Sentiment analysis

author: Wayne LI

Sentiment analysis:

Goal: Sentiment analysis is a method for extracting the information from the vast amount of texts. We can understand the feelings ,rate, emotion of a certain service, products like movies, shopping,etc. We can also analyze the reason of this comments and realize what is the next step that we proceed. It can be used to help with decision making. It makes the process easier and more efficient.

In this practice, we use the dataset from kaggle (https://www.kaggle.com/datasets/miriamodeyianypeter/sentiment-analysis-amazon-product-reviews/data). It is about the comment of a product on amazon. We use it to do the sentiment analysis. By doing so, we can understand the overall rate of a product we are interested in ,and the manufacturers can also know the direction of improving their products and provides better experience or service.

Main steps:

1. We will first import our datasets and do some data cleaning

2. Use tokenizer to turn the text data,which is difficult to understand for the computer, to integers with each represents a word

3. use padding to make equal length comment (to reduce error and satisfy the input for model)

4. use embedding to change the word to a certain vector to fed to our RNN model

5. train test split

6. train RNN network (using simpleRNN,LSTM,GRU to compare)

7. we can use the model to do further predicts to reduce time

8. finally, we can try to find out what is the reason and what can we do to improve it for the next products

import data:

This is our raw data. We have our comments X and sentiment y,which will later be used in m
trainings(as this is a supervised model)

review_body	review_date	sentiment
Great love it	2015-08-31	1
Lots of ads Slow processing speed Occasionally shuts down apps WIFI keeps having authentication issues Was cheap for a tablet and now i know why.	2015-08-31	0
Excellent unit. The versatility of this tablet, besides being competitively priced is a solution to the elderly. Poor eyesight and physical disabilities associated with age and using the supporting add on features allows the user	2015-08-31	1
I bought this on Amazon Prime so I ended up buying the 16gb one for $95. The camera is okay and I love the edit features you can do to the pictures. Amazon/Netflix/Huliu all run smoothly on this tablet. The internet runs a	2015-08-31	1
All Amazon products continue to meet my expectations	2015-08-31	1
Good product. I like it 😀	2015-08-31	1
This kindle works well but the battery goes dead more quickly than you'd expect. Really it is a version of the previous Kindle that weighs less. The internet browser works considerably better than the previous but still has issu	2015-08-31	1
I really enjoy my new kindle, it is easy to use and offers plenty of useful and entertaining activities	2015-08-31	1
It's what I wanted and performs perfectly. Everything works great, the menu system is awful. I'd go 5 stars if the menu system were better.	2015-08-31	1
Made well. Like looks and style	2015-08-31	1
Yes it met all of my needs 5 Stars	2015-08-31	1
We have the old kindles but this one does a lot more. Love it	2015-08-31	1
Great product	2015-08-31	1

Comment Preprocessing:

The comments we have cannot be used directly into the model and analysis later on. Thus, our first step is to do some NLP preprocessing. This step is really important in the overall tasks as it will affect the effectiveness and interpretability of the final results.

```
#preprocessing
#turn into lower case
data['review_body'] = data['review_body'].astype(str).str.lower()

#delete <br>,<p> html tag
data['review_body'] = data['review_body'].apply(lambda x: BeautifulSoup(x, "html.parser").get_text())

#delete number,comma,punctuation mark or special characters,emogi
data['review_body'] = data['review_body'].apply(lambda x: re.sub(r'[^a-zA-Z\s]', '', x))

#remove extra space
data['review_body'] = data['review_body'].apply(lambda x: " ".join(x.split()))

#return words without contraction (縮寫) like it's-->it is
data['review_body'] = data['review_body'].apply(lambda x: contractions.fix(x))

#remove stopwords(停用詞)(common words that is not important like the,is,are,I)
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
data['review_body'] = data['review_body'].apply(lambda x: " ".join(
    [word for word in x.split() if word not in stop_words]))

#lemmatization(詞性還原)like running->run/better->good
lemmatizer = WordNetLemmatizer()
data['review_body'] = data['review_body'].apply(
    lambda x: " ".join([lemmatizer.lemmatize(word) for word in x.split()]))
```

