Amazon
Business Analytics Capstone Final Report

Jiao Cong, Gabriella Giancola, John Lin, Zhongting Wang
Lally School of Management, Rensselaer Polytechnic Institute
Dr. T Ravichandran
May 6, 2022

Contents

1 Introduction

1.1 Executive Summary

Amazon's current recommendation system on their website utilizes an item to item collaborative filtering method which takes into account a user's past behaviors on the website to attempt to predict and recommend what the user might need [1]. The purpose of the Rensselaer Polytechnic Institute (RPI) team was to build a more robust system that would utilize descriptive features of a searched item, the brand, the keywords, and even use parts of the images of the product to recommend similar items to the user. In doing this, the team was able to create a model that used the item description, brand, and keywords, but unfortunately fell short in the image model analysis aspect.

Before the team was able to create a model, an Exploratory Data Analysis (EDA) was done to determine underlying trends in the data. Through this, the team was able to determine the more important features in the dataset, and a thorough cleaning was executed on the data to ensure efficiency in the data when the model was run. Next, research was conducted to determine the best Natural Language Processing (NLP) technique to use that would result in the best recommendation system we could make. Techniques such as stemming, semantic reasoning, cosine similarity, and Bert were looked at, but the model ultimately used cosine similarity and Bert. Cosine similarity was essential to the model because it is a metric used to measure how similar documents are based on the angle between them. Then Bert was used with Bert Tokenizer because Bert has a wide variety of usages which is where Tokenizer was found. This method used Pytorch pretrained Bert and then Tokenizer was used to encode words into data that the model could use. The Hugging Face package in Tokenizer was utilized to convert text into a tensor "vector."

The modeling approach used for the final model was done in five major steps. First, it was crucial to define the target product, which was determined to be the product ID. Then, since the dataset was very large, the team subset the data into the same category and country as the product. This helped in analyzing similar products. The third step was to encode the target product to be compared into machine processable values. This was done because machine learning tools cannot interpret text in the data's original form so Tokenization was done using the Hugging Face package. The fourth step of this was to calculate the cosine distance between the target variable and the compared products. Using cosine distance, the model produced a score for the team to determine how similar the compared products were. Finally, the last step was to select the products that were similar to the target product based on the cosine score calculated. This model created the mean measures for the average cosine similarity score between each text column to represent similarity.

Ultimately, the team made a model that found cosine similarity scores between products in the dataset, and the value that was input in the model. There was no accuracy metric found for the model as a whole due to time constraints. The team attempted to use a variety of functions that were part of sentence transformers to find the model's accuracy, but nothing resulted in working to the standards of the team. For future work on this project, the team would recommend trying to implement an F-score to determine the model's accuracy. F-scores do show a model's accuracy, but more research would need to be done to implement this successfully to determine if this was truly the best fit for the model and data that was worked with in this project.

1.2 Problem Statement and Background

Amazon utilizes a system to recommend items to users based on previous purchases or search history. This is item to item collaborative filtering and means that user's purchases and rated items are used to recommend similar items to buyers [1]. The Amazon project's main goal for the team is to build the most robust recommendation system in the Amazon online shopping portal. In doing this, feature abstraction will be utilized to find similar products among a few different product attributes. The team had a goal to create a model to obtain similar products based on item description, images, brands, and even keywords. Doing this, the model will be able to recommend similar products in all categories and work just as accurately, and even better than the current system that is used. This model will be created in Python and use an input of an item ID. Then the algorithm would tell the model to produce a number of items that were similar to the item that was input. The results of the model would be what the system would recommend the user.

1.3 Project Timeline

There were multiple milestones the team identified in the beginning of the project that were goals to accomplish by the completion of the Amazon project. The first and most important aspect of the project was sorting through the metadata that was presented to the team by the Amazon team. In doing this, the team had to look at what categories were present in the data to get a sense of what data they would be working with. The next milestone was to build the metadata model and ensure it was working accurately. This model would be built using Python, and the way the team intended it to work was that a user would be able to input an item ID and then the result would be a list of items that were closely related to the item.

Similarly, the team intended to create a model for image data that would be able to be used to abstract information from images to be able to recommend items based on this aspect as well. This would create an even more accurate model. Unfortunately, the team was unable to continue with this aspect of the model creation so all efforts were directed towards making the metadata model work as accurately as possible. The final model was to combine all data and information collected and have a working model by the end of the project. This meant to have a model that

would recommend items as accurately as possible based on features in the dataset resulting in a similarity score.

