## **Managing Big Data Final Project** *Team 4(04): Frank Fan, Carl Xi, Yaping Zhang, Jie Zhu*

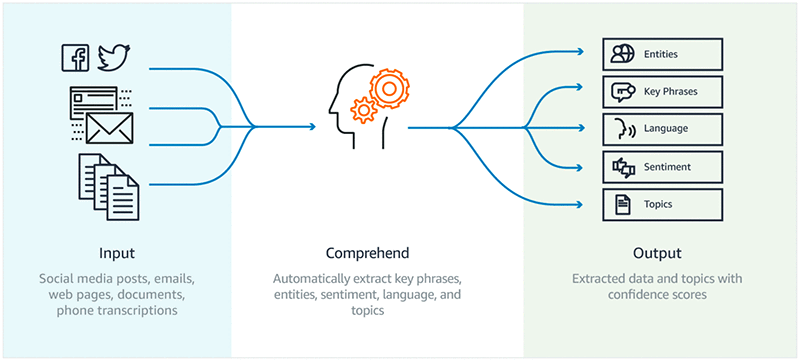
## **Introduction to Amazon Comprehend**

### What is Amazon Comprehend:

Amazon Comprehend utilizes NLP(natural language processing) to draw insights from the content of text documents. Amazon Comprehend is able to process any text file in UTF-8 format. Because Amazon Comprehend

uses a pre-trained model to examine and analyze a document or set of documents to gather insights about it. This model is continuously trained on a large body of text so that there is no need for us to provide training data.

### How does it work:



Amazon Comprehend takes social media posts, emails, web pages, phone transcriptions or any other documents you want to analyze, and automatically develops insights by extracting key phrases, identifying languages and recognizing entities and sentiments.

### Key Features:

# Detecting the Dominant Language

Amazon Comprehend identifies the dominant language in a document. Amazon Comprehend can identify over 100 languages.

# Detecting Named Entities

Amazon Comprehend returns a list of entities, such as people, places, and locations, identified in a document.

## Keyphrase Extraction

Amazon Comprehend extracts key phrases that appear in a document. For example, a document about a basketball game might return the names of the teams, the name of the venue, and the final score.

* Sentiment Analysis

Amazon Comprehend determines the emotional sentiment of a document. Sentiment can be positive, neutral, negative, or mixed.

* Customized Classification

Apart from the features above, Amazon Comprehend also allows us to create a customized text classification model. For example, This function enables us to automatically categorize input inquiry data by the problem type based on how the customer described the problem. It is very simple to create a custom model: we just need to provide documents for each of the labels we want to use, and Amazon Comprehend will train on these labels and documents to set the classifier.

## **Amazon Comprehend Workshop**

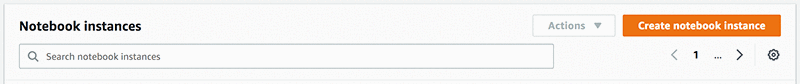
We would love to share this part with our MSBA cohorts, who plan to use NLP for their capstone project.

### Set up the environment:

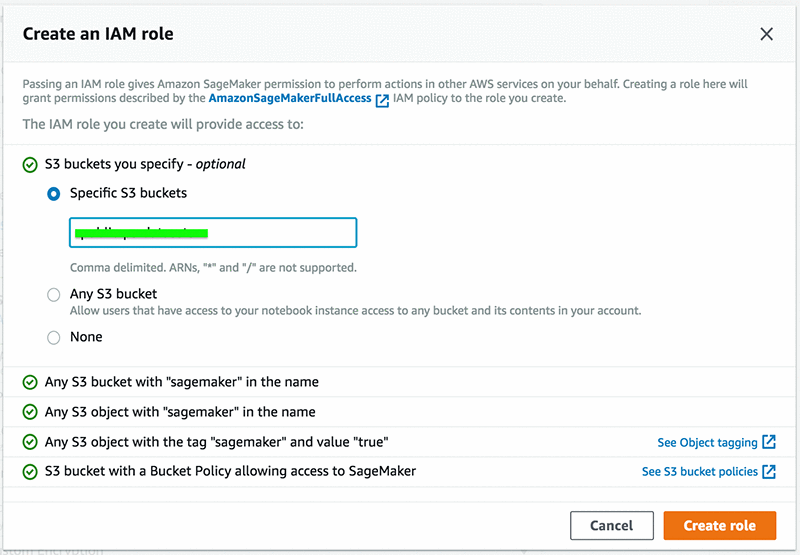
After experimenting with different approaches to use Amazon Comprehend, we found the easiest way is to use Amazon Comprehend using a Jupyter Notebook based on Amazon SageMaker. The advantage is that we get to use Amazon Comprehend in a Python 3 environment that we are familiar with.

* Step 1: Set up your Amazon SageMaker notebook

From the AWS Management Console, choose Services and then Amazon SageMaker under Machine Learning, and in the Amazon SageMaker console, under Notebook, choose Notebook instances. Now choose the Create Notebook Instance.



In the settings, we can specify the size based on your needs. But we need to create an IAM role that gives access to any necessary Amazon S3 buckets.



After successfully creating an IAM role, just click on Create notebook instance. After a few minutes, the notebook instance will be ready to use.

