Customer Review Analysis and Response Generation

Samridhi Agrawal
University of Washington Bothell
Bothell, WA United States
sagra15@uw.edu

ABSTRACT

Today consumers expect brands to engage them in a personalized way. To deliver superior customer experience, brands need to capture feedback through all channel of interactions like email, voice, chat and reviews and contextually respond to it. In order to create a personalized response to a customer review, we need customer intent. In my project, I use natural language understanding (Topic modelling and Sentiment Analysis) to generate customer intent. Working on OPOSUM dataset obtained by down sampling the Amazon product dataset done by [1]. The dataset contains Amazon reviews from six product categories: Laptop Bags, Bluetooth Headsets, Boots, Keyboards, Televisions, and Vacuums. Methodologies used in the project will be NLTK library and BERT for sentiment analysis, LDA , BERT for topic modelling. This response system will increase customer satisfaction (CSAT Score) for brands to build customer trust and loyalty.

KEYWORDS

Topic modelling, Sentiment Analysis, TF-IDF Vectorizer, LDA, BERT, Text Generation

1 INTRODUCTION

In [1], introduces a neural framework for summarization of online product reviews using aspect extractor, sentiment predictor and summary generation. Aspect extraction which aims to find specific features like battery life, sound quality, ease of use and identify expressions that discuss them (2) sentiment prediction which determines the sentiment orientation (positive or negative) on the aspects found in the first step, and (3) summary generation which presents the identified opinions to the user. To understand the extraction of important topics, [2] proposes tagging Latent Dirichlet Allocation (LDA) which defines the latent topics from online user experience and automatically tag the resource based on the most likely tags derived from the latent topics identified. LDA is most popular probabilistic topic model for diverse class of applications. This [3] provides two types of LDA – scalable disk based LDA and memory based LDA. In [4], compares the performance of lexicon-based sentiment analysis and polarity score based with LDA as topic modelling method. Related to [5] , in which sentiment analysis has been done on document as well as word level, we will also perform sentiment analysis and topic modelling together.

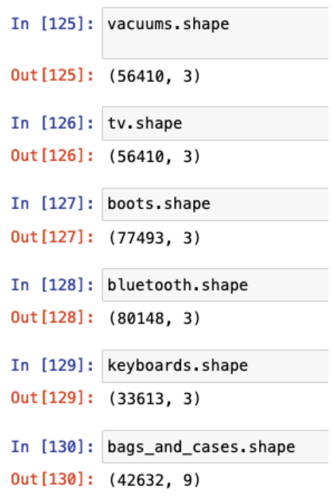
2 METHODS

2.1 ARCHITECTURE

In Figure 2, shows the flow of the project from processing the raw data to provide a response to customer reviews. The project is being done in Python 3.6 using Jupyter notebooks. In the following sections, we will be discussing the individual steps.

2.2 DATA PREPROCESSING

The data is taken by down sampling the Amazon Product reviews. Processing the text files into data frames and creating three features' named product ID, rating and reviews. The dimensions of data frames are below



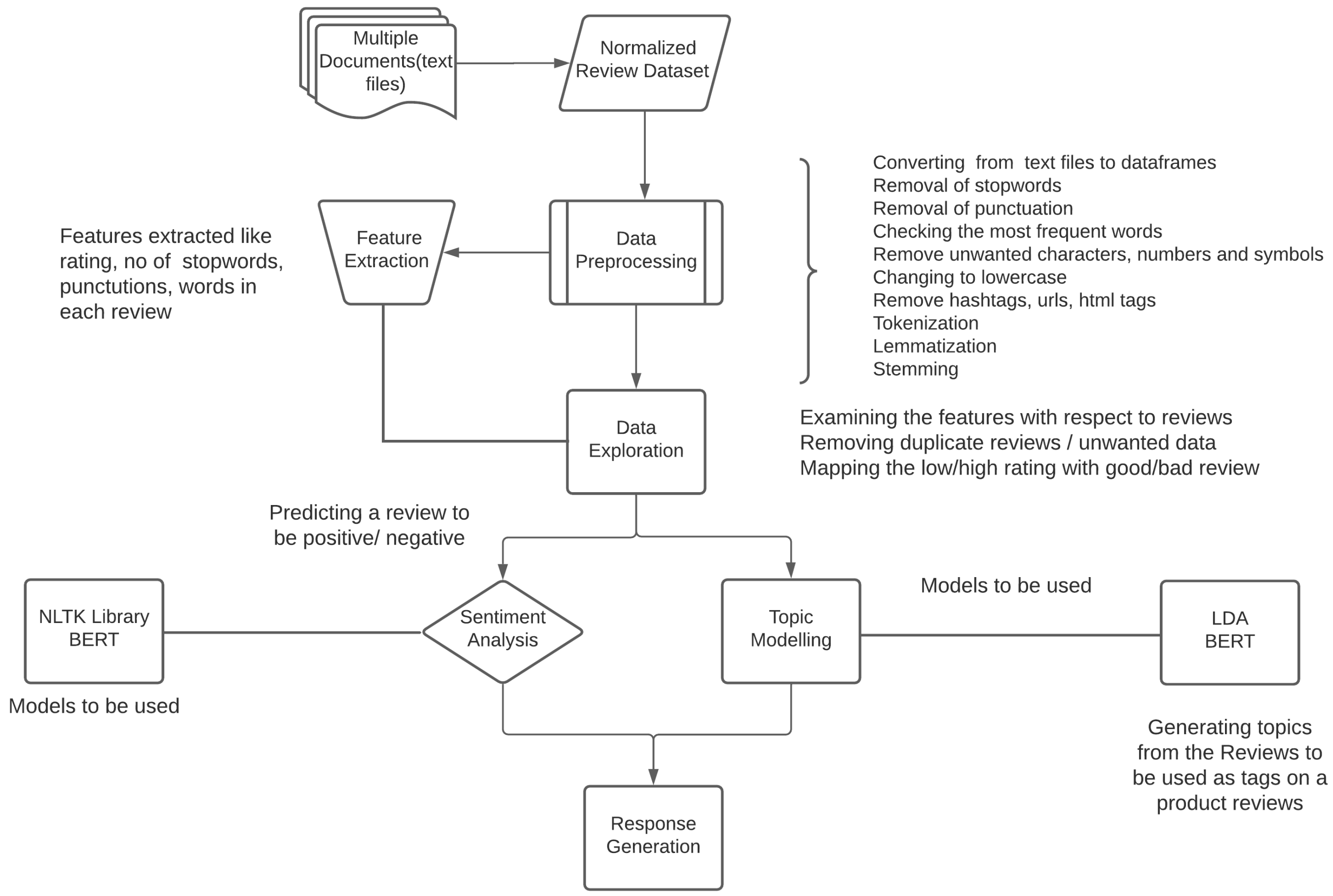
```
In [125]: vacuums.shape
Out[125]: (56410, 3)

In [126]: tv.shape
Out[126]: (56410, 3)

In [127]: boots.shape
Out[127]: (77493, 3)

In [128]: bluetooth.shape
Out[128]: (80148, 3)

In [129]: keyboards.shape
Out[129]: (33613, 3)

In [130]: bags_and_cases.shape
Out[130]: (42632, 9)
```

