

## **Providing Key Insights to Flipkart Stakeholders Utilizing NLP Techniques**

Alex Ibrahim, Leslie Juarez, Chinonye Mgboji, Jessica Nguyen

School of Information, University of Texas

I 320D: Text Mining and NLP Essentials

Professor Abhijit Mishra

May 1, 2025

## **Providing Key Insights to Flipkart Stakeholders Utilizing NLP Techniques**

In a world driven by technology, online shopping platforms have become the norm for consumers around the world. With this in mind, understanding customer experiences has become more important than ever. Through our project, we aim to explore this topic utilizing a variety of NLP (Natural Language Processing) techniques to provide valuable insights to advancing understanding of user reviews.

Due to the nature of our research, we opted to use a dataset with scraped data from a popular e-commerce site called Flipkart. This data was chosen due to its valuable characteristics for our overall project goal, which included product reviews and sentiment categories for a variety of items on the site.

Techniques we deemed helpful in our analysis began with training a classification model to classify user reviews by sentiment category. Then, this data would be analyzed using Latent Dirichlet Allocation (LDA) to identify global trends for reviews and products by extracting relevant words for a product to find a common theme. Keyword extraction aids in determining what was most frequently discussed about a product. Topic analysis, on the other hand, works to find patterns between the reviews to highlight overall consumer feelings toward a particular product.

### **Data Description**

To gain insights into consumers' general thoughts, we used a CSV file containing 205,053 reviews on various FlipKart products, ranging anywhere from clothing, electronics, home decor, and more. These reviews were web-scraped from Flipkart's website in 2022 and uploaded to Kaggle (where we retrieved the dataset) under the Open Database License. It contains six

columns that include the product's name, the price, a one-to-five rating, the customer's review, a summary of the customer's thoughts, and a sentiment label based on the summary.

To pre-process the data, we began by removing any reviews with missing entries, as the dataset's large size would allow a few to be dropped without severely affecting our overall analysis. Additionally, we checked for and dropped any duplicate entries to prevent misleading results. We decided to remove the price, rating, and product name columns because our main focus was extracting keywords from the reviews themselves and finding the most popular topics in them. We also chose to remove the reviews that were classified with a neutral sentiment, as their lack of passion from the customer tells us significantly less about the best and worst aspects of the products. From there, we converted the sentiment column to a binary label system with a 1 for the positive sentiment and a 0 for the negative.

We then used various regular expressions to clean the text, allowing us to remove non-alphanumeric characters, convert digits into word form (1 to one), remove double spaces, and more. This was an important step as it limited the noisy text in the reviews. Additionally, we converted the reviews to lowercase letters, removed unnecessary stop words, and lemmatized the text to limit any noisy data further and ensure that the text we'd be reviewing is relevant to our analysis. After processing the data, we finally combined the review and summary columns into one processed review column for simplification.

After these lengthy pre-processing steps, we found that most of the reviews were positive, as 147,171 had a positive sentiment label and only 24,401 had a negative one, making approximately 85% of the reviews used in our analysis positive. Additionally, the average length of the reviews when fully processed was about 7 words.

## Methodology

After pre-processing and labeling, the data was split into three subsets: test, train, and validation. To evaluate the model's performance, 20% of the dataset was put aside to be the test set. The remaining data was split again to create a train (75%) and validation set (25%), resulting in a final split of 60% for training, 20% for validation, and for 20% testing. The reviews and sentiment labels were then separated and the splits were then saved using Pickle to be reused again later.

To convert the reviews into numbers our models could understand, we explored four different vectorization techniques: CountVectorizer, TF-IDFVectorizer, GloVe Embeddings, and Sentence Transformers. For each vectorizer, we trained three different classification models and evaluated their performance with the following metrics: accuracy, precision, recall, and F1-score - with precision of both classes being deemed as the most important. The metrics were then summarized into classification reports. The classification models used were Logistic Regression, Support Vector Machines (SVM), and Multi-Layer Perceptron (MLP) neural network.

### Vectorizers

#### *CountVectorizer*

CountVectorizer uses Bag-of-Words (BoW), which represents the text by counting how often each word appears. It creates a sparse vector where each element corresponds to a word in the vocabulary, which reflects the frequency of the word in the document. We initialized the vectorizer, limiting it to keep only the top 5,000 most common words in our dataset, and then fitted onto the training set to learn the vocabulary and used to transform the validation and test sets to maintain consistency (see Appendix A, Figure A1). This vectorizer was chosen as a

baseline, and offers a simple, effective way to capture word presence and frequency patterns in our dataset.

