Zevi Project Headstart

-Manish Borthakur

Data Analysis

After analysing the dataset, I got to know that it is of various products with the following data:

(1) Name
(2) Description
(3) Brand
(4) Categories
(5) hierarchicalCategories
(6) Type
(7) Price
(8) Image
(9) Url
(10) Free_shipping
(11) Rating
(12) Popularity
(13) objectID

There are a total of 21469 data points.

The Model

For an initial model, I used just the name and description. In this model, I implemented Latent Semantic Index.
[https://medium.com/analytics-vidhya/semantic-search-engine-using-nlp-cec19e8cfa7e]

First, I cleaned the description texts so that they become lists of simple lower-cased words without punctuations, digits or special characters. Then, I made a dictionary out of this data. I used the Dictionary class of the Gensim library for this. Now, a lot of the words in the dictionary are useless for training the models, as they are present in almost every document. Thus, I removed those words from the descriptions which occur in 50% of the descriptions.

Next, I used the Bag of Words method to build a corpus using the dictionary. In corpus, each data point represents the words present in a description along with the frequency of that word in that description. Doc2bow function of Dictionary can be used to easily perform this. This corpus is used to make the LSI model.

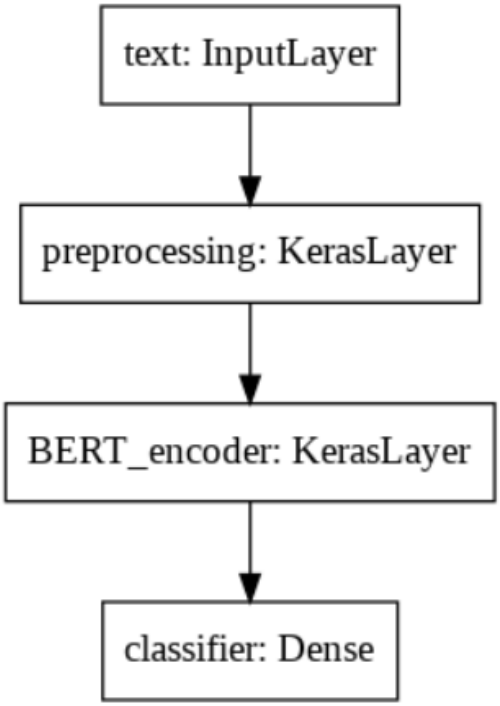
To build the LSI model, I first made a Term frequency-Inverse Document Frequency(Tf-Idf) model using corpus, which gives the most important words in each description. Then, I passed this to an LSI model with a default of 200 features. I used the MatrixSimilarity class of Gensim

so that I can find similarities between my query and each of the datapoints later(using cosine similarity). Now, I was ready to make suggestion lists from queries.

For making the predictions, I first made a function which took in a query and gave out 5 suggestions using this. In this function, first the sentence is cleaned using the previous function used, then fed into the dictionary to make a Bag of Words datapoint(like in corpus). This was passed to the Tf-Idf and LSI models. Then, MatrixSimilarity was used to find similarities with the datapoints and the top 5 datapoints were taken(sorting the list and taking the first 5 elements). The list was given in a pandas dataframe. This function can be used for searching any query. This model works fine for specific queries, where a user knows what they want and type in words close to an actual product. The words are compared to descriptions of items, which assigns relevance to them based on how many similar words there are. But this model fails to find meaning in the queries themselves. So, users who do not know what they want cannot find a satisfactory item.

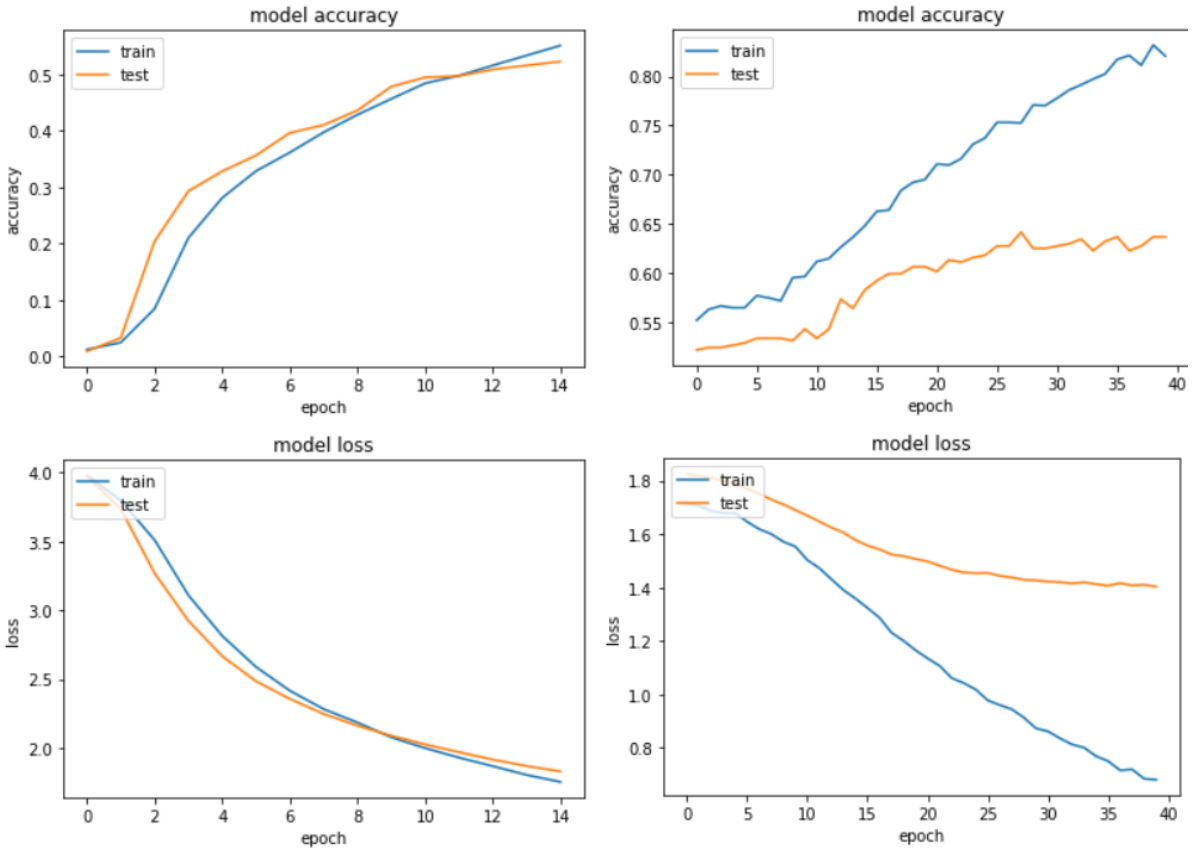
To improve the model, I tried to integrate the category of products. I considered the bottom-most level of category (39 distinct categories). I gave an index to these categories and took the first 4 words of every description from the cleaned data I had for the training set. Due to memory constraints of Google Colab, I could not use a lot of data. So I jumbled the data and took 10% of it. Otherwise, every description can be used and also, we can take a lot of slices of words from the description too.