* Step 2: Attach Policy

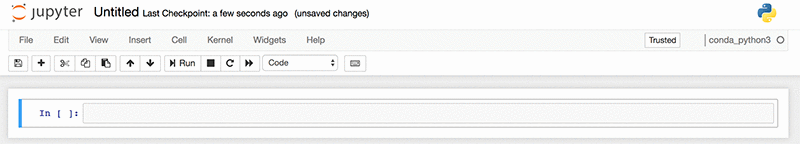
In the notebook settings, we need to choose IAM role ARN to open the IAM role attached to the notebook instance. After clicking on Attach policies, we just need to search for ComprehendFullAccess to attach it.





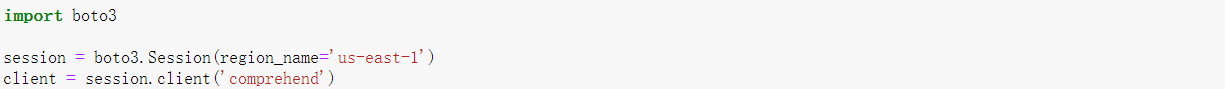
* Step 3: Create a notebook

After successfully launching the notebook instance, we should choose new and choose conda\_python3 to create a notebook in Python 3 environment.



* Step 4: Connect to Amazon Comprehend

Then we can use the AWS SDK for Python SDK (Boto3) to connect to Amazon Comprehend from your Python code base. Boto is the Amazon Web Services (AWS) SDK for Python. It enables Python developers to create, configure, and manage AWS services, such as S3 and Amazon Comprehend. Using the following command, we import boto3 and connect to Amazon Comprehend in a specified AWS Region using the boto3 client.



Remember, we need to specify the AWS region in order to connect to Amazon Comprehend. Now we are ready to go.

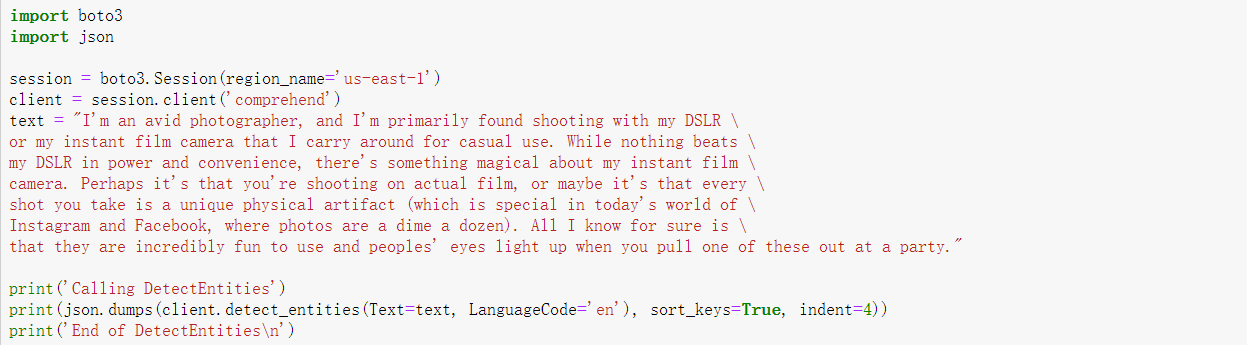
### Detect the dominant language:

Amazon Comprehend is able to identify text written in over 100 languages and returns the dominant language with a confidence score to support that a language is dominant. By calling the detect\_dominant\_language function on the text you want to analyze, Amazon comprehend will output the dominant language and the corresponding confidence score.



### Entity Recognition:

By calling the Amazon Comprehend detect\_entities function, it will return the named entities ("People," "Places," "Locations," etc.) that are automatically categorized based on the provided text, as well as a confidence score and count.

