Parser

November 4, 2019

[2]:
```
import numpy as np
import pandas as pd
import nltk
import string
```

[112]:
```
text = input()
```

```
cd boy
```

1 Preprocessing

Hence in the Preprocessing phase we do the following in the order below:-

1. Begin by removing the html tags
2. Remove any punctuations or limited set of special characters like , or . or # etc.
3. Check if the word is made up of english letters and is not alpha-numeric
4. Convert the word to lowercase
5. Finally we apply Lemmetizing approach.

After which we collect the words by parser.

[113]:
```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won\'t", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
```

```
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

[131]:
```
# Combining all the above stundents
sent = "this is a #sample @sentence"
sent = re.sub(r"http\S+", "", sent)
sent = decontracted(sent)
sent = re.sub("\S*\d\S*", "", sent).strip()
sent = re.sub('[^A-Za-z1-9]+', ' ', sent) # To check if everything is alpha␣
↪numeric
```

[132]:
```
from nltk.tokenize import word_tokenize
#nltk.download('punkt')
sent = word_tokenize(sent)
sent
```

[132]:
```
['this', 'is', 'a', 'sample', 'sentence']
```

2 Lemmatization

[116]:
```
from nltk import pos_tag
from nltk.corpus import wordnet
from nltk import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

[142]:
```
def get_simple_pos(tag):
  if tag.startswith('J'):
    return wordnet.ADJ
  elif tag.startswith('N'):
    return wordnet.NOUN
  elif tag.startswith('V'):
    return wordnet.VERB
  elif tag.startswith('R'):
    return wordnet.ADV
  else:
    return wordnet.NOUN
```

[143]:
```
# nltk.download('averaged_perceptron_tagger')
# nltk.download('wordnet')
def clean_review(words):
    output_words = []
    for w in words:
        pos = pos_tag([w])
        clean_word = lemmatizer.lemmatize(w , pos = get_simple_pos(pos[0][1]))
        output_words.append(clean_word.lower())
```

```
    output_words = ' '.join(word for word in output_words)
    return output_words
```

[133]:
```
documents = clean_review(sent)
documents
```

[133]:
```
'this be a sample sentence'
```

[ ]:
```

```

3 Object Standardization

[167]:
```
lookup_dict = {'rt':'Retweet', 'dm':'direct message', "awsm" : "awesome", "luv"␣
 ↪:"love", "cooool" : "cool", "asap":"as soon as possible", "2morrow":␣
 ↪"tomorrow", "u": "you", "ur":"your", "xam": "exam","bday":"birthday", "r8":␣
 ↪"right", "gr8":"great", "btw": "by the way", "gn":"good night", "ttyl":␣
 ↪"talk to you latter", "r": "are", "gr8": "great", "2mrw": "tomorrow", "2day":
 ↪ "today", "dbt":"doubt"}
def standardize(input_text):
    words = input_text.split()
    new_words = []
    for word in words:
        if word.lower() in lookup_dict:
            word = lookup_dict[word.lower()]
        new_words.append(word)
    new_text = " ".join(new_words)
    return new_text
```

[151]:
```
standardize("RT this is a retweeted tweet by Mr.X")
```

[151]:
```
'Retweet this is a retweeted tweet by Mr.X'
```

4 Spell Checker

[122]:
```
import rShivam Bansale
from collections import Counter

def words(text): return re.findall(r'\w+', text.lower())

WORDS = Counter(words(open('big.txt', encoding='utf-8').read()))

def P(word, N=sum(WORDS.values())):
    "Probability of `word`."
```

```
    return WORDS[word] / N

def correction(word):
    "Most probable spelling correction for word."
    return max(candidates(word), key=P)

def candidates(word):
    "Generate possible spelling corrections for word."
    return (known([word]) or known(edits1(word)) or known(edits2(word)) or␣
 ↪[word])

def known(words):
    "The subset of `words` that appear in the dictionary of WORDS."
    return set(w for w in words if w in WORDS)

def edits1(word):
    "All edits that are one edit away from `word`."
    letters    = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(word[:i], word[i:])    for i in range(len(word) + 1)]
    deletes    = [L + R[1:]               for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces   = [L + c + R[1:]           for L, R in splits if R for c in␣
 ↪letters]
    inserts    = [L + c + R               for L, R in splits for c in letters]
    return set(deletes + transposes + replaces + inserts)

def edits2(word):
    "All edits that are two edits away from `word`."
    return (e2 for e1 in edits1(word) for e2 in edits1(e1))
```

[123]:
```
correction('speling')
```

[123]: 'spelling'

[124]:
```
correction('korrectud')
```

[124]: 'corrected'

[159]:
```
def preprocess(text):
    # Combining all the above stundents
    text = re.sub(r"http\S+", "", text)
    text = decontracted(text)
    text = re.sub('[^A-Za-z1-9]+', ' ', text) # To check if everything is alpha␣
 ↪numeric

    # code for standardization
    text = standardize(text)
```

```
    # Tokenize
    text = word_tokenize(text)

    # code for Lemmatization
    text = clean_review(text)

    text = word_tokenize(text)

    # Code for spell checker
    parsed_text = []
    for word in text:
        pos = pos_tag([word])
        if get_simple_pos(pos[0][1]) is wordnet.NOUN:
            parsed_text.append(word)
        else:
            parsed_text.append(correction(word))
    return parsed_text
```

[169]:
```
result = preprocess("It's really gr8 2mrw is ur xam, dm me if u have any dbt,␣
↪ttyl, gn")
```

[173]:
```
print(result)
```

```
['it', 'is', 'really', 'great', 'tomorrow', 'is', 'your', 'exam', 'direct',
'message', 'me', 'if', 'you', 'have', 'any', 'doubt', 'talk', 'to', 'you',
'latter', 'good', 'night']
```

[ ]:
```

```