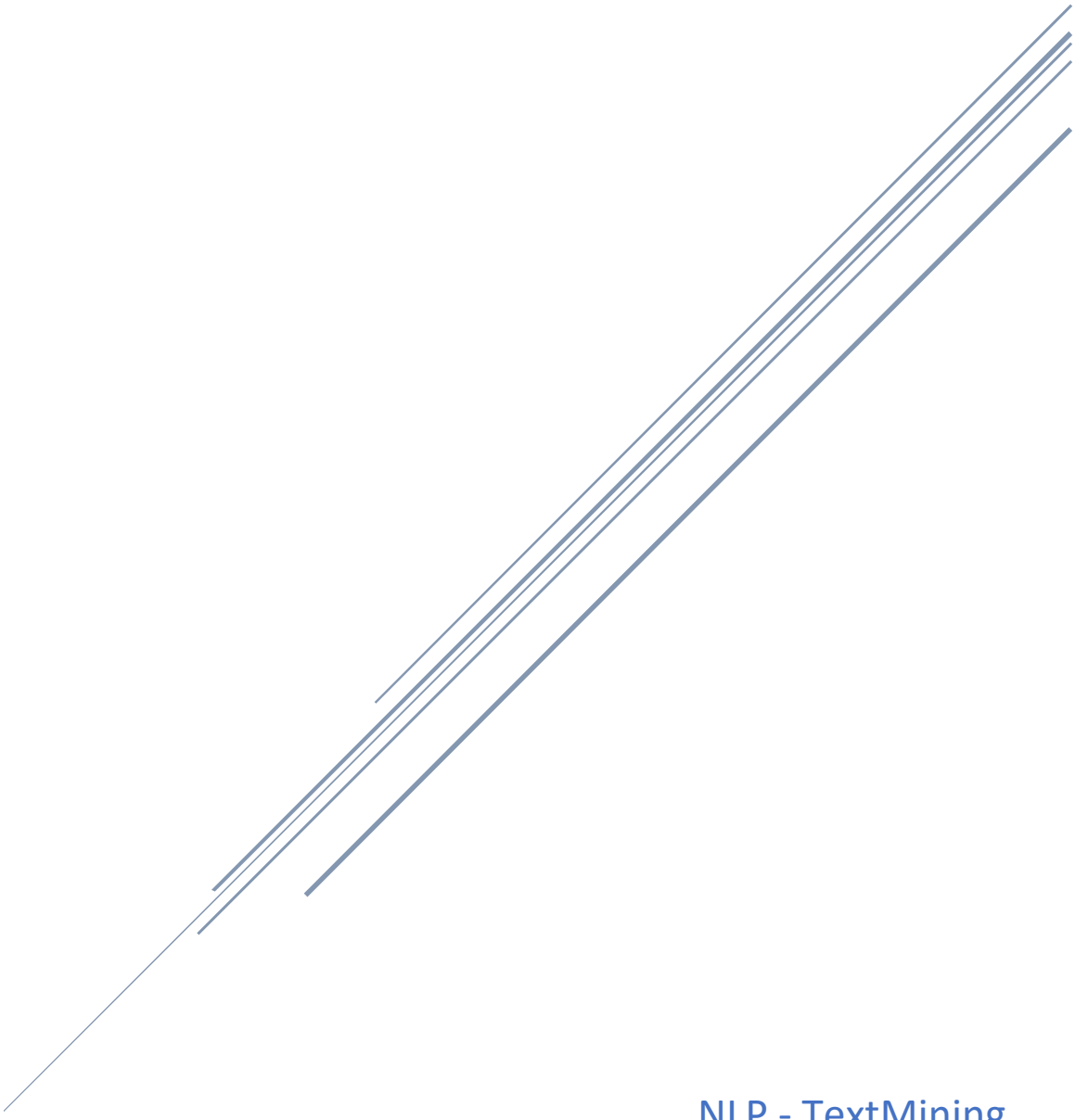
Sentiment Analysis Project

Classifying Amazon Reviews



NLP - TextMining

Done By: Jailan aljizawi and Daad Alfaleh

In this Project will go through a Natural Language Processing Python Project creating a Sentiment Analysis classifier with NLTK's VADER and Huggingface Roberta Transformers. The project is to classify the seniment of amazon customer reviews. We will look at the difference between model outputs from the two packages and compare the results. Seniment analysis is an important tool for data scientists to use in laguage modeling.

