**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)

Text

Description automatically generated

**Remove Whitespace**

def remove\_whitespace(text):

  text = ' '.join(text.split())

  return text

Graphical user interface, text, application

Description automatically generated**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())

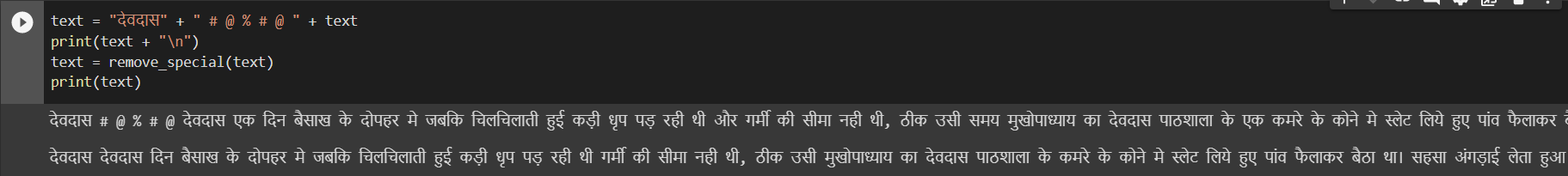
  text = text.split()

  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

Graphical user interface, text

Description automatically generated

**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 = []

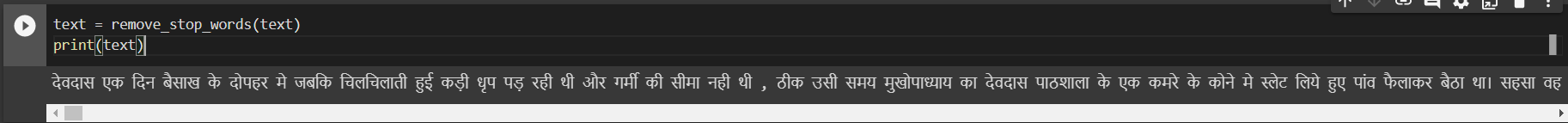
  for w in word\_tokens:

      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)

Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated

**Tokenization**

def tokenization(text):

  tokenized\_text = tokenize(text, 'hi')

  return tokenized\_text

A screenshot of a computer

Description automatically generated with medium confidence

**Stemming**

def stemming(text):

  ps = PorterStemmer()

  text = text.split()

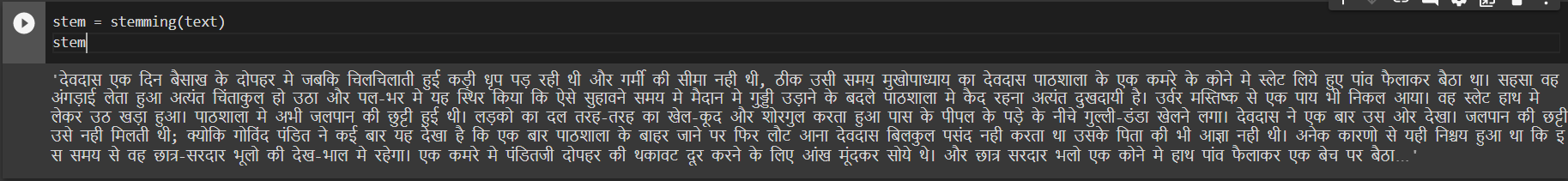
  output = ""

  for i in text:

    res = ps.stem(i)

    output += (str(res) + " ")

  return output



**Text Normalization**

def text\_normalization(text):

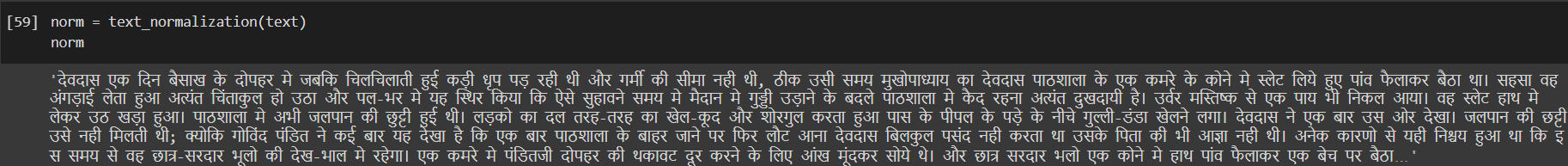
  # remove\_nuktas = False

  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)

A screenshot of a computer

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Complete Execution**

Text

Description automatically generated with medium confidence

**Graphical user interface, text, application

Description automatically generated**

**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.