Experiment 1

Aim

Perform pre-processing of Text on any dataset

Theory

Text pre-processing is traditionally an important step for natural language processing (NLP) tasks. It transforms text into a more digestible form so that machine learning algorithms can perform better.

List of Text Pre-processing Steps

Based on the general outline above, we performed a series of steps under each component.

1. Remove HTML tags
2. Remove extra whitespaces
3. Convert accented characters to ASCII characters
4. Expand contractions
5. Remove special characters
6. Lowercase all texts
7. Convert number words to numeric form
8. Remove numbers
9. Remove stopwords
10. Lemmatization
11. Tokenisation
12. Stemming
13. Normalisation
14. POS Tagging

Remove HTML Tags

If the reviews or texts are web scraped, chances are they will contain some HTML tags. Since these tags are not useful for our NLP tasks, it is better to remove them.

Convert Accented Characters

Words with accent marks like “latté” and “café” can be converted and standardized to just “latte” and “cafe”, else the NLP model will treat “latté” and “latte” as different words even though they are referring to same thing.

Expand Contractions

Contractions are shortened words, e.g., don’t and can’t. Expanding such words to “do not” and “cannot” helps to standardize text.

Treatment for Numbers

One of the steps involve the conversion of number words to numeric form, e.g., seven to 7, to standardize text.Or you can also remove the numbers. Removing numbers may make sense for sentiment analysis since numbers contain no information about sentiments. However, if our NLP task is to extract the number of tickets ordered in a message to our chatbot, we will definitely not want to remove numbers.

Stopwords

Stopwords are very common words. Words like “we” and “are” probably do not help at all in NLP tasks such as sentiment analysis or text classifications. Hence, we can remove stopwords to save computing time and efforts in processing large volumes of text.

Lemmatization

Lemmatization is the process of converting a word to its base form, e.g., “caring” to “care”.

Tokenisation

It is about splitting strings of text into smaller pieces, or “tokens”. Paragraphs can be tokenized into sentences and sentences can be tokenized into words.

Stemming:

It is the process of reducing inflection in words (e.g. troubled, troubles) to their root form (e.g. trouble). The “root” in this case may not be a real root word, but just a canonical form of the original word.

Normalisation:

A highly overlooked preprocessing step is text normalization. Text normalization is the process of transforming a text into a canonical (standard) form. For example, the word “gooood” and “gud” can be transformed to “good”, its canonical form. Another example is mapping of near identical words such as “stopwords”, “stop-words” and “stop words” to just “stopwords”.

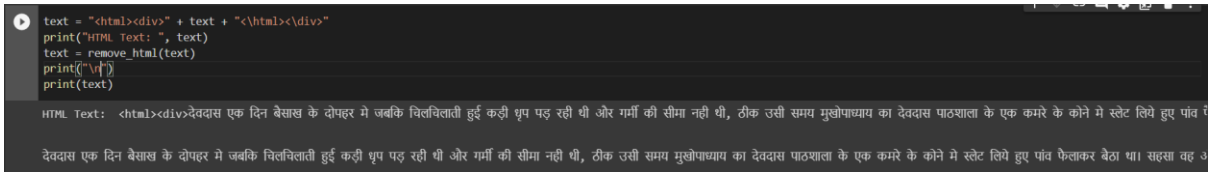
Parts of Speech Tagging

Understand parts of speech can make difference in determining the meaning of a sentence. Part of Speech (POS) often requires look at the proceeding and following words and combined with either a rule-based or stochastic method. It can than be combined with other processes for more feature engineering.