Fig. 1. Customers review dataset for vacuums, tv, boots, Bluetooth, keyboards, bags and cases

At every step of preprocessing, evaluating the words frequency regularly is beneficial as it provides an idea of important words and next step to remove the unwanted data. Following are the steps for preprocessing text data:

A. Removal of stop words – Using the nltk library, removing the stop words in English like the, not, have, also etc. from the review text and counting the no of stop words as a feature extraction for each text.

B. Removal of punctuation – Using the nltk library 'punkt' consist of the also punctuations. Removing the punctuation from the


Multiple Documents(text files)
Normalized Review Dataset
Data Preprocessing
Converting from text files to dataframes
Removal of stopwords
Removal of punctuation
Checking the most frequent words
Remove unwanted characters, numbers and symbols
Changing to lowercase
Remove hashtags, urls, html tags
Tokenization
Lemmatization
Stemming
Feature Extraction
Features extracted like rating, no of stopwords, punctutions, words in each review
Data Exploration
Examining the features with respect to reviews
Removing duplicate reviews / unwanted data
Mapping the low/high rating with good/bad review
Predicting a review to be positive/ negative
Sentiment Analysis
NLTK Library BERT
Models to be used
Topic Modelling
Models to be used
LDA BERT
Generating topics from the Reviews to be used as tags on a product reviews
Response Generation

review text and counting the no of punctuation as a feature extraction for each text.

Fig. 2. Architecture Diagram

C. Remove unwanted characters, numbers and symbols – using regular expressions to remove any unwanted characters, numbers and symbols from the review text.

D. Changing to lowercase – making the entire review text into lower case.

E. Remove hashtags, URLs, html tags

F. Tokenization – Splitting the sentences of words into individual tokens.

G. Lemmatization and Stemming - Lemmatization reduces any given word to its base-form which means reducing multiple forms of a word to a single word. Stemming removes the last few characters of the word. Lemmatization is more effective than stemming but stemming the word is faster. Using the 'spacy' library for lemmatization and 'nltk' snowball stemmer for stemming the data. Counting the number of stop words, punctuation, numeric data, wordcount, hashtags as a part of feature extraction.

2.3 DATA EXPLORATION

In this section, we will explore the data using various libraries matplolib, seaborn, word cloud. Word cloud library visualizes the important words evaluating the frequency of words showing it in multiple fonts. Visualizing the bags_and_cases reviews after preprocessing

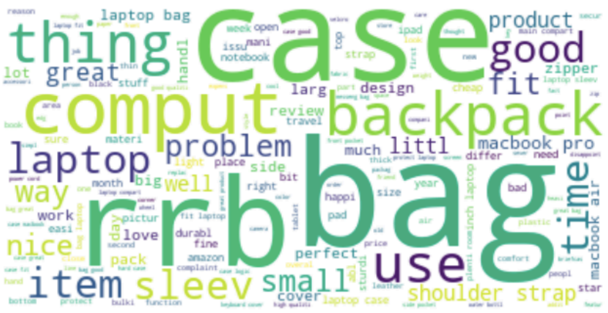


Fig. 3. bags_and_cases reviews visualization using wordcloud

Analyzing the data with the features extracted with the matplotlib:

A. Remove the review which has high number of stopwords (an outlier)
B. Dropping the duplicate reviews
C. Checking the hashtags, numerical data
D. Checking the ratings with number of reviews
E. Based on rating divide the reviews into two categories as 1-3 as bad review and 4-5 as good review.

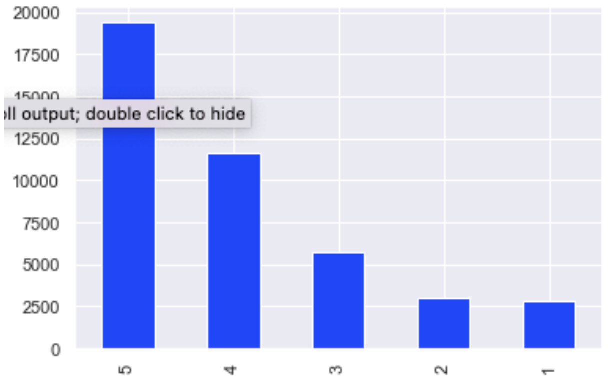


Fig. 4. Reviews vs Rating (1 - 5) graph

2.4 SENTIMENT ANALYSIS

Sentiment analysis is the process to extract emotions which can be positive, negative, or neutral. For example, a text such as "I found my new laptop bag very useful and trendy" shows a positive emotion. In the project, we have implemented sentiment analysis with two approaches, (a) using the NLTK library (Ref), and (b) using BERT pre-trained model. NLTK library is a natural language processing library which contains many APIs and labeled datasets to help you perform text analysis. We implemented "Polarity" and "Subjectivity" through a library "TextBlob" for our project. Polarity is defined in the range of [-1,1] which means (-1) refers to a negative emotion while (+1) refers to a positive emotion. Subjectivity is defined in the range of [0,1] , as measure of whether a text is a fact or an opinion in terms of a score value where (0) being closest to a fact while the higher the score(1) the more it becomes an opinion. As in fig 5, the polarity of reviews is increasing in ascending order from rating of 1 to 5 which is justifying the fact that low rating of the product is a negative sentiment with higher polarity score of high rating of 5. Similar to polarity, subjectivity score increases from rating 1 to 5 shows how subjectivity of text changes from only facts to more opinionized reviews.

