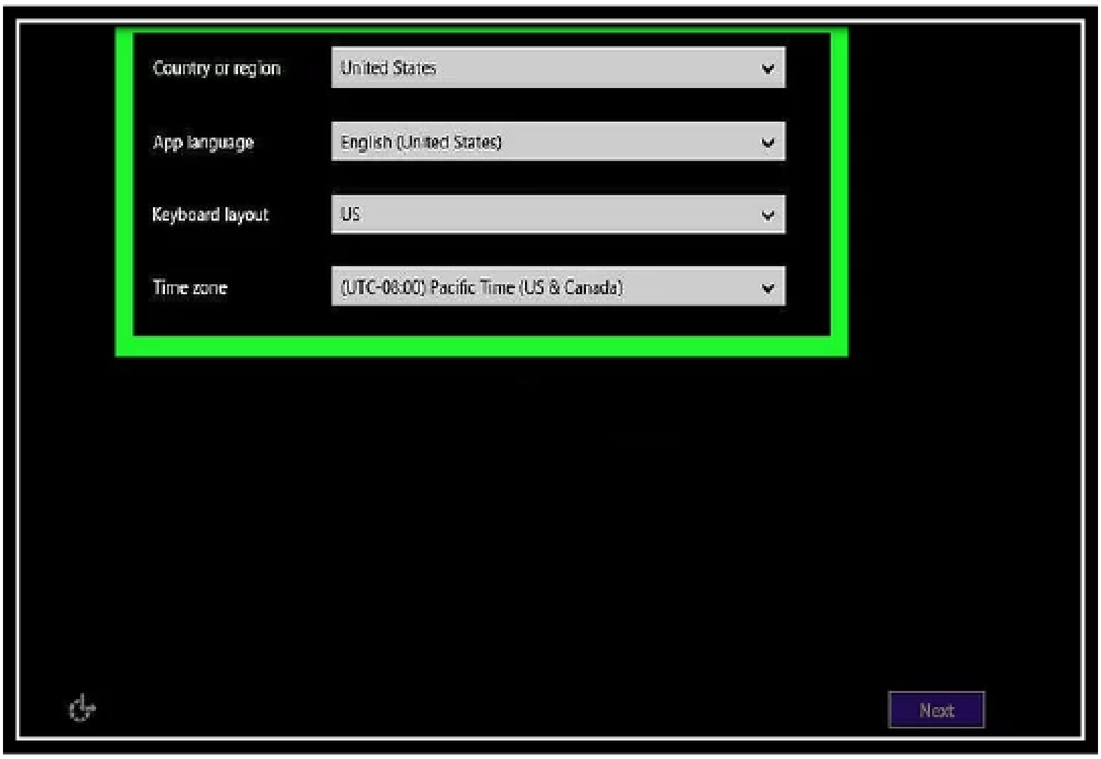
IF YOU LIVE IN (OR IF A BUSINESS, YOUR PRINCIPAL PLACE OF BUSINESS IS IN) THE UNITED STATES, SECTION 1 OF THE ADDITIONAL TERMS CONTAINS A BINDING ARBITRATION CLAUSE AND CLASS ACTION WAIVER. IT AFFECTS YOUR RIGHTS TO RESOLVE A DISPUTE WITH MICROSOFT. PLEASE READ IT.

This is a license agreement between you and Microsoft Corporation (or, based on where you live, or if a business, where your principal place of business is located, one of its affiliates) that describes your rights to use the pre-release software named above. If you are entering into this agreement

☑ I accept the license terms

Figure 6: Step 10

Step 11: Follow the instruction and set up the region, Language, layout, time zone and location. Click next. Here and now windows 10 is ready for use. Figure 7.



Country or region: United States

App language: English (United States)

Keyboard layout: US

Time zone: (UTC-08:00) Pacific Time (US & Canada)

Figure 7: Step 11

2.2.2 Anaconda Installation Process

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system conda.

Step 1: The first step is to go to the website https://www.anaconda.com/download/ and download Anaconda as shown in Figure 8.

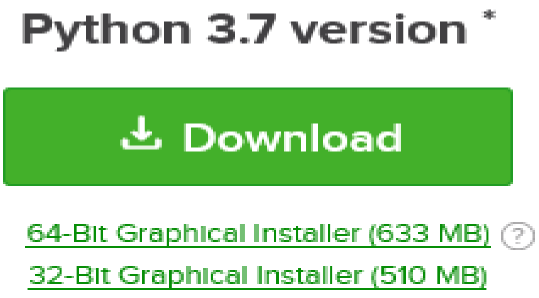
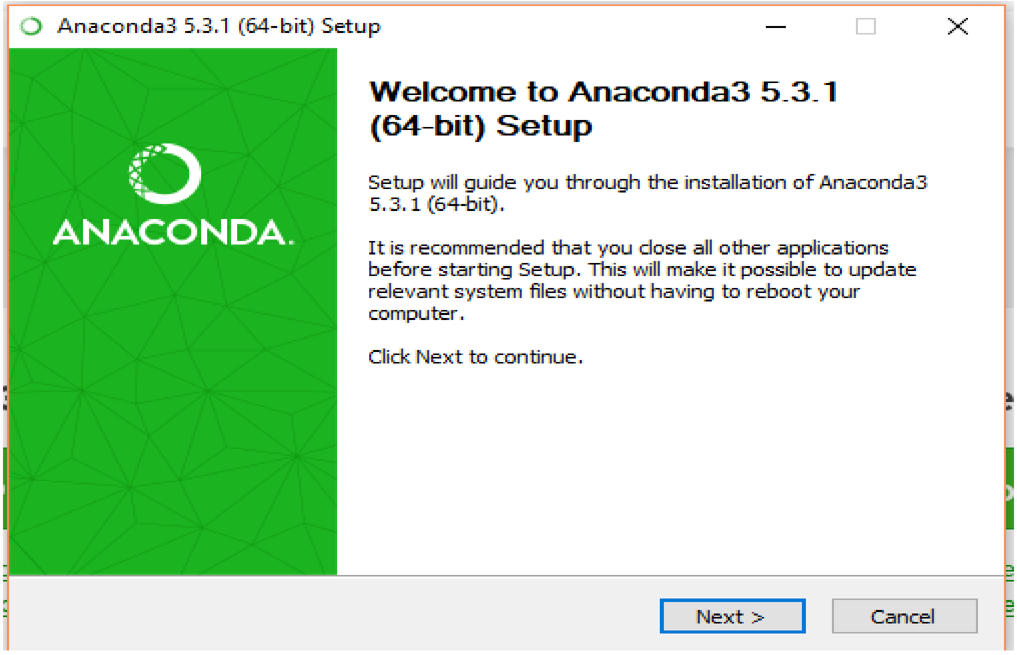

