01_preprocessing

September 29, 2021

0.1 Imports

[2]:
```
import os, tarfile, sys
from pathlib import Path
from time import time
from pprint import pprint
from collections import Counter

import numpy as np
from numpy.random import choice
import pandas as pd

import spacy

from gensim.models.word2vec import LineSentence
from gensim.models.phrases import Phrases, Phraser
```

0.1.1 Settings

[3]:
```
pd.set_option('float_format', '{:,.2f}'.format)
np.random.seed(42)
```

[4]:
```
LANGUAGES = ['en', 'es']
language_dict = dict(zip(LANGUAGES, ['English', 'Spanish']))
```

[5]:
```
def format_time(t):
    m, s = divmod(t, 60)
    h, m = divmod(m, 60)
    return '{:02.0f}:{:02.0f}:{:02.0f}'.format(h, m, s)
```

0.2 Preprocess Data

0.2.1 TED 2013 English & Spanish

[6]:
```
SOURCE = 'TED'
FILE_NAME = 'TED2013'
DATA_DIR = Path('..', 'data')
```

Data source: http://opus.nlpl.eu/TED2013.php

[7]:
```
filename = DATA_DIR / 'TED' / 'TED2013.en'
print(filename.read_text()[:500])
```

```
http://www.ted.com/talks/stephen_palumbi_following_the_mercury_trail.html
There's a tight and surprising link between the ocean's health and ours, says
marine biologist Stephen Palumbi. He shows how toxins at the bottom of the ocean
food chain find their way into our bodies, with a shocking story of toxic
contamination from a Japanese fish market. His work points a way forward for
saving the oceans' health -- and humanity's.
fish,health,mission blue,oceans,science
899
Stephen Palumbi: Following
```

0.2.2 Tokenize & Clean Sentences

Models expect data provided as a single sentence per line. We'll remove punctuation after using spaCy's parser to tokenize the input text.

[8]:
```
def read_sentences(path, min_sent_length=3):
    stats = pd.DataFrame()
    sentences = []
    skipped, word_count = 0, 0

    with path.open() as source:
        for sentence in source:
            # remove short sentences and urls (for TED data)
            n_words = len(sentence.split())
            if n_words < min_sent_length or sentence.startswith('http:///'):
                skipped += 1
            else:
                word_count += n_words
                sentences.append(sentence.strip())

    stats = pd.Series({'Sentences': len(sentences),
                       '# Words': word_count,
                       'Skipped': skipped})
    return sentences, stats
```

[9]:
```
def clean_sentences(sents, nlp, path, lang):
    exclude = ['PUNCT', 'SYM', 'X']
    start = time()
    vocab = Counter()
    sents = nlp.pipe(sents)
    d = []
    with open(path / 'ngrams_1.txt'.format(language), 'a') as f:
        for i, sent in enumerate(sents):
```

```
            if i % 20000 == 0 and i > 0:
                print(i, end=' ')
            d.extend([[i, w.text, w.pos_] for w in sent])
            clean_sentence = [w.text.lower() for w in sent if w.pos_ not in␣
↪exclude]
            vocab.update(clean_sentence)
            f.write(' '.join(clean_sentence) + '\n')

    vocab = pd.Series(vocab).sort_values(ascending=False).to_frame('count')
    with pd.HDFStore(path.parent / 'vocab.h5') as store:
        store.put('/'.join([lang, 'vocab']), vocab)
        store.put('/'.join([lang, 'tokens']), pd.DataFrame(d,␣
↪columns=['sent_id', 'token', 'pos']))
    duration = time() - start
    print('\n\tDuration: ', format_time(duration))
```

[12]:
```
sentences, stats = {}, pd.DataFrame()

for language in LANGUAGES:
    source_path =  DATA_DIR / SOURCE / '{}.{}'.format(FILE_NAME, language)
    sentences[language], stats[language_dict[language]] =␣
↪read_sentences(source_path)

    print(language, end=': ')
    target_path = Path('vocab', SOURCE, language)
    if not target_path.exists():
        target_path.mkdir(parents=True, exist_ok=True)

    clean_sentences(sentences[language], spacy.load(language), target_path,␣
↪language)
```

```
en: 20000 40000 60000 80000 100000 120000 140000
        Duration:  00:07:01
es: 20000 40000 60000 80000 100000 120000 140000
        Duration:  00:06:49
```

0.2.3 Corpus Summary Stats

[14]:
```
stats.applymap(lambda x: '{:,d}'.format(x))
```

[14]:

	English	Spanish
# Words	2,640,928	2,548,942
Sentences	152,729	151,850
Skipped	5,166	6,045

[15]:
```
with pd.HDFStore(Path('vocab', SOURCE, 'vocab.h5')) as store:
    store.put('stats', stats)
```

0.2.4 Inspect Result

[16]:
```
sentences['en'][:3]
```

[16]:
```
["There's a tight and surprising link between the ocean's health and ours, says
marine biologist Stephen Palumbi. He shows how toxins at the bottom of the ocean
food chain find their way into our bodies, with a shocking story of toxic
contamination from a Japanese fish market. His work points a way forward for
saving the oceans' health -- and humanity's.",
 'Stephen Palumbi: Following the mercury trail',
 'It can be a very complicated thing, the ocean.']
```

[17]:
```
sentences['es'][:3]
```

[17]:
```
['Existe una estrecha y sorprendente relación entre nuestra salud y la salud del
océano, dice el biologo marino Stephen Palumbi. Nos muestra, através de una
impactante historia acerca de la contaminación tóxica en el mercado pesquero
japonés, como las toxinas de la cadena alimenticia del fondo oceánico llegan a
nuestro cuerpo.',
 'Stephen Palumbi: Siguiendo el camino del mercurio.',
 'El océano puede ser una cosa muy complicada.']
```

0.2.5 Create n-grams

[14]:
```
def create_ngrams(language, max_length=3):
    """Using gensim to create ngrams"""

    path = Path('vocab', SOURCE, language)
    n_grams = pd.DataFrame()
    start = time()
    for n in range(2, max_length + 1):
        print(n, end=' ')

        sentences = LineSentence(str(path / 'ngrams_{}.txt'.format(n-1)))
        phrases = Phrases(sentences, threshold=100, min_count=10)

        s = pd.Series({k.decode('utf-8'): v for k,
                       v in phrases.export_phrases(sentences)})
        s = s.to_frame('score').reset_index().rename(
            columns={'index': 'phrase'}).assign(length=n)

        n_grams = pd.concat([n_grams, s])
        grams = Phraser(phrases)
        sentences = grams[sentences]

        with open(path / 'ngrams_{}.txt'.format(n), 'w') as f:
            for sentence in sentences:
```

```
                f.write(' '.join(sentence) + '\n')

    n_grams = n_grams.sort_values('score', ascending=False)
    n_grams.phrase = n_grams.phrase.str.replace('_', ' ')
    n_grams['ngram'] = n_grams.phrase.str.replace(' ', '_')

    with pd.HDFStore(Path(path.parent / 'vocab.h5')) as store:
        store.put('/'.join([language, 'ngrams']), n_grams)

    print('\n\tDuration: ', format_time(time() - start))
    print('\tngrams: {:,d}\n'.format(len(n_grams)))
    print(n_grams.groupby('length').size())
```

[15]:
```
for language in LANGUAGES:
    print('\n', language, end=' ')
    create_ngrams(language)
```

```
 en 2 3
        Duration:  00:00:36
        ngrams: 483

length
2    433
3     50
dtype: int64

 es 2 3
        Duration:  00:00:37
        ngrams: 508

length
2    462
3     46
dtype: int64
```

[ ]:
```

```