Our Report on The Project

The imports we are going to import pandas as PD we're gonna import numpy it for some plotting will import matplotlib pipe plot as PLT and we're going to import seaborn as SNS let's set a set a style sheet that we'll use for our plots and then we're going to just to start out here import NLTK which is that natural language toolkit will be using for the start let's go ahead and read in our data . we're going to read in from the input directory there is this reviews that CSV and after we read this in we can show here just a head command on this that in this data set we have each row is the unique ID we have the product ID user ID profile name and then the real interesting stuff here is the score so this is out of a one to five star rating how many stars the reviewer gave this item and then the text it's a little small to see here but you can see if we just do the text row and show the first one it's actual text with the review that was written by the reviewer for this product and we're going to be running our sentiment analysis on this row of data in this entire data set but actually this data set if we print the shape is quite large there are almost half a million reviews here and just for time sake let's downsample this data set so we can do that pretty simply by just doing a head command on this and taking the first 500 rows and then if we print the data frame shape after that we'll see that it's 500 rows but you could scale up this project very easily to all half a million products if you wanted to run a more intense analysis so then well just put the data frame head command here so we can see and reference back to what columns we have available to us now let's do some quick EDA just to get an idea of what this data set looks like so we'll take this score column which we know to be a value between one and five and we're going to do a value counts on this this gives us the number of times each score occurs and then we'll sort the index and we'll do just a bar plot of this kind will be a bar plot and the title is going to be count of reviews by stars and let's also do a fig size 10 by 5 plot that and let's add a label to it so I'm going to do some line breaks here to clean this up and we're going to call this our axis and then we'll be setting the ex label as review stars and we will do PLT show all right so we can see here that most of the reviews are actually five stars but then it kind of goes down it has a little uptick in the number of one star reviews we have so the very biased towards positive reviews in our data set that's good to know before we go any further so the next thing we'll do is just some basic and NLTK stuff and we'll start by just taking one example review so let's do example equals is text column and just pick the 50th value as an example and we'll print this example alright So what did this person say they said this oatmeal is not good it's mushy soft I don't like it Quaker oats is like OK so seems to be negative sentiment here but before we get into that let's just see some of the stuff that NLTK can do out-of-the-box NLTK can kept tokenize this sentence so let's taste this example and run NTK word tokenizer and all that basically does is splits this into the parts of each word in the sentence now it looks pretty clear clean at the beginning but then you can see that I don't do an end apostrophe T split so this is a little bit smarter than just splitting on spaces in the text and this will give us our actual tokenized results in natural language processing often you need to convert the text into some format that the computer can interpret and tokenizing that is the way that you do it so let's make this the tokens and then take the tokens and let's just show the first 10 so we can remember what this looks like

all right so now once we have these tokens another thing NLTK can do out-of-the-box is actually find the part of speech for each of these words so let's run NLTP KKS West tag for part of speech tagging and we'll run this on each of these tokens now we can see that we have each token and we also have its part of speech so these part of speech values our codes and we can actually load up an example page that has some examples of what each abbreviation means so let's go back here to our example not here so you could see that oatmeal is N and in our part of speech tagging that means it's a noun a singular noun so each of the values in this text has now been given its part of speech so let's call this tagged and then let's just show the first ten again as our example now it can actually we can take it the next step from this and take these tags part of speech and put them into entities so NLTK we can do a chunk on this and then in each chunk so this takes the recommended name entity chunker to chunk the given list of tokens so it takes these tokens and actually will group them into chunks of text so let's run it on this to see what it looks like what are we getting here , so we need to store this because we're running in a notebook and then we'll actually run entities dot P print for pretty print of this so you can see that it's chunked this into a sentence and noted here that this is an organization some other interesting stuff about the text that can be extracted out automatically using NLTK alright so that's just a basic primer about NLTK but we want to get into sentiment analysis so we're going to start by using Vader Vader stands for what does Vader stand for it stands for I wrote up here valence aware dictionary and sentiment reasoner so this approach essentially just takes all the words in our sentence and it has a value of either positive negative or neutral for each of those words and it combines up it just doesn't math equation for all the words it'll add up to tell you how positive negative or neutral that the statement is based on all those words now one thing that keep in mind is this approach does not account for relationships between words which in human speech is very important .
but at least it's a good start so we also removed something called stop words that really don't have a positive or negative feeling to them they're just for the structure of the sentence alright so let's do some sentiment analysis using this Vader approach we're going to do from NLTK sentiment sentiment import sentiment intensity analyzer and then we're gonna also import from T QDM notebook import TQ DM this is just a progress bar tracker for when we're going to do some loops on this data,then we're going to make our sentiment analyze our object by calling this sentiment intensity analyzer creating it and calling it sia,this needs to be from TQ DM notebook QDM and now we have our sentiment intensity analyzer object we can run this on text and see what the sentiment is based on the words!
We can see that this fader approach has made this has tagged this negative as zero this these are scales from zero to one so zero negative neutral .3 and positive .682 so mostly positive now there's also this compound score which is an aggregation of negative neutral and positive this count compound compound value is from - 1 to positive one representing how negative to positive it is but if you want more detail you can take the breakdown of this negative neutral and positive so it did a good job it made this I tagged this as being mostly positive let's try the opposite so SA a polarity scores of this is the worst thing ever all right now we see that the polarity school polarity score for this is mostly negative and neutral and nothing positive and then this compound score is netpoint- .62 so more on the negative side than positive very interesting now we can run SA on our example like that we had before remember our example which was this oatmeal comment let's run that on the oatmeal comment and see what it is OK so it's pretty high neutral but also some negative and the overall compound score is negative no positive score so we want to run this polarity score when the polarity score on the entire data set