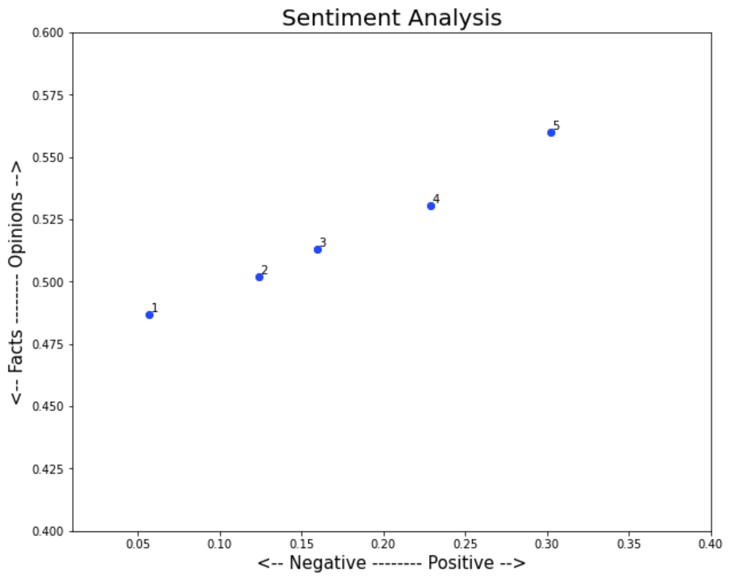


Fig. 5. Polarity vs subjectivity score w.r.t. Rating (1 - 5)

In fig 6, we can see those reviews with less than polarity score of 0 are low as compared to right side of the figure of polarity score between 0-1. This dataset is a bit imbalanced as there are more good reviews than bad reviews.

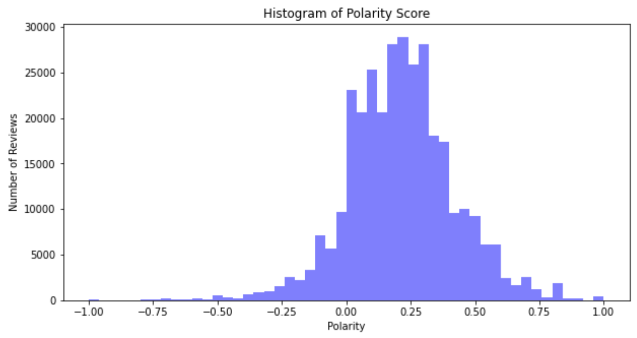


Fig. 6. Reviews vs Polarity

In fig 7, the subjectivity score is balanced across the dataset because the curve of the distribution of dataset is symmetric.

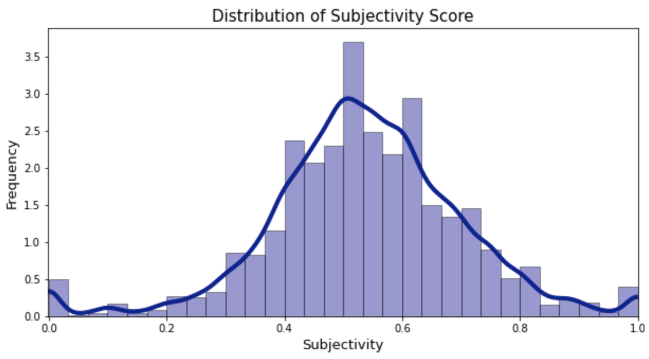


Fig. 7. Distribution of subjectivity score in range [0,1]

In fig 8, the scatter graph shows polarity vs subjectivity for good reviews and bad reviews. The blue scatters point are less as compared to orange dots.

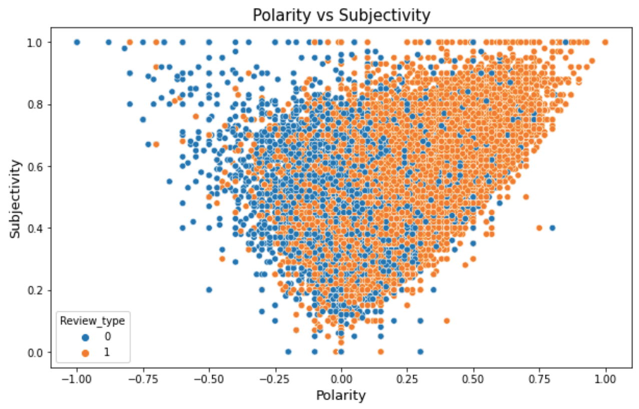
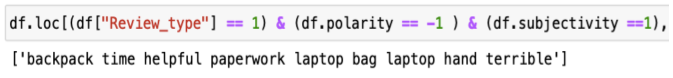


Fig. 8. Polarity vs Subjectivity

There are also datapoints in the dataset where the scores are not aligned with the ratings of the product. For example,

1. Rating = 0 , polarity = 1
2. Rating = 0 , polarity = 1, subjectivity = 1
3. Rating = 1 , polarity = -1, subjectivity = 1

In these cases, we are trusting the polarity score for finding the sentiment, because it can be human error does not provide the correct rating. These can also be the outliers which may need to removed.



```
df.loc[(df["Review_type"] == 1) & (df.polarity == -1 ) & (df.subjectivity ==1),
['backpack time helpful paperwork laptop bag laptop hand terrible']
```

Fig. 9. An example where review has high rating, but polarity score is -1. We can see that the review says "helpful" and "terrible" at the same time.

