1 - Pre_processing

November 30, 2021

1 Process Overview

2 Terminology

- Tokenization:
  - A sentence contain be broken down in elements or units of semantics. These are referred to as tokens. Tokens can be words, numbers, symbols, punctuation marks, etc. The process of generating tokens is called tokenization.
- Stop words:
  - Stop words are the most common words in a natural language. For the purposes of analyzing text data, stop words do not, generally, add value and are removed from the corpus.
- Stemming:
  - Stemming is a rudimentary rule-based process of stripping the suffixes (“ing”, “ly”, “es”, “s” etc) from a word.
- Lemmatization:
  - Lemmatization is an organized & step by step procedure of obtaining the root form of the word, it makes use of vocabulary (dictionary importance of words) and morphological analysis (word structure and grammar relations).

[ ]:

[ ]:
```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from textblob import TextBlob
from textblob import Word
```

[ ]:
```
blob = TextBlob('Do not taste or eat raw dough or batter that is meant to be␣
 ↪baked or cooked. This includes dough or batter for cookies, cakes, pies,␣
 ↪biscuits, pancakes, tortillas, pizza, or crafts. Do not let children taste␣
 ↪raw dough or batter or play with dough at home or in restaurants.')
print(blob.words)
print(' ')
print(blob.sentences)
```

[ ]:
```
blob = TextBlob("Spellling whil typing is hardd for me")
blob_corrected  = blob.correct()
print(blob_corrected.string)
```

[ ]:
```
nlp_word = Word("disease")
print(nlp_word.definitions)
```

2.1 A quick overview of common pre-processing tasks

[ ]:
```
import os    # This is the python built-in version
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
import re    # This is the python built-in version of regex
#nltk.data.path.append(os.path.join(os.getcwd(), "nltk_data"))

# Sample text - from CDC website
text = 'Minnesota officials found E. coli O157:H7 in a package of leftover␣
 ↪Josie's Organics baby spinach collected from a sick person's home. Five␣
 ↪people in this outbreak reported eating spinach in the week before they got␣
 ↪sick and 1 reported Josie's Organics brand.'
print('Original Text --------------------',text)
print(' ')
print(' ')
print('Length of Text ------------------',len(text))
print(' ')
print(' ')

from nltk.tokenize import word_tokenize
# Tokenization in NLP is the process by which a large quantity of text is␣
 ↪divided
# into smaller parts called tokens.

# Split text into words using NLTK
words = word_tokenize(text)
print('Tokenized words --------------------',words)
print(' ')
print(' ')

from nltk.tokenize import sent_tokenize

# Split text into sentences
sentences = sent_tokenize(text)
print('Tokenized sentences --------------------',sentences)
print(' ')
print(' ')
```

```
# List stop words - Words that do not contribute to the meaning of a phrase
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
print('stop words --------------------',stop_words)
print(' ')
print(' ')
print('no stopwords --------------------',stopwords.words("english"))
print(' ')
print(' ')


# Reset text
text = "What is public health? It's about more than just responding to disease␣
 ↪outbreaks. The standard definition of public health used for over 100 year␣
 ↪was developed by C.-E.A. Winslow. Is this a good definition to keep using?"


# Normalize it
text = re.sub(r"[^a-zA-Z0-9]", " ", text.lower())

# Tokenize it
words = text.split()
print('lowercase & tokenized --------------------',words)
print(' ')
print(' ')

# Remove stop words
words = [w for w in words if w not in stopwords.words("english")]

print('no stop words --------------------',words)
print(' ')
print(' ')

from nltk.stem.porter import PorterStemmer

# different forms of the same "word"
input1 = 'List listed lists listing listings'
words1 = input1.lower().split(' ')
words1
porter = nltk.PorterStemmer()
print([porter.stem(t) for t in words1])
print(' ')
print(' ')

# Reduce words to their stems
stemmed = [PorterStemmer().stem(w) for w in words]
```

```
print('stemmed -------------------',stemmed)
print(' ')
print(' ')

from nltk.stem.wordnet import WordNetLemmatizer

# Reduce words to their root form
lemmed = [WordNetLemmatizer().lemmatize(w) for w in words]
print('lemmed -------------------',lemmed)
print(' ')
print(' ')

# Lemmatize verbs by specifying pos
lemmed = [WordNetLemmatizer().lemmatize(w, pos='v') for w in lemmed]
print('lemmed verbs -------------------',lemmed)
print(' ')
print(' ')
```

2.2 Objective

This notebook is intended to demonstrate some of the typical pre-processingthat occurs in NLP. The dataset constructed will have 2 columns - a drug review and a good/bad rating. Our assumption for a problem statement is: Can we determine the rating a person will apply based on the text in their review. As such, it is a classification problem.