so basically looping through this data frame we have every text field we wanted run this and grab the polarity scores and we can do that with a simple loop so we're going to do 4D which is going to be just our row or we can just say four row and QDM DF dot itter rows and then our total is going to be yeah so then this should work I think and then we're going to take um the row text and this will be tech our text and then we're also gonna take our we're gonna call it my ID which is the Rose ID card column and then let's just break here to make sure we have this correctly ,this is going to be for I row and TQM literature tuples it arose and then we'll also make the total of this the length of the data frame so that when we see our progress bar it's out of 500 we're going to want some way to store these results so let's make a dictionary called res for results and every time we loop through we'll take my ID we'll store it in the my ID part of the dictionary the polarity squirrel score polarity score of the text right and then yeah that's it let's run this so really fast it ran it's done and now we have this result a dictionary with each ID the negative neutral positive and compound score of each but we want to store this into a pandas data frame because that's easier to work with let's do that really quickly by just running PD dataframe on this dictionary pandas can take in a dictionary pretty easily except for it's oriented the wrong way so let's just quickly run a dot T on this which will flip everything horizontally and now we have an index which is our ID and then our negative neutral positive in our compound score for the sentiment for each of those values alright let's call this vaders that's our feeder's result and then let's also let's take this vaders and let's reset the index and rename that index as our ID so we can merge this onto our original data frame and then we're gonna take vaders so we'll vaders will now be this and then we'll also take vaders and we're going to merge it on our original data frame and how we'll do a left merge so now basically we have our data frame but with our scores and we also have all the other values from our original data set including the text so if I run a head on this you could see now we have sentiment score and metadata all right so let's see let's see if this in general is in line with what we would expect so we're going to make some assumptions here about our data that if the score of the item that the reviewer gave it is a five star review it's probably going to be more positive of text than if it was a score of 1 that one star review is gonna have more negative connotation than a 5 star review and we can do that by just doing a simple bar plot so let's use seaborn I think I imported seaborn yeah imported seaboard before and we're going to do a bar plot of this data where data is vaders let's call this the plot better results and our X value is going to be the score which remember is the star review of the the person and then compound is going to be our Y value and that's the - 1 to positive one overall sentiment of the of the text then let's set the title to be compound score by Amazon well by each score that was given it's more and mowell by each score that was given it's more and more positive of text respectively and that's that's good that just kind of validates what we are looking for we can even break this down instead of looking at the compound score we can look at the positive neutral negative scores for each so we're gonna do that by doing something like SNS barplot data is vaders again X's score again and then let's do the positive and see what this looks like all right so this is the positive score and let's actually make three of these side-by-side left being positive neutral and then the negative to the right and we'll do that with matplotlib subplots so this will make a one by three grid of our results and we will call this axis put this first one here which will be our positive then we want our neutral and then we want the negative and this is going to be in position 1/2 and three and then let's also set the title so we remember what these are positive neutral and negative and plot show this this needs to be a X equals and I need to change each of these but there we go now we have let's see what we have here let's make this a little less wide we have the positive positivity is higher as the score is higher in terms of stars the neutrals kind