2.5 TOPIC MODELING

Topic modelling is the process of statistical modelling for finding abstract topics from a collection of documents. It is an unsupervised machine learning algorithms that can process a set of documents and extract similar words and phrases out of them. It is done by mapping the words with document and documents with the topics using a matrix. It is one of the best practices to analyze text data which contains huge number of documents to monitor. In our project, customer reviews are very important to analyze to know what is lacking, required, and enhanced to improve customer experience. I have performed topic modelling using (a) LDA with count vectorizer (b) LDA using genism library (c) Using BERT embeddings. To produce topics from the text data, we need to provide number of topics we are expecting from the data.

Topic modelling process converts all the text into a huge document called corpus. The text gets converted into vectors so that the model can be applied on the data.

2.6 RESPONSE GENERATION

This section uses sentiments and topics created for each customer review in the above sections to generate a response for the customer.

3 EXPERIMENT AND RESULTS

I have used the dataset for building the model to predict the sentiment of the texts. Before sending my reviews into the model, the text needs to be merged.

3.1 SENTIMENT ANALYSIS

I have used the dataset for building the model to predict the sentiment of the texts. Before sending my reviews into the model, the text needs to be converted into vectors using TF-IDF vectorizer. I have used four models i.e. Logistic regression, GaussianNB, MultinomialNB, BernoulliNB.

Table 1. *Model results with metrics*

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
LR	83%	0.85	0.91	0.83
GaussianNB	74%	0.86	0.73	0.74
MultinomialNB	78%	0.78	0.95	0.78
BernoulliNB	78%	0.85	0.83	0.78

Logistics Regression performs the best with accuracy of 83 percent as compared to others models.

In fig. 10. AUC-ROC curve plotting the true positive rate with the false positive rate. From the figure, we can see that the logistic regression has higher roc score than the other models and it can distinguish the positive class in the dataset correctly.

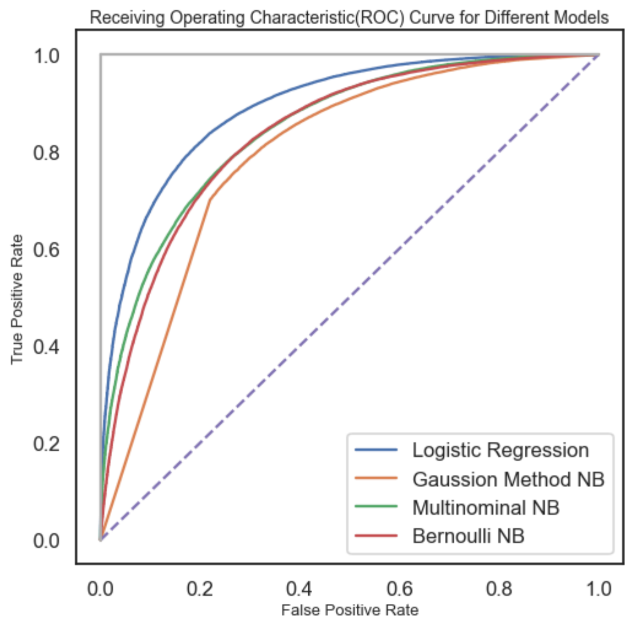


Fig. 10. AUC-ROC curve for all models

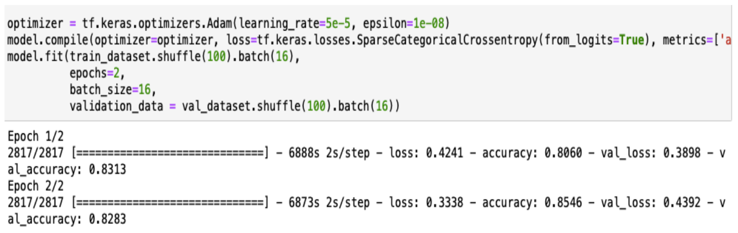
I have also experimented to build the model using ensemble methods such as bagging, boosting, and stocking. Ensemble methods are used to enhance the accuracy of the model. Ensemble methods includes RandomForestClassifier, BaggingClassifier with Logistic Regression, VotingClassifier which combines multiple classifiers, and XGBoostClassifier. In Table 2, the results for ensemble models show that there was no material increase in accuracy.

Table 2. *Ensemble methods results with metrics*

Model	Accuracy	Precision	Recall	F1-score
BaggingClassifier with LR	82%	0.82	0.95	0.82
VotingClassifier	82%	0.83	0.95	0.82
XGB	79%	0.79	0.97	0.79

The second approach to sentiment analysis is using pretrained models. A pretrained models are developed at google and open to use so that NLP researchers don't have to pretrain the model from scratch. It requires high computational resources to build these models. BERT[6] is a transformer-based model where it only acts as an encoder and transformer consist of both encoder and decoder. BERT is used as pre-trained language representations and it can solve multiple NLP problems like Question and Answering, classifications, sentence prediction given the first sentence. It is better than the existing methods on NLP tasks. It stands for *Bidirectional Encoder Representation from Transformers*. In[7] performs sentiment analysis using BERT for Indonesian mobile app reviews which gives the accuracy of 84% on pre-trained models.

