review_analysis

August 15, 2020

[372]:
```
import pandas as pd
```

[373]:
```
df = pd.read_csv('feedback_week1.csv').iloc[:,0:4]
```

[374]:
```
df.isnull().sum()
```

[374]:
```
User full name    0
Email address     0
Date              0
Feedback          0
dtype: int64
```

[375]:
```
n=dataset.shape[0]
```

[376]:
```
import nltk
```

[377]:
```
reviews=dataset['Feedback']
```

[378]:
```
df['Feedback'] =df['Feedback'].astype(str)
```

0.0.1 Data Cleanup process

[379]:
```
#lower case
df['Feedback'] = df['Feedback'].apply(lambda x: " ".join(x.lower() for x in x.
 ↪split()))
```

[380]:
```
## remove punctuation
df['Feedback'] = df['Feedback'].str.replace('[^\w\s]','')
```

[ ]:
```

```

[381]:
```
## remove stopwords

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

```
df['Feedback'] = df['Feedback'].apply(lambda x: " ".join(x for x in x.split()␣
↪if x not in stop))
```

[382]:
```
#special character
df['Feedback'] = df['Feedback'].str.replace('[^\w\s]','')
```

[383]:
```
#remove new line
df = df.replace('\n','', regex=True)
```

[384]:
```
## regular expression cleanup
import re

REPLACE_NO_SPACE = re.compile("[.;:!\'?,\"()\[\]]")
REPLACE_WITH_SPACE = re.compile("(<br\s*/><br\s*/>)|(\-)|(\/)")
DIGIT_REMOVE = re.compile(r"(^|\W)\d+")
NEW_LINE_REMOVE = re.compile("\n")


def preprocess_reviews(reviews):
    reviews = [REPLACE_NO_SPACE.sub("", line.lower()) for line in reviews]
    reviews = [REPLACE_WITH_SPACE.sub(" ", line) for line in reviews]
    reviews = [DIGIT_REMOVE.sub("", line) for line in reviews]

    return reviews

df['Feedback'] = preprocess_reviews(df['Feedback'])
```

[385]:
```
#### Lemmatization
```

[386]:
```
def get_lemmatized_text(corpus):
    from nltk.stem import WordNetLemmatizer
    lemmatizer = WordNetLemmatizer()
    return [' '.join([lemmatizer.lemmatize(word) for word in review.split()])␣
↪for review in corpus]

df['Feedback'] = get_lemmatized_text(df['Feedback'])
```

[387]:
```
### stemming
```

[388]:
```
# st = PorterStemmer()
# df['reviews.text'] = df['reviews.text'].apply(lambda x: " ".join([st.
↪stem(word) for word in x.split()]))
```

0.0.2 Frequently used words

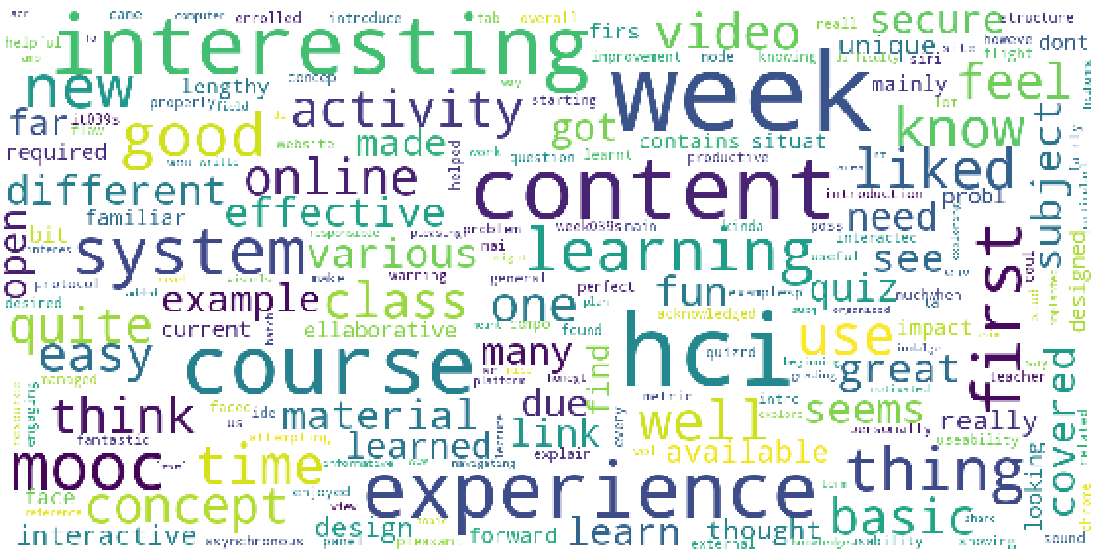
```
[389]: most = pd.Series(' '.join(df['Feedback']).split()).value_counts()[:10]
       most
```