of flat and the negative goes down it becomes less negative of a comment as the star review becomes higher great this just confirms what we would hope to see and shows that Vader is valuable in having this connection between the score of the text and sentiment score and that it does relate to the actual with the actual rating review of the reviewers let's do a tight layout just because I see some overlapping here of of the review of the why axis labels but I think this is good alright so now we're going to take it up a notch our previous model just looked at each word in the sentence or in the review and scored each word individually but like we mentioned before human language depends a lot about a lot on context so if I say something we'll see a sentence that could have negative words actually could be sarcastic or related to other words in which way it makes it a positive seaman so this Vader model wouldn't pick up on that sort of relationship between words but more and more recently these transformer based deep learning models have become very popular because they can pick up on that context so we're going to use from hugging face which is one of the leaders in these types of models and gathering them and making them easily available we're going to import from Transformers now this is hugging faces library you could pip install Transformers to get this on your local machine or of course you can just running in a G Collab notebook like we are right now so let me make sure this works from Transformers we're gonna import our auto tokenizer now this is gonna tokenize similar to what we showed NLTK can do and then from Transformers we're going to import auto model four let's autocomplete here for sequence classification you could see that there are a lot of different types of models that hugging face has and then we're also going to import from side pie special softmax which we will apply to the outputs because they don't have softmax applied and this will smooth out between zero and one alright special spell that correctly all right and then we're going to pull it in a very specific model that has been pre trained on a bunch of data um bit for sentiment exactly like we're trying to do this is provided by hugging face and when we run the auto tokenizer and the auto model sequence classification methods and load it from a pre trained model it'll pull down the model weights that have been stored and this is really great because we're essentially doing transfer learning this model was trained on a bunch of Twitter comments that were labeled and we don't have to retrain the model at all we can just use these trained weights and apply it to our data set and see what comes out so anytime you do this the first time you will see that it needs to download all of the weights this is expected and now that's finished now we have a model in a tokenizer that we can apply to the text so let's remember what our example was before now this is this oatmeal comment and our polarity score from the old type of model look like this let's call this the theater results on example remember negative neutral and we want to run this though on using the Roberta model that we've pulled so we just need to take a few steps so before we can run it on Alberta model and that's first thing is encoding the text so we're going to take our tokenizer that we pulled in and we're going to apply it to this example and return turn
We'll understand we'll call this our encoded text then we're going to take that and we're going to run our model on it it's just that simple so we're going to take this encoded text run our model on it and this will be our output and then you remember how we so this was the output looks like it's a tensor with our results and then we're going to take that output take it from being a tensor and make it into numpy so that we can store locally so let's detach this and then numpy and store this as scores and then the last thing we're going to do is just apply that softmax to the these scores that we imported softmax layer like this now if you print our scores we see that we have three different values in a numpy array now these are similar to the last type of model that we ran so basically this is the negative the neutral and the positive score for this text so let's make it a

scores dictionary where we're will store this and we'll put in Roberta negative is going to be the first value and then we'll just do like this negative neutral and positive and this will be 012 and we'll print this scores dictionary you come here there we go alright so the Roberta model much more than the Vader model thinks that this comment is negative which from reading it seems to make sense this is a very negative review of this product so already here we can see sort of how much more powerful Roberta is than just a Vader model let's go ahead and run this on the entire data set like we did before with the Vader model so we can do this pretty easily by just making a function out of the code that we did before called polarity scores Roberta where it takes in an example our code did before and it runs all of this and it just returns scores dictionary so now we could run this on one example of text and get this scores dictionary like we had written the code for above and we're going to iter iterate over the data set just like we did before so let's take this the code from above where we iterated and we have our we're going to call this our better result because we'll still run the Vader text on this and then we'll also have our Roberto results which is going to be the polarity scores Roberta function that we had written on this text and we'll break here after the first one just to see how it ran on the 1st iteration through so we see we have our veda results and we also have our Roberta results see exactly what we wanted and we also want to combine these so the way we can combine two dictionaries is there's a way to do it with the newer version of Python but we're running an older version so we'll just do it like this and we'll call this both that's both results let's also go and rename this from negative neutral and positive to be explicitly named that they're invader results and I'm just going to copy this code which will basically rename these to Vader and then the key name instead of right now it's just negative positive and then we will combine these two OK so we're running that for one iteration it looks like our results look good and we want to now just run it through all 500 examples so I'm going to take out this break and it's going to run through oh we also need to um actually store this into the dictionary that we're going to store with the ID and we're going to store both now here's I know this is going to break because I read this before but I wanted to just show as an example when it does break alright it did break on one of these examples because the text had some issues with it and it wasn't able to run through the Roberta model so instead of debugging this all right now and it has to do with the size of the text itself there's certain size of text that's just too big for the model to handle we will skip those and we will skip those by adding a try except clause here so it'll run through except for when this runtime error occurs you did not just print out a message so we we know that it broke so now we have good now we're going to rerun your account and get OK so that's done running you can see that did break for two examples oh I'd have account to open like now you start it and it would have worked device but you installed it without typing so I was using these representatives and get the e-mail that you connected to get help I could have run a lot faster but it works for this case just to run it on a CPU