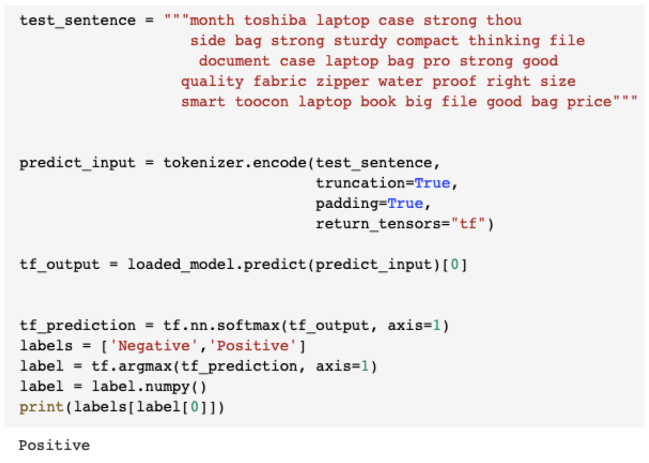
I have used Bert-base uncased model to classify the customer reviews into positive and negative sentences. The base model requires 12GB of RAM and 'uncased' is case insensitive. I have used Google Colab to run the model because it requires GPU, and my dataset contains 56K number of customer reviews which is huge. Before training of the model, we need to split the model into training and validation dataset. The text data needs to be encoded using Bert tokenizer and then to be converted into TensorFlow dataset. BERT model contains 12-layer, 768-hidden, 12-heads, 110M parameters. In fig. 11, we can see the model training gives an accuracy of 85 percent which is the better than previous model results.


```
optimizer = tf.keras.optimizers.Adam(learning_rate=5e-5, epsilon=1e-08)
model.compile(optimizer=optimizer, loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['a
model.fit(train_dataset.shuffle(100).batch(16),
          epochs=2,
          batch_size=16,
          validation_data = val_dataset.shuffle(100).batch(16))
```

```
Epoch 1/2
2817/2817 [==============================] - 6888s 2s/step - loss: 0.4241 - accuracy: 0.8060 - val_loss: 0.3898 - v
al_accuracy: 0.8313
Epoch 2/2
2817/2817 [==============================] - 6873s 2s/step - loss: 0.3338 - accuracy: 0.8546 - val_loss: 0.4392 - v
al_accuracy: 0.8283
```

Fig. 11. Pre-trained model results in 85% accuracy

In fig 12, model is tested against a customer review which outputs in 'negative' sentiment by using the pretrained Bert model.

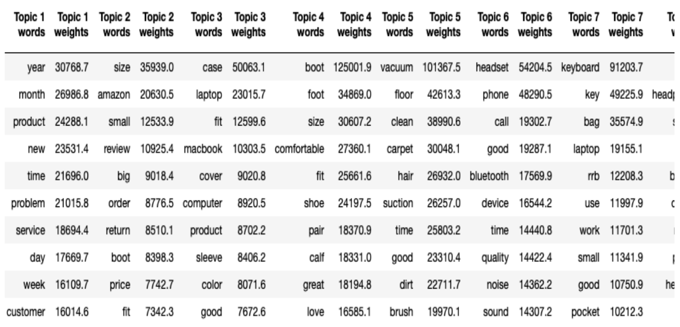

```
test_sentence = """month toshiba laptop case strong thou
                   side bag strong sturdy compact thinking file
                    document case laptop bag pro strong good
                  quality fabric zipper water proof right size
                  smart toocon laptop book big file good bag price"""


predict_input = tokenizer.encode(test_sentence,
                                 truncation=True,
                                 padding=True,
                                 return_tensors="tf")

tf_output = loaded_model.predict(predict_input)[0]


tf_prediction = tf.nn.softmax(tf_output, axis=1)
labels = ['Negative','Positive']
label = tf.argmax(tf_prediction, axis=1)
label = label.numpy()
print(labels[label[0]])
```

```
Positive
```

Fig. 12. Testing the Bert model with an example

3.2 TOPIC MODELING

I have experimented with multiple number of topics. In fig 13, method (a) generated 10 topics with the scores.



Topic 1 words	Topic 1 weights	Topic 2 words	Topic 2 weights	Topic 3 words	Topic 3 weights	Topic 4 words	Topic 4 weights	Topic 5 words	Topic 5 weights	Topic 6 words	Topic 6 weights	Topic 7 words	Topic 7 weights	To w
year	30768.7	size	35939.0	case	50063.1	boot	125001.9	vacuum	101367.5	headset	54204.5	keyboard	91203.7	
month	26986.8	amazon	20630.5	laptop	23015.7	foot	34869.0	floor	42613.3	phone	48290.5	key	49225.9	headp
product	24288.1	small	12533.9	fit	12599.6	size	30607.2	clean	38990.6	call	19302.7	bag	35574.9	s
new	23531.4	review	10925.4	macbook	10303.5	comfortable	27360.1	carpet	30048.1	good	19287.1	laptop	19155.1	
time	21696.0	big	9018.4	cover	9020.8	fit	25661.6	hair	26932.0	bluetooth	17569.9	rrb	12208.3	b
problem	21015.8	order	8776.5	computer	8920.5	shoe	24197.5	suction	26257.0	device	16544.2	use	11997.9	c
service	18694.4	return	8510.1	product	8702.2	pair	18370.9	time	25803.2	time	14440.8	work	11701.3	
day	17669.7	boot	8398.3	sleeve	8406.2	calf	18331.0	good	23310.4	quality	14422.4	small	11341.9	p
week	16109.7	price	7742.7	color	8071.6	great	18194.8	dirt	22711.7	noise	14362.2	good	10750.9	he
customer	16014.6	fit	7342.3	good	7672.6	love	16585.1	brush	19970.1	sound	14307.2	pocket	10212.3	

Fig. 13. LDA from sklearn with number of topics(k) as 10

Let's write our understanding of the topics created.

- Topic 1 says new year time product
- Topic 2 says boots to return because of small size
- Topic 3 says laptop fits in case and color god
- Topic 4 says boots comfortable and fit
- Topic 5 says vacuums clean carpet, hair, good suction
- Topic 6 says Bluetooth quality is good
- Topic 7 says bag is small for laptop
- Topic 8 says headphone battery life great