✓ 3.3s

Preprocessing results:

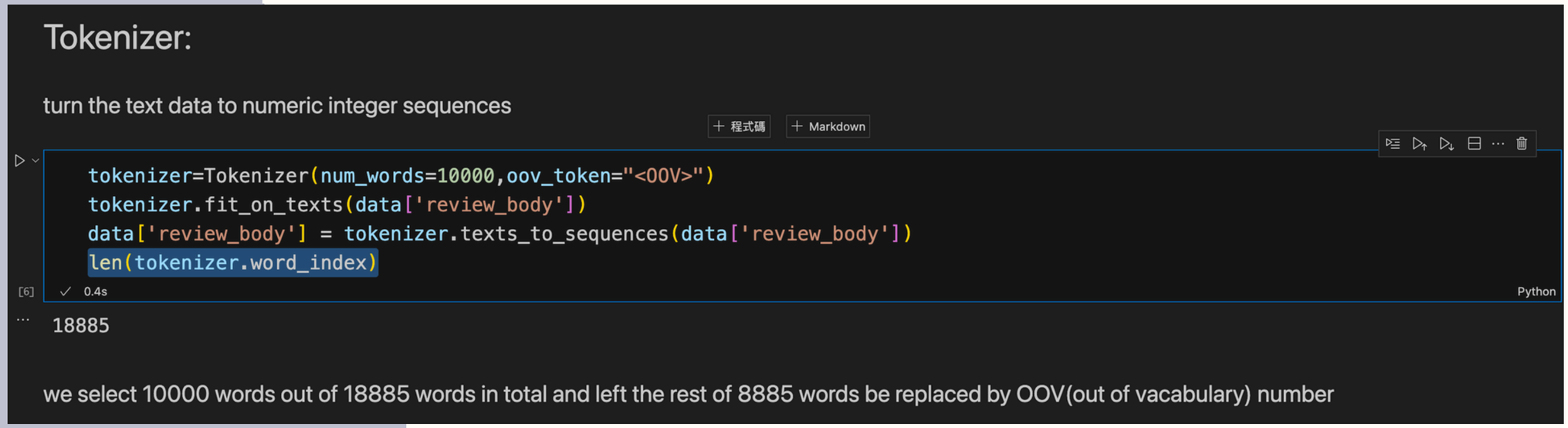
```
#result
data['review_body']
```

✓ 0.0s

```
0                                                   great love
1          lot adsslow processing speedoccasionally shuts...
2          excellent unit versatility tablet besides comp...
3          bought amazon prime ended buying gb one camera...
4                    amazon product continue meet expectation
                                  ...
30841      videoidmopbgsrllyi purchased original kindle f...
30842      writing review benefit experienced kindle fire...
30843      purchased kindle grandma wanted simple way vid...
30844      bought tablet fire hd best buy day came planne...
30845       impressive piece hardware regret girlfriend love
Name: review_body, Length: 30846, dtype: object
```

Tokenizer:

Our next step is tokenizing the texts. Turn it into computer-understandable number sequences. Each integer represents a word



The num_words limits the words that we will use in this part. I set it to 10000 means the functions will turrn the words into corresponding integers. It will choose the most frequent 10000 words and set the vocab not in this range to OOV(out of vacabulary). The word_index will remain all the words that appear in the comments no matter what num_words you set.