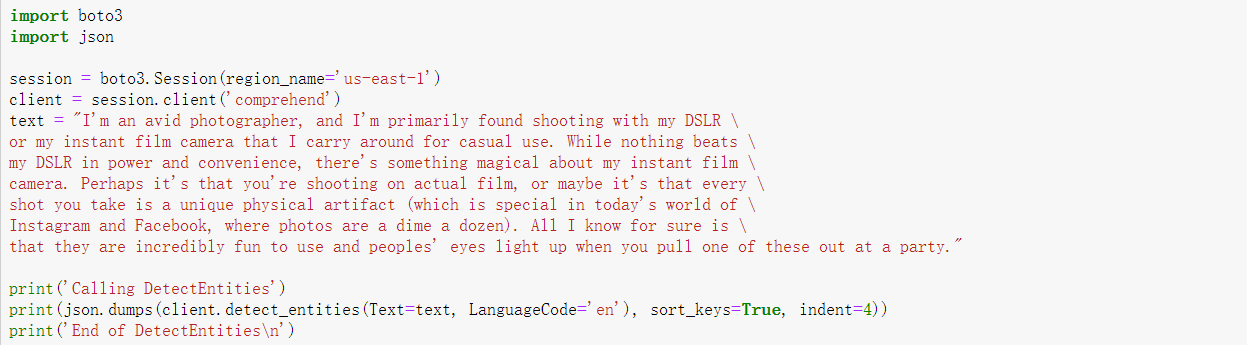


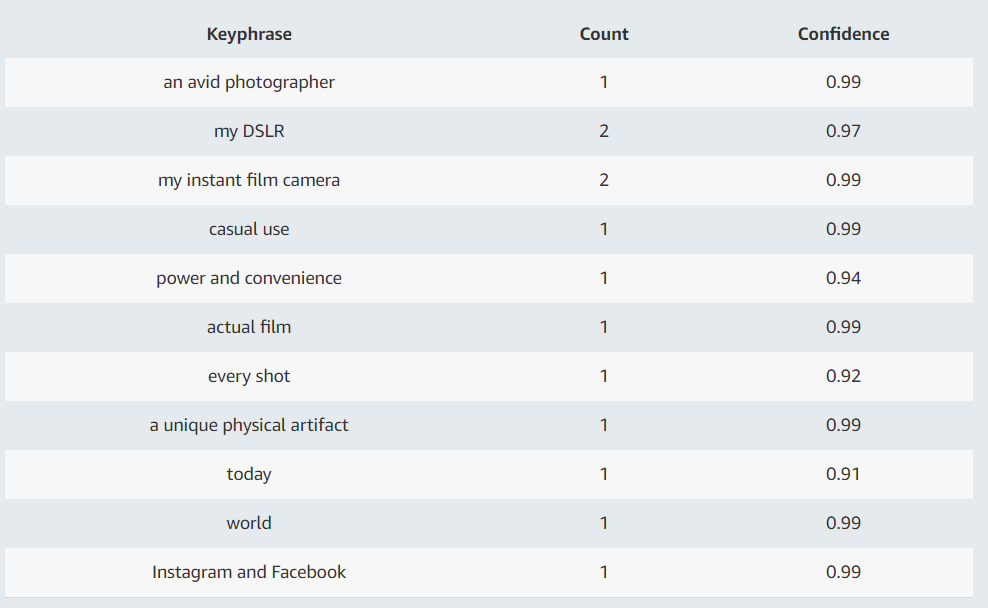
Result Table:



### Key Phrase Extraction:

By calling the Amazon Comprehend detect\_entities function, it will return the key phrases or talking points and a confidence score to support that this is a key phrase.

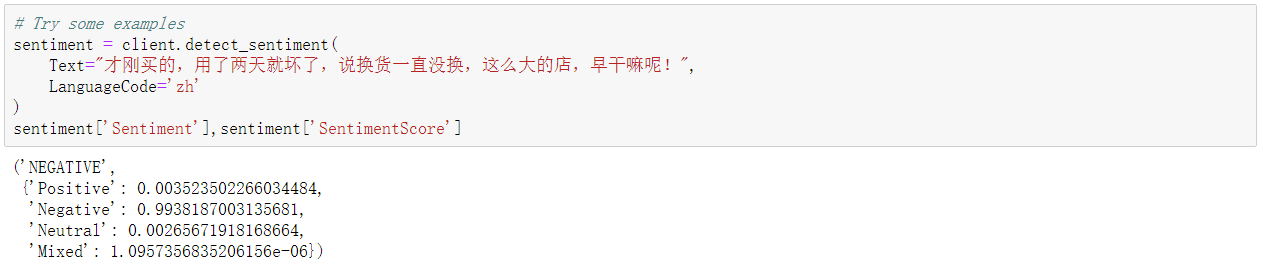




### Sentiment Analysis:

And last but not least, Amazon Comprehend is able to detect the sentiment of our input text. While using, we need to specify the language that we want to analyze. And the output will be the sentiment (positive, negative, mixed and neutral) and its confidence score.

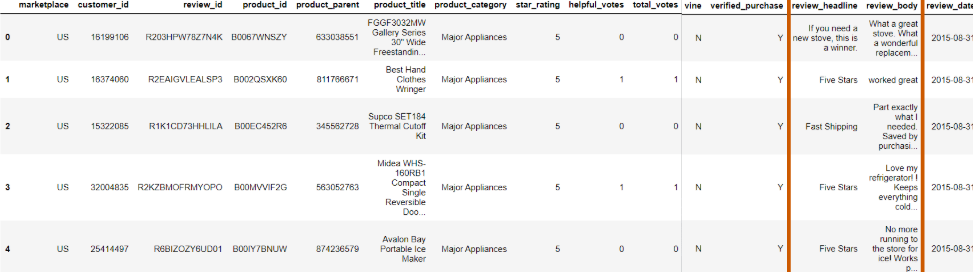




## **Amazon Reviews Data**

We then used Amazon Customer Reviews as our dataset to try out Amazon Comprehend as a big data analytic tool. Amazon Customer Reviews (a.k.a. Product Reviews) is one of Amazon’s iconic products. In a period of over two decades since the first review in 1995, millions of Amazon customers have contributed over a hundred million reviews to express opinions and describe their experiences regarding products on the Amazon.com website. Over 130+ million customer reviews are available to researchers as part of this dataset.