2.3 Pre-requisite work

1. In a browser, navigate to https://archive-beta.ics.uci.edu/
2. Download the Drug Review Dataset (Drugs.com)

The dataset provides patient reviews on specific drugs along with related conditions and a 10 star patient rating reflecting overall patient satisfaction.

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

2.3.1 Load the data into a dataframe

[ ]:
```
import pandas as pd
```

[ ]:
```
latuda = pd.read_csv('https://raw.githubusercontent.com/jimcody2014/nlp_cdc/
↪main/data/latuda.csv')
```

[ ]:
```
latuda.columns
```

2.4 Pre-processing

```
[ ]: !pip install contractions
!pip install pyspellchecker

import contractions
#from pyspellchecker import SpellChecker
import string
import re

import nltk
#nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
```

```
[ ]: # Add a column 'target' based on the rating column
latuda['target'] = latuda['rating'].apply(lambda x: 'Good' if x >= 6 else 'Bad')
latuda.columns
```

```
[ ]: # Drop columns
drop_columns = {'drugName', 'condition','date', 'usefulCount', 'rating'}
latuda = latuda.drop(columns = drop_columns)
```

2.4.1 Contractions

```
[ ]: latuda['remove_ctr'] = latuda['review'].apply(lambda x: [contractions.fix(word)␣
 ↪for word in x.split()])
latuda.head(25)
```

```
[ ]: # change no_contract back to a string
latuda["review_new"] = [' '.join(map(str, l)) for l in latuda['remove_ctr']]
latuda.head()
```

An alternative - Regular Expressions

```
[ ]: latuda2 = pd.read_csv('https://raw.githubusercontent.com/jimcody2014/nlp_cdc/
 ↪main/data/latuda.csv')
drop_columns = {'drugName', 'condition','date', 'usefulCount', 'rating'}
latuda2 = latuda2.drop(columns = drop_columns)

#import re
#import string

def clean_text_round1(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\w*\d\w*', '', text)
```

```
    text = re.sub('[‘’""…]', '', text)
    text = re.sub('\n', '', text)
    return text

round1 = lambda x: clean_text_round1(x)

# Let's take a look at the updated text
data_clean = pd.DataFrame(latuda2.review.apply(round1))
data_clean
```

2.4.2 Tokenization

[ ]:
```
sample = 'This is a sentence ready to be tokenized'
print(word_tokenize(sample))
```

[ ]:
```
sample = 'This is a paragraph ready to be tokenized.  It has two sentences.'
print(sent_tokenize(sample))
```

[ ]:
```
latuda.columns
```

[ ]:
```
latuda['tokenized'] = latuda['review_new'].apply(word_tokenize)
latuda.head()
```

2.4.3 Noise cleaning (spacing, lowercase, special characters)

[ ]:
```
latuda['lower'] = latuda['tokenized'].apply(lambda x: [word.lower() for word in␣
↪x])
latuda.head()

# This could have been done before the data was tokenized.
# latuda['lower'] = latuda['review'].str.lowercase()
```

2.4.4 Use the string package

[ ]:
```
print(string.punctuation)
```

[ ]:
```
punc = string.punctuation
latuda['lower'] = latuda['lower'].apply(lambda x: [word for word in x if word␣
↪not in punc])
latuda.head()
```

[ ]:
```
def clean_text_round1(text):
    #text = text.lower()
    #text = re.sub('\[.*?\]', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub('[‘’""…]', '', text)
```

```
    text = re.sub('\n', '', text)
    return text

round1 = lambda x: clean_text_round1(x)

latuda.review = pd.DataFrame(latuda.review.apply(round1))
latuda.head()
```

2.4.5 Stop words

[ ]:
```
# Import the nltk stopwords library and sewt it to english

nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

latuda['no_stopwords'] = latuda['lower'].apply(lambda x: [word for word in x if␣
↪word not in stop_words])
latuda.head()
```

2.4.6 Stemming/ Lemmanization

[ ]:
```
nltk.download('averaged_perceptron_tagger')
```

[ ]:
```
latuda['pos_tags'] = latuda['no_stopwords'].apply(nltk.tag.pos_tag)
latuda.head()
```

[ ]:
```
nltk.download('wordnet')
from nltk.corpus import wordnet
```

[ ]:
```
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

[ ]:
```
latuda['wordnet_pos'] = latuda['pos_tags'].apply(lambda x: [(word,␣
↪get_wordnet_pos(pos_tag)) for (word, pos_tag) in x])
latuda.head()
```

[ ]:
```
from nltk.stem import WordNetLemmatizer

# There are multiple nltk stemmer methods : PorterStemmer(),␣
↪SnowballStemmer('english')
```

[ ]:
```
wnl = WordNetLemmatizer()
latuda['lemmatized'] = latuda['wordnet_pos'].apply(lambda x: [wnl.
↪lemmatize(word, tag) for word, tag in x])
latuda.head()
```