2 Problem Statement

2.1 Algorithms and Shortcomings of Current Amazon System

Amazon currently uses a recommendation system that uses item to item collaborative filtering [1]. This uses a customer's past purchases and rated items and matches them to recommend similar items. Amazon also utilizes their items frequently bought together algorithm which ultimately encourages the customer to buy more in that transaction. They use a system that really looks at browsing history and recently viewed items to try to predict what the user is looking to buy the next time they go to the Amazon website. Although this system works well, Amazon was looking for a model that is a more robust version of this recommendation system so customers would be viewing a variety of items that are similar to their searches and previous purchases. Amazon is a powerhouse in the online shopping market and is always looking for ways to advance their algorithms and methods to increase profit and customer satisfaction.

2.2 Comparison of Methods and Final Selection

The model developed for this project uses a sentence transformer in order to encode the data for cosine scoring. In using this method, the team is able to produce an output of products in the dataset that are similar to the initial product that was input in the model, based on a similarity score from the model. Sentence transformers use different models such as Pytorch, Transformers, and Hugging Face as tools to compare sentences and phrases to determine the similarity between them. This model mostly utilizes the Hugging Face model to perform these tasks. Specifically, the package that is used in this model is the 'mpnet' base as it is a pre-trained model and works faster than previous models that were created, such as Bert or TensorFlow [2].

2.3 Data

The data that was presented to the team was a large set of the metadata that contained a variety of different products that Amazon sells. It was divided by different categories that items belonged to and the different countries that sold each of the products. In addition to the metadata, the team was also given a large dataset of images that included 2D original images, its small vision, and also 3D models of products that are sold on the Amazon website. Both the metadata and the image data were intended to be used simultaneously to improve and produce a robust recommendation system for Amazon.

2.4 Cleaning

Looking at the data, there were many initial issues that the team wanted to clean up to facilitate the modeling process, which would enhance the accuracy of the model. First, all lists in the data were converted to data frames, and then all the data frames were combined. Next, all columns that did not have enough data or too many missing values were removed. There were some columns that contained the same information as other columns, such as ‘Market Place Country” and ‘Domain.’ Since this data was repeated, it would not influence decisions when one was removed so the team made the decision to remove columns with the same information. The data was then flattened, which means it was cleaned, and then put into a relational format to compare the data.

Next, the ‘Product Type’ feature was analyzed because this was a significant column containing information that would essentially be crucial for the final model analysis. In looking at this feature, the team removed any product that did not contain enough data, so the baseline was that if a product had less than 200 entries, it was not necessary and could be removed. This was due to the fact that there was so much data in the initial data file. The next step was to create a text column to combine ‘Bullet Points,’ ‘item names,’ and ‘keywords’ to make a single column that would be useful to the model when it would be run in the future. Label encoder was then utilized to fit the feature ‘Product type.’ Finally, the features of ‘Item ID,’ ‘Text,’ and ‘Label’ were dropped and the data was officially cleaned to the standards of the team and was ready to use in the model that was created.

3 Observations and Research

3.1 Research Conducted

Initially looking for methods to use for the model was time intensive and required a lot of research. The methods looked at when deciding what model to build for the metadata required a lot of Natural Language Processing (NLP) methods due to the variety of words and different phrases and categories that were in the metadata. Some of the methods that were found were stemming, semantic reasoning, Parts of Speech (POS) tagging, and WordNet. Stemming is the process of extracting the base form of a word such as ‘eat,’ which would be taken from words like ‘eating,’ ‘eats,’ or ‘eaten’ [3]. This was not used because the team was not looking to extract the base words from the model, but were looking to compare items in a row to other items to determine how similar they would be based on all the columns in the data. The next NLP method looked at was Semantic Reasoning. This uses the system to infer new facts from existing data based on rules that are set, but this again was not what would help in making the model run smoothly so the team did not continue with this method [4]. The next method that was looked at briefly was POS Tagging. POS tagging marks up words in text for a particular part of speech

based on context, but it was not necessary for the model because it is used for text reading mostly [5]. The last method was WordNet. WordNet looks at words based on semantic relations in which grouping of words is better [6]. In the case of this project, semantic reasoning was not needed because this was not used.

When these methods were unsuccessful and not what would be necessary to produce a robust model, the team looked at Cosine Similarity, Pytorch, Bert, and Bert Tokenizer. These were methods that became crucial to the success of the model. Cosine Similarity was an important metric that was used because it measures how similar documents and words are in a vector form based on the angle between them [7]. This was helpful in determining how closely related objects were in the dataset. Using Pytorch, it was a useful tensor library that utilized deep learning applications to compute graphic processing units and was again, important when running our model [8]. Next, the team looked at Bert and decided it would be very beneficial to the model to use because it is a bidirectional encoder that is pre-trained with a large dataset to look for accuracy and checks for context in words [9]. Since there were multiple types of Bert to use and test, the team figured this would give a variety of ways to test the dataset and model. The team was successful when creating a model because of this. Finally, Bert Tokenizer was looked at. Bert Tokenizer uses both Pytorch and Bert to encode words into data that the model would be able to use, which made the model run better as well.