Output:

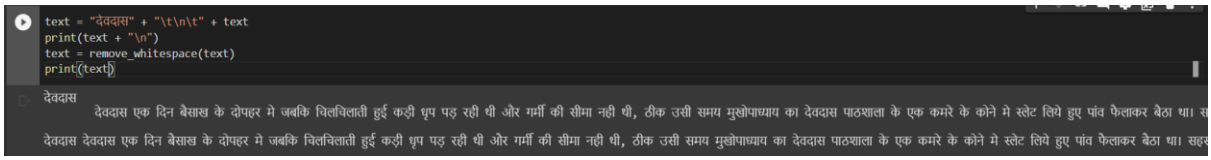
Remove HTML Tags

```
def remove_html(text):
  soup = BeautifulSoup(text, 'lxml')
  text = soup.get_text()
  return str(text)
```



Remove Whitespace

```
def remove_whitespace(text):
  text = ' '.join(text.split())
  return text
```



Accented to ASCII

```
def accented_to_ascii(text):
  try:
      text = unicode(text, 'utf-8')
  except (TypeError, NameError): # unicode is a default on python 3
      pass
  text = unicodedata.normalize('NFD', text)
  text = text.encode('ascii', 'ignore')
  text = text.decode("utf-8")
  return str(text)
```

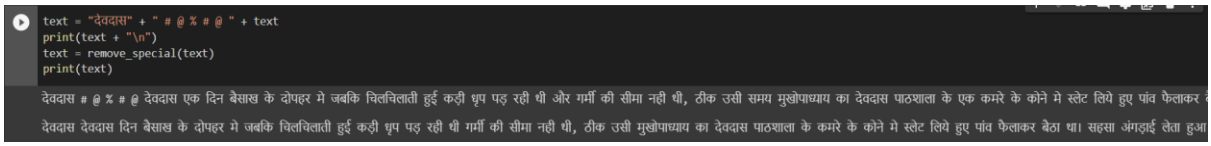
Expand Contractions

```
def expand_contractions(text):
  # N/A for Hindi
  expanded_words = []
  for word in text.split():
    expanded_words.append(contractions.fix(word))
  expanded_text = ' '.join(expanded_words)
  return expanded_text
```

Remove Special Characters

```
def remove_special(text):
  text = text.split()
  text = ' '.join(x for x in text if not x.isalnum())
  text = text.split()
  special_char_list = ["$", "@", "#", "&", "%"]
```

```
  text = " ".join([k for k in text if k not in special_char_list])
  text = ' '.join(text.split())
  return text
```



Text to Lowercase

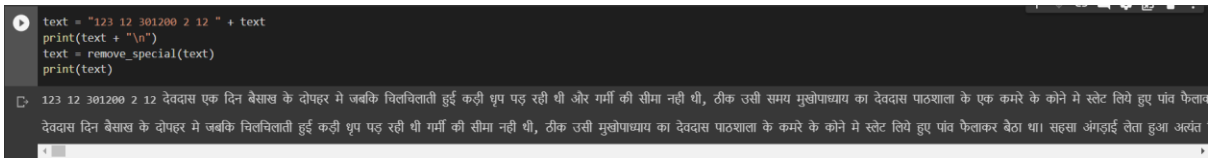
```
def text_to_lowercase(text):
  # N/A for Hindi
  text = text.lower()
  return text
```

Numerical Word to Number

```
def number_word_to_numeric(text):
  text = text.split()
  output = ""
  for i in text:
    try:
      res = w2n.word_to_num(i)
    except:
      res = i
    output += (str(res) + " ")
  output = output.rstrip()
  return output
```

Remove Number

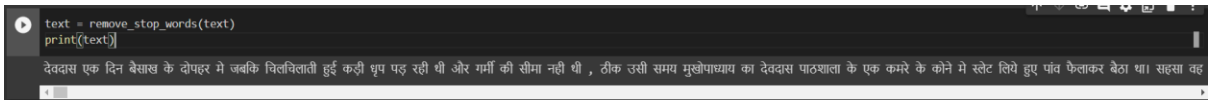
```
def remove_number(text):
  res = ' '.join([i for i in text if not i.isdigit()])
  return res
```