### ***TF-IDF Vectorizer***

TF-IDF furthers the BoW approach by weighing each word in accordance with their importance in a document, using the dataset as context. It calculates the term frequency-inverse document frequency (TF-IDF), highlighting more unique or informative terms over common terms. TF-IDF was also limited to the top 5,000 words, and then fitted onto the training set to transform the validation and test sets with the learned weights (see Appendix A, Figure A2).

TF-IDF was chosen due to its ability to emphasize discriminative terms, helping the model focus on words that are more indicative of sentiment.

### ***GloVe Embeddings***

Global Vectors for Word Representation (GloVe) uses pre-trained word embeddings that map words into a dense vector space, where semantically similar words are positioned closer together. We used the ‘glove-wiki-gigaword-100’ model to convert entire reviews into usable input vectors. We created a function that tokenized each review into individual words, retrieved the corresponding GloVe vectors when available, and computed the mean of those vectors. If no valid tokens were found in the GloVe vocabulary, the function returned a zero vector of appropriate length. The resulting embeddings were then stacked into NumPy arrays using `np.apply()` and `np.vstack()` for efficient batch processing across the training, validation, and test sets (see Appendix A, Figure A3). GloVe introduces semantic understanding into the models, which enable them to capture the relationships between similar words, unlike frequency-based methods.

GloVe improves the semantic understanding, but is limited by the static word representations. In product reviews, words have different meanings depending on the context, for example, a laptop being “light” can be a positive feature, whilst a dark jacket that is too “light” is a negative feature.

### ***Sentence Transformers***

Sentence Transformers provides dense, context-aware embeddings for full sentences using transformer-based neural networks. Sentence Transformers encodes entire reviews into single fixed-length vectors that account for word order, syntax, and contextual relationships. In our implementation, we used the ‘all-MiniLM-L6-v2’ pre-trained model, where the review texts were first converted to lists, and then passed to the model’s encoding to generate 768-dimensional embeddings. The embeddings were then stacked into NumPy arrays using `np.vstack()` to form consistent feature matrices for the training, validation, and test sets (see Appendix A, Figure A4). Sentence transformers allows us to capture deeper contextual information and sentence-level meaning, which makes it effective for subtle sentiment cues like sarcasm.

### **Classification Models**

In order to evaluate the effectiveness of each vectorization technique, we used Logistic Regression, Support Vector Machines (SVM), and Multi-Layer Perceptron (MLP) neural networks, which are classification models that are commonly used in NLP tasks.

Logistic Regression is a linear classifier that’s widely used in binary classification due to its simplicity and speed. It was used to test which vectorizations were effective and gave us a baseline to compare other models as well.

SVM performs well with high-dimensional data like our reviews, and was especially effective with TF-IDF, where it helped to identify between positive and negative reviews using weighted word importance. It handles short and more neutral-sounding reviews better than Logistic Regression.

MLP neural networks are used for their ability to model non-linear relationships, and in our case, denser inputs like GloVe and Sentence Transformers. It's most effective for reviews where sentiments are implied rather than directly stated, for example, "better than I thought".

### **Latent Dirichlet Allocation (LDA)**

LDA is an unsupervised NLP technique that is used to create a set number of topics from a document or set of documents. The goal is to be able to sort initial documents into the newly formed topics to create human-interpretable groups of documents. For example, using LDA to sort a collection of books into genres. However, as this is an unsupervised method, there isn't a guarantee that the topics will be interpretable – which is why a coherence score is used to measure the effectiveness of a topic.

### **Figure 5**

*Formula for U-Mass coherence score*

$$C_{UMass}(w_i, w_j) = \log \frac{D(w_i, w_j) + 1}{D(w_i)},$$

*Note.* U-Mass coherence formula, where  $D(w_i, w_j)$  is the frequency of the two words seen together in document(s)  $D$ , and  $D(w_i)$  is the frequency of  $w_i$  in document(s)  $D$ . From

Zvornicanin, E. (2025, February 28). When coherence score is good or bad in topic modeling?  
 [Online Image] Baeldung <https://www.baeldung.com/cs/topic-modeling-coherence-score>

Coherence score (specifically U-Mass coherence score) is represented in Figure 5. where  $D$  is a document and  $w$  is a word. The goal of the formula is to find the probability of a word( $i$ ) occurring with another word( $j$ ) for all instances of that word( $i$ ) occurring in the document. In other words, the frequency of a word appearing, but only when accompanied by another specific word. The result of this is that a lower (U-Mass becomes more negative as it improves) score is achieved when a word very commonly occurs with another word while rarely appearing with other words. This maximizes the occurrence of a word in a certain topic and minimizes its occurrence outside of it.