Python 3.7 version *
Download
64-Bit Graphical Installer (633 MB)
32-Bit Graphical Installer (510 MB)

Figure 8: Download Icon

Step 2: After downloading, search the file in downlaods option and then double click. The file was with “Anaconda3-5.3.1-Windows-x86_64.exe”. After this process start.

Step 3: Install wizard open click on next button as shown in Figure 9.



Welcome to Anaconda3 5.3.1 (64-bit) Setup

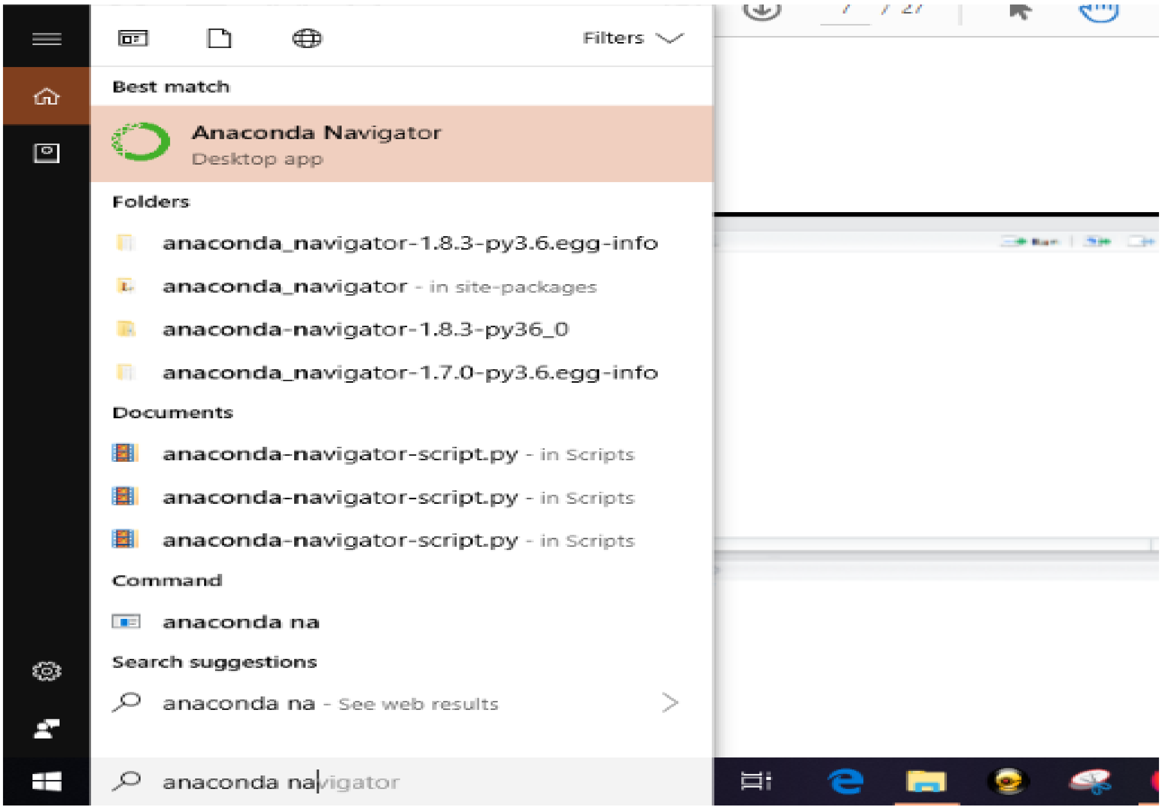
Setup will guide you through the installation of Anaconda3 5.3.1 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 9: Installation Wizard

Step 4: Go to Start button, Search for Anaconda Navigator and launch it as shown in figure 10.