### Features:

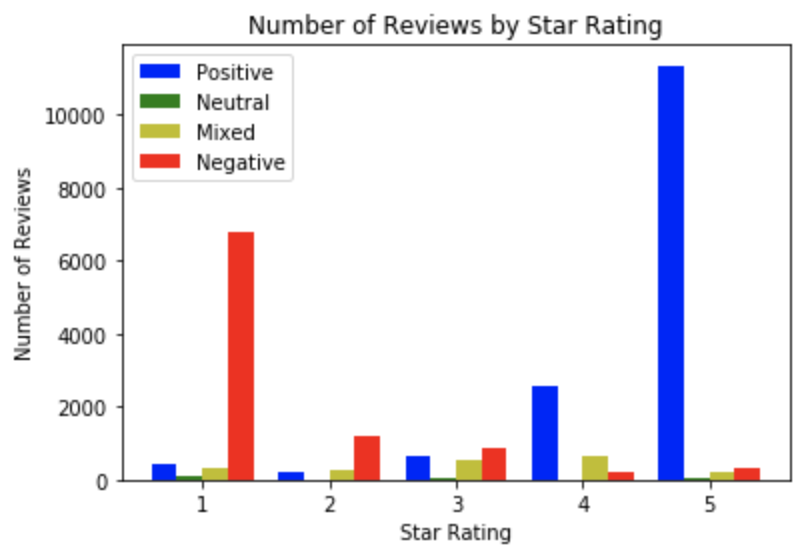


Most of the features are intuitive. Among these, ‘vine’ means vine review. Basically, Amazon Vine invites the most trusted reviewers on Amazon to post opinions about new and pre-release items to help their fellow customers make informed purchase decisions. Amazon invites customers to become Vine Voices based on their reviewer rank, which is a reflection of the quality and helpfulness of their reviews as judged by other Amazon customers.

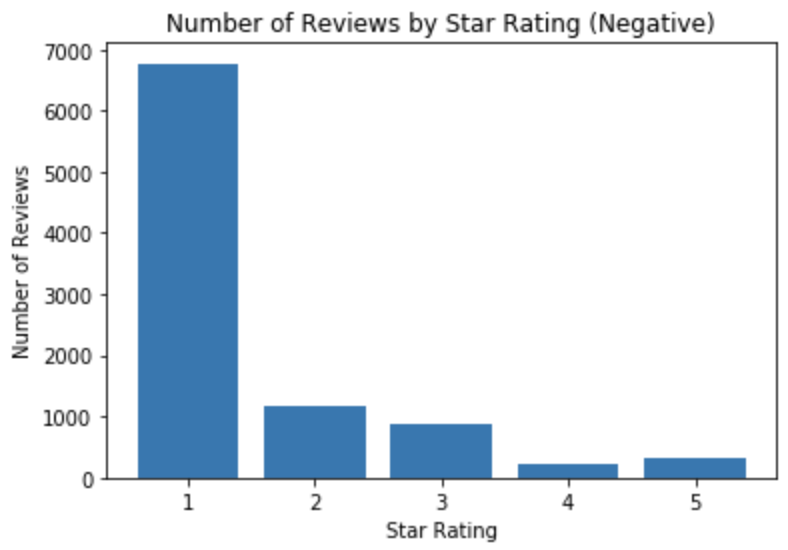
### Data Cleaning:

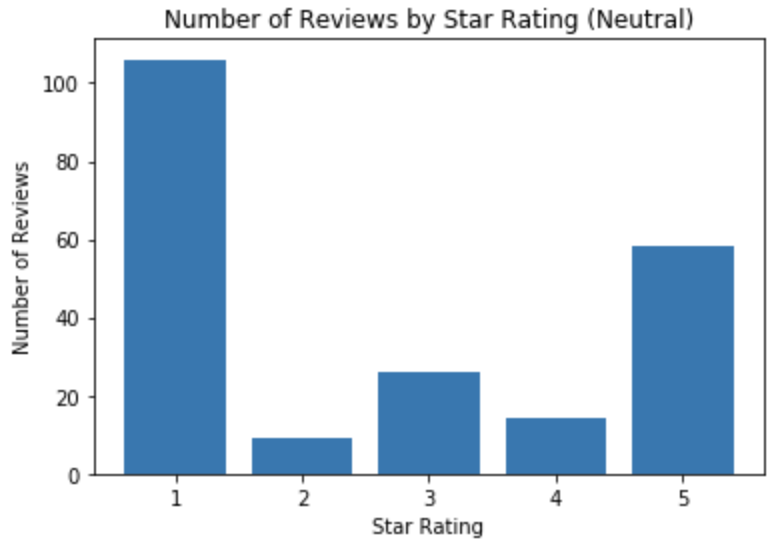
We filtered out all reviews that are under 10 characters and over 5000 characters, the Amazon Comprehend max character length. And we only kept the purchases happening in the US market for convenience.

## **Sentiment Irregularities**

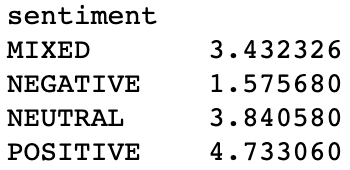
With everything setup, we can now take a look at the average ratings by category and sentiment. Firstly, we compiled the review sentiments by star rating for the Major Appliances category, as shown below:

From visual inspection, we can immediately see that inflection point between positive and negative review occur at 3-star reviews, which makes sense. What doesn’t make sense is that there are positive reviews with very low star ratings and negative reviews with very high star ratings. In addition, mixed reviews, which by definition should be around 2-4 in ratings, are sporadically all over the place. If we break apart the above graph into histograms by each rating, this effect is seen even more clearly:

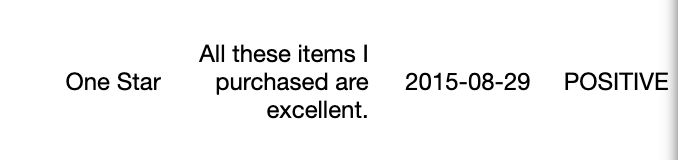




With the average star rating at 3.8853, it seems like the average scores of each sentiment fall within expected norms.