Remove Stopwords

```
def remove_stop_words(text):
  stop1 = open('drive/My Drive/SEM8/NLP/stopwords_1.txt')
  stop2 = open('drive/My Drive/SEM8/NLP/stopwords_2.txt')
```

```
  stop_words1 = []
  stop_words2 = []

  for x in stop1:
    stop_words1.append(x)

  for x in stop2:
    stop_words2.append(x)

  stop_words = stop_words1 + stop_words2
  stop_words = list(set(stop_words))

  word_tokens = word_tokenize(text)
  filtered_sentence = []

  for w in word_tokens:
      if w not in stop_words:
          filtered_sentence.append(w)

  filtered_sentence = ' '.join(filtered_sentence)
  return filtered_sentence
```



Lemmatization

```
def lemmatization(text):
  nlp = stanza.Pipeline(lang='hi', processors='tokenize, pos, lemma')
  doc = nlp(text)
  parsed_text = {'word':[], 'lemma':[]}
  for sent in doc.sentences:
      for wrd in sent.words:
          parsed_text['word'].append(wrd.text)
          parsed_text['lemma'].append(wrd.lemma)
  return pd.DataFrame(parsed_text)
```

```
lemm.head(15)
```

1 to 15 of 15 entries Filter

index	word	lemma
0	देवदास	देवदास
1	एक	एक
2	दिन	दिन
3	बैसाख	बैसाख
4	के	का
5	दोपहर	दोपहर
6	मे	मा
7	जबकि	जबकि
8	चिलचिलाती	चिलचिलाई
9	हुई	हो
10	कड़ी	कड़ा
11	धूप	धूप
12	पड़	पड
13	रही	रह
14	थी	था

Tokenization

```
def tokenization(text):
  tokenized_text = tokenize(text, 'hi')
  return tokenized_text
```

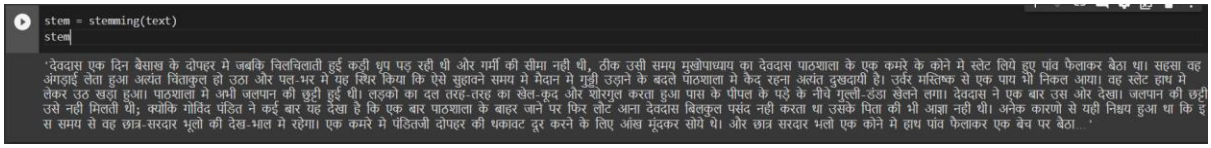
```
tok = tokenization(text)
t = pd.DataFrame()
t['token'] = tok
t
```

1 to 25 of 20000 entries Filter

index	token
0	_देवदास
1	_एक
2	_दिन
3	_बैसाख
4	_के
5	_दोपहर
6	_मे
7	_जबकि
8	_
9	चिल
10	चिल
11	ा
12	ती
13	_हुई
14	_कड़ी
15	_ध
16	ू
17	प
18	_पड़
19	_रही

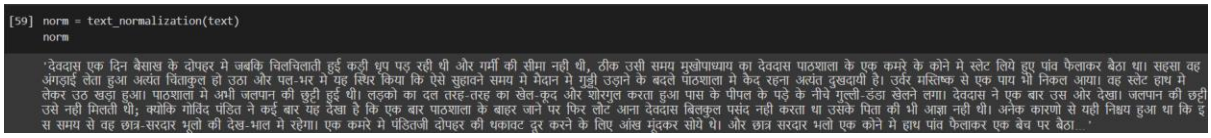
Stemming

```
def stemming(text):
  ps = PorterStemmer()
  text = text.split()
  output = ""
  for i in text:
    res = ps.stem(i)
    output += (str(res) + " ")
  return output
```