This method is similar to clustering, but it generates two mapping internally (1) document-word and (2) word-topics to create document-topics mapping. In fig 14, we can see that it is difficult to analyze some of the topics generated.

```
[(0,
  '0.026*"floor" + 0.021*"clean" + 0.020*"carpet" + 0.018*"hair" + '
  '0.014*"suction" + 0.014*"easy" + 0.013*"dirt" + 0.012*"dyson" + '
  '0.012*"brush" + 0.012*"small"'),
 (1,
  '0.034*"vacuum" + 0.013*"vac" + 0.012*"rrb" + 0.009*"screen" + '
  '0.009*"picture" + 0.009*"time" + 0.008*"problem" + 0.008*"good" + '
  '0.008*"clean" + 0.007*"hardwood"'),
 (2,
  '0.071*"boot" + 0.033*"size" + 0.027*"foot" + 0.017*"shoe" + 0.015*"fit" + '
  '0.014*"comfortable" + 0.013*"calf" + 0.011*"warm" + 0.011*"great" + '
  '0.011*"sock"'),
 (3,
  '0.023*"good" + 0.017*"sound" + 0.017*"great" + 0.014*"quality" + '
  '0.012*"cable" + 0.011*"remote" + 0.010*"rrb" + 0.009*"movie" + '
  '0.008*"speaker" + 0.008*"easy"'),
 (4,
  '0.061*"vacuum" + 0.020*"time" + 0.017*"product" + 0.015*"year" + '
  '0.015*"good" + 0.013*"month" + 0.012*"great" + 0.012*"new" + 0.011*"day" + '
  '0.011*"amazon"'),
 (5,
  '0.048*"picture" + 0.042*"great" + 0.034*"color" + 0.028*"price" + '
  '0.024*"love" + 0.021*"boot" + 0.018*"black" + 0.017*"good" + 0.015*"size" +
  '0.012*"robot"')]
```

Fig. 14. LDA from genism with number of topics(k) as 6

In fig 15, the topics generated through LDA is visualized by the library PyLDAvis. Intertopic distance for the topics is shown on the left side of the figure and most salient terms with their weights per topic in descending order is displayed on the right side. I have added the topics generated for each review in the final dataframe to be used in response generation.

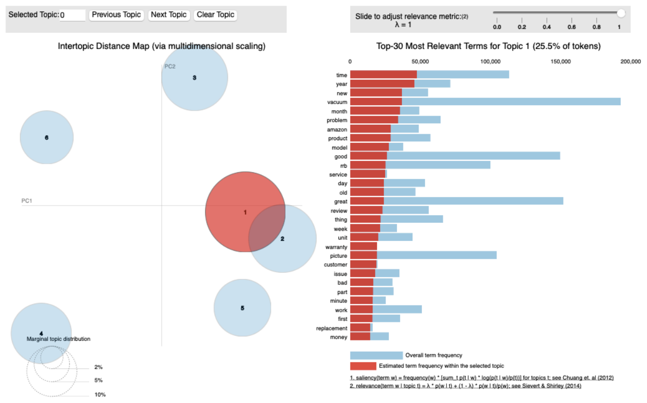
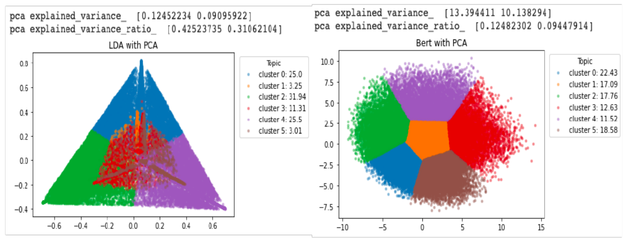


Fig. 14. PyLDAvis library to visualize the topics

Third approach is implemented to check the accuracy of LDA with BERT. I have used word embedding for both LDA and Bert. In LDA, vector form is the probability of a word occurring in several documents. For Bert, we used sentence transformers to create embeddings which uses pretrained models by encoding the input data. Our dataset contains 56k rows, so to apply word embedding, dimensionality reduction is required to provide better results. For dimensionality reduction, I have used PCA (principal component analysis) and T-Sne (t-distributed stochastic neighbor embedding). A silhouette score is calculated for LDA and BERT with their dimensionality reductions embedding. Silhouette score is similarity measure of points in a cluster compared to other clusters. Possible values for SC [-1,0,1]. Value 0 refers to overlapping clusters. If the value is negative, the datapoint is assigned to the wrong cluster.

In fig 15, visualization of Bert is better and clear than LDA using PCA dimensionality reduction for number of topics taken as 6.



```
pca explained_variance_  [0.12452234 0.09095922]
pca explained_variance_ratio_  [0.42523735 0.31062104]
```

```
pca explained_variance_  [13.394411 10.138294]
pca explained_variance_ratio_  [0.12482302 0.09447914]
```

Fig. 15. Visualization of LDA and BERT embedding after dimensionality reduction using PCA

In fig 16, visualization of Bert is better and clear than LDA using T-Sne dimensionality reduction.

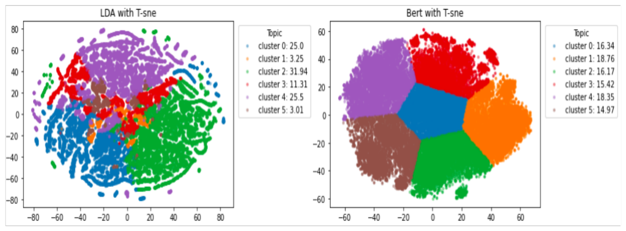


Fig. 16. Visualization of LDA and BERT embedding after dimensionality reduction using T-Sne

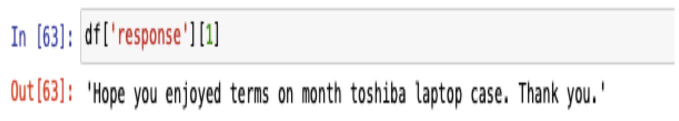
From table 2, we can see that Bert performance is better than LDA as silhouette score for PCA and TSNE for Bert is better. In [8] performed topic modelling using LDA and Bert, and resulted that Bert performs better than LDA.

Table 3. *Model results with Silhouette score*

	BERT Silhouette score	LDA Silhouette score
Raw	0.05213876	0.395832
with PCA	0.3522803	0.2730971
with Tsne	0.38252714	0.017188

3.3 RESPONSE GENERATION

I have created a sample of responses manually and associated with the topic depending on the sentiment positive or negative. This needs to be done automatically through NLP. I have done some research on how to generate text data through NLP. This work can be continued as the future work.



```
In [63]: df['response'][1]
Out[63]: 'Hope you enjoyed terms on month toshiba laptop case. Thank you.'
```

Fig. 17. Response for a positive customer review

4 CONCLUSION AND DISCUSSION

The service will help enterprises analyze customer reviews and create personalized responses to increase customer engagement. As per market research 80% of the consumers will increase brand loyalty with personalized content. The main technical challenge of the service is each model instance training took 4-5 hours to complete. We addressed this challenge through dimensionality reduction with limited improvements. The BERT model requires GPU's to process training data which is not available in local devices. We adopted Google Colab to rent on demand GPU units for Bert model execution saving time and resources.