I used the smallest available BERT model (L-2 H-128 A-2). I made the following model:-



The preprocessing layer uses Tensorflow Hub to process the input string for BERT. The Dense layer has 39 nodes and I used Categorical Crossentropy as the loss function and Categorical Accuracy as the metric. I used AdamW as the optimizer.

The model trained well initially but started to overfit as I trained the model more. I reached a training accuracy of 82% and validation accuracy of 64% before I stopped. Looking at the graphs, I think training with a larger training data can fix this. Also, the 1st 4 words may not contain much information about a product, so more samples from the same description has to be taken. Still, trying for some sample inputs, the model is giving some good results.



(Epochs 1-15) & (Epochs 15-55)

A few examples:-

1) (Input: Macbook, Output: Computers & Tablets)
2) (Input: Guitar, Output: Musical Instruments)
3) (Input: Nokia, Output: Cell Phones)

After this, I made the final model which uses the saved models to give a ranked list. First, the query is inserted to the LSI model and matches are found using MatrixSimilarity. After that, the BERT model finds out the category the query belongs to. Then among the matches earlier, I increased the relevance of those products who had the same category as the prediction. I increased them by the probability found by the BERT model.

A few examples:-

```
search_similar_products('mouse')
```

	Relevance	Name
0	106.44	Logitech - Anywhere Mouse MX Wireless Laser Mo...
1	106.13	Logitech - M310 Wireless Optical Mouse - Peaco...
2	105.81	ROCCAT - Taito Control Mouse Pad - Black/Blue
3	105.71	Razer - Goliathus Speed Mouse Pad - Black/green
4	105.10	Logitech - Marathon Mouse M705 Wireless Laser ...

```
search_similar_products('dell')
```

	Relevance	Name
0	107.04	Dell - Inspiron 15.6" Laptop - Intel Core i3 -...
1	104.34	Dell - Inspiron 3459 23.8" Touch-Screen All-In...
2	101.57	Dell - Inspiron Desktop - Intel Pentium - 4GB ...
3	101.25	Dell - Inspiron 15.6" Laptop - Intel Core i7 -...
4	100.85	Dell - Inspiron Desktop - Intel Core i7 - 16GB...

```
search_similar_products('table')
```

	Relevance	Name
0	82.70	Insignia™ - 9" Table Fan - Blue
1	82.70	Insignia™ - 9" Table Fan - Mint
2	82.70	Insignia™ - 9" Table Fan - Gray
3	75.62	Insignia™ - 10" Table Fan - Black
4	75.62	Insignia™ - 10" Table Fan - Pink

```
search_similar_products('mobile')
```

	Relevance	Name
0	96.55	Boost Mobile - $45 Re-Boost Prepaid Phone Card
1	96.55	Boost Mobile - $35 Re-Boost Prepaid Phone Card
2	85.85	Virgin Mobile - $25 Top-Up Card
3	79.69	SIMPLE Mobile - $60 ReUp Prepaid Card
4	64.72	Socket mobile® - SocketScan™ S800 (formerly 8C...

Future Works

There are still a lot of problems in this model. If we do not enter a query which has a word in any product description, no object will turn up in MatrixSimilarity. This problem has to be solved by suggesting products of the predicted category when no other prediction is there. Other methods of finding similar products also have to be found.

```
search_similar_products('nokia')
```

Relevance	Name

```
search_similar_products('tv')
```

Relevance	Name

Right now, only the lowermost categories were used. Using the topmost categories can give much more specific results as there are around 1100 categories. There is a lot of data to train this easily as well(as suggested before for the previous model).

The same model can be used for “type” too.

Another model that can be integrated is a classifier which predicts what price range the query seeks. The training data for this can be created by forming a corpus of words similar to “cheap” and mixing them with other words from descriptions.

Popularity and ratings are also important parameters which can be used to increase the relevance of some products over others. This can be included in the same way I included the relevance of “category”.