Best match

Anaconda Navigator
Desktop app

Folders

anaconda_navigator-1.8.3-py3.6.egg-info

anaconda_navigator - in site-packages

anaconda-navigator-1.8.3-py36_0

anaconda_navigator-1.7.0-py3.6.egg-info

Documents

anaconda-navigator-script.py - in Scripts

anaconda-navigator-script.py - in Scripts

anaconda-navigator-script.py - in Scripts

Command

anaconda na

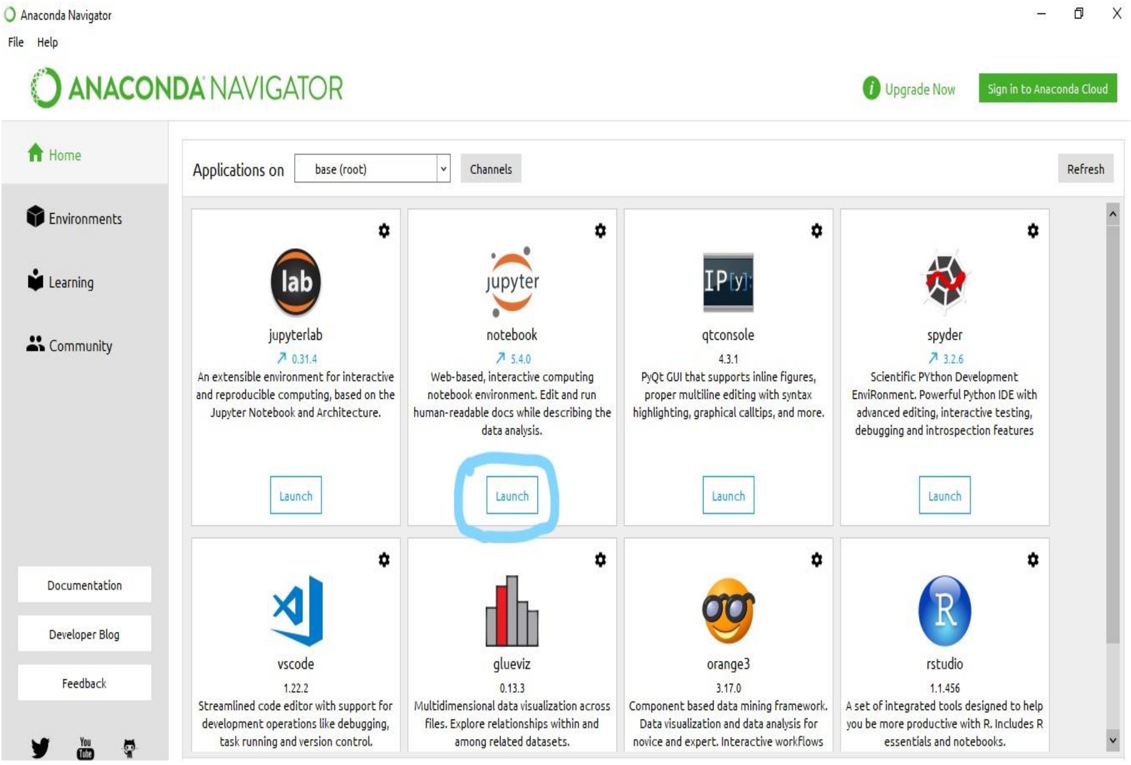
Search suggestions

anaconda na - See web results

anaconda navigator

Figure 10: Launch Anaconda Navigator

Step 5: Launch Jupyter Notebook from Anaconda Navigator



jupyterlab
0.31.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

notebook
5.4.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder
3.2.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

vscode
1.22.2
Streamlined code editor with support for development operations like debugging, task running and version control.

glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.

orange3
3.17.0
Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows

rstudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 11: Launch Jupyter Notebook

Step 6: Now, Jupyter Notebook is ready to use as show` `zn in figure 12.

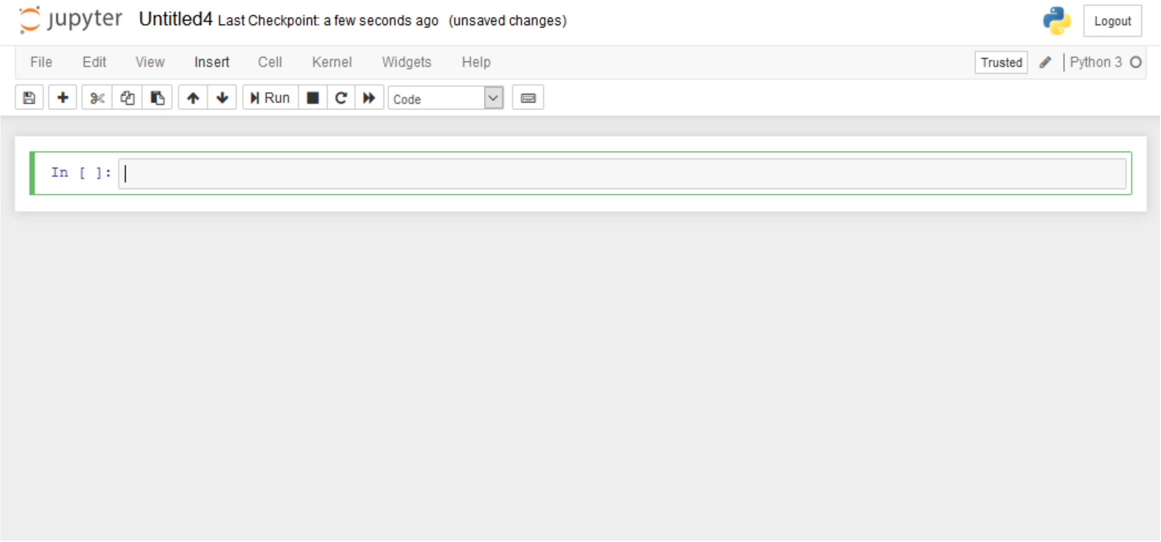


Figure: 12 Step 6

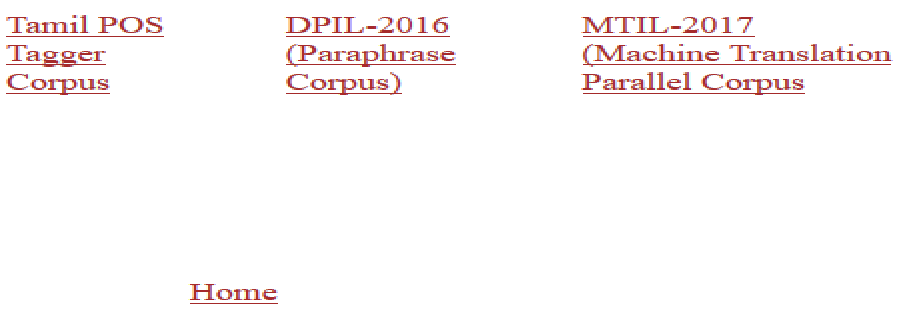
2.2.3 Other Software

- Microsoft Word
- Microsoft Excel
- Lucid Chart

3. Implementation

3.1 Data Source

The data requested from Amrita Cen NLP Group.. It can be requested from https://nlp.amrita.edu/nlpcorpus.html as shown below in figure.