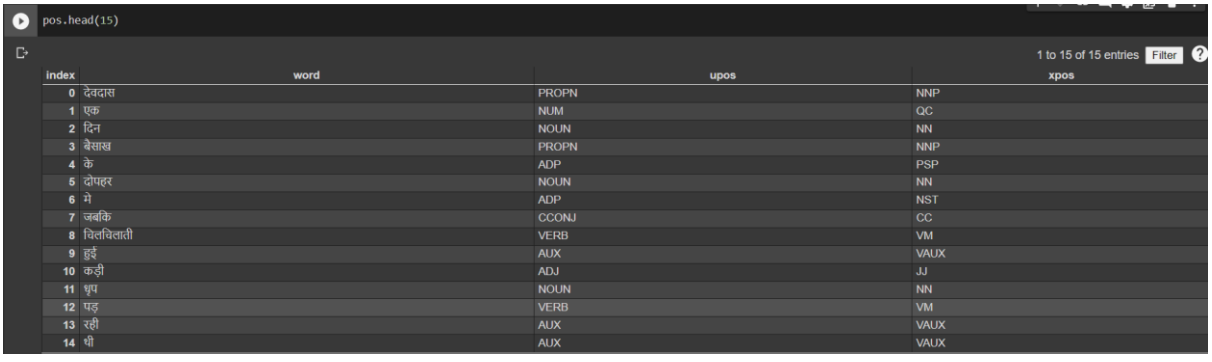
Text Normalization

```
def text_normalization(text):
  # remove_nuktas = False
  factory = IndicNormalizerFactory()
  normalizer = factory.get_normalizer("hi")
  text = normalizer.normalize(text)
  return text
```



POS

```
def pos(text):
  nlp = stanza.Pipeline(lang='hi', processors='tokenize, pos, lemma')
  doc = nlp(text)
  parsed_text = {'word':[], 'upos':[], 'xpos':[]}
  for sent in doc.sentences:
      for wrd in sent.words:
          parsed_text['word'].append(wrd.text)
          parsed_text['upos'].append(wrd.upos)
          parsed_text['xpos'].append(wrd.xpos)
  return pd.DataFrame(parsed_text)
```



index	word	upos	xpos
0	देवदास	PROPN	NNP
1	एक	NUM	QC
2	दिन	NOUN	NN
3	बैसाख	PROPN	NNP
4	के	ADP	PSP
5	दोपहर	NOUN	NN
6	मे	ADP	NST
7	जबकि	CCONJ	CC
8	चिलचिलाती	VERB	VM
9	हुई	AUX	VAUX
10	कड़ी	ADJ	JJ
11	धूप	NOUN	NN
12	पड़	VERB	VM
13	रही	AUX	VAUX
14	थी	AUX	VAUX

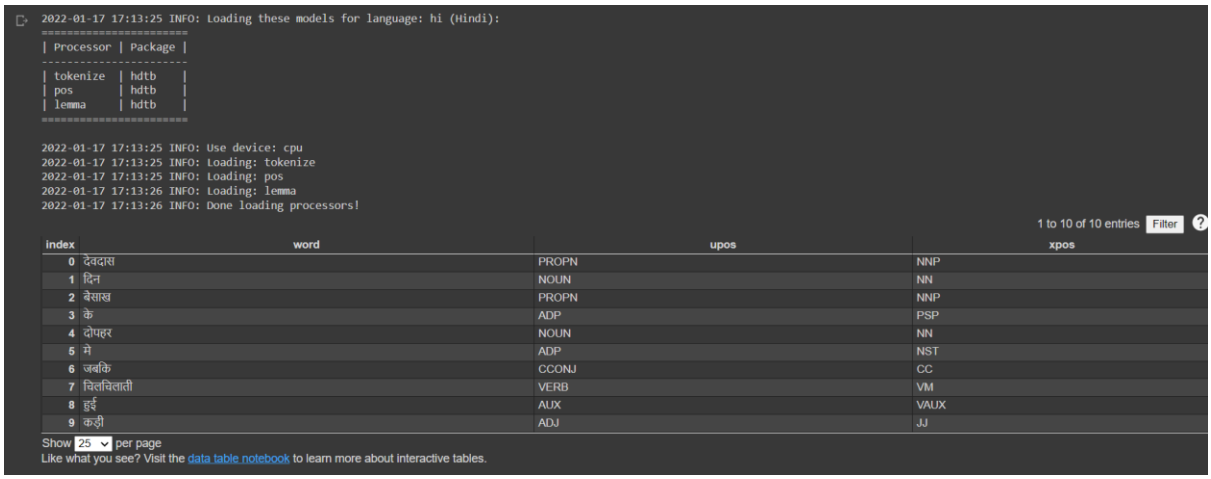
```
pos = pos(text)
pos
```

```
2022-01-17 16:41:38 INFO: Loading these models for language: hi (Hindi):
=======================
| Processor | Package |
-----------------------
| tokenize  | hdtb    |
| pos       | hdtb    |
| lemma     | hdtb    |
=======================

2022-01-17 16:41:38 INFO: Use device: cpu
2022-01-17 16:41:38 INFO: Loading: tokenize
2022-01-17 16:41:39 INFO: Loading: pos
2022-01-17 16:41:39 INFO: Loading: lemma
2022-01-17 16:41:39 INFO: Done loading processors!
```

