PROCESS NOTES

WILL BE DONE

- Porter Stemming and Lemmatizing (both available in NLTK) might be used to treat "messages", "message", and "messaging" as the same word
- dublicate check for whole data set might be needed

DONE

- all phrases are lowered
- punctuations in phrases are erased
- empty phrase rows returned to NA then erased
- stop words (nltk library pre created stop words list is used) in phrases are erased
- after stop word clean empty phrase rows returned to NA then erased

Get train data under folder data.

In [3]:
```
import numpy as np
import pandas as pd
df=pd.read_csv('data/train.csv',sep="\t")
df[71:77]
```

Out[3]:

	Unnamed: 0	PhraseId	SentenceId	Phrase	Sentiment
71	71	87	3	of Ismail Merchant 's work	2
72	72	88	3	Ismail Merchant 's work	2
73	73	89	3	Ismail Merchant 's	2
74	74	91	3	Merchant 's	2
75	75	94	3	work	2
76	76	95	3	, I suspect , would have a hard time sitting t...	1

Remove column "Unnamed: 0"

In [4]:
```
df = df.drop('Unnamed: 0', 1)
df[71:77]
```

Out[4]:

	PhraseId	SentenceId	Phrase	Sentiment
71	87	3	of Ismail Merchant 's work	2
72	88	3	Ismail Merchant 's work	2
73	89	3	Ismail Merchant 's	2
74	91	3	Merchant 's	2
75	94	3	work	2
76	95	3	, I suspect , would have a hard time sitting t...	1

Replace capitals to non-capitals.

In [5]:
```
df.Phrase=df.Phrase.str.lower()
df[71:77]
```

Out[5]:

	PhraseId	SentenceId	Phrase	Sentiment
71	87	3	of ismail merchant 's work	2
72	88	3	ismail merchant 's work	2
73	89	3	ismail merchant 's	2
74	91	3	merchant 's	2
75	94	3	work	2
76	95	3	, i suspect , would have a hard time sitting t...	1

Remove punctuations.

In [6]:
```
df.Phrase = df.Phrase.str.replace('[^\w\s]','')
df[71:77]
```

Out[6]:

	PhraseId	SentenceId	Phrase	Sentiment
71	87	3	of ismail merchant s work	2
72	88	3	ismail merchant s work	2
73	89	3	ismail merchant s	2
74	91	3	merchant s	2
75	94	3	work	2
76	95	3	i suspect would have a hard time sitting thr...	1

Remove stopwords by using pre-created list of nltk library.

In [7]:
```
#import stopword list from nltk
from nltk.corpus import stopwords
stopwords_set = set(stopwords.words('english'))

#define a map to delete stopwords under each phrase
def stop_erase(x):
    proper_words = []
    words = x.split()
    for word in words:
        if not word in stopwords_set:
           proper_words.append(word)

    return " ".join(proper_words)

df.Phrase=df.Phrase.map(stop_erase)
df[71:77]
```

Out[7]:

	PhraseId	SentenceId	Phrase	Sentiment
71	87	3	ismail merchant work	2
72	88	3	ismail merchant work	2
73	89	3	ismail merchant	2
74	91	3	merchant	2
75	94	3	work	2
76	95	3	suspect would hard time sitting one	1

Replace empty phrase rows to NaN then delete.

In [8]:
```
print("size of data with empty phrase rows=",len(df.index))

#define a map to change empty phrases to NaN
def change_nan(x):
    if pd.isnull(x):
        return np.nan
    if len(x) == 0:
        return np.nan
    return x

df['Phrase'] = df.Phrase.map(change_nan)

#remove NaN rows of df
df=df.dropna()

#reset indexing
df=df.reset_index(drop=True)

print("size of data after empty phrase rows deleted=",len(df.index))
```

```
size of data with empty phrase rows= 125049
size of data after empty phrase rows deleted= 124125
```

Afer clean, the same same phrases with differen sentiment scores created as follows.

In [9]:
```
df[10:12]
```

Out[9]:

	PhraseId	SentenceId	Phrase	Sentiment
10	18	1	good goose	2
11	22	1	good goose	3

The dataset is in order so that same phrases are one under the other. Use that property instead of search whole list to find duplicate (searching whole list might be needed in case such same phrase groups on different locations of list). In such same phrase groups assign the first one as main by changing its sentiment score to group average and change other phrases to NaN to remove later from the list. Before doing that sentiment scores should be converted into float data type from intiger(as a default).

In [10]:
```
print("data type of sentiment scores= ",df.Sentiment.dtype)

#convert sentiment scores to float to write averaged ones
df.Sentiment=df.Sentiment.astype(float)
df.dtypes
print("data type of sentiment scores after conversion= ",df.Sentiment.dtype)

#phrase list holds a single phrase to compare it to the next one if same
#sum_score holds the sum of same phrases sentiments'
#count_same_phrase holds the number of same phrases
phrase=[]
sum_score=0
count_same_phrase=0

for index,row in df.iterrows():
    #if no phrase is get to previously to compare, get this phrase for next ph
rase comparings
    if len(phrase)==0:
        phrase.append(row.Phrase)
        sum_score+=row.Sentiment
        count_same_phrase+=1
    else:
        #if this phrase is same with phrase get previously,
        #add sentiment score to sum_score and increment count_same_phrase to c
alculate average later
        if row.Phrase==phrase[0]:
            sum_score+=row.Sentiment
            count_same_phrase+=1
        #if this phrase is different the same phrase group is now end
        #calculate average score and assign it to same phrase group's first se
ntiment
        #assingn as NaN all sentiments in group other than first one
        #change phrase list element with this phrase for next comparings
        else:
            df.loc[(index-count_same_phrase),'Sentiment']=sum_score/count_same
_phrase
            while (count_same_phrase > 1):
                df.loc[(index-count_same_phrase+1),'Phrase']=np.nan
                count_same_phrase-=1
            sum_score=row.Sentiment
            count_same_phrase=1
            phrase[0]=row.Phrase
```

```
data type of sentiment scores=  int64
data type of sentiment scores after conversion=  float64
```

In [11]:
```
df[10:12]
```

Out[11]:

	PhraseId	SentenceId	Phrase	Sentiment
10	18	1	good goose	2.5
11	22	1	NaN	3.0

Remove NaN assigneds and reindex data.

In [12]:
```
df=df.dropna()
df=df.reset_index(drop=True)
df.head()
```

Out[12]:

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	series escapades demonstrating adage good goos...	1.0
1	2	1	series escapades demonstrating adage good goose	2.0
2	5	1	series	2.0
3	6	1	escapades demonstrating adage good goose	2.0
4	9	1	escapades	2.0

This cleaned data is sort of vocabulary list. Save file as cleandf.csv under data folder

In [13]:
```
df.to_csv("data/cleandf.csv", sep='\t')
```