Tamil POS Tagger Corpus

DPIL-2016 (Paraphrase Corpus)

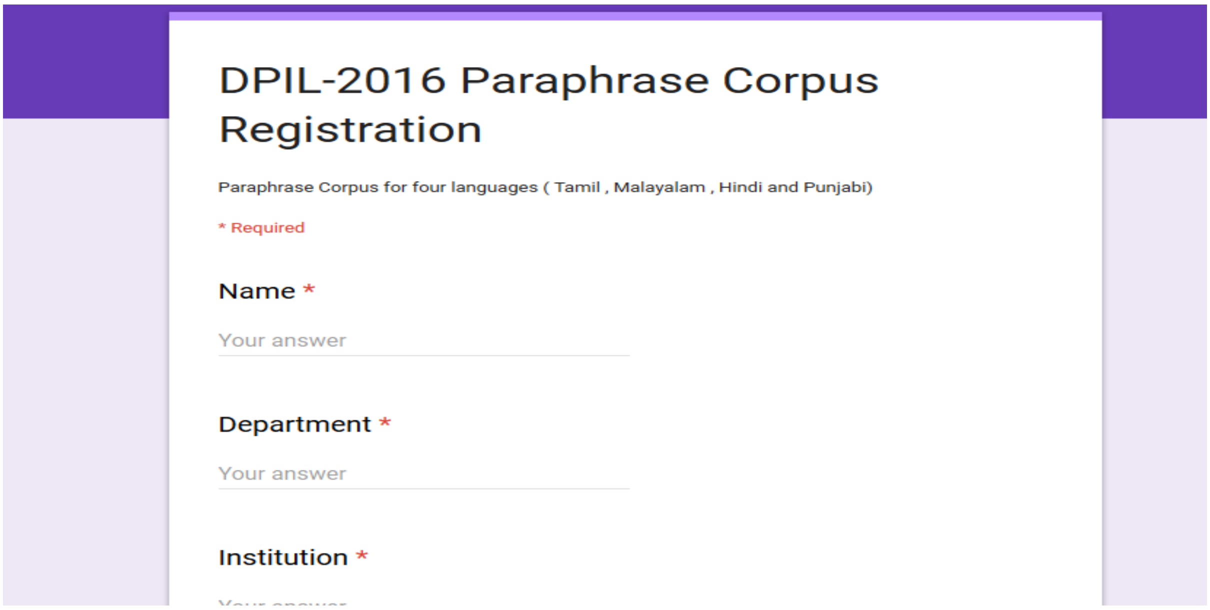
MTIL-2017 (Machine Translation Parallel Corpus

Home

Figure 13: Data Source

3.2 Data Extraction and Pre-Processing

Step 1: Fill the below form (figure 14) to request the dataset.



DPIL-2016 Paraphrase Corpus Registration

Paraphrase Corpus for four languages (Tamil , Malayalam , Hindi and Punjabi)

* Required

Name *

Your answer

Department *

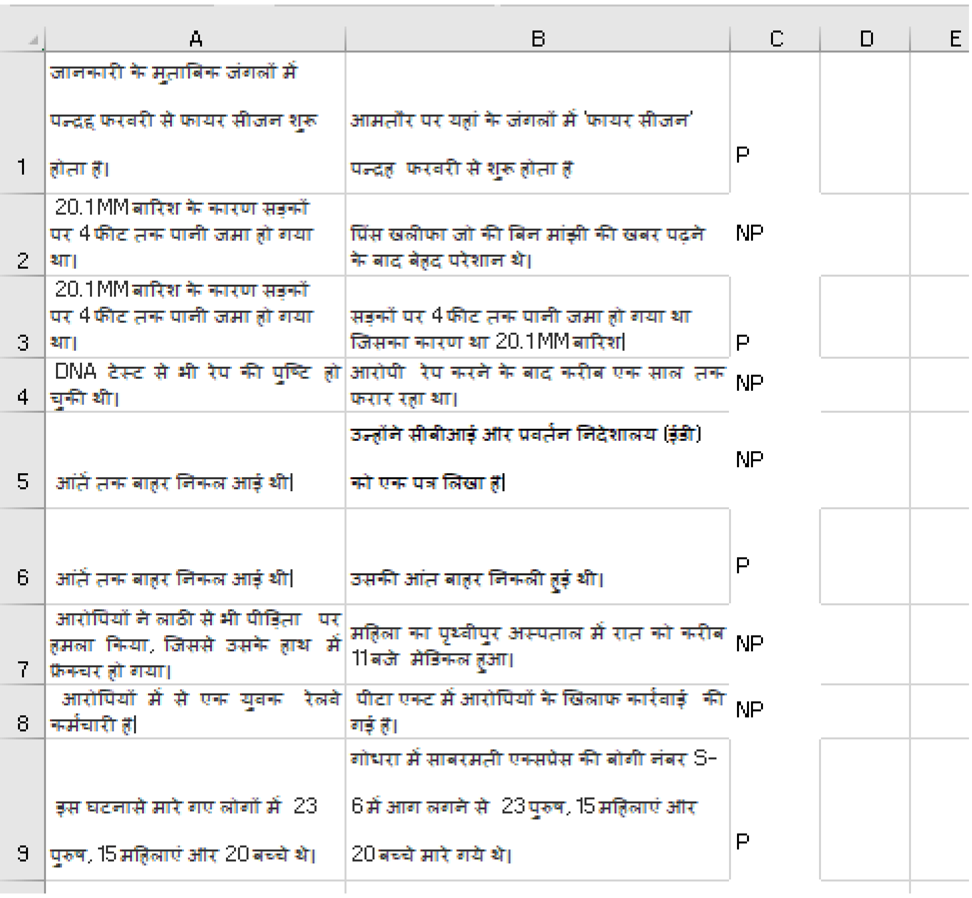
Your answer

Institution *

Your answer

Figure 14: Data Requesting Form

Step 2: Final Data in Figure 15.