Complete Execution

```
processed_text = remove_html(text)
processed_text = remove_whitespace(processed_text)
processed_text = remove_special(processed_text)
processed_text = remove_number(processed_text)
processed_text = remove_stop_words(processed_text)
processed_text = lemmatization(processed_text)
processed_text = tokenization(processed_text)
processed_text = stemming(processed_text)
processed_text = text_normalization(processed_text)
processed_text = pos(processed_text)
processed_text.head(10)
```

```
2022-01-17 17:02:09 INFO: Loading these models for language: hi (Hindi):
=======================
| Processor | Package |
-----------------------
| tokenize  | hdtb    |
| pos       | hdtb    |
| lemma     | hdtb    |
=======================

2022-01-17 17:02:09 INFO: Use device: cpu
2022-01-17 17:02:09 INFO: Loading: tokenize
2022-01-17 17:02:09 INFO: Loading: pos
2022-01-17 17:02:09 INFO: Loading: lemma
2022-01-17 17:02:09 INFO: Done loading processors!
```



```
2022-01-17 17:13:25 INFO: Loading these models for language: hi (Hindi):
=======================
| Processor | Package |
-----------------------
| tokenize  | hdtb    |
| pos       | hdtb    |
| lemma     | hdtb    |
=======================

2022-01-17 17:13:25 INFO: Use device: cpu
2022-01-17 17:13:25 INFO: Loading: tokenize
2022-01-17 17:13:25 INFO: Loading: pos
2022-01-17 17:13:26 INFO: Loading: lemma
2022-01-17 17:13:26 INFO: Done loading processors!
```

1 to 10 of 10 entries Filter

index	word	upos	xpos
0	देवदास	PROPN	NNP
1	दिन	NOUN	NN
2	बैसाख	PROPN	NNP
3	के	ADP	PSP
4	दोपहर	NOUN	NN
5	मे	ADP	NST
6	जबकि	CCONJ	CC
7	चिलचिलाती	VERB	VM
8	हुई	AUX	VAUX
9	कड़ी	ADJ	JJ

Show 25 per page

Like what you see? Visit the data table notebook to learn more about interactive tables.