3.2 Exploratory Data Analysis

3.2.1 Basic Information

There are 16 listings, each with 28 columns and 9,232 rows. All columns contain non-numerical values. *Table 1* below displays the names of the columns in the data set that the team used.

Feature Names			
brand	item_id	item_name	product_type
domain_name	model_name	color	bullet_point
item_dimensions	item_weight	material	pattern
style	marketplace	model_year	fabric_type
item_shape	country	product_description	marketplace
spin_id	main_image _id	other_image_id	3dmodel_id
color_code	finish_type	item_keywords	model_number

Table 1: Feature Names

There are five columns that are described below that were the most important to recognize and explain.

Feature Descriptions:

1. ‘Bullet_point’: The bullet points shown when a user clicks into a product page.
2. ‘Marketplace’: There are 6 values that describe the various places users can buy items from: 'Amazon', 'AmazonFresh', 'AmazonGo', 'PrimeNow', 'WholeFoods', 'Woot'
3. ‘Domain name’: 27 URLs
4. ‘Node’: 2 values – ID and name

   Ex. [{'node_id': 5159512051, 'node_name': '/Categories/Hardware/Door Hardware & Locks/Door Levers'}]
5. ‘Product type’: The different categories of the products

3.2.2 Feature Selection

Dropped Features

There were 20 features that were dropped because they contained information that was either redundant or insignificant to the model, which can be seen in the table below. Six of the below features are image related. Those are bolded in *Table 2* below.

Dropped Features			
item_dimension	color	item_weight	material
pattern	style	marketplace	model_year
fabric_type	item_shape	product_description	country
marketplace	**main_image_id**	**other_image_id**	**spin_id**
3dmodel_id	**color_code**	**finish_type**	model_number

Table 2: Dropped Features

Retained Features

There were 8 features that were relevant to the analysis of the original dataset using similarity matching. These 8 features were retained and are listed in *Table 3* below.

Retained Features			
brand	item_id	item_name	product_type
item_keywords	domain_name	model_name	bullet_point

Table 3: Retained Features

3.2.3 Missing Values

With the model the team chose to find product similarity, missing values caused issues with the algorithm. Due to this issue, all rows with missing values were dropped in order to only include data that could be run through the model to increase success.

3.2.4 Flattening Data

Flattening data refers to the act of flattening semi-structured data, such as name-value pairs in JSON, into separate columns where the name becomes the column name that holds the values in the rows [10]. This is similar to cleaning the data to make it easier to fit a model. The images below show the flattened data in *Figure 1*.

	item_dimensions	brand	bullet_point	color	item_id
0	{'height': {'normalized_value': {'unit': 'inch...	[{'language_tag': 'es_ES', 'value': 'Pinzon by...	[{'language_tag': 'es_ES', 'value': 'Sábana ba...	[{'language_tag': 'es_ES', 'value': 'Azul'}]	B00MGSFTLE
1	NaN	[{'language_tag': 'en_IN', 'value': 'Klepe'}]	[{'language_tag': 'hi_IN', 'value': 'आउटर मटीर...	[{'language_tag': 'hi_IN', 'standardized_value...	B07WRK7NGQ
2	NaN	[{'language_tag': 'en_IN', 'value': 'Amazon Br...	[{'language_tag': 'en_IN', 'value': 'Snug fit ...	[{'language_tag': 'en_IN', 'standardized_value...	B08511RJ3W
3	NaN	[{'language_tag': 'en_IN', 'value': 'Amazon Br...	[{'language_tag': 'en_IN', 'value': 'Snug fit ...	[{'language_tag': 'en_IN', 'standardized_value...	B08511NVLY

	brand	bullet_point	item_id	item_name
0	Pinzon by Amazon	Sábana bajera ajustable - Dimensiones: 90 x 20...	B00MGSFTLE	Pinzon by Amazon 'Everyday' Spannbetttuch, Bau...
1	Klepe	आउटर मटीरियल: PU	B07WRK7NGQ	Klepe Men's Dark Grey Running Shoes-11 UK (45 ...
2	Amazon Brand - Solimo	Snug fit for Samsung Galaxy M21, with perfect ...	B08511RJ3W	Amazon Brand - Solimo Designer Two Different P...
3	Amazon Brand - Solimo	Snug fit for Samsung Galaxy M21, with perfect ...	B08511NVLY	Amazon Brand - Solimo Designer Wooden Blocks T...
4	Allegro Coffee	Kosher	B078ZMFWC3	Allegro Tea, Tea Spice Puer Yunnan Organic, 0....