	A	B	C	D	E
1	जानकारी के मुताबिक जंगलों में पन्दह फरवरी से फायर सीजन शुरू होता है।	आमतौर पर यहां के जंगलों में 'फायर सीजन' पन्दह फरवरी से शुरू होता है	P		
2	20.1MM बारिश के कारण सड़कों पर 4 फीट तक पानी जमा हो गया था।	प्रिंस खलीफा जो की बिन मांझी की खबर पढ़ने के बाद बेहद परेशान थे।	NP		
3	20.1MM बारिश के कारण सड़कों पर 4 फीट तक पानी जमा हो गया था।	सड़कों पर 4 फीट तक पानी जमा हो गया था जिसका कारण था 20.1MM बारिश।	P		
4	DNA टेस्ट से भी रेप की पुष्टि हो चुकी थी।	आरोपी रेप करने के बाद करीब एक साल तक फरार रहा था।	NP		
5	आंतें तक बाहर निकल आई थी।	उन्होंने सीबीआई और प्रवर्तन निदेशालय (ईडी) को एक पत्र लिखा है।	NP		
6	आंतें तक बाहर निकल आई थी।	उसकी आंत बाहर निकली हुई थी।	P		
7	आरोपियों ने लाठी से भी पीड़िता पर हमला किया, जिससे उसके हाथ में फ्रैक्चर हो गया।	महिला का पृथ्वीपुर अस्पताल में रात को करीब 11बजे मेडिकल हुआ।	NP		
8	आरोपियों में से एक युवक रेलवे कर्मचारी है।	पीटा एक्ट में आरोपियों के खिलाफ कार्रवाई की गई है।	NP		
9	इस घटनासे मारे गए लोगों में 23 पुरुष, 15 महिलाएं और 20 बच्चे थे।	गोधरा में साबरमती एक्सप्रेस की बोगी नंबर S- 6 में आग लगने से 23 पुरुष, 15 महिलाएं और 20 बच्चे मारे गये थे।	P		

Figure 15: Dataset

Step 3: import data in Jupyter Notebook.

3.3 Implementation

3.3.1 Packages and library for Hindi dataset:

```
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.corpus import wordnet as wn
from nltk.corpus import stopwords
from pyiwn import pyiwn
import re, math
from collections import Counter
import re, math
from collections import Counter
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tag import tnt
from nltk.corpus import indian
nltk.download('indian')
from pandas import ExcelWriter
from pandas import ExcelFile
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, recall_score, precision_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import VotingClassifier
import matplotlib.pyplot as plt
```

3.3.2 Code for importing Dataset in Jupyter Notebook

```
data = pd.read_excel("E:/NCI/Thesis/TestHindi.xls",header=None)
```

3.3.3 Code for K-Nearest Neighbors

```
classifier = KNeighborsClassifier(n_neighbors=5)

AdaBoost = AdaBoostClassifier(base_estimator= classifier,n_estimators=400,learning_rate=1)

AdaBoost = AdaBoostClassifier(n_estimators=400,learning_rate=1,algorithm='SAMME')

Ada = AdaBoost.fit(X_train,y_train)

prediction = Ada.predict(X_test)
```

3.3.4 Code for Decision Tree

```
clf = DecisionTreeClassifier(random_state=0)

bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2), n_estimators=600, learning_rate=1)

model_real = bdt.fit(X_train, y_train)

predict_real = model_real.predict(X_test)
```

4. Perfomance Evaluation

4.1 K-Nearest Neighbor Performance Evaluation

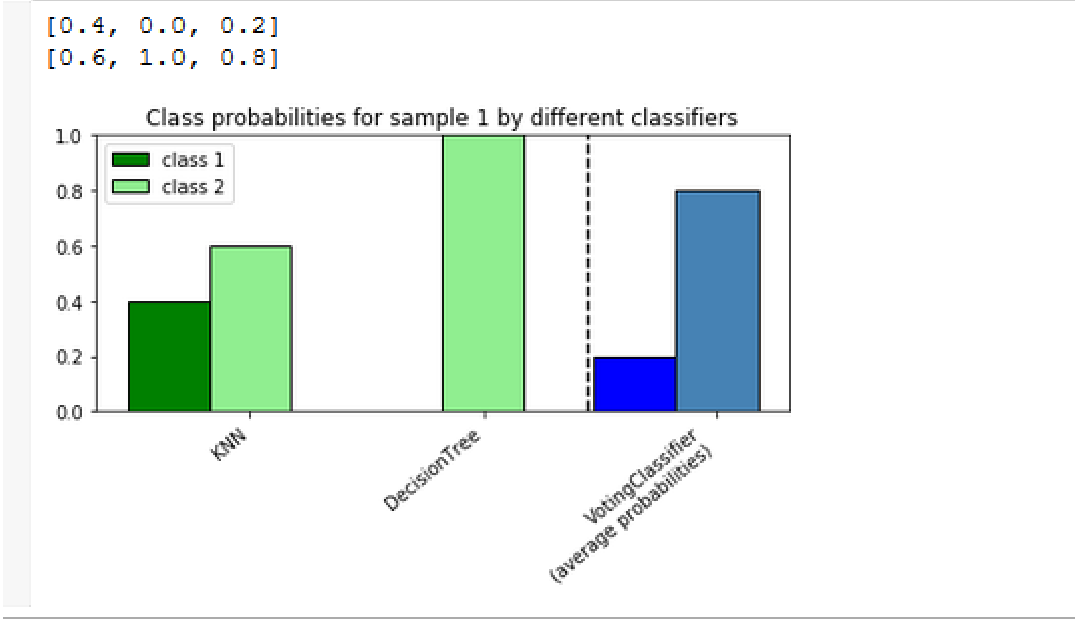
```
print(confusion_matrix(y_test, prediction))

accuracy_score = accuracy_score(y_test, prediction)

print(accuracy_score)

f1_score = f1_score(y_test, prediction, average='macro')

print(f1_score)

recall_score = recall_score(y_test, prediction, average='macro')

print(recall_score)

precision_score = precision_score(y_test, prediction, average='macro')

print(precision_score)

print('Average accuracy: %0.2f +/- (%0.1f) %%' % (accuracy_score.mean()*100, accuracy_score.std()*100))

print('Average Precision: %0.2f +/- (%0.1f) %%' % (precision_score.mean()*100, precision_score.std()*100))

print('Average Recall: %0.2f +/- (%0.1f) %%' % (recall_score.mean()*100, recall_score.std()*100))

print('Average F1-Score: %0.2f +/- (%0.1f) %%' % (f1_score.mean()*100, f1_score.std()*100))
```