In future scope and work for the service, we recommend diversifying the training dataset to multiple retail product categories to enhance the model performance. The scope of the service can expand to create multiple instances for multiple E-commerce providers such as Walmart, Amazon, Costco. The automatic response generation application can be improved to include NLP for text generation thereby increasing customer engagement. We also recommend creating an universal annotation pipeline which takes random sample of results and annotates it for false positives/ true positive to generate precision values/ FM scores to continuously monitor model behavior across all instances.

5 APPENDIX

5.1 SYSTEM SETUP

Install Jupyter notebook version 6.4.6 and necessary library through pip. For example, pip install pandas

Download the code from the zip uploaded.

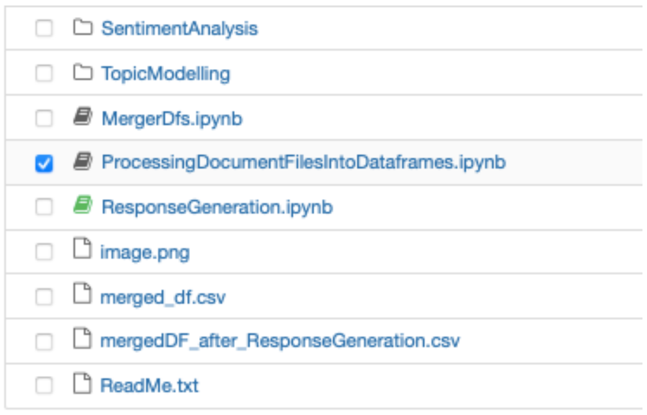
5.2 ACCESS TO DATA SETS

Download the dataset from the https://paperswithcode.com/dataset/oposum

5.3 DIRECTORY NAVIGATION

5.3.1 Processing Input Steps

(a) Start the execution of the project by running "ProcessingDocumentFilesIntoDataframes.ipynb" which inputs document files and create dataframes. This will save the input dataframes into folder "inputCSVs".



☐ SentimentAnalysis
☐ TopicModelling
☐ MergerDfs.ipynb
☑ ProcessingDocumentFilesIntoDataframes.ipynb
☐ ResponseGeneration.ipynb
☐ image.png
☐ merged_df.csv
☐ mergedDF_after_ResponseGeneration.csv
☐ ReadMe.txt

Fig. 18. File hierarchy - ProcessingDocumentFilesIntoDataframes

(b) Go to folder “Preprocessing”, start executing files type-"Preprocessing_bags_and_cases.ipynb" which cleans the input file and outputs the "bags_and_cases_processed.csv" in the same folder.

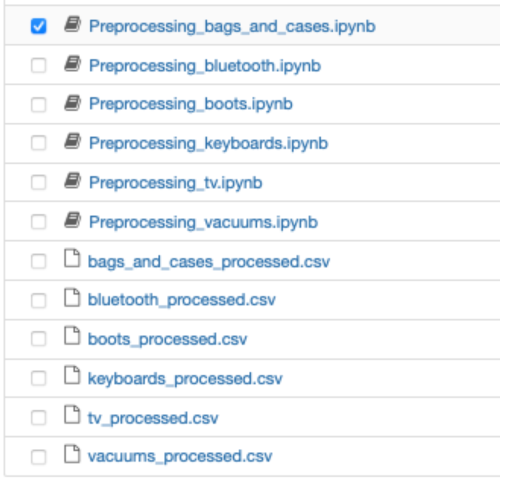

Preprocessing_bags_and_cases.ipynb
Preprocessing_bluetooth.ipynb
Preprocessing_boots.ipynb
Preprocessing_keyboards.ipynb
Preprocessing_tv.ipynb
Preprocessing_vacuums.ipynb
bags_and_cases_processed.csv
bluetooth_processed.csv
boots_processed.csv
keyboards_processed.csv
tv_processed.csv
vacuums_processed.csv

Fig. 19. File hierarchy - Preprocessing_bags_and_cases

(c) Go to folder “DataExploration”, run the files type-"DataExploration_bags_and_cases.ipynb" which does data exploration on the processed data and removes unwanted data and output "bags_and_cases_after_data_exploration.csv" in the same folder.

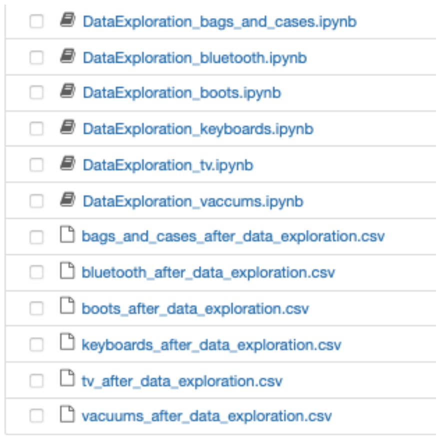

DataExploration_bags_and_cases.ipynb
DataExploration_bluetooth.ipynb
DataExploration_boots.ipynb
DataExploration_keyboards.ipynb
DataExploration_tv.ipynb
DataExploration_vaccums.ipynb
bags_and_cases_after_data_exploration.csv
bluetooth_after_data_exploration.csv
boots_after_data_exploration.csv
keyboards_after_data_exploration.csv
tv_after_data_exploration.csv
vacuums_after_data_exploration.csv

Fig. 20. File hierarchy - DataExploration_bags_and_cases

5.3.2 Merge dataframes

Run file "MergerDfs.ipynb" which merges the above files and create a single dataframe for the project.