Figure 1: Flattened Data

3.2.5 Categories

There were a lot of categories to work with in this dataset so to facilitate this, we removed categories that had less than 199 entries. This was to decrease the dataset to make it easier to work with and to pick out the categories that were relevant to the analysis of the data. The number of categories started at 576 and went down to 67 after this process was performed. The maximum number of entries in a category is ‘cellular_phone_case’ with 64,853, and the minimum number of entries in a category is ‘pantry’ with 201. These values can be seen in *Appendix 2*. Below in *Figure 2* is a graph to visualize the variety of categories and how many entries they had.

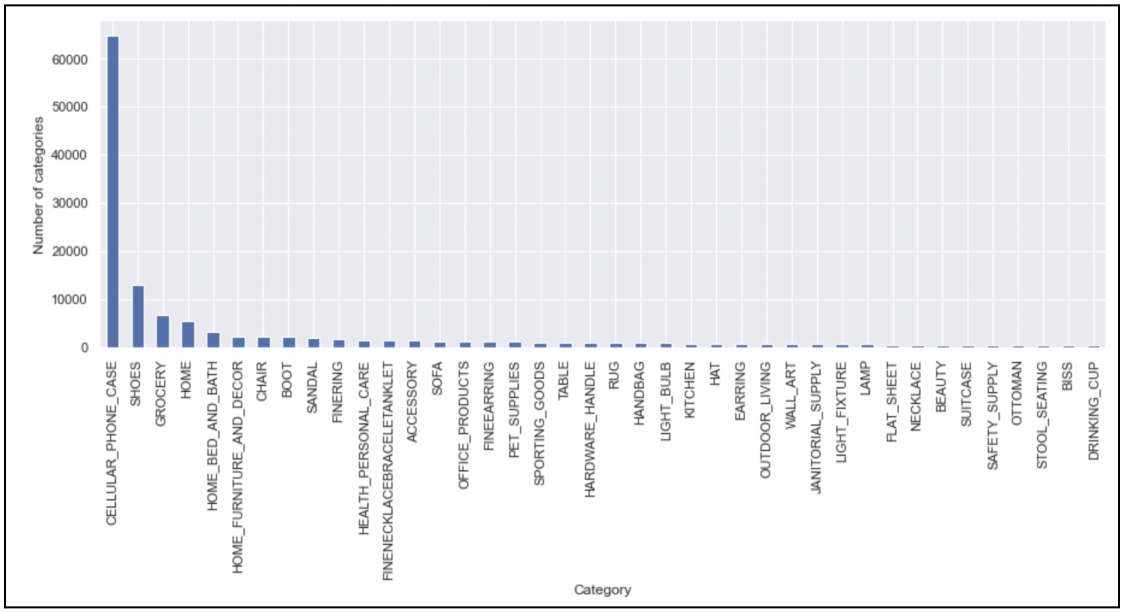
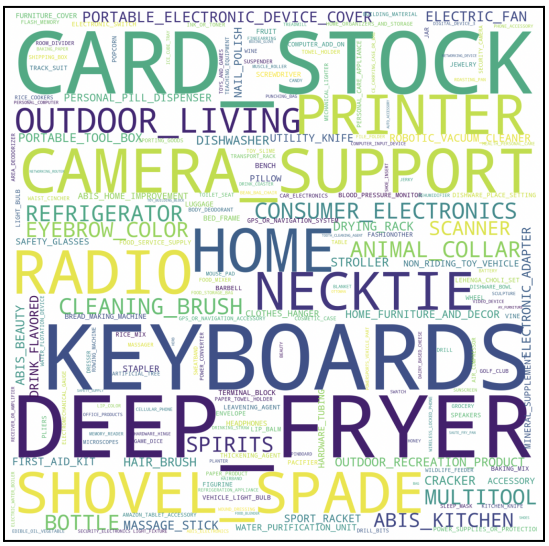


Figure 2: Category Entry Count

3.2.6 Word Cloud

Word Clouds were utilized in this project because they are a simple visual to show how important a word or feature is in a dataset. The larger the word in the Word Cloud, the more used it is in the set of data. These are shown below in *Figure 3* in terms of the Categories and Keywords that were used in the dataset.

Category



Keywords

Figure 3: Category and Keyword Word Clouds

4 Solution Approach

4.1 Metadata Model

The goal of the project is to build the most robust recommendation system for Amazon that the team could possibly make. The team was given a very large dataset that contained a variety of products available on Amazon, and they were categorized by what each product was, which would make it easy to group them.