Results / Conclusion

let's actually take the results of this looks like it looks similar to what we did before so it's code which takes these results runs of transforms on it let's call this results yeah so for me I have a user for the course and these emerged back on the main data set because I did this last time now if I do a head command on this you can see now we have our picture can you start recording all four of them for each row to complete set really quickly let's compare scores yeah just like because you are silent for many times or part of times model and we can do this using seaborn's pair plot I think this would be a nice way to look at it so pair plot lets us see so comparison between you have this uh so I'll show you here by users connected to it also we're just going to

provide it the variables we want it to look at and those will be this Vader negative neutral positive and the let's remove the feeder compound so I don't think that's really needed to compare and we're also provided the Roberta columns and we're saying these are the ones variables we want to compare let's also color it so the hue of color of each dot is going to be Star Wars and let's also give it a palette something where we can easily see the difference between each values OK so this is theirs I think actually this combining compare that's what we're doing now OK so a lot going on here but one thing that we notice here is the five star reviews are this purplish color and if we look at Vader the positive reviews are more so to the right on for these five star reviews for the Roberta model you can see it's way over to the right and then you can see that there are some correlations between the Roberta model exactly username this other one model is a little loose Jane Burnham and used each of these so if you look here there was this positive and neutral like the rubber to my very high scores for the five stars is one stars are very low typing installation so that's pretty cool let's also review some examples this is going to be pretty cool because now that we have sentiment score and they should of the review we can look and see where the model maybe does the opposite of what we did so one way we can do that is we just take this results data frame they run the lines where there's a one star review this issue that you're username was another one star it was the list sort the values in there by this e-mail means for me right but let's make this false have a list score is the score will be the rating of the value being one it will appear at the top and then we'll take the text of that and just do a values and print out the top value

we don't need the model is more of a positive you have excessively you need to work in groups well this will help us this statement so that's interesting it makes sense I think it's the same thing with the with the Vader score so look at the most positive no we've got to score for a one star rating it says so we cancelled the order it was cancelled without any problem that is a positive note so they actually were used the word positive that without any problem seems positive but it is a negative review and it's been a little sarcastic I guess hello I have passport type of model which is only looking at a bag of words for all of this sentence and in the score of each word individually let's also look at negative sentiment five stars and let's do this with the rubber model first

we'll switch to a five star review and we'll look at the top negative senses it says this was so delicious but too bad I ate them too fast in game two puns my faults OK so it is sort of a negative sentiment but a positive review that's kind of funny that that one came up and then we'll do the exact same thing for the data so it's looking happens to be the exact same one so the but did you face any issue and you were doing it actually is a negative sentiment for a positive review .

Step 0. Read in Data and NLTK Basics

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.style.use('ggplot')

import nltk
```

In [2]:
```
# Read in data
df = pd.read_csv('../input/amazon-fine-food-reviews/Reviews.csv')
print(df.shape)
df = df.head(500)
print(df.shape)
```

```
(568454, 10)
(500, 10)
```

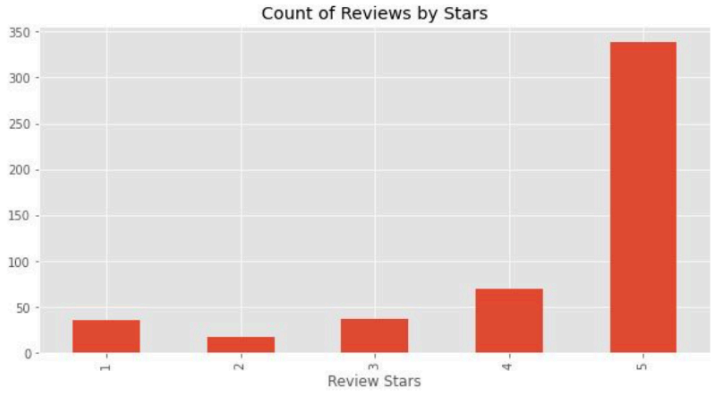
In [3]:
```
df.head()
```

Out[3]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	13038624
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	13469760
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	12190176
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	13079232
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	13507776

Quick EDA

In [4]:
```
ax = df['Score'].value_counts().sort_index() \
    .plot(kind='bar',
          title='Count of Reviews by Stars',
          figsize=(10, 5))
ax.set_xlabel('Review Stars')
plt.show()
```



Basic NLTK

In [5]:
```
example = df['Text'][50]
print(example)
```

