Processing using NLTK

Uzair Ahmad

What is text preprocessing in NLP and why it is important?

Text preprocessing is a first step in any text based classification. In large corpus, text comes from various resources which contain noise along with valueable information. If we use unprocessed data directly with the models, it model will peform badly and give unpredictable result.

Steps to install NLTK

Mac/Unix

From the terminal:

1. Install NLTK: run `pip install -U nltk`
2. Test installation: run `python` then type `import nltk`

Windows

1. Install NLTK: http://pypi.python.org/pypi/nltk
2. Test installation: `Start>Python35`, then type `import nltk`

Download NLTK data

```
import nltk
nltk.download()
```

1) Convert all words to lower case

Lower and upper case for same words is treated as different by the models, so we need to convert all words to lower case or uppper case

```
def toLowerCase(text):
    return text.lower() #changes all upper case alphabet to lower case
```

```
text = 'Did you catch the bus ? Are you frying an egg ?'
print(f'Before - {text}')
print(f'After  - {toLowerCase(text)}')

```

```
Before - Did you catch the bus ? Are you frying an egg ?
After  - did you catch the bus ? are you frying an egg ?
```

2) Removal of URLs

Regular expression or RegEx in Python is denoted as RE (REs, regexes or regex pattern) are imported through re module. The functions in this module let you check if a particular string matches a given regular expression.

```
import re
def removeURLs(text):

    text = re.sub(r"http\S+", "", text) # replaces URLs starting with http
    text = re.sub(r"www.\S+", "", text) # replaces URLs starting with wwe
    return text
```

```
text = 'https://thistutorialisawesome.com is best site'
print(f'Before - {text}')
print(f'After  - {removeURLs(text)}')
```

```
Before - https://thistutorialisawesome.com is best site
After  -  is best site
```

3) Removal of punctuations

Punctuation don't add any value to overall text and can be removed without impacting

```
import string
string.punctuation # checking punctuations
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
def removePunctuation(text):
    return "".join([char for char in text if char not in string.punctuation])
```

```
text = 'Is that seriously how you spell my name? I am not joking.'
print(f'Before - {text}')
print(f'After  - {removePunctuation(text)}')
```

```
Before - Is that seriously how you spell my name? I am not joking.
After  - Is that seriously how you spell my name I am not joking
```

4) Removal of stopwords

In sentiment analysis, stopwords like pronouns can overweight actual words expressing emotions, which leads to poor for perfomance of model. So stopwords should be removed as a part of text preprocessing

```python
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stopword = nltk.corpus.stopwords.words('english')
print(stopword[0:500:25])
```

```
['i', 'herself', 'been', 'with', 'here', 'very', 'doesn', 'won']
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/afsarequebal/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```python
def removeStopwords(text):
    return " ".join([word for word in re.split('\W+', text)
        if word not in stopword])
```

```python
text = 'Is that seriously how you spell my name? I am not joking.'
print(f'Before - {text.lower()}')
print(f'After  - {removeStopwords(text.lower())}')
```

```
Before - is that seriously how you spell my name? i am not joking.
After  - seriously spell name joking
```

5) Tokenization - It is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.

NLTK contains a module called tokenize() which further classifies into two sub-categories: Word tokenize: We use the word_tokenize() method to split a sentence into tokens or words. Sentence tokenize: We use the sent_tokenize() method to split a document or paragraph into sentences.

```python
nltk.download('punkt')
from nltk.tokenize import sent_tokenize, word_tokenize
def sentenceTokenize(text):
    return sent_tokenize(text)

```

```
def wordTokenize(text):
    return word_tokenize(text)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/afsarequebal/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
text = 'Its a wonderful day in Boston. I really like NLP. I will be visting some nearby park today.'
print(f'Before - {text}')
print(f'After  - {sentenceTokenize(text.lower())}')
```

```
Before - Its a wonderful day in Boston. I really like NLP. I will be visting some nearby
After  - ['its a wonderful day in boston.', 'i really like nlp.', 'i will be visting som
```

```
text = 'Its a wonderful day in Boston. I really NLP. I will be visting some nearby park today.'
print(f'Before - {text}')
print(f'After  - {wordTokenize(text.lower())}')
```

```
Before - Its a wonderful day in Boston. I really NLP. I will be visting some nearby park
After  - ['its', 'a', 'wonderful', 'day', 'in', 'boston', '.', 'i', 'really', 'nlp', '.'
```

6) Stemming

Stemming is the process of reducing words to their stem/roots. It help in removing redundant words. For eg. root of connect, connected, connecting, connection is same. We will be using NLTK to perform this task

```
nltk.download('wordnet')
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
wn = nltk.WordNetLemmatizer()
ps = nltk.PorterStemmer()
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/afsarequebal/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
def performStemming(text):
    return" ".join([ps.stem(word) for word in re.split('\W+', text)])
```

```
text = 'It am troubled by riding bicyle daily. I have no troubles taking a ride in lyft to college.'
print(f'Before - {text}')
print(f'After  - {performStemming(text.lower())}')
```

```
Before - It am troubled by riding bicyle daily. I have no troubles taking a ride in lyft
After  - it am troubl by ride bicyl daili i have no troubl take a ride in lyft to colleg
```

7) Lemmatization

It is similar to stemming except the lemmatized word belongs to the language. It also allows us to specify verb or noun to be used as parameter.

```
def performLemmatization(text):
    return" ".join([wn.lemmatize(word,'v') for word in re.split('\W+', text)])
```

```
text = 'It am troubled by riding bicyle daily. I have no troubles taking a ride in lyft to college.'
print(f'Before - {text}')
print(f'After  - {performLemmatization(text.lower())}')
```

```
Before - It am troubled by riding bicyle daily. I have no troubles taking a ride in lyft
After  - it be trouble by rid bicyle daily i have no trouble take a ride in lyft to coll
```

Exercise:

Write a Python program using NLTK to:

1. Tokenize the following text into words:

I am a fourth-year bachelor's student in Data Science and AI, an exciting field that combines cutting-edge technology with the power of data to solve real-world problems!

2. Remove stopwords from the tokenized words.
3. Apply **stemming** to the filtered words using the Porter Stemmer.
4. Apply **lemmatization** to the filtered words..
5. Explain the difference between stemming and lemmatization using your results.