GradProject

May 24, 2023

[ ]:
```
Dutch = 'https://www.gutenberg.org/cache/epub/39181/pg39181-images.html'
English = "https://www.gutenberg.org/cache/epub/1004/pg1004-images.html"
Finnish = "https://www.gutenberg.org/cache/epub/12546/pg12546.html"
German = "https://www.gutenberg.org/cache/epub/8085/pg8085.html"
Italian = "https://www.gutenberg.org/cache/epub/1000/pg1000-images.html"
Spanish = "https://www.gutenberg.org/cache/epub/57303/pg57303-images.html"

languages = [Dutch, German, Italian, English, Spanish, Finnish]
names = ['Dutch', 'German', 'Italian', 'English', 'Spanish', 'Finnish']
```

[ ]:
```
import gensim
from gensim.models import word2vec
import nltk
from nltk.tokenize import sent_tokenize, word_tokenize
import gensim.downloader as api
import numpy as np
from urllib.request import urlopen
import re
from nltk.corpus import stopwords
import pandas as pd
from nameof import nameof
```

[ ]:
```
def data_tokenizer(url, language, encoding = 'utf-8'):
    with urlopen(url) as file:
        divine_comedy = file.read().decode(encoding)

    f = divine_comedy.replace("\n", " ")
    f = re.sub(r'[^\w\s]', '', f)
    data = []
    for i in sent_tokenize(f):
        temp = []

    # tokenize the sentence into words
        for j in word_tokenize(i):
            temp.append(j.lower())

        data.append(temp)
    stop_words = set(stopwords.words(language))
    stopped = [[i for i in j if i not in stop_words] for j in data]
    return stopped
```

[ ]:
```
var_holder = {}
for i in languages:
     langname = names[languages.index(i)]
     var_holder['tokenized_' + langname]= data_tokenizer(i, langname)

locals().update(var_holder)
```

[ ]:
```
tokenized_languages = [tokenized_Dutch, tokenized_German, tokenized_Italian,␣
 ↪tokenized_English, tokenized_Spanish, tokenized_Finnish]
tokenized_names = 'tokenized_'+pd.Series(names)
```

[ ]:
```
for tk in tokenized_languages:
    print(tokenized_names[tokenized_languages.index(tk)],": ", tk[0][:10])
```

```
tokenized_Dutch :  ['doctype', 'html', 'html', 'langnl', 'head', 'meta',
'charsetutf8style', 'pgheader', 'div', 'pgfooter']
tokenized_German :  ['doctype', 'html', 'html', 'langdeheadmeta',
'charsetutf8style', 'pgheader', 'div', 'pgfooter', 'div', 'all']
tokenized_Italian :  ['doctype', 'html', 'html', 'langit', 'head', 'meta',
'charsetutf8style', 'pgheader', 'div', 'pgfooter']
tokenized_English :  ['doctype', 'html', 'html', 'langen', 'head', 'meta',
'charsetutf8style', 'pgheader', 'div', 'pgfooter']
tokenized_Spanish :  ['doctype', 'html', 'html', 'langes', 'head', 'meta',
'charsetutf8style', 'pgheader', 'div', 'pgfooter']
tokenized_Finnish :  ['doctype', 'html', 'html', 'langfiheadmeta',
'charsetutf8style', 'pgheader', 'div', 'pgfooter', 'div', 'all']
```

[ ]:
```
for tk in tokenized_languages:
    print(len(tk[0]))
```

```
44471
57427
71994
74499
71623
90944
```

[ ]:
```
def skipgram(language):
    return gensim.models.Word2Vec(language, min_count = 1,
                                  vector_size = 100,sg = 1).wv
def cbow(language):
    return gensim.models.Word2Vec(language, min_count = 1,
                                  vector_size = 100, sg = 0).wv
```

```
[ ]: skipgram_English = skipgram(tokenized_English)
     cbow_English = cbow(tokenized_English)
     skipgram_Dutch = skipgram(tokenized_Dutch)
     cbow_Dutch = cbow(tokenized_Dutch)
     skipgram_German = skipgram(tokenized_German)
     cbow_German = cbow(tokenized_German)
     skipgram_Italian = skipgram(tokenized_Italian)
     cbow_Italian = cbow(tokenized_Italian)
     skipgram_Spanish = skipgram(tokenized_Spanish)
     cbow_Spanish = cbow(tokenized_Spanish)
     skipgram_Finnish = skipgram(tokenized_Finnish)
     cbow_Finnish = cbow(tokenized_Finnish)
```

```
[ ]: print(skipgram_English.index_to_key[:10])
     print(cbow_English.index_to_key[:10])
```

```
['p', 'classnoindent', 'thou', 'one', 'unto', 'upon', 'thee', 'thy', 'said',
'us']
['p', 'classnoindent', 'thou', 'one', 'unto', 'upon', 'thee', 'thy', 'said',
'us']
```

```
[ ]: print(skipgram_Dutch.index_to_key[:10])
     print(cbow_Dutch.index_to_key[:10])
```

```
['p', 'classregel', 'classp2a', 'den', 'classp3a', 'a', 'pginternal',
'classnoot', 'gij', 'div']
['p', 'classregel', 'classp2a', 'den', 'classp3a', 'a', 'pginternal',
'classnoot', 'gij', 'div']
```

```
[ ]: print(skipgram_German.index_to_key[:10])
     print(cbow_German.index_to_key[:10])
```

```
['p', 'sprach', 'sah', 'drum', 'the', 'wohl', 'schon', 'mehr', 'gleich', 'wer']
['p', 'sprach', 'sah', 'drum', 'the', 'wohl', 'schon', 'mehr', 'gleich', 'wer']
```

```
[ ]: print(skipgram_Italian.index_to_key[:10])
     print(cbow_Italian.index_to_key[:10])
```

```
['p', 'sì', 'de', 'quel', 'me', 'così', 'poi', 'là', 'quando', 'già']
['p', 'sì', 'de', 'quel', 'me', 'così', 'poi', 'là', 'quando', 'già']
```

```
[ ]: print(skipgram_Spanish.index_to_key[:10])
     print(cbow_Spanish.index_to_key[:10])
```

```
['si', 'div', 'classpagenuma', 'tan', 'hacia', 'aquel', 'así', 'pues', 'pa',
'modo']
['si', 'div', 'classpagenuma', 'tan', 'hacia', 'aquel', 'así', 'pues', 'pa',
'modo']
```

```
[ ]: print(skipgram_Finnish.index_to_key[:10])
     print(cbow_Finnish.index_to_key[:10])
```

```
['p', 'mi', 'ma', 'näin', 'stylemargintop', 'sa', 'jo', 'mun', 'kaikki', 'mut']
['p', 'mi', 'ma', 'näin', 'stylemargintop', 'sa', 'jo', 'mun', 'kaikki', 'mut']
```