Code

```
import nltk

# !pip install stanza
# !pip install indic-nlp-library
# !pip install indic-nlp-datasets
# !pip install inltk
# !pip install spacy
# !pip install contractions
# !pip install word2number
# nltk.download('punkt')
```

```
# nltk.download('wordnet')
# nltk.download('stopwords')
# nltk.download('indian')

from nltk.corpus import indian

nltk.corpus.indian.words("hindi.pos")
from indicnlp.normalize.indic_normalize import IndicNormalizerFactory

from idatasets import load_devdas
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
import spacy
import pandas as pd

# import stanfordnlp

from bs4 import BeautifulSoup
import unicodedata
import contractions
from word2number import w2n
import re
import stanza

from inltk.inltk import setup
from inltk.inltk import tokenize

# setup('hi')
# stanfordnlp.download('hi')
# stanfordnlp.download('en')
# stanza.download('hi')



def remove_html(text):
    soup = BeautifulSoup(text, "lxml")
    text = soup.get_text()
    return str(text)



def remove_whitespace(text):
    text = " ".join(text.split())
    return text



def accented_to_ascii(text):
    try:
        text = unicode(text, "utf-8")
    except (TypeError, NameError):  # unicode is a default on python 3
        pass
    text = unicodedata.normalize("NFD", text)
    text = text.encode("ascii", "ignore")
    text = text.decode("utf-8")
    return str(text)
```

```
def expand_contractions(text):
    # N/A for Hindi
    expanded_words = []
    for word in text.split():
        expanded_words.append(contractions.fix(word))
    expanded_text = " ".join(expanded_words)
    return expanded_text


def remove_special(text):
    text = text.split()
    text = " ".join(x for x in text if not x.isalnum())
    text = text.split()
    special_char_list = ["$", "@", "#", "&", "%"]
    text = " ".join([k for k in text if k not in special_char_list])
    text = " ".join(text.split())
    return text


def text_to_lowercase(text):
    # N/A for Hindi
    text = text.lower()
    return text


def number_word_to_numeric(text):
    text = text.split()
    output = ""
    for i in text:
        try:
            res = w2n.word_to_num(i)
        except:
            res = i
        output += str(res) + " "
    output = output.rstrip()
    return output


def remove_number(text):
    res = " ".join([i for i in text if not i.isdigit()])
    return res


def remove_stop_words(text):
    stop1 = open("drive/My Drive/College/stopwords_1.txt")
    stop2 = open("drive/My Drive/College/stopwords_2.txt")

    stop_words1 = []
    stop_words2 = []

    for x in stop1:
        stop_words1.append(x)

    for x in stop2:
        stop_words2.append(x)

    stop_words = stop_words1 + stop_words2
```

```
    stop_words = list(set(stop_words))

    word_tokens = word_tokenize(text)
    filtered_sentence = []

    for w in word_tokens:
        if w not in stop_words:
            filtered_sentence.append(w)

    filtered_sentence = " ".join(filtered_sentence)
    return filtered_sentence



def lemmatization(text):
    nlp = stanza.Pipeline(lang="hi", processors="tokenize, pos, lemma")
    doc = nlp(text)
    parsed_text = {"word": [], "lemma": []}
    for sent in doc.sentences:
        for wrd in sent.words:
            parsed_text["word"].append(wrd.text)
            parsed_text["lemma"].append(wrd.lemma)
    return pd.DataFrame(parsed_text)



def tokenization(text):
    tokenized_text = tokenize(text, "hi")
    return tokenized_text



def stemming(text):
    ps = PorterStemmer()
    text = text.split()
    output = ""
    for i in text:
        res = ps.stem(i)
        output += str(res) + " "
    return output



def text_normalization(text):
    # remove_nuktas = False
    factory = IndicNormalizerFactory()
    normalizer = factory.get_normalizer("hi")
    text = normalizer.normalize(text)
    return text



def pos(text):
    nlp = stanza.Pipeline(lang="hi", processors="tokenize, pos, lemma")
    doc = nlp(text)
    parsed_text = {"word": [], "upos": [], "xpos": []}
    for sent in doc.sentences:
        for wrd in sent.words:
            parsed_text["word"].append(wrd.text)
            parsed_text["upos"].append(wrd.upos)
            parsed_text["xpos"].append(wrd.xpos)
    return pd.DataFrame(parsed_text)
```

```
text = load_devdas()
paragraphs = list(text.data)
text = " ".join(paragraphs)
text
```

Conclusion

We have successfully performed text pre-processing tasks on a given piece of non-English text.