Word_index:

```
{'<OOV>': 1,
 'love': 2,
 'kindle': 3,
 'fire': 4,
 'great': 5,
 'tablet': 6,
 'one': 7,
 'use': 8,
 'like': 9,
 'hd': 10,
 'amazon': 11,
 'would': 12,
 'book': 13,
 'get': 14,
 'good': 15,
 'device': 16,
 'easy': 17,
 'new': 18,
 'apps': 19,
 'time': 20,
 'screen': 21,
 'much': 22,
 'work': 23,
 'really': 24,
 'bought': 25,
...
 'description': 997,
 'tap': 998,
 'he': 999,
 'together': 1000,
 ...}
```

Tokenize result:

```
data['review_body']
```

[8] ✓ 0.0s

```
0                                                      [5, 2]
1        [57, 7618, 1131, 7619, 1042, 7620, 131, 3420, ...
2        [102, 245, 1373, 6, 1132, 7622, 975, 1144, 195...
3        [25, 11, 66, 746, 259, 101, 7, 51, 594, 2, 163...
4                                  [11, 26, 757, 460, 193]
                                    ...
30841    [1, 73, 136, 3, 4, 145, 155, 3, 4, 239, 136, 7...
30842    [1178, 215, 1054, 1270, 3, 4, 110, 1043, 404, ...
30843    [73, 3, 1459, 93, 326, 77, 68, 1047, 28, 738, ...
30844    [25, 6, 4, 10, 86, 58, 53, 209, 1801, 11, 1039...
30845                          [1382, 573, 825, 856, 1363, 2]
Name: review_body, Length: 30846, dtype: object
```

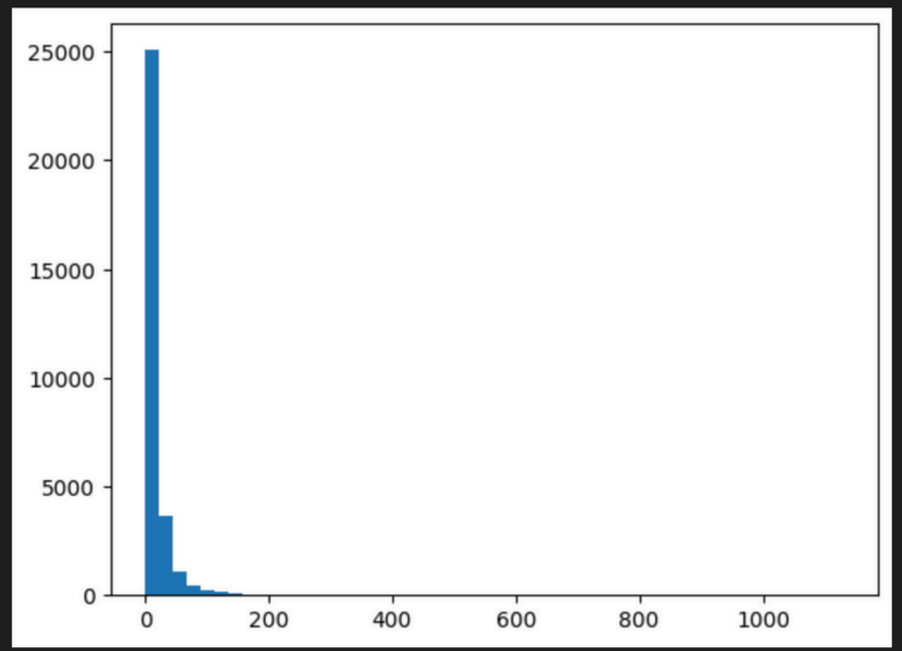
Padding:

To make sure each comment have equal length , we need padding.Each comment has a length of maxlen. If greater than it, the comment will be cut into maxlen long and if less than it the function will fill in certain value (usually 0) to make the equal length.

To decide which maxlen to use, we will first see how long is each comment to avoid lossing useful information.

```
sequence_lengths = [len(seq) for seq in data['review_body']]
plt.hist(sequence_lengths, bins=50)
plt.show()
```

✓ 0.1s



We use histogram to display the length of each comment and the result shows that almost every comment is less than 50. Thus, it will be our maxlen here to minimize the information losses.