4.2 Decision Tree

```
print(confusion_matrix(y_test, predict_real))
accuracy_score = accuracy_score(y_test, predict_real)
print(accuracy_score)
f1_score = f1_score(y_test, predict_real, average='macro')
print(f1_score)
recall_score = recall_score(y_test, predict_real, average='macro')
print(recall_score)
precision_score = precision_score(y_test, predict_real, average='macro')
print(precision_score)
print('Average accuracy: %0.2f +/- (%0.1f) %%' % (accuracy_score.mean()*100, accuracy_score.std()*100))
print('Average Precision: %0.2f +/- (%0.1f) %%' % (precision_score.mean()*100, precision_score.std()*100))
print('Average Recall: %0.2f +/- (%0.1f) %%' % (recall_score.mean()*100, recall_score.std()*100))
print('Average F1-Score: %0.2f +/- (%0.1f) %%' % (f1_score.mean()*100, f1_score.std()*100))
```

5. Results

K-Nearest Neighbor performed better than Decision Tree with Adaptive Boosting on the Hindi dataset used in this project but the different is significant. But According to the Voting classifier result and above graph referred to Figure 16 we can say that Decision Tree will be better algorithm for text classification.



```
[0.4, 0.0, 0.2]
[0.6, 1.0, 0.8]
```