Upon closer inspection, the reasoning behind illogical sentiment-rating pairings become evident. Reviews are written by humans, and humans make mistakes. Take this review below for an example. With an extremely positive comment, the one-star rating can only be explained by human error. In fact, it becomes apparent that this is the one factor that data has a hard time encapsulating.

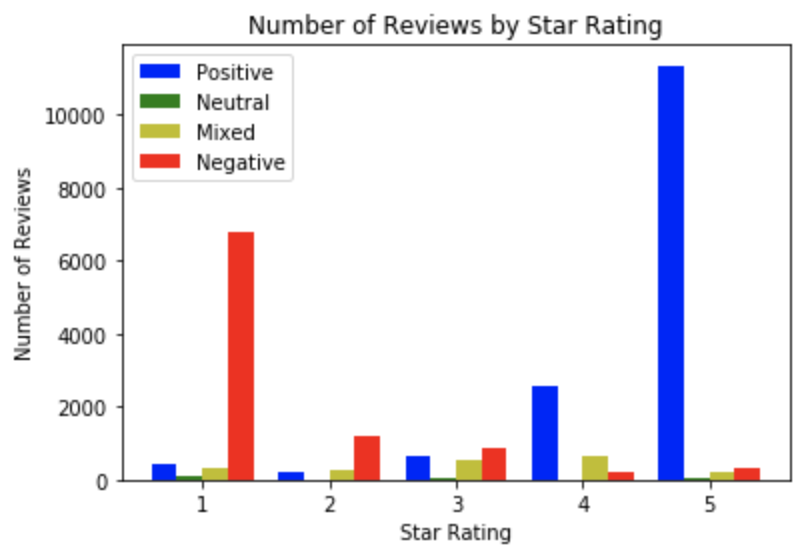


As we will see when we examine another category, other factors also come into play that makes gives Amazon Comprehend a lot of space for improvement.

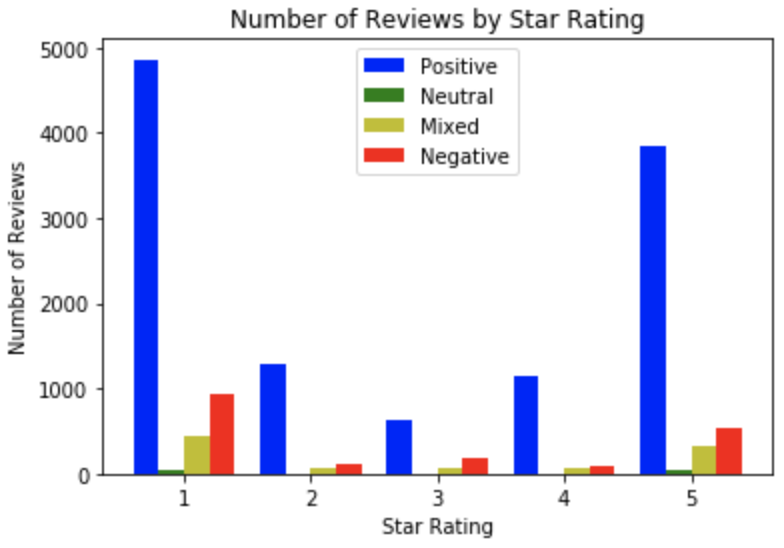
1. **Cross-Category Analysis**

Let’s take a look at how ratings by sentiment differ by category:

**Major Appliances :**

****

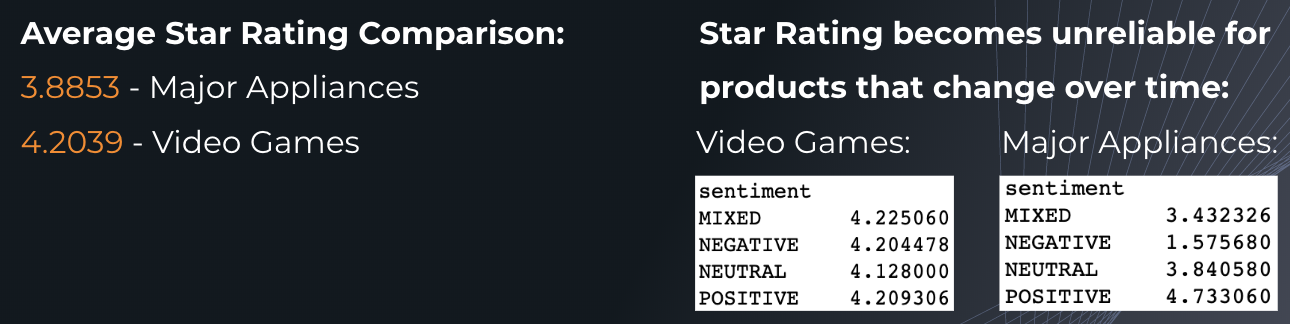
**Video Games:**

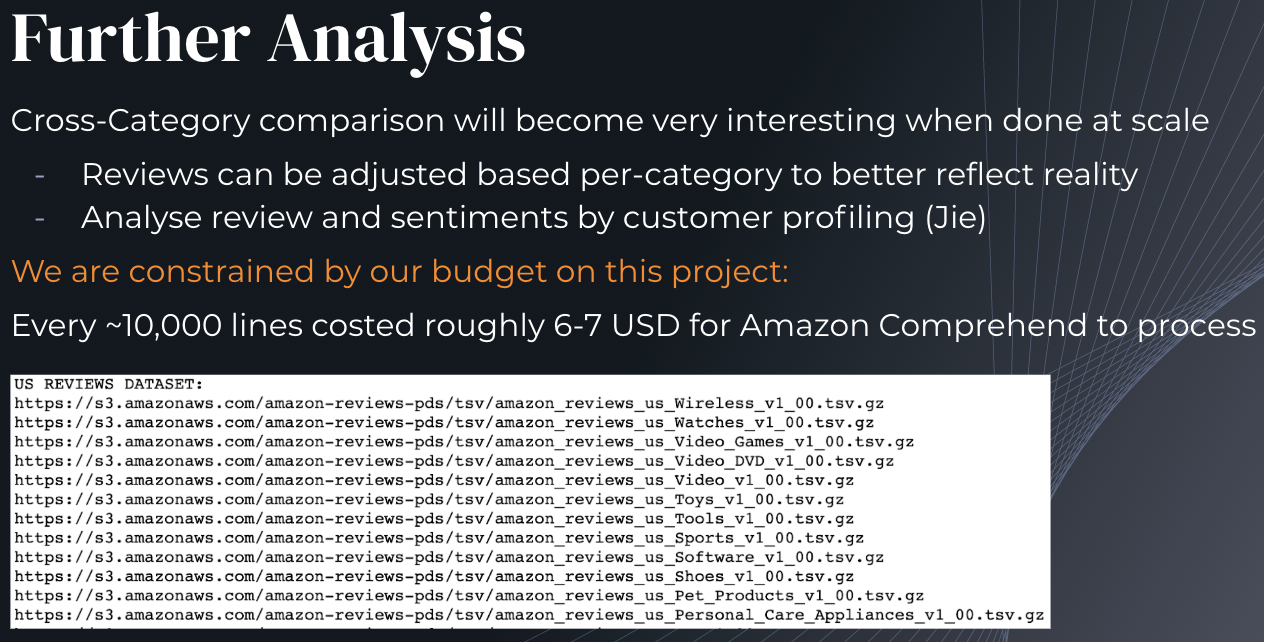
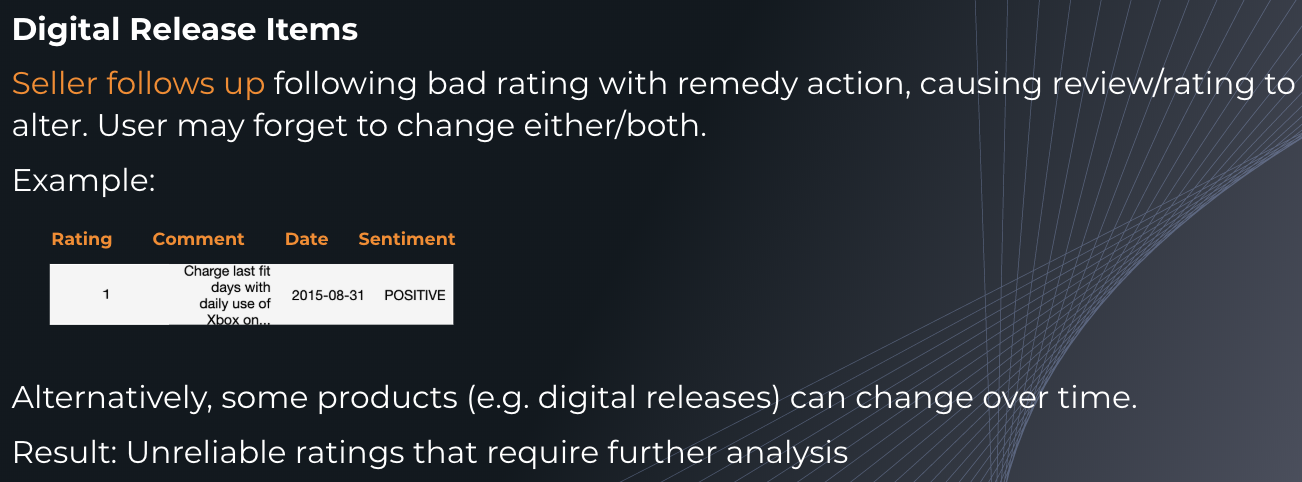
****

Right off the bat, we can see that sentiment analysis for Video Games is completely screwed up. Positive review forms a concave parabola from 1-5 star ratings. Rating Expectations greatly vary depending on the product category. This could be due to various factors, such as:

* availability of choice (20,000 racing games vs 5 washing machines)
* ease of review (finding scratches on product vs visual glitch in-game)
* tolerance level (you can restart a game but not a product)

Here are some snippet from our presentation that further identify this problem. In summary, due to the change in digital products after release, reviews scores and sentiment scores are unreliable for digital products.