The first part of the model, which is the function of ‘subset’, is a function that splits the data into product type and domain name in order to produce a more specific dataset for creating the model. Product type allows for products that are more similar, and domain name ensures that customers are only recommended products on the country’s website that they are buying from, for example if they are in Canada, they are only buying from the Canadian webstore. The subset function also divides the target and the rest of the dataframe. It returns the target and the rest of the dataframe in a list. This allows the target to not be compared to itself later in the model. The next function of the model is to encode the target. This function encodes the entire target object,

which is the product that we are finding other similar products for. It loops through the columns in target and encodes each column. This is stored as a list of columns, and each of these columns are encoded as a numeric tensor object that is later used for cosine similarity. This is essential in finding the similar products among the dataset.

The next function of the model is to encode the dataframe which encodes the rest of the dataframe, and at this point, the dataframe is the dataframe with the target removed. It encodes all the columns, similar to previously, and then the columns are put into a consolidated list. An example of this is pictures in *Appendix 3*. The model continues with the function of cosine. This function compares the target variable to each row in the dataframe, which is all the other products. This then computes cosine scores for these rows and creates a list of each of the row's cosine scores. The model then converts the values into numpy variables, as the previous numpy values could not be put in a dataframe, so the values have to be converted in order to become numpy arrays instead of tensorflow objects.

The model then has to build a new dataframe. This is done with all of the cosine scores in the rows instead of the original text data. This process also calculates the means across each row as the final scoring measure for the row. Essentially, this has to be sorted into a readable format, so the next part of the model sorts the previously created dataframe with all of the cosine scores by their mean values. It sorts the means from highest score to lowest in order to find the highest correlated products, and place those items at the top of the list. This also removes all of the scores from before and only adds means to the dataframe with all of the text data. Finally, the model recommends a product. It combines all of the previous functions together. This eventually returned the sorted dataframe. Ultimately, using this dataframe can find the closest products to the original product that was input and can recommend the most similar products to the customer.

4.1.1 Algorithms Used and Not Used

Initially, the team tried using STSB Roberta Larget, but this method was very slow and did not produce an accurate output. There was an original attempt to also build a model by encoding manually, but this proved to be a tedious task. The team determined that it would be more efficient to use Sentence Transformer, which was easier to run and also ran faster than previous efforts had proved to run. The team then tried to encode a model using just Hugging Face, but there was limited knowledge and time constraints when this was attempted. The team decided that using Sentence Transformer was the best solution because it had a Hugging Face package already integrated into it and it proved to be much easier and accurate when in use.

4.1.2 Inputs and Outputs

The inputs of this model were the target variable, which in this case was whatever the product code was, and it utilized the original dataframe of the model. The outputs of the model were a

few different features. First there was the target that was chosen, which is what the target looks like as a dataframe. The second is the ‘sorted’ feature which is that final dataframe that is organized with the mean cosine scores. This feature takes the top 10 or 50 products from the dataframe and displays the top 10 or 50 most similar products to the initial input product. Finally, there is a ‘score_df’ column that is displayed that shows the scores of each of the products to facilitate in showing which products are the most similar to the input.

4.1.3 Issues

There were a couple of main issues that the team ran into when creating the model for the proposed recommendation system. The first was figuring out how Tensorflow and Pytorch work. These were new tools for the team so there was a lot of research conducted on them to determine how exactly they work, and the best way to produce the most accurate results. Originally, these were used by trying to implement them without a package that does it automatically. This was overcome by finding a package that did the calculation automatically to reduce the manual coding done, ultimately reducing human error in the process. There was another issue that was run into throughout the coding process. Tensorflow gave errors that were not easy to understand when we were using that method, and resulted in very broad errors that did not have straightforward solutions. The team was unsure how to go about these issues, as there was nothing specific listed in the error. Even with the outreach and help of those who were knowledgeable in this domain served the team no luck so we had to figure out a way around this. Our solution was to not use Tensorflow.

4.1.4 Results

The team was successful in building a model that found similar items to the product that was input in the model. The model created a list of values in order from most similar to least similar based on the features in the dataset. Unfortunately, due to time constraints and the COVID-19 pandemic that the team fell victim to, a proper analysis to determine the accuracy of the model was unable to be executed.

4.2 Image Data Model

In this project, Amazon provided abundant image data in this project including 2D original images, its small vision, and 3D models. To make it easier and faster to study the image data, the team chose a low-pixel version of the image. The only difference between it and the original image is that their pixels are no bigger than 256 pixels. In the small pixeles vision, there are 256 folders and 398,468 items. Comparing the number of products, we found that some products have more than one image describing the details so making a model to differentiate between multiple images and a single image for each product was a difficult task. However, the team only had the main image ID in the product listings. So it was decided to use the product cover illustration as the object of the image data.