Padding results:

from the graph we see the comments are usually less than 50 words so we will choose 50 as our maxlen

```
data['review_body'] =list(pad_sequences(data['review_body'], maxlen=50))
data['review_body']
```

✓ 0.0s

```
0         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2         [3169, 551, 67, 3421, 252, 65, 665, 110, 534, ...
3         [1374, 7, 3, 13, 167, 85, 35, 7, 438, 212, 73,...
4         [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
                                ...
30841     [85, 11, 289, 7610, 3293, 2971, 3025, 66, 169,...
30842     [190, 404, 24, 361, 68, 19, 65, 69, 2, 30, 13,...
30843     [19, 62, 52, 1459, 213, 178, 133, 3, 556, 289,...
30844     [1008, 3455, 1051, 71, 2444, 1, 633, 41, 83, 7...
30845     [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
Name: review_body, Length: 30846, dtype: object
```

Train_test_split:

I use train_test_split to get the train and test dataset as this is a classify task. The stratify=y parameter make sure we have same distribution on y=0,1 on the training test dataset.

train,test split

```
x = np.array(data['review_body'].tolist())
y=data['sentiment']
```

✓ 0.0s

```
#I will use 20% as test data
X_train,X_test,y_train,y_test=train_test_split(x,y,stratify=y,test_size=0.2,random_state=42)
```

✓ 0.0s

Model building:

Next, I will use embedding to change each word to certain vector and use rnn model(simple rnn,LSTM,GRU) to compare the result

	simplernn	LSTM	GRU
test_loss	0.369073	0.297372	0.441721
test_accuracy	0.896434	0.900648	0.894814
time	65.43	226.76	203.32

SimpleRNN:

```
start=time.time()
model=Sequential([
    Embedding(input_dim=10000,output_dim=128,input_length=50),
    SimpleRNN(64, activation='relu'),
    Dense(128,activation='relu'),
    Dropout(0.2),
    Dense(1,activation='sigmoid'),
])
model.compile(optimizer=Adam(learning_rate=0.0001),loss='binary_crossentropy',metrics=['accuracy'])
model.fit(X_train,y_train,batch_size=32,epochs=10,validation_split=0.2)
end=time.time()
print(f"training time: {end - start:.2f} seconds")
```

LSTM:

```
start=time.time()
model = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=50),
    LSTM(128, activation='relu'),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid'),
])
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train,batch_size=32,epochs=10,validation_split=0.2)
end=time.time()
print(f"training time: {end - start:.2f} seconds")
```

GRU:

```
start = time.time()
model = Sequential([
    Embedding(input_dim=10000, output_dim=128, input_length=50),
    GRU(128, activation='relu'),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid'),
])
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=32, epochs=10, validation_split=0.2)
end = time.time()
print(f"training time: {end - start:.2f} seconds")
```

By comparing, we know that simplernn,LSTM,GRU RNN model performs almost equally on the test set. They have similar accuracy ~~90%. But LSTM have much less loss than the other two. This might because it is better on the long comments. However, it also takes the longest time to train.

Next:

I want to use other methods to understand what cause the positive and negative comments,I will use shap and wordcloud to analyze it.

Shap:

The shap value could evaluate the effects of a certain features to the predicted results.In this case, the word with the larger shap values mean has positive effects in the positive category. The words with high negative values in shap is good for predicting negative categories.