```
[389]: week           47
       content        46
       hci            36
       learning       28
       course         28
       interesting    26
       system         25
       video          23
       mooc           21
       good           21
       dtype: int64
```

0.0.3 Word cloud

```
[390]: from wordcloud import WordCloud, STOPWORDS
       import matplotlib.pyplot as plt
       stopwords = set(STOPWORDS)
```

```
[391]: def show_wordcloud(col, title = None):
           wordcloud = WordCloud(
               background_color='white',
               stopwords=stopwords,
               max_words=500,
               max_font_size=40,
               scale=3,
               random_state=1
           ).generate(str(col))

           fig = plt.figure(1, figsize=(10, 10))
           plt.axis('off')
           if title:
               fig.suptitle(title, fontsize=10)
               fig.subplots_adjust(top=2.3)

           plt.imshow(wordcloud)
           plt.show()


       if __name__ == '__main__':
           show_wordcloud(df['Feedback'])
       #     show_wordcloud(df['review'])
```



0.1 Adjectives used

```
[392]: feelings = []
       for sentence in df['Feedback']:
           # each sentence is either a list of words or a list of (word, POS tag)␣
        ↪tuples
           for word, pos in nltk.pos_tag(sentence.split()): # remove the call to nltk.
        ↪pos_tag if `sentence` is a list of tuples as described above
               if pos in ['JJR', "JJ", "JJS", "RB","RBR","RBS"]: # feel free to add␣
        ↪any other noun tags
                   feelings.append(word)
```

```
[393]: most = pd.Series(' '.join(feelings).split()).value_counts()[:10]
       most
```

```
[393]: really         21
       also           19
       good           18
       first          17
       content        15
       well           12
       interesting    11
       helpful        10
       new             9
       different       9
       dtype: int64
```

0.1.1 Sentiment scoring

```
[394]: def senti(x):
           return TextBlob(x).sentiment

       df['senti_score'] = df['Feedback'].apply(senti)

       df.senti_score.head()
```

```
[394]: 0                                     (0.4, 0.4)
       1      (0.3954545454545455, 0.5126262626262627)
       2      (0.4642857142857143, 0.5476190476190476)
       3                                   (0.375, 0.55)
       4    (0.16666666666666669, 0.41666666666666663)
       Name: senti_score, dtype: object
```

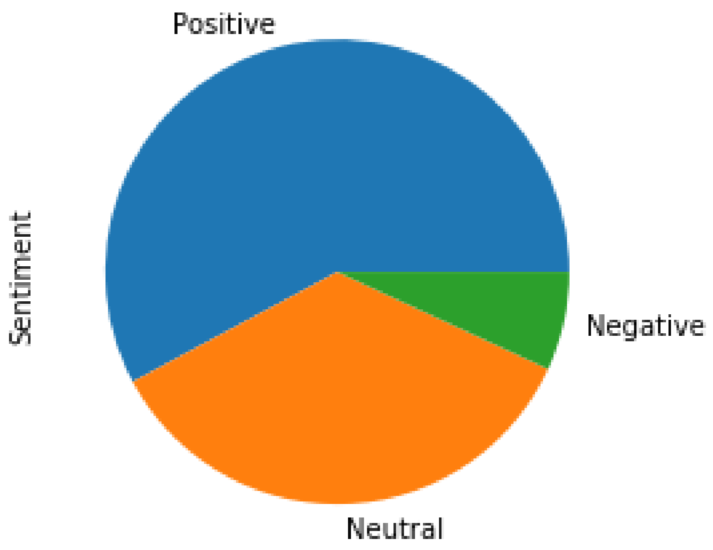
```
[395]: import pandas as pd
       import nltk
       nltk.download('vader_lexicon')
       from nltk.sentiment.vader import SentimentIntensityAnalyzer

       #load in the sentiment analyzer
       sia=SentimentIntensityAnalyzer()

       #apply the analyzer over each comment
       df['polairty scores'] =df['Feedback'].apply(lambda x: sia.
        ↪polarity_scores(x)['compound'])
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/bibhuti/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
[396]: def sentiment(score):
           if score<0 :
               return "Negative"
           elif score<=0.5:
               return "Neutral"
           elif score>0.5:
               return "Positive"
```

```
[397]: sentiment_list=[]
       for a in df['polairty scores']:
           sentiment_list.append(sentiment(a))
       df['Sentiment']=sentiment_list
```

[398]:
```
df['Sentiment'].value_counts().plot(kind="pie")
plt.show()
```



[399]:
```
df.to_csv('review_analysis.csv',index=False)
```