6. Appendix

```
# coding: utf-8

# In[1]:


import warnings
warnings.filterwarnings("ignore")


# In[2]:



import pandas as pd
import nltk
#nltk.download()
from nltk.tokenize import word_tokenize
#nltk.download('stopwords')

#importing Dataset
data = pd.read_excel("E:/NCI/Thesis/TestHindi.xls",header=None)

#Data Preprocessing
tokenized0=[]
tokenized1=[]
paraphrased=[]
for index,i in data.iterrows():
    tokenized0.append(word_tokenize(i[0]))
    tokenized1.append(word_tokenize(i[1]))
    paraphrased.append(i[2])

tokenizedDF = pd.DataFrame(
    {0: tokenized0,
     1: tokenized1,
     2:paraphrased
    })

from nltk.corpus import stopwords

stop = stopwords.words('hindi')



tokenizedDF[0]=tokenizedDF[0].apply(lambda x: [item for item in x if item not in stop])
tokenizedDF[1]=tokenizedDF[1].apply(lambda x: [item for item in x if item not in stop])



#Stemming and Lemmatization
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer



ps = PorterStemmer()
#example_words = ["python","pythoner","pythoning","pythoned","pythonly"]
#for w in tokenizedDF[0]:
 #   print(ps.stem(w))

new_text = "It is important to by very pythonly while you are pythoning with python. All pythoners have pythoned poorly at least once."
words = word_tokenize(new_text)

stem=['कों','ौ','ै','ा','ी','ू','ो','े','्','ि','ु','ं','ॅ','कर','ाओ','िए','ाई','ाए','ने','नी','ना','ते','ीं','ती','ता','ाॅ','ां','ों','ें','ाकर','ाइए','ाईं','ाया','ेगी','ेगा','ोगी','ोगे','ाने','ाना','ाते','ाती','ाता','तीं','ाओं','ाएं','ुओं','ुएं','ुआं']
tokenizedDF[0]=tokenizedDF[0].apply(lambda x: [item for item in x if item not in stem])
#for w in tokenizedDF[0]:
    #print(PorterStemmer)
    #print(ps.stem(w))
```

```
# In[ ]:



from nltk.corpus import wordnet as wn
from nltk.corpus import stopwords
from pyiwn import pyiwn

iwn= pyiwn.IndoWordNet('hindi')

#row = tokenizedDF.iloc[0] #Just taking the first row, you can put a loopover
#print(row)
# terms1=tokenizedDF[0].iloc[0]
# print(terms1)
# terms2=tokenizedDF[1].iloc[0]
# print(terms2)




sims = []

Synonyms1=[]
Synonyms2=[]
j=0
for index,i in tokenizedDF.iterrows():
    #terms1=tokenizedDF[0].iloc[index]
    terms1=tokenizedDF[0].iloc[index]
    syn1 = []
    syn2=  []
    #print(terms1)
    for word1 in terms1:
        try:
            syn1.append(iwn.synsets(word1)[0])
        except:  #if wordnet is not able to find a synset for word1
            sims.append([0 for i in range(0, len(terms1))])
            continue
    Synonyms1.append(syn1)
    print(Synonyms1)
    terms2=tokenizedDF[1].iloc[index]
    #print(terms1)
    for word2 in terms2:
        try:
            syn2.append(iwn.synsets(word2)[0])
        except:  #if wordnet is not able to find a synset for word1
            sims.append([0 for i in range(0, len(terms2))])
            continue
    Synonyms2.append(syn2)

newinput_list1=[]
newinput_list2=[]
from nltk import ngrams
for index,i in tokenizedDF.iterrows():
    terms1=tokenizedDF[0].iloc[index]
    terms2=tokenizedDF[1].iloc[index]
    newinput_list1.append(list(zip(terms1, terms1[1:])))
    newinput_list2.append(list(zip(terms2, terms2[1:])))
newinput_list1
newinput_list2



# In[6]:

#Feature Extraction
import re, math
from collections import Counter

WORD = re.compile(r'\w+')

def get_cosine(vec1, vec2):
     intersection = set(vec1.keys()) & set(vec2.keys())
     numerator = sum([vec1[x] * vec2[x] for x in intersection])

     sum1 = sum([vec1[x]**2 for x in vec1.keys()])
     sum2 = sum([vec2[x]**2 for x in vec2.keys()])
     denominator = math.sqrt(sum1) * math.sqrt(sum2)

     if not denominator:
        return 0.0
     else:
        return float(numerator) / denominator

def text_to_vector(text):
    output=[]
    vect1={}
    for k in text:
        if k not in output:
            output.append(k)
    for i in output:
        count = 0
        for j in text:
            if i == j:
                count=count+1
        if i in vect1:
            vect1[i].append(count)
        else:
            vect1[i]= count
        #count=str(count)
        #vect1.append(i+':'+count)
        #print(Counter({vect1}))
    return vect1
#       words = WORD.findall(text)
#       print(words)
#       return Counter(words)

cosine=[]
for index,i in tokenizedDF.iterrows():
    terms1=[]
    terms2=[]
    terms1=tokenizedDF[0].iloc[index]
    terms2=tokenizedDF[1].iloc[index]
    #text1 = ['जानकारी', 'मुताबिक','जानकारी', 'जंगलों', 'पन्द्रह्', 'फरवरी', 'फायर', 'सीजन', 'शुरू','जानकारी', 'जंगलों', 'पन्द्रह्']
    #text2 = ['आमतौर', 'जंगलों', '"फायर"', 'सीजन', '""', 'पन्द्रह', 'फरवरी', 'शुरू']
    # text1 = 'जानकारी के मुताबिक जंगलों में पन्द्रह् फरवरी से फायर सीजन शुरू होता है। '
    # text2 = 'आमतौर पर यहां के जंगलों में  सीजन पन्द्रह  फरवरी से शुरू होता है'
    vector1 = text_to_vector(terms1)
    #print(vector1)
    vector2 = text_to_vector(terms2)
    #print(vector2)
    cosine.append(get_cosine(vector1, vector2))

print ('Cosine:', cosine)



# In[7]:



import re, math
from collections import Counter

WORD = re.compile(r'\w+')

def get_cosine(vec1, vec2):
    intersection = set(vec1.keys()) & set(vec2.keys())
    numerator = sum([vec1[x] * vec2[x] for x in intersection])

    sum1 = sum([vec1[x]**2 for x in vec1.keys()])
    sum2 = sum([vec2[x]**2 for x in vec2.keys()])
    denominator = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominator:
        return 0.0
    else:
        return float(numerator) / denominator

def text_to_vector(text):
    words = WORD.findall(text)
    return Counter(words)

import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
sim=[]
for index,i in tokenizedDF.iterrows():
    text1=data[0].iloc[index]
    text2=data[1].iloc[index]
#     text1 = 'जानकारी मुताबिक जंगलों में पन्द्रह् फरवरी से फ'
#     text2 = 'आमतौर पर यहां के जंगलों में  सीजन पन्द्रह  फरवरी से शुरू होता है'
    #text1 = ['जानकारी', 'मुताबिक','जानकारी', 'जंगलों', 'पन्द्रह्', 'फरवरी', 'फायर', 'सीजन', 'शुरू','जानकारी', 'जंगलों', 'पन्द्रह्']
    #text2 = ['आमतौर', 'जंगलों', "'फायर'", 'सीजन', "''", 'पन्द्रह', 'फरवरी', 'शुरू']
    corpus = [text1, text2]
    vectorizer = TfidfVectorizer(min_df=1)
    vec_1 = vectorizer.fit_transform(corpus).toarray()[0]
    vec_2 = vectorizer.fit_transform(corpus).toarray()[1]
    sim.append(np.dot(vec_1, vec_2.T) / (np.linalg.norm(vec_1) * np.linalg.norm(vec_2)))