4.2.1 Algorithms Used and Not Used

The team tried neural network models for image classifications. Most projects and related work used neural networks as the tool to classify images. The general steps of neural network models are using images as an input. The second step is using a neural network to classify the images in the different categories, of which there were 76 categories in this project. Then another neural network is used to model the similarity score between pair images. Jaccard Similarity is a common way to measure this, and then choosing pair images which are greater than 0.5. The last step is selecting the most similar products.

In the field of image recommendation, other papers tend to recommend images using Tuned Perceptual Retrieval (PR), Complementary Nearest Neighbor Consensus (CNNC), Gaussian Mixture Models (GMM), Markov Chain (MCL), and Texture Agnostic Retrieval (TAR). CNNC, GMM, TAR, and PR are easy to train, but CNNC and GMM are hard to test [11]. On the other hand, PR, GMM, and TAR are hard to generalize. Since data consists of images, the neural network should be a method that is worth it to try after finding research related to the success of it.

4.2.2 Issues

The main issue with building an image model was the time constraint. The team had little experience with handling image data, which took some time to settle on how to best build an image model. The team started off by considering adding the image information to the metadata model as an attribute. This idea was not realized due to technical constraints. Therefore, it was recognized that a separate image model was necessary to build. Unfortunately, due to time constraints and limited capabilities, there was no longer enough time or resources to build a new image model and combine it with the metadata model. Therefore, it was decided to move away from building an image model, and put all focus into the metadata model.

5 Conclusions and Takeaways

5.1 Conclusions

Overall, the model ended up working okay, but it would have worked better if some columns were weighted properly. Unfortunately, this was not possible without an accuracy metric of how well the model worked because an accuracy score could have helped the team gauge how each column was relevant to the importance of the final model. For example, the column ‘brand name’ was weighing too heavily, but without the accuracy metric the team was unable to test how weighing it would affect the model because there was no way to test if more relevance would make the model better or not. In this case, if there was more time, the team could have

weighed the columns and eventually tested to see how accurate each attempt at different column weights would change the accuracy of the model as a whole.

5.2 Future Improvements

In the future, more experimentation could have also improved the model outputs. If there was more time, more testing could have been conducted to improve the accuracy of the model and determine what specific features improve the accuracy. Learning more about this would help weigh the features to figure out which features were more important to the overall model, and which were less important. This could have helped to make the model the most robust it could be. Another aspect that could be continued and improved to assist in resulting in a better overall model would be creating an image data model that would eventually be connected with the metadata model that was created. This could have improved the accuracy of the model and the outcome of how the model worked in general. In terms of finding an accuracy score of the model, the team attempted to use a variety of functions that were part of sentence transformers, but nothing ended up working to the standards of the team. It could be useful to look at F-scores in the future because they are ways to show the model's accuracy, but this was an idea that was not tried yet. It could be what the model uses to calculate accuracy, if implemented correctly and with enough research conducted though. Ultimately, in the future, more time and resources would have been beneficial, but there was only a semester to work on this project.

5.3 Takeaways

As a team, we learned a lot in terms of what methods to use for NLP and when to use them. The team learned about how text similarity metrics are computed, in regards to cosine similarity. This was useful in the success of our project. We also learned how data is not necessarily clean in real life examples and how much extra work is done to rebuild the data in a form that is easy to work with. A good portion of the time this semester was spent cleaning the dataset to make our model run smoothly. Another aspect that was learned throughout this project was how to handle the speed of a model. The best solution might not always work because it could take way too long to run, for the amount of computing power each of us had on our computers.

6 Appendix

Appendix 1: Missing Value Count Chart

Below is a chart of the count of null values in the dataset.

```
# Finding the null values.
print(df.isnull().sum())
```

```
brand                59
bullet_point      16135
item_id               0
item_name             0
product_type          0
item_keywords     21014
domain_name           0
model_name        66126
dtype: int64
```