****

****

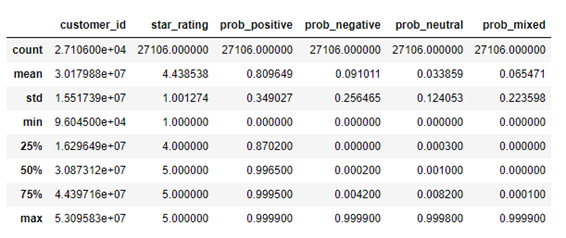
1. **User Analysis**
   1. **Data Used**

In this Amazon Reviews Database, there are 46 US reviews datasets, ranging from wireless products to apparels. In addition, there are 5 multilingual reviews datasets from Amazon’s US, UK, Japan, France, and Germany websites. A complete list of these datasets can be found at<https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt>.

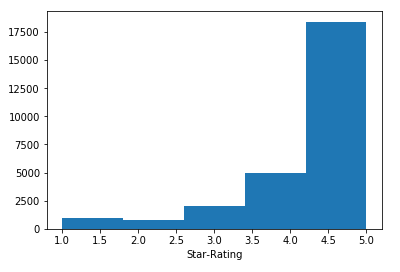
For user analysis, we choose 4 US reviews datasets, including Major Appliances, Furniture, Watches, and Musical Instruments. After being concatenated, there are 2,752,353 rows in total. For the purpose of analyzing user behavior, we only keep customers who have more than 30 reviews in these product categories. This leaves us with 27,106 reviews and 569 unique customers.

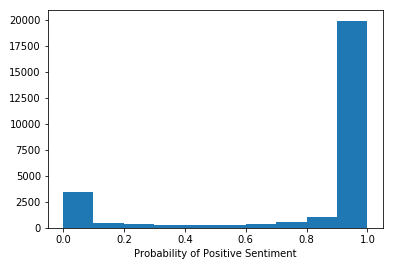
* 1. **Data Description**

After dataset concatenation, we append sentiment results and sentiment statistics. Sentiment statistics consist of positive sentiment probability, negative sentiment probability, neutral sentiment probability, mixed sentiment probability. These statistics measure the probability of the review is one of these sentiment results and should add up to 1 for each review. We will focus primarily on star-rating and positive-probability in this analysis.

****

As is shown above, for these four product categories, the average star-rating is 4.44 stars, and the average probability of positive sentiment is 80.96%. The *star\_rating* variable ranges from 1 to 5, while the *prob\_positive* covers a variety of 0.00% to 99.99%. Most of the reviews have 5 stars and a probability of positive sentiment over 99%. Histograms for *star\_rating* and *prob\_positive* also confirm this finding.

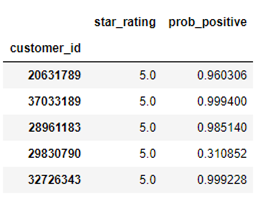
****

****

* 1. **User Behaviors**

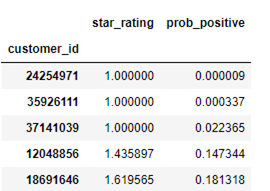
After grouping data by *customer\_id*, we have the following interesting findings.

**i) Highest Star-Rating Givers**

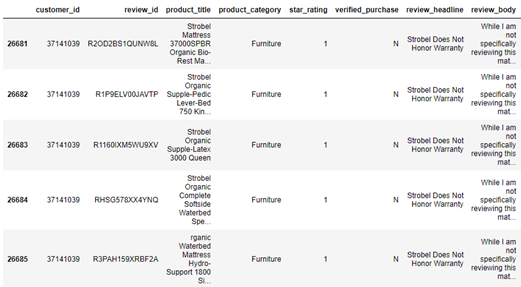
****

Among all 569 customers with more than 30 reviews, 51 customers always give 5 stars, and most of them have also very high average positive sentiment probability. However, some customers have low positive sentiment probability even though they give 5 stars frequently. In other words, they rate 5-star even when they are not that satisfied. A good example here is customer 29830790, who has an average *star\_rating* of 5.0 but with an average *prob\_positive* of only 31.09%.

**ii) Lowest Star-Rating Givers**

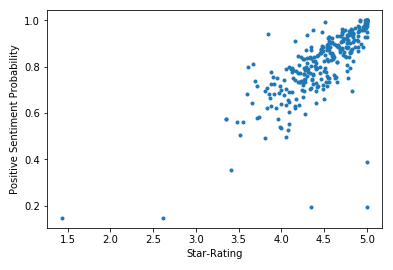
****

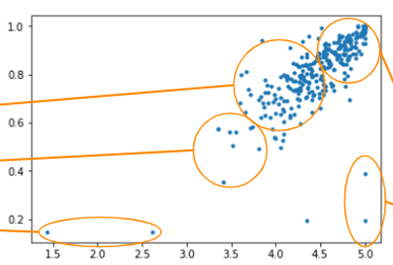
There are 3 customers always giving 1-star. Among these three, customer 24254971 and 35926111 only reviewed watches, while customer 37141039 only reviewed mattresses in the furniture product category. All their reviews are without Verified Purchase. Since it makes no sense for a customer to buy more than 30 mattresses from one single brand, we conclude these reviews are made on purpose by competitors.