print(sim)



# In[11]:


data


# In[8]:


from nltk.tag import tnt
from nltk.corpus import indian
nltk.download('indian')
train_data = indian.tagged_sents('hindi.pos')
#print(train_data)
tagged_words_1=[]
```

```
tagged_words_2=[]

tnt_pos_tagger = tnt.TnT()
for index,i in tokenizedDF.iterrows():
    tnt_pos_tagger.train(train_data)
    text1=tokenizedDF[0].iloc[index]
    text2=tokenizedDF[1].iloc[index]
#text=['जानकारी', 'मुताबिक', 'जंगलों', 'पन्द्रह्', 'फरवरी', 'फायर', 'सीजन', 'शुरू']
    tagged_words_1.append(tnt_pos_tagger.tag(text1))
    tagged_words_2.append(tnt_pos_tagger.tag(text2))
print(tagged_words_1)


# In[9]:


FinalDataFrame = pd.DataFrame(
    {'Column1': tokenized0,
     'Column2': tokenized1,
     'IsParaphrased':paraphrased,
     'TaggedWordsCol1':tagged_words_1,
     'TaggedWordsCol2':tagged_words_2,
     'TF-IDF Score':sim,
     'Cosine Similarity':cosine,
     'N-gramCol1':newinput_list1,
     'N-gramCol2':newinput_list2,
     'Synonyms_Col1':Synonyms1,
     'Synonyms_Col2':Synonyms2
    })


# In[50]:


from pandas import ExcelWriter
from pandas import ExcelFile
writer = ExcelWriter('E:/NCI/Thesis/FinalDataFrame.xlsx')
```

```
FinalDataFrame.to_excel(writer,'Sheet1',index=False)
writer.save()


# In[3]:


FinalDataFrame = pd.read_excel("E:/NCI/Thesis/FinaldataFrame.xlsx")


# In[4]:


from sklearn.preprocessing import LabelEncoder
for column in FinalDataFrame.columns:
    if FinalDataFrame[column].dtype == type(object):
        le = LabelEncoder()
        FinalDataFrame[column] = le.fit_transform(FinalDataFrame[column])
FinalDataFrame.dtypes

# cols = FinalDataFrame.select_dtypes(exclude=['float']).columns

# FinalDataFrame[cols] = FinalDataFrame[cols].apply(pd.to_numeric, downcast='float', errors='coerce')
FinalDataFrame.dtypes


# In[5]:

#Data Modelling
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, recall_score, precision_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import AdaBoostClassifier
```

```
X = FinalDataFrame[['Column1', 'Column2', 'Cosine Similarity', 'N-gramCol1', 'N-gramCol2',
'Synonyms_Col1', 'Synonyms_Col2', 'TF-IDF Score', 'TaggedWordsCol1',
'TaggedWordsCol2']]
y = FinalDataFrame['IsParaphrased']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
AdaBoost = AdaBoostClassifier(base_estimator=
classifier,n_estimators=400,learning_rate=1)
AdaBoost = AdaBoostClassifier(n_estimators=400,learning_rate=1,algorithm='SAMME')
Ada = AdaBoost.fit(X_train,y_train)
prediction = Ada.predict(X_test)

# clf = classifier.fit(X_train, y_train)

# target_pred = clf.predict(X_test)

print(confusion_matrix(y_test, prediction))

accuracy_score = accuracy_score(y_test, prediction)
print(accuracy_score)

f1_score = f1_score(y_test, prediction, average='macro')
print(f1_score)

recall_score = recall_score(y_test, prediction, average='macro')
print(recall_score)

precision_score = precision_score(y_test, prediction, average='macro')
print(precision_score)

print('Average accuracy: %0.2f +/- (%0.1f) %%' % (accuracy_score.mean()*100,
accuracy_score.std()*100))
print('Average Precision: %0.2f +/- (%0.1f) %%' % (precision_score.mean()*100,
precision_score.std()*100))
print('Average Recall: %0.2f +/- (%0.1f) %%' % (recall_score.mean()*100,
recall_score.std()*100))
```

```
print('Average F1-Score: %0.2f +/- (%0.1f) %%' % (f1_score.mean()*100,
f1_score.std()*100))




# In[6]:


import matplotlib.pyplot as plt
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix, recall_score, precision_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X = FinalDataFrame[['Column1', 'Column2', 'Cosine Similarity', 'N-gramCol1', 'N-gramCol2',
'Synonyms_Col1', 'Synonyms_Col2', 'TF-IDF Score', 'TaggedWordsCol1',
'TaggedWordsCol2']]
y = FinalDataFrame['IsParaphrased']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

clf = DecisionTreeClassifier(random_state=0)

bdt = AdaBoostClassifier(DecisionTreeClassifier(max_depth=2), n_estimators=600,
learning_rate=1)

model_real = bdt.fit(X_train, y_train)

predict_real = model_real.predict(X_test)


print(confusion_matrix(y_test, predict_real))


accuracy_score = accuracy_score(y_test, predict_real)
```

```
print(accuracy_score)


f1_score = f1_score(y_test, predict_real, average='macro')
print(f1_score)


recall_score = recall_score(y_test, predict_real, average='macro')
print(recall_score)

precision_score = precision_score(y_test, predict_real, average='macro')
print(precision_score)


print('Average accuracy: %0.2f +/- (%0.1f) %%' % (accuracy_score.mean()*100, accuracy_score.std()*100))
print('Average Precision: %0.2f +/- (%0.1f) %%' % (precision_score.mean()*100, precision_score.std()*100))
print('Average Recall: %0.2f +/- (%0.1f) %%' % (recall_score.mean()*100, recall_score.std()*100))
print('Average F1-Score: %0.2f +/- (%0.1f) %%' % (f1_score.mean()*100, f1_score.std()*100))



# In[7]:


from sklearn.ensemble import VotingClassifier
import matplotlib.pyplot as plt
import numpy as np

Voting = VotingClassifier(estimators = [('knn', classifier), ('DT', clf)], voting = 'soft')

probas = [c.fit(X_train, y_train).predict_proba(X_test) for c in (classifier, clf, Voting)]
print(probas)

class1_1 = [pr[0, 0] for pr in probas]
class2_1 = [pr[0, 1] for pr in probas]
```

```
print(class1_1)
print(class2_1)

N = 3  # number of groups
ind = np.arange(N)  # group positions
width = 0.35  # bar width

fig, ax = plt.subplots()

# bars for classifier
p1 = ax.bar(ind, np.hstack(([class1_1[:-1], [0]])), width,
            color='green', edgecolor='k')
p2 = ax.bar(ind + width, np.hstack(([class2_1[:-1], [0]])), width,
            color='lightgreen', edgecolor='k')

# bars for VotingClassifier
p3 = ax.bar(ind, [0, 0, class1_1[-1]], width,
            color='blue', edgecolor='k')
p4 = ax.bar(ind + width, [0,  0, class2_1[-1]], width,
            color='steelblue', edgecolor='k')

# plot annotations
plt.axvline(1.8, color='k', linestyle='dashed')
ax.set_xticks(ind + width)
ax.set_xticklabels(['KNN', 'DecisionTree', 'VotingClassifier\n(average probabilities)'],
rotation=40, ha='right')



plt.ylim([0, 1])
plt.title('Class probabilities for sample 1 by different classifiers')
plt.legend([p1[0], p2[0]], ['class 1', 'class 2'], loc='upper left')
plt.tight_layout()
plt.show()
```