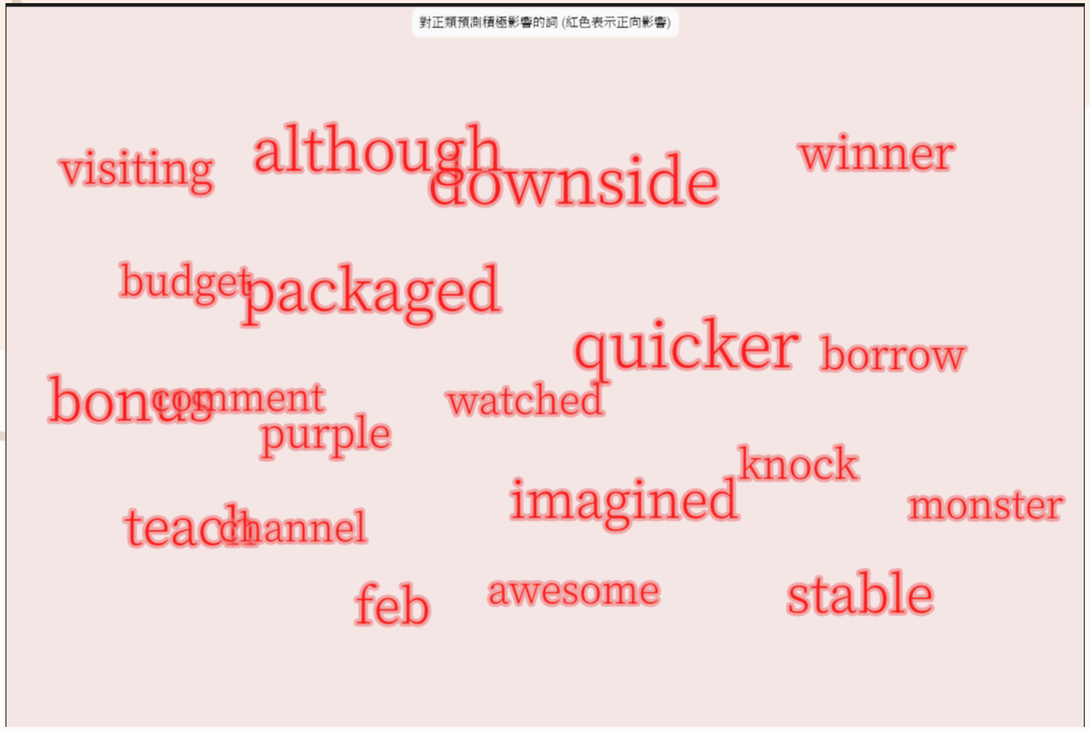
I will extract the words based on the result category and calculate the mean shap value of these words finally I will draw a wordcloud based on the value of the mean shap of each word.

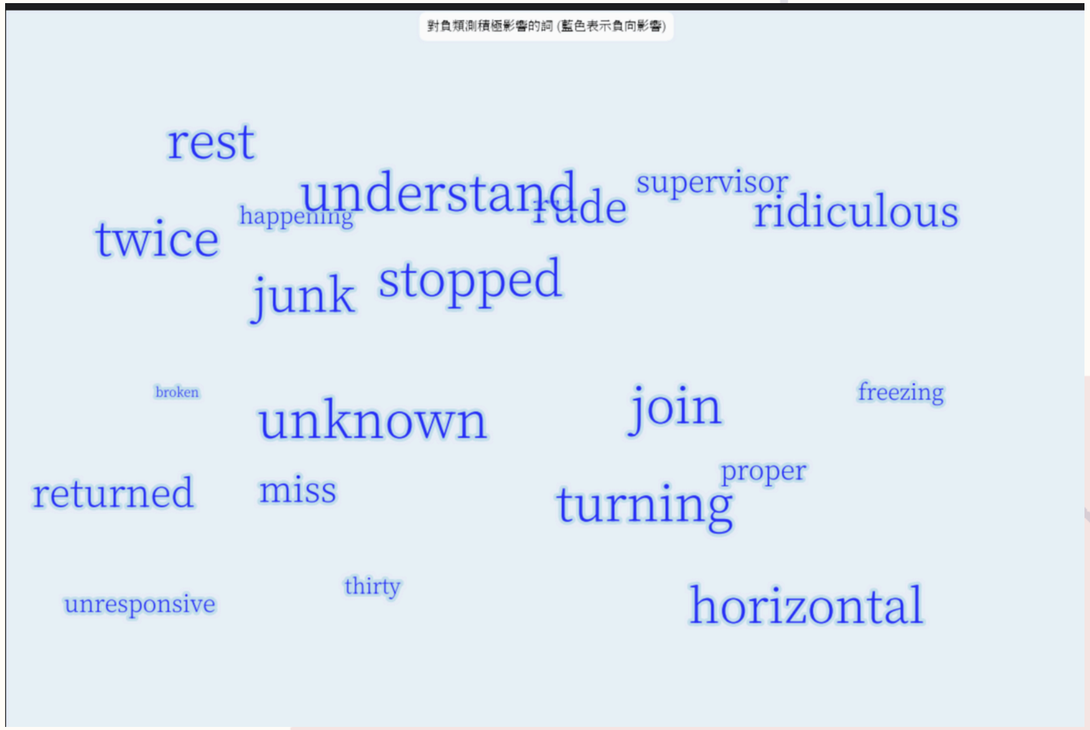
Wordcloud result:

Positive words: ['downside', 'quicker', 'although', 'packaged', 'bonus', 'stable', 'teach', 'imagined', 'feb', 'winner', 'visiting', 'borrow', 'purple', 'knock', 'budget', 'monster', 'watched', 'awesome', 'channel', 'comment']
Negative words: ['broken', 'thirty', 'freezing', 'happening', 'unresponsive', 'proper', 'supervisor', 'miss', 'returned', 'ridiculous', 'rude', 'junk', 'horizontal', 'twice', 'stopped', 'rest', 'turning', 'join', 'unknown', 'understand']

Matplotlib wordcloud result:

Positive Words: ['downside', 'quicker', 'although', 'packaged', 'bonus', 'stable', 'teach', 'imagined', 'feb', 'winner', 'visiting', 'borrow', 'purple', 'knock', 'budget', 'monster', 'watched', 'awesome', 'channel', 'comment']
Negative Words ['broken', 'thirty', 'freezing', 'happening', 'unresponsive', 'proper', 'supervisor', 'miss', 'returned', 'ridiculous', 'rude', 'junk', 'horizontal', 'twice', 'stopped', 'rest', 'turning', 'join', 'unknown', 'understand']





I use two methods to create my wordcloud, wordcloud and matplotlib. This is because the wordcloud library pops up the ttf problems. As you can see, the wordcloud makes better and prettier graphs and it has easier parameters to adjust your plot.
The results show that the words that affect the positive emotions are stable,quicker,packaged,bonus,etc.
For the negative emotions we have broken,ridiculous,rude,junk,unresponsive.

We can see that some words might be associated with positive or negative comments, but we still can't really understand what happen. To further dicuss it, we can see what words combinations usually go with these words . To do so, I will continue using ngrams and spacy to analyze the most common word combinations in the positive and negative category.

Ngrams:
Ngrams is used to see how frequent a certain word combinations appears in the comments.
We can use ngrams to see what causes positive,negative comments, since appear more frequently t
more likely is the
main reasons.

positive:

📌 最常見的正面 n-grams:

	n-gram	Positive Frequency
239	kindle fire	291
127	fire hd	247
100	easy use	113
300	love kindle	105
334	new kindle	78
389	prime day	71
203	great product	60
304	love new	55
15	amazon prime	54
30	battery life	53
525	year old	52
295	love fire	50
510	work great	48
333	new fire	42
343	old kindle	42
415	really like	41
446	still learning	39
302	love love	38
209	great tablet	38
202	great price	36
405	read book	34
361	original kindle	33
371	play game	31
116	far good	31
512	work well	30
335	new one	30
5	absolutely love	29
326	much better	27
409	reading book	27
367	perfect size	24

negative:

📌 最常見的負面 n-grams:

	n-gram	Negative Frequency
515	kindle fire	390
325	fire hd	325
106	battery life	159
446	google play	138
239	customer service	126
670	new one	94
750	play store	82
67	app store	79
685	old kindle	74
182	cannot get	59
686	old one	59
669	new kindle	58
36	amazon prime	56
301	every time	54
1131	year old	54
730	original kindle	45
771	prime day	45
972	thought would	44
56	android tablet	43
23	amazon app	43
1030	user friendly	42
483	hold charge	41
647	much better	37
804	read book	37
1053	waste money	37
899	special offer	36
346	first one	36
807	reading book	35
488	home screen	35
501	ipad mini	35

