03_financial_news_preprocessing

September 29, 2021

1 Financial News - Preprocessing for word2vec

See notebook lda_financial_news for download instructions.

1.1 Imports

[1]:
```
import os, tarfile, sys, json
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 seaborn as sns

import spacy
from spacy.lang.en import English

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

1.1.1 Settings

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

[3]:
```
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)
```

[4]:
```
stop_words = set(pd.read_csv('http://ir.dcs.gla.ac.uk/resources/
 ↪linguistic_utils/stop_words',
                             header=None,
                                    squeeze=True).tolist())
```

1.1.2 Paths

[6]:

```
data_path = Path('..', 'data', 'us-financial-news')
```

[5]:

```
results_path = Path('results', 'financial_news')
if not results_path.exists():
    results_path.mkdir(exist_ok=True)
```

[ ]:

```
article_path = results_path / 'articles.txt'
clean_article_path = results_path / 'articles_clean.txt'
```

1.2 Load Data

[7]:

```
section_titles = ['Press Releases - CNBC',
                  'Reuters: Company News',
                  'Reuters: World News',
                  'Reuters: Business News',
                  'Reuters: Financial Services and Real Estate',
                  'Top News and Analysis (pro)',
                  'Reuters: Top News',
                  'The Wall Street Journal &amp; Breaking News, Business,␣
 ↪Financial and Economic News, World News and Video',
                  'Business &amp; Financial News, U.S &amp; International␣
 ↪Breaking News | Reuters',
                  'Reuters: Money News', 'Reuters: Technology News']
```

[8]:

```
articles = []
counter = Counter()
start = time()
for f in fin_news_path.glob('*/**/*.json'):
    article = json.load(f.open())
    if article['thread']['section_title'] in set(section_titles):
        text = article['text'].lower().split()
        counter.update(text)
        articles.append(' '.join([t for t in text if t not in stop_words]))

print(f'Done loading {len(articles):,.0f} articles in␣
 ↪{format_time(time()-start)}')
```

```
Done loading 125,964 articles in 00:02:23
```

[9]:

```
article_path.write_text('\n'.join(articles))
```

[9]: 418422851

1.3 Clean Financial News Articles

[10]:
```
articles = article_path.read_text().split('\n')
len(articles)
```

[10]:
```
125964
```

1.3.1 Sentence Boundary Detection

[11]:
```
def clean_doc(d):
    doc = []
    for sent in d.sents:
        s = [t.text.lower() for t in sent if not
        any([t.is_digit, not t.is_alpha, t.is_punct, t.is_space])]
        if len(s) > 5 or len(sent) < 100:
            doc.append(' '.join(s))
    return doc
```

[12]:
```
nlp = English()
sentencizer = nlp.create_pipe("sentencizer")
nlp.add_pipe(sentencizer)
```

[13]:
```
clean_articles = []
iter_articles = (article for article in articles)
for i, doc in enumerate(nlp.pipe(iter_articles, batch_size=100, n_process=8),␣
 ↪1):
    if i % int(len(articles) / 100) + 1 == 0:
        print(f'{i / len(articles):.1%}', end=' ', flush=True)
    clean_articles.extend(clean_doc(doc))
```

[14]:
```
clean_article_path.write_text('\n'.join(clean_articles))
```

[14]:
```
356650931
```

1.3.2 Corpus Stats

[15]:
```
len(clean_articles)
```

[15]:
```
2986105
```

[16]:
```
vocab = Counter()
sent_length = []
for sentence in clean_articles:
    tokens = sentence.lower().split()
    sent_length.append(len(tokens))
    vocab.update(tokens)
```

[17]:

```
len(sent_length)
```

[17]:

```
2986105
```

[18]:

```
pd.Series(sent_length).describe(percentiles=np.arange(.1, 1, .1).round(1))
```

[18]:

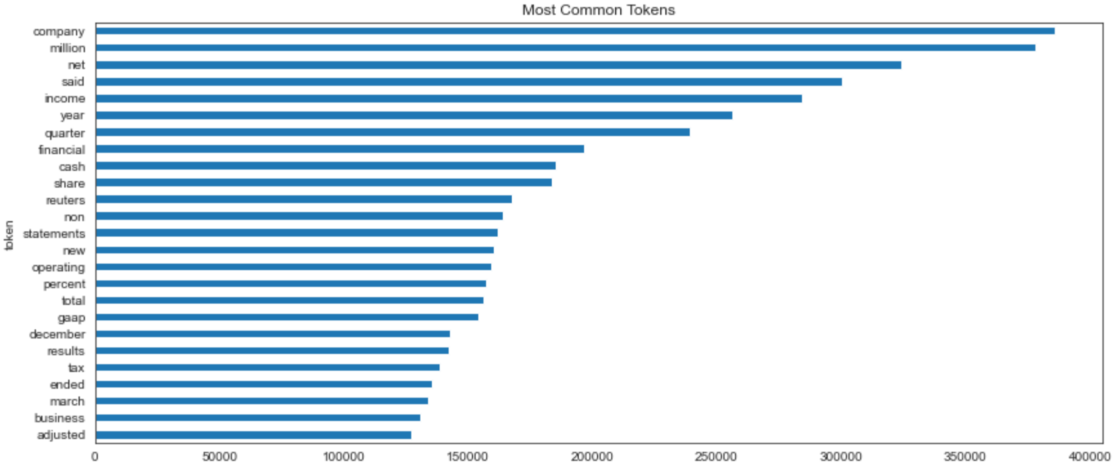
```
count   2,986,105.00
mean           15.36
std            20.68
min             0.00
10%             4.00
20%             6.00
30%             8.00
40%            10.00
50%            12.00
60%            14.00
70%            17.00
80%            20.00
90%            25.00
max         6,910.00
dtype: float64
```

[19]:

```
most_common = (pd.DataFrame(vocab.most_common(), columns=['token', 'count'])
               .pipe(lambda x: x[~x.token.str.lower().isin(stop_words)]))
```

[20]:

```
most_common.head(25).set_index('token')['count'].sort_values().plot.
 ↪barh(title='Most Common Tokens', figsize=(14, 6));
```



1.3.3 Inspect Result

[21]:

```
clean_articles[:3]
```

[21]:

```
['duluth allete inc',
 'nyse ale today reported earnings share net income million operating revenue
billion',
 'results share net income million operating revenue billion']
```

1.4 Create n-grams

[22]:

```
max_length = 3
```

[23]:

```
n_grams = pd.DataFrame()
start = time()
sentences = LineSentence(clean_article_path.as_posix())
for n in range(2, max_length + 1):
    print(n, end=' ')
    if n>2:
        sentences = LineSentence((results_path / f'articles_{n-1}_grams.txt').
 ↪as_posix())
    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 (results_path / f'articles_{n}_grams.txt').open('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(' ', '_')

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

```
2 3
        Duration:  00:09:31
        ngrams: 43,678
```

```
length
2    26965
3    16713
dtype: int64
```

[24]:

```
n_grams.groupby('length').apply(lambda x: x.nlargest(10, 'score'))
```

[24]:

```
                                           phrase      score  length  \
length
2      23317             hidradenitis suppurativa 182,360.93       2
       17916                        citigate dewe 182,360.93       2
       13440           mukhammadsharif mamatkulov 182,360.93       2
       11037                pracha hariraksapitak 182,360.93       2
       24230                     navesh chitrakar 181,947.41       2
       25079                         nidaa tounes 181,947.41       2
       12947                      koustav samanta 181,947.41       2
       18997               axalimogene filolisbac 181,947.41       2
       14914                  shilpa jamkhandikar 181,855.77       2
       10972                         krispy kreme 181,855.77       2
3      6433                  dana reizniece ozola 189,347.18       3
       13583  daniela palmieri jessica fitzgerald 189,347.18       3
       11954          hgst sandisk tegile upthere 189,347.18       3
       10424       carson elder beerman herberger 189,347.18       3
       7974        freshfields bruckhaus deringer 189,347.18       3
       5611              writing fransiska nangoy 189,347.18       3
       13342   strategist janney montgomery scott 189,347.18       3
       4506               spokesman baik tae hyun 188,917.82       3
       9167         hormone transcon pth transcon 188,917.82       3
       10597                   castello di casole 188,917.82       3

                                            ngram
length
2      23317             hidradenitis_suppurativa
       17916                        citigate_dewe
       13440           mukhammadsharif_mamatkulov
       11037                pracha_hariraksapitak
       24230                     navesh_chitrakar
       25079                         nidaa_tounes
       12947                      koustav_samanta
       18997               axalimogene_filolisbac
       14914                  shilpa_jamkhandikar
       10972                         krispy_kreme
3      6433                  dana_reizniece_ozola
       13583  daniela_palmieri_jessica_fitzgerald
       11954          hgst_sandisk_tegile_upthere
       10424       carson_elder_beerman_herberger
```

7974	freshfields_bruckhaus_deringer
5611	writing_fransiska_nangoy
13342	strategist_janney_montgomery_scott
4506	spokesman_baik_tae_hyun
9167	hormone_transcon_pth_transcon
10597	castello_di_casole