```
This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the way to go.
```

In [6]:
```
tokens = nltk.word_tokenize(example)
tokens[:10]
```

Out[6]:
```
['This', 'oatmeal', 'is', 'not', 'good', '.', 'Its', 'mushy', ',', 'soft']
```

In [7]:
```
tagged = nltk.pos_tag(tokens)
tagged[:10]
```

Out[7]:
```
[('This', 'DT'),
 ('oatmeal', 'NN'),
 ('is', 'VBZ'),
 ('not', 'RB'),
 ('good', 'JJ'),
 ('.', '.'),
 ('Its', 'PRP$'),
 ('mushy', 'NN'),
 (',', ','),
 ('soft', 'JJ')]
```

In [8]:
```
entities = nltk.chunk.ne_chunk(tagged)
entities.pprint()
```

```
(S
  This/DT
  oatmeal/NN
  is/VBZ
  not/RB
  good/JJ
  ./.
  Its/PRP$
  mushy/NN
  ,/,
  soft/JJ
  ,/,
  I/PRP
  do/VBP
  n't/RB
  like/VB
  it/PRP
  ./.
  (ORGANIZATION Quaker/NNP Oats/NNPS)
  is/VBZ
  the/DT
  way/NN
  to/TO
  go/VB
  ./.)
```

Step 1 :

Step 1. VADER Seniment Scoring

We will use NLTK's `SentimentIntensityAnalyzer` to get the neg/neu/pos scores of the text.

- This uses a "bag of words" approach:
  1. Stop words are removed
  2. each word is scored and combined to a total score.

In [9]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm

sia = SentimentIntensityAnalyzer()
```

```
/opt/conda/lib/python3.7/site-packages/nltk/twitter/__init__.py:20: UserWarning: The twytho
n library has not been installed. Some functionality from the twitter package will not be a
vailable.
  warnings.warn("The twython library has not been installed. "
```

In [10]:
```
sia.polarity_scores('I am so happy!')
```

Out[10]:
```
{'neg': 0.0, 'neu': 0.318, 'pos': 0.682, 'compound': 0.6468}
```

In [11]:
```
sia.polarity_scores('This is the worst thing ever.')
```

Out[11]:
```
{'neg': 0.451, 'neu': 0.549, 'pos': 0.0, 'compound': -0.6249}
```

In [12]:
```
sia.polarity_scores(example)
```

Out[12]:
```
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

In [13]:
```
# Run the polarity score on the entire dataset
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    text = row['Text']
    myid = row['Id']
    res[myid] = sia.polarity_scores(text)
```

```
100% 500/500 [00:00<00:00, 901.30it/s]
```

In [14]:
```
vaders = pd.DataFrame(res).T
vaders = vaders.reset_index().rename(columns={'index': 'Id'})
vaders = vaders.merge(df, how='left')
```

In [15]:
```
# Now we have sentiment score and metadata
vaders.head()
```

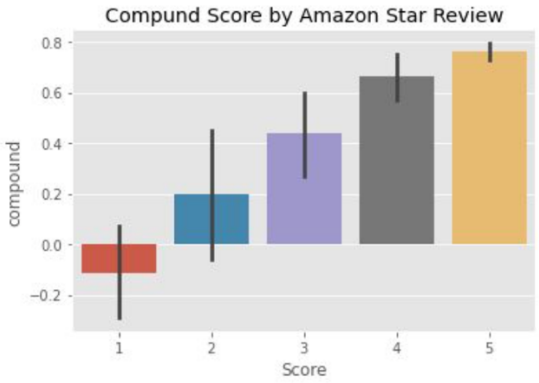
Out[15]:

	Id	neg	neu	pos	compound	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfu
0	1	0.000	0.695	0.305	0.9441	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	2	0.079	0.853	0.068	-0.1027	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0
2	3	0.091	0.754	0.155	0.8265	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1
3	4	0.000	1.000	0.000	0.0000	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3
4	5	0.000	0.552	0.448	0.9468	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0

Plot VADER results

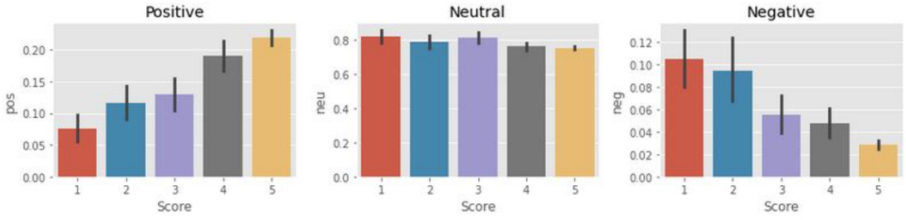
In [16]:

```
ax = sns.barplot(data=vaders, x='Score', y='compound')
ax.set_title('Compund Score by Amazon Star Review')
plt.show()
```



In [17]:

```
fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0])
sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1])
sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2])
axs[0].set_title('Positive')
axs[1].set_title('Neutral')
axs[2].set_title('Negative')
plt.tight_layout()
plt.show()
```



Step 3. Roberta Pretrained Model

- Use a model trained of a large corpus of data.
- Transformer model accounts for the words but also the context related to other words.

In [18]:
```
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax
```

In [19]:
```
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

```
Downloading: 100% 747/747 [00:00<00:00, 23.7kB/s]

Downloading: 100% 878k/878k [00:00<00:00, 2.29MB/s]

Downloading: 100% 446k/446k [00:00<00:00, 818kB/s]

Downloading: 100% 150/150 [00:00<00:00, 5.03kB/s]

Downloading: 100% 476M/476M [00:22<00:00, 23.4MB/s]
```

In [20]:
```
# VADER results on example
print(example)
sia.polarity_scores(example)
```

```
This oatmeal is not good. Its mushy, soft, I don't like it. Quaker Oats is the way to go.
```

Out[20]:
```
{'neg': 0.22, 'neu': 0.78, 'pos': 0.0, 'compound': -0.5448}
```

In [21]:
```
# Run for Roberta Model
encoded_text = tokenizer(example, return_tensors='pt')
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
    'roberta_neg' : scores[0],
    'roberta_neu' : scores[1],
    'roberta_pos' : scores[2]
}
print(scores_dict)
```

```
{'roberta_neg': 0.9763551, 'roberta_neu': 0.020687457, 'roberta_pos': 0.0029573673}
```

In [22]:
```
def polarity_scores_roberta(example):
    encoded_text = tokenizer(example, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
        'roberta_neg' : scores[0],
        'roberta_neu' : scores[1],
        'roberta_pos' : scores[2]
    }
    return scores_dict
```

In [23]:
```
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    try:
        text = row['Text']
        myid = row['Id']
        vader_result = sia.polarity_scores(text)
        vader_result_rename = {}
        for key, value in vader_result.items():
            vader_result_rename[f"vader_{key}"] = value
        roberta_result = polarity_scores_roberta(text)
        both = {**vader_result_rename, **roberta_result}
        res[myid] = both
    except RuntimeError:
        print(f'Broke for id {myid}')
```

In [23]:

```
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    try:
        text = row['Text']
        myid = row['Id']
        vader_result = sia.polarity_scores(text)
        vader_result_rename = {}
        for key, value in vader_result.items():
            vader_result_rename[f"vader_{key}"] = value
        roberta_result = polarity_scores_roberta(text)
        both = {**vader_result_rename, **roberta_result}
        res[myid] = both
    except RuntimeError:
        print(f'Broke for id {myid}')
```

```
100% 500/500 [01:42<00:00, 3.62it/s]
```

```
Broke for id 83
Broke for id 187
```

In [24]:

```
results_df = pd.DataFrame(res).T
results_df = results_df.reset_index().rename(columns={'index': 'Id'})
results_df = results_df.merge(df, how='left')
```