****

**iii) Customers Reviewed in Multiple Categories**

In order to analyze customers’ reviews across multiple categories, we further limit the data to reviews by customers who reviewed in 2 or more categories. This leaves us with 285 customers and 14,442 reviews. We then plot a Star-Sentiment plot using these reviews.

****

**iv) Customers are separated into the following groups.**



## **TextBlob Introduction**

## Aside from Amazon Comprehend, there are other NLP tools like:

· NLTK

· Spacy

· Stanford Core NLP

· TextBlob

They have their own advantages and disadvantages and can be utilized on different cases, such as what kinds of text analysis you want to perform and what your data looks like.

However, in general, TextBlob is easier to use because it has a nicer user interface and its documentation is thoroughly explained. There, it is more new-user friendly and we will emphasize on this new and popular NLP tool, TextBlob.

First of all, it is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

Secondly, TextBlob is built on the shoulders of NLTK, google translator etc. to serve various features like:

· Part-of-speech tagging

· Noun phrase extraction

· Sentiment analysis

· Classification (by using Naive Bayes, Decision Tree)

· Language translation and detection powered by Google Translate

· Spelling correction

· Tokenization (splitting text into words and sentences)

· Word and phrase frequencies

· Parsing

· n-grams

· Word inflection (pluralization and singularization) and lemmatization

· Add new models or languages through extensions

· WordNet integration

And we will use sentiment analysis from TextBlob in Pyspark environment to analysis our Amazon review data and compare the results with the two.

**Sentiment Analysis Using TextBlob:**

The general pipeline of sentiment analysis is: data collection, text preparation, sentiment detection, sentiment classification and result presentation. And today we will go over those steps using PySpark and TextBlob.

· Step 1: Data Collection

After setting up Spark context and SparkSession, we can load the same Amazon review data as we used in Amazon Comprehend.

· Step 2: Text Preparation

o remove non-ASCII characters

o fixed abbreviation

o remove irrelevant features:

Specifically, here we need to convert all text to lowercase, remove hyperlinks, remove @mentions, remove punctuation, remove numeric 'words' and remove non a-z 0-9 characters and words shorter than 1 characters.

· Step 3: Sentiment Detection

After text preparation, we then can get a cleaner text data for sentiment detection. In TextBlob, the textblob.sentiments module contains two sentiment analysis implementations. The first one is PatternAnalyzer (based on the pattern library) and the second one is NaiveBayesAnalyzer (an NLTK classifier trained on a movie reviews corpus).

Here, we will use PatternAnalyzer to detect the sentiment. It will return results as a named tuple of the form: Sentiment (polarity, subjectivity, [assessments]). Specifically, polarity is float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement. Subjectivity, ranging from 0 to 1, generally refer to personal opinion, emotion or judgment whereas objective refers to factual information.

After defining polarity as our sentient score, we can get the 10 rows of results.

· Sentiment Classification:

Instead of using the NaiveBayesAnalyzer implemented in TextBlob (reason: 1: it only has two categories: negative and positive, and here, it’s better we also have the neutral category; 2: it is trained based on movie review data, and it’s different from the Amazon review data( mostly are about various products) , and using it as the training data can give some bias; 3: the naïve bayes has an assumption: features are independent from each other, and it’s very likely not true in our case.), we can just classify sentiment categories based on their sentiment score.

·

**Compare TextBlob with Amazon Comprehend：**

|  |  |  |
| --- | --- | --- |
|  | Amazon Comprehend | TextBlob |
| Positive | **6696** | **6911** |
| Neutral | **69** | **2331** |
| Negative | **2306** | **758** |
| Mixed | **749** |  |

After getting the count of each sentiment category, we can see that TextBlob gives about the same amount of positive detections as Amazon Comprehend. It gives more Neutral results and fewer negative results. Since TextBlob can’t be used to detect mixed feelings, it gives 0 result for mixed sentiment detections.

|  |  |  |
| --- | --- | --- |
| **Review Text** | **Amazon Comprehend** | **TextBlob** |
| Love my refrigerator! ! Keeps everything cold. will recommend! | Positive | Neutral |
| AS advertised | Positive | Neutral |
| It's not worth 22 dollars, I've heard it became of some value just not that high. | Negative | Neutral |
| Did the job but didn't match the original gray wheels. | Mixed | Positive |
| Cheap knock-off. Don’t waste your time | Negative | Positive |

However, simple count summary value can’t help us determine which tool is better. We need to dive down to the review text itself. After randomly choosing some reviews, we find out that:

Amazon Comprehend has a better sentiment detection accuracy. The possible reasons are that it’s based on Machine Learning while TextBlob(PatternAnalyzer) is based on dictionary) and Amazon comprehend has a custom set of entities or text classification models that are tailored uniquely to text data. Further, since Amazon Comprehend has the level “Mixed”, it can better classify reviews. By comparison, the only advantage of TextBlob we can find is that it is free.