Appendix 2: Value Count in Various Categories

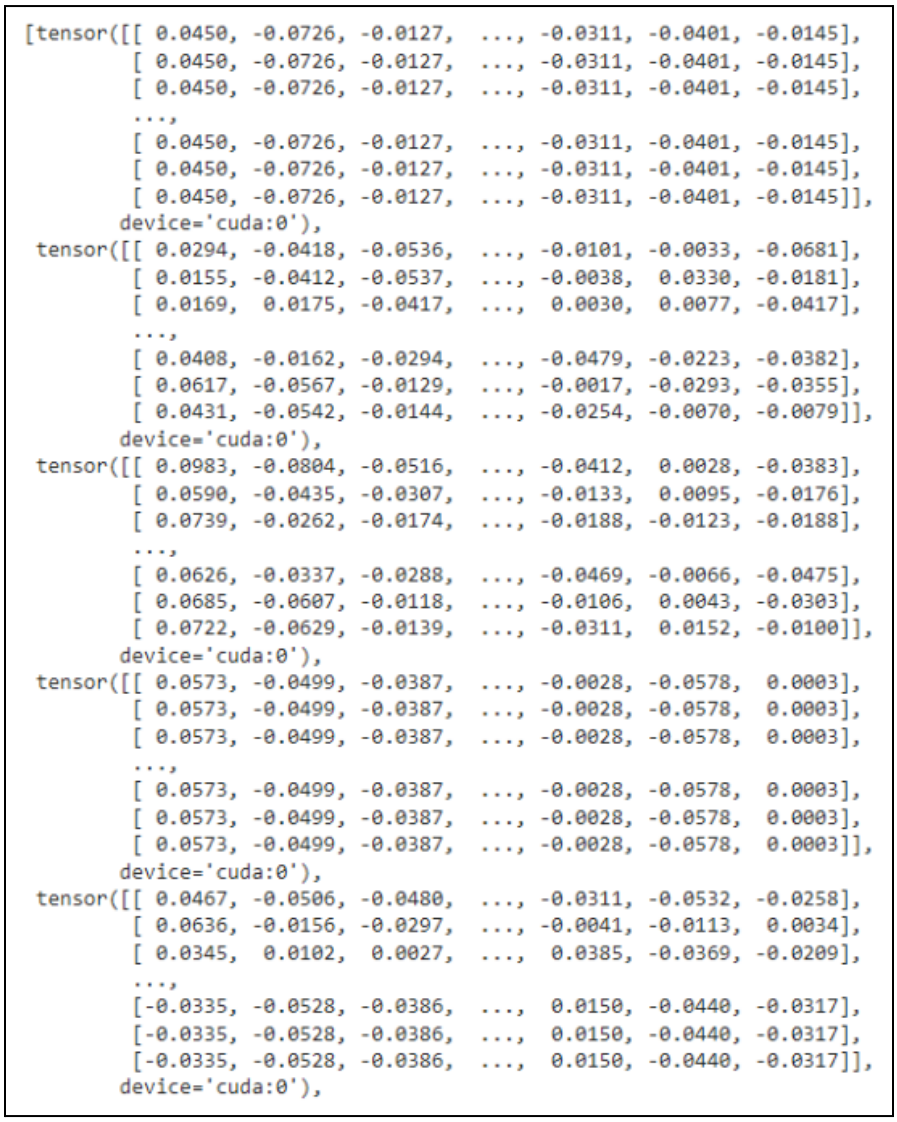
Below is the value count of different product types within the dataset.

```
top_df['product_type'].value_counts()
```

```
CELLULAR_PHONE_CASE    64853
SHOES                  12965
GROCERY                 6546
HOME                    5264
HOME_BED_AND_BATH       3082
                       ...
BED                      221
CLEANING_AGENT           214
BATTERY                  207
COMPUTER_ADD_ON          204
PANTRY                   201
Name: product_type, Length: 67, dtype: int64
```