Spacy:

However,sometimes, more frequent words might not show ideas or reasons, so some other methods can also be used, like spacy(a powerful NLP library). Spacy is like ngrams, it will show the most frequent words in the positive and negative categories. But it is more powerful than ngrams. Unlike ngrams, which just split the words and find the nearest vocab. Spacy will also tag each words with types: like nouns, verbs,adjectives and show a more meaning analysis or pictures of the reasons.

📌 Positive: [('kindle fire', 259), ('fire hd', 231), ('easy use', 111), ('love love', 74), ('prime day', 71), ('great product', 61), ('love kindle', 58), ('new kindle', 54), ('battery life', 53), ('year old', 52), ('love fire', 51), ('work great', 46), ('love new', 46), ('new fire', 42), ('old kindle', 37), ('great price', 36), ('amazon prime', 33), ('love great', 31), ('original kindle', 30), ('play game', 29)]
📌Negative: [('kindle fire', 363), ('fire hd', 294), ('battery life', 159), ('customer service', 125), ('old kindle', 68), ('play store', 66), ('app store', 62), ('year old', 53), ('original kindle', 45), ('prime day', 42), ('new kindle', 40), ('hold charge', 40), ('user friendly', 36), ('special offer', 36), ('home screen', 35), ('lock screen', 34), ('android tablet', 33), ('reading book', 33), ('battery last', 31), ('play game', 30)]

```
positive_bigram_counts.most_common(20)
```
✓ 0.0s

```
[('kindle fire', 259),
 ('fire hd', 231),
 ('easy use', 111),
 ('love love', 74),
 ('prime day', 71),
 ('great product', 61),
 ('love kindle', 58),
 ('new kindle', 54),
 ('battery life', 53),
 ('year old', 52),
 ('love fire', 51),
 ('work great', 46),
 ('love new', 46),
 ('new fire', 42),
 ('old kindle', 37),
 ('great price', 36),
 ('amazon prime', 33),
 ('love great', 31),
 ('original kindle', 30),
 ('play game', 29)]
```

```
negative_bigram_counts.most_common(20)
```
✓ 0.0s

```
[('kindle fire', 363),
 ('fire hd', 294),
 ('battery life', 159),
 ('customer service', 125),
 ('old kindle', 68),
 ('play store', 66),
 ('app store', 62),
 ('year old', 53),
 ('original kindle', 45),
 ('prime day', 42),
 ('new kindle', 40),
 ('hold charge', 40),
 ('user friendly', 36),
 ('special offer', 36),
 ('home screen', 35),
 ('lock screen', 34),
 ('android tablet', 33),
 ('reading book', 33),
 ('battery last', 31),
 ('play game', 30)]
```

Result:

With the ngrams and spacy, we could find that:

1.The product we are having here is kindle fire, a tablet released by amazon.

2.The customers are satisfied with the experience of the tablets, it is easy to use, it may has discounts and sells cheaper on amazon prime day.

3.However, some points that make customers unsatisfied are the battery problems(maybe because battery last time),customer service, and the apple store(amazon develops by itself not using GoogleStore or Appstore from apple), the diversity and choices of apps and games may be limited.
Maybe for some users, lock screen is another issue.

4.Connect to the wordcloud above, we can say that the bonus might be related to the amazon prime activities and the rude or unresponsive might refer to the customer service of amazon.

5.With wordcloud and ngrams or spacy, we can have a bigger picture of how a certain products is, we can use this information to understand the overall rate of it and draw new solutions or improvements for the next version.

What's Next:

1.We can use it to build a recommendation systems or analyze systems of a certain products and expand the idea into other usages like: movie,trip choice, etc.

2.In addition to the comments from others, we can learn how to build a recommendation systems using personal preferences , viewing times or clicking times.

Thank you!