Compare Scores between models

In [25]:

```
results_df.columns
```

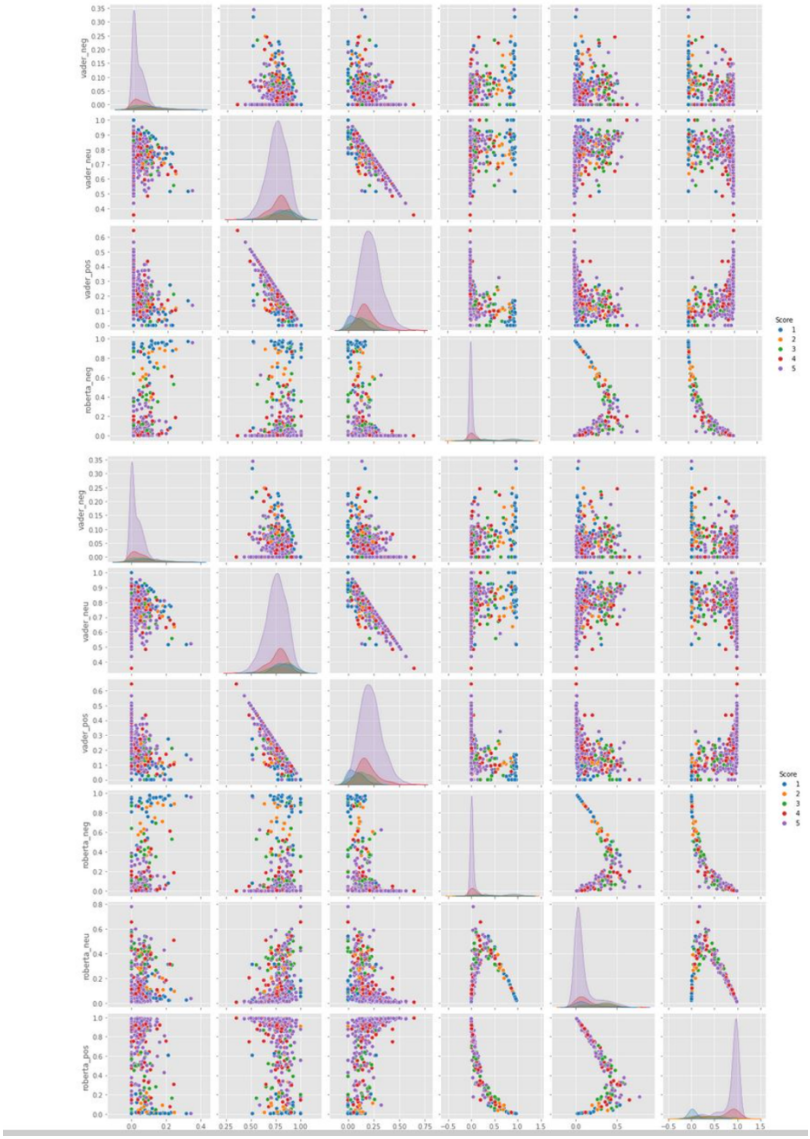
Out[25]:

```
Index(['Id', 'vader_neg', 'vader_neu', 'vader_pos', 'vader_compound',
       'roberta_neg', 'roberta_neu', 'roberta_pos', 'ProductId', 'UserId',
       'ProfileName', 'HelpfulnessNumerator', 'HelpfulnessDenominator',
       'Score', 'Time', 'Summary', 'Text'],
      dtype='object')
```

Step 3. Combine and compare

In [26]:

```
sns.pairplot(data=results_df,
             vars=['vader_neg', 'vader_neu', 'vader_pos',
                   'roberta_neg', 'roberta_neu', 'roberta_pos'],
             hue='Score',
             palette='tab10')
plt.show()
```



In [27]:
```
results_df.query('Score == 1') \
    .sort_values('roberta_pos', ascending=False)['Text'].values[0]
```

Out[27]:
```
'I felt energized within five minutes, but it lasted for about 45 minutes. I paid $3.99 for
this drink. I could have just drunk a cup of coffee and saved my money.'
```

In [28]:
```
results_df.query('Score == 1') \
    .sort_values('vader_pos', ascending=False)['Text'].values[0]
```

Out[28]:
```
'So we cancelled the order.  It was cancelled without any problem.  That is a positive not
e...'
```

In [29]:
```
# nevative sentiment 5-Star view
```

In [30]:
```
results_df.query('Score == 5') \
    .sort_values('roberta_neg', ascending=False)['Text'].values[0]
```

Out[30]:
```
'this was sooooo deliscious but too bad i ate em too fast and gained 2 pds! my fault'
```

In [31]:
```
results_df.query('Score == 5') \
    .sort_values('vader_neg', ascending=False)['Text'].values[0]
```

Out[31]:
```
'this was sooooo deliscious but too bad i ate em too fast and gained 2 pds! my fault'
```

Extra: The Transformers Pipeline

- Quick & easy way to run sentiment predictions

In [32]:
```
from transformers import pipeline

sent_pipeline = pipeline("sentiment-analysis")
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english (http
s://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english)
```

```
Downloading: 100% 629/629 [00:00<00:00, 20.9kB/s]

Downloading: 100% 255M/255M [00:12<00:00, 23.0MB/s]

Downloading: 100% 48.0/48.0 [00:00<00:00, 1.42kB/s]

Downloading: 100% 226k/226k [00:00<00:00, 897kB/s]
```

In [33]:
```
sent_pipeline('I love sentiment analysis!')
```

Out[33]:
```
[{'label': 'POSITIVE', 'score': 0.9997853636741638}]
```

In [34]:
```
sent_pipeline('Make sure to like and subscribe!')
```

Out[34]:
```
[{'label': 'POSITIVE', 'score': 0.9991742968559265}]
```

In [35]:
```
sent_pipeline('booo')
```

Out[35]:
```
[{'label': 'NEGATIVE', 'score': 0.9936267137527466}]
```