Appendix 3: Encode Dataframe, Tensor Example

Below is an example of a list of a column encoded. A single column begins with the word ‘tensor’ on the left of the image.



```
[tensor([[ 0.0450, -0.0726, -0.0127,  ..., -0.0311, -0.0401, -0.0145],
         [ 0.0450, -0.0726, -0.0127,  ..., -0.0311, -0.0401, -0.0145],
         [ 0.0450, -0.0726, -0.0127,  ..., -0.0311, -0.0401, -0.0145],
         ...,
         [ 0.0450, -0.0726, -0.0127,  ..., -0.0311, -0.0401, -0.0145],
         [ 0.0450, -0.0726, -0.0127,  ..., -0.0311, -0.0401, -0.0145],
         [ 0.0450, -0.0726, -0.0127,  ..., -0.0311, -0.0401, -0.0145]],
        device='cuda:0'),
 tensor([[ 0.0294, -0.0418, -0.0536,  ..., -0.0101, -0.0033, -0.0681],
         [ 0.0155, -0.0412, -0.0537,  ..., -0.0038,  0.0330, -0.0181],
         [ 0.0169,  0.0175, -0.0417,  ...,  0.0030,  0.0077, -0.0417],
         ...,
         [ 0.0408, -0.0162, -0.0294,  ..., -0.0479, -0.0223, -0.0382],
         [ 0.0617, -0.0567, -0.0129,  ..., -0.0017, -0.0293, -0.0355],
         [ 0.0431, -0.0542, -0.0144,  ..., -0.0254, -0.0070, -0.0079]],
        device='cuda:0'),
 tensor([[ 0.0983, -0.0804, -0.0516,  ..., -0.0412,  0.0028, -0.0383],
         [ 0.0590, -0.0435, -0.0307,  ..., -0.0133,  0.0095, -0.0176],
         [ 0.0739, -0.0262, -0.0174,  ..., -0.0188, -0.0123, -0.0188],
         ...,
         [ 0.0626, -0.0337, -0.0288,  ..., -0.0469, -0.0066, -0.0475],
         [ 0.0685, -0.0607, -0.0118,  ..., -0.0106,  0.0043, -0.0303],
         [ 0.0722, -0.0629, -0.0139,  ..., -0.0311,  0.0152, -0.0100]],
        device='cuda:0'),
 tensor([[ 0.0573, -0.0499, -0.0387,  ..., -0.0028, -0.0578,  0.0003],
         [ 0.0573, -0.0499, -0.0387,  ..., -0.0028, -0.0578,  0.0003],
         [ 0.0573, -0.0499, -0.0387,  ..., -0.0028, -0.0578,  0.0003],
         ...,
         [ 0.0573, -0.0499, -0.0387,  ..., -0.0028, -0.0578,  0.0003],
         [ 0.0573, -0.0499, -0.0387,  ..., -0.0028, -0.0578,  0.0003],
         [ 0.0573, -0.0499, -0.0387,  ..., -0.0028, -0.0578,  0.0003]],
        device='cuda:0'),
 tensor([[ 0.0467, -0.0506, -0.0480,  ..., -0.0311, -0.0532, -0.0258],
         [ 0.0636, -0.0156, -0.0297,  ..., -0.0041, -0.0113,  0.0034],
         [ 0.0345,  0.0102,  0.0027,  ...,  0.0385, -0.0369, -0.0209],
         ...,
         [-0.0335, -0.0528, -0.0386,  ...,  0.0150, -0.0440, -0.0317],
         [-0.0335, -0.0528, -0.0386,  ...,  0.0150, -0.0440, -0.0317],
         [-0.0335, -0.0528, -0.0386,  ...,  0.0150, -0.0440, -0.0317]],
        device='cuda:0'),
```

Appendix 4: References

[1] Hardesty, L. (2022, January 26). *The history of Amazon's recommendation algorithm*. Amazon Science. Retrieved May 3, 2022, from https://www.amazon.science/the-history-of-amazons-recommendation-algorithm.

[2] *Sentence-transformers/all-mpnet-base-V2 · hugging face*. Hugging Face. (n.d.). Retrieved May 4, 2022, from https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

[3] TutorialsPoint. (n.d.). *What is Stemming?* Stemming & lemmatization. Retrieved May 3, 2022, from https://www.tutorialspoint.com/natural_language_toolkit/natural_language_toolkit_stemming_lemmatization.htm.

[4] What is semantic reasoning?: Fundamentals with Oxford Semantic Technologies. Oxford Semantic Technologies. (n.d.). Retrieved May 3, 2022, from https://www.oxfordsemantic.tech/fundamentals/what-is-semantic-reasoning#:~:text=Semantic%20reasoning%20is%20the%20ability,a%20form%20of%20Semantic%20AI.

[5] Johnson, D. (2022, March 8). *POS tagging with NLTK and chunking in NLP [examples]*. POS Tagging. Retrieved May 3, 2022, from https://www.guru99.com/pos-tagging-chunking-nltk.html.

[6] Verma, Y. (2021, November 11). *A complete guide to using wordnet in NLP applications*. Analytics India Magazine. Retrieved May 3, 2022, from https://analyticsindiamag.com/a-complete-guide-to-using-wordnet-in-nlp-applications/.

[7] Han, J., & Pei, J. (n.d.). *Cosine similarity*. ScienceDirect. Retrieved May 3, 2022, from https://www.sciencedirect.com/topics/computer-science/cosine-similarity.

[8] Jwalapuram, N. (2021, April 12). *Pytorch Library: What is Pytorch Library for deep learning* |. Analytics Vidhya. Retrieved May 3, 2022, from https://www.analyticsvidhya.com/blog/2021/04/a-gentle-introduction-to-pytorch-library/.

[9] Kamath, U., Graham, K. L., & Emara, W. (2022). Bidirectional encoder representations from Transformers (Bert). *Transformers for Machine Learning*, 43–70. https://doi.org/10.1201/9781003170082-3.

[10] *Data flattening and data unflattening*. Firebolt Glossary. (n.d.). Retrieved May 4, 2022, from https://www.firebolt.io/glossary-items/data-flattening-and-data-unflattening#:~:text=Data%20flattening%20usually%20refers%20to,nested%20structure%20to%20relational%20data.

[11] Chen, L., Yang, F., & Yang, H. (n.d.). Image-based Product Recommendation System with Convolutional Neural Networks. https://doi.org/http://cs231n.stanford.edu/reports/2017/pdfs/105.pdf.