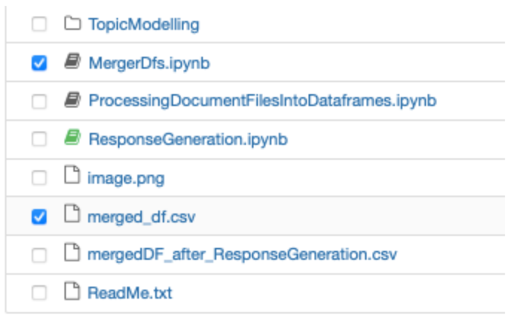

TopicModelling
MergerDfs.ipynb
ProcessingDocumentFilesIntoDataframes.ipynb
ResponseGeneration.ipynb
image.png
merged_df.csv
mergedDF_after_ResponseGeneration.csv
ReadMe.txt

Fig. 21. File hierarchy - MergerDfs

5.3.3 Sentiment Analysis

Go to folder "SentimentAnalysis", run "Sentiment_Analysis_NLTK_CustomerReviews.ipynb" and "Sentiment_Analysis_BERT_CustomerReviews.ipynb" to predict the sentiments and a new file "mergedDF_after_SentimentAnalysis.csv" will be generated in the same folder.

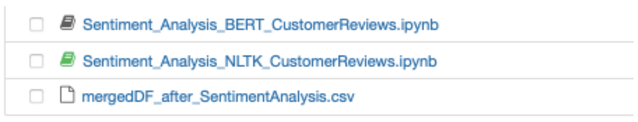

Sentiment_Analysis_BERT_CustomerReviews.ipynb
Sentiment_Analysis_NLTK_CustomerReviews.ipynb
mergedDF_after_SentimentAnalysis.csv

Fig. 22. File hierarchy - SentimentAnalysis

5.3.4 Topic Modelling

Go to folder "TopicModelling", run "LDA_and_BERT_for_CustomerReviews.ipynb" to create the topics and a new file "mergedDF_after_TopicModelling.csv" will be generated in the same folder.

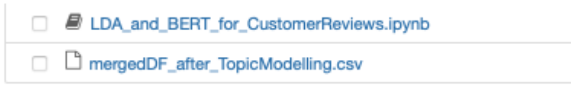

LDA_and_BERT_for_CustomerReviews.ipynb
mergedDF_after_TopicModelling.csv

Fig. 22. File hierarchy - TopicModelling

5.3.5 Response Generation

In the main folder, run "ResponseGeneration.ipynb" that used df after topic modelling, to create response for each customer review and outputs "mergedDF_after_ResponseGeneration.csv".

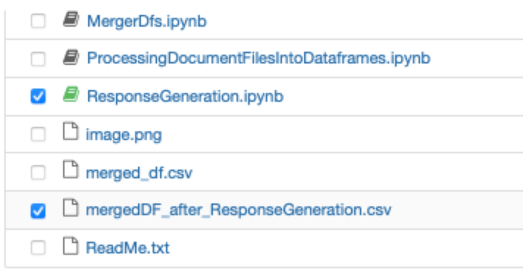

MergerDfs.ipynb
ProcessingDocumentFilesIntoDataframes.ipynb
ResponseGeneration.ipynb
image.png
merged_df.csv
mergedDF_after_ResponseGeneration.csv
ReadMe.txt

Fig. 23. File hierarchy - ResponseGeneration

The results of LDA will be saved in the "results" folder. The architecture of the project is available in "image.png".

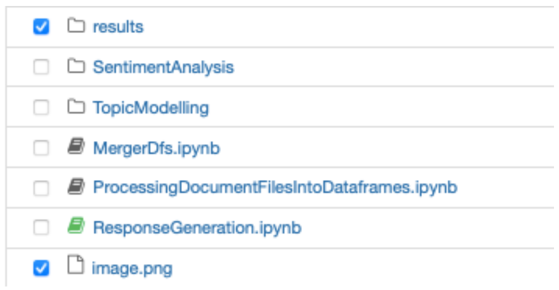


Fig. 23. File hierarchy – results and image

6 REFERENCES

[1] S. Angelidis and M. Lapata, "Summarizing Opinions: Aspect Extraction Meets Sentiment Prediction and They Are Both Weakly Supervised." 2018.

[2] E. Diaz-Aviles, M. Georgescu, A. Stewart, and W. Nejdl, "LDA for On-the-Fly Auto Tagging," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, 2010, pp. 309–312. doi: 10.1145/1864708.1864774.

[3] Z. Xue, "Scalable Text Analysis," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, p. 835. doi: 10.1145/3018661.3022750.

[4] Y. Yiran and S. Srivastava, "Aspect-Based Sentiment Analysis on Mobile Phone Reviews with LDA," in *Proceedings of the 2019 4th International Conference on Machine Learning Technologies*, 2019, pp. 101–105. doi: 10.1145/3340997.3341012.

[5] A. Farkhod, A. Abdusalomov, F. Makhmudov, and Y. I. Cho, "Lda-based topic modeling sentiment analysis using topic/document/sentence (Tds) model," *Applied Sciences (Switzerland)*, vol. 11, no. 23, Dec. 2021, doi: 10.3390/app112311091.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *CoRR*, vol. abs/1810.04805, 2018, [Online]. Available: http://arxiv.org/abs/1810.04805

[7] K. S. Nugroho, A. Y. Sukmadewa, H. Wuswilahaken DW, F. A. Bachtiar, and N. Yudistira, "BERT Fine-Tuning for Sentiment Analysis on Indonesian Mobile Apps Reviews," in *6th International Conference on Sustainable Information Engineering and Technology 2021*, New York, NY, USA: Association for Computing Machinery, 2021, pp. 258–264. [Online]. Available: https://doi-org.offcampus.lib.washington.edu/10.1145/3479645.3479679

[8] A. Abuzayed and H. Al-Khalifa, "BERT for Arabic Topic Modeling: An Experimental Study on BERTopic Technique," *Procedia Computer Science*, vol. 189, pp. 191–194, 2021, doi: https://doi.org/10.1016/j.procs.2021.05.096.