nlp

April 10, 2023

```
[ ]: # import libraries
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]: # create dataframe with the first 10000 rows
     news = pd.read_csv ('abcnews-date-text.csv', nrows = 10000)
```

```
[ ]: # sanity checks
     news.shape
```

```
[ ]: news.head()
```

```
[ ]: news.tail()
```

```
[ ]: # number of characters in each sentence
     news['headline_text'].str.len().hist()
```

News headlines range from 15 characters to 55 characters

```
[ ]: # generate a histogram of word numbers
     def plot_word_number_histogram (text):
         text.str.split().\
         map(lambda x: len(x)). \
         hist()
```

```
[ ]: plot_word_number_histogram (news['headline_text'])
```

Number of words in a headline ranges from 4 to 9 words

```
[ ]: # word length in a headline
     def plot_word_length_histogram (text):
         text.str.split(). \
         apply (lambda x : [len(i) for i in x]). \
         map (lambda x : np.mean(x)). \
         hist()
```

```
[ ]: # plot word length
     plot_word_length_histogram (news ['headline_text'])
```

Stopwords are the most common words

```
[ ]: pip install --user -U nltk
```

```
[ ]: import nltk
     from nltk.corpus import stopwords
     stop = stopwords.words('english')
```

```
[ ]: # create a list of words
     new = news['headline_text'].str.split()
     new = new.values.tolist()
     corpus = [word for i in new for word in i]
```

```
[ ]: from collections import defaultdict
     new_dict = defaultdict(int)
     for word in corpus:
         if word in stop:
             new_dict[word] += 1
```

```
[ ]: # create a plot of the top 12 words
     top = sorted (new_dict.items(), key = lambda x : x[1], reverse = True)[:12]
     x, y = zip (*top)
     plt.bar (x, y)
```

```
[ ]: # create bar chart of words that are not stopwords
     from collections import Counter

     counter = Counter (corpus)
     most = counter.most_common()
     x = []
     y = []
     for word, count in most [:40]:
         if word not in stop:
             x.append (word)
             y.append (count)
     sns.barplot (x = y, y = x)
```

WordCloud

```
[ ]: pip install --user -U wordcloud
```

```
[ ]: from wordcloud import WordCloud, STOPWORDS
     stopwords = set (STOPWORDS)
```

[ ]:
```
# create a function to display a word cloud
def display_wordcloud (text):
    wordCloud = WordCloud (
        background_color = 'white',
        stopwords = stopwords,
        max_words = 100,
        max_font_size = 30,
        scale = 3,
        random_state = 1
    )
    wordCloud = wordCloud.generate (str(text))
    fig = plt.figure (1, figsize = (12, 12))
    plt.axis ('off')

    plt.imshow (wordCloud)
    plt.show()
```

[ ]:
```
display_wordcloud (corpus)
```

Ngrams

Contiguous sequence of n words. bigram is a continuous sequence of two words.

[ ]:
```
from nltk.util import ngrams
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from collections import Counter

def plot_ngrams (text, n = 2):
    stop = set (stopwords.words('english'))

    new_text = text.str.split()
    new_text = new_text.values.tolist()
    corpus = [word for i in new_text for word in i]

    def _get_top_ngrams (corpus, n = None):
        vec = CountVectorizer (ngram_range = (n,n)).fit (corpus)
        bag_of_words =vec.transform (corpus)
        sum_words = bag_of_words.sum (axis = 0)
        words_freq = [(word, sum_words[0, idx])
                      for word, idx in vec.vocabulary_.items()
                     ]
        words_freq = sorted (
            words_freq,
            key = lambda x : x[1],
            reverse = True
                            )
        return words_freq [:10]
    top_n_grams = _get_top_ngrams (text, n)[:10]
    x, y = map (list, zip (*top_n_grams))
    sns.barplot (x = y, y = x)
```

```
[ ]: plot_ngrams (news['headline_text'], 2)
```

Sentiment Analysis

```
[ ]: pip install --user -U textblob
```

```
[ ]: from textblob import TextBlob

     def polarity (text):
         return TextBlob(text).sentiment.polarity
```

```
[ ]: news['polarity_score'] = news['headline_text'].apply (lambda x : polarity(x))
     news['polarity_score'].hist()
```

```
[ ]: def sentiment (x):
         if x < 0:
             return 'neg'
         elif x > 0:
             return 'pos'
         else:
             return 'neu'
```

```
[ ]: news['polarity'] = news['polarity_score'].map (lambda x : sentiment(x))
     plt.bar (
         news.polarity.value_counts().index,
         news.polarity.value_counts()
     )
```

```
[ ]: # positive news
     news[news['polarity'] == 'pos']['headline_text'].head(10)
```

```
[ ]: # negative news
     news[news['polarity'] == 'neg']['headline_text'].head(10)
```