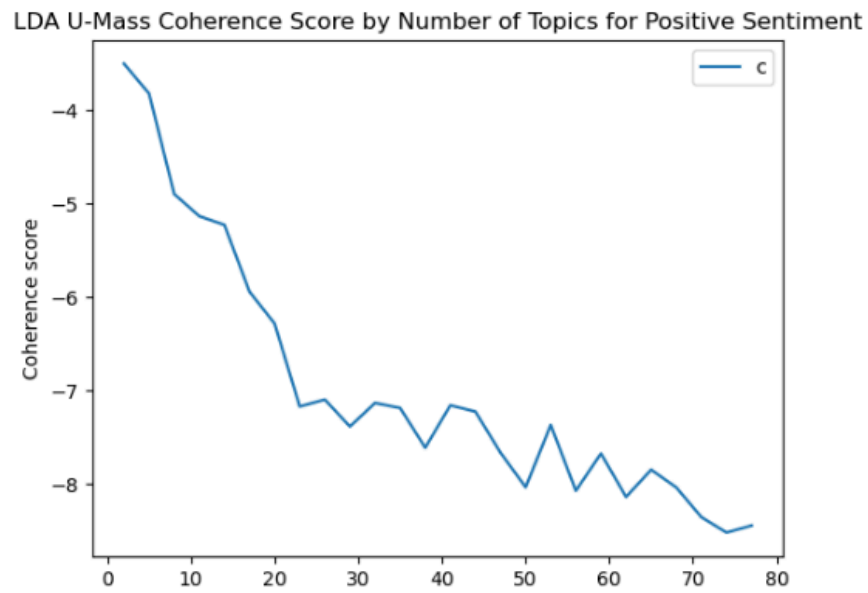
To determine the number of topics to use for a model, you create multiple models with a different number of topics and then find the coherence score for each of them. Best practice is to select the number of topics where the coherence score stops receiving rapid improvements. This is as past this point, improvements will be marginal, while increasing the number of topics generally makes the results less useful (ex. dividing 10 books into 10 topics isn't useful for anyone).

For our purposes, we created an LDA model for both our positive sentiment reviews and negative sentiment reviews (separately). We generated the coherence score for topics 2 to 80, taking 3 topic steps. This can be seen in Figures 6 & 7 below. Based on this, we determined that 26 topics for positive sentiment reviews and 29 for negative sentiment reviews was the optimal topic number. These topics were then interpreted by a human for results.

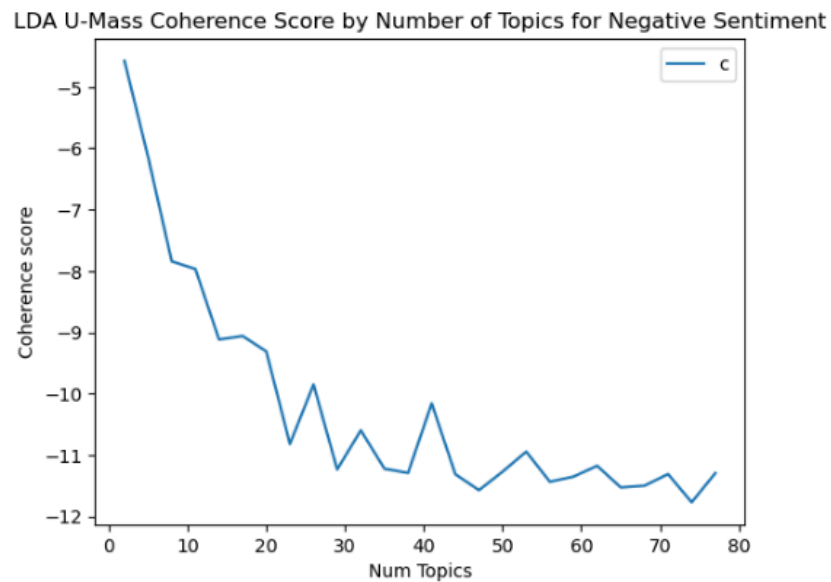


**Figure 6**

*Coherence Score for LDA topic numbers 2 to 80 (3 topic steps) for **positive** reviews*

**Figure 7**

*Coherence Score for LDA topic numbers 2 to 80 (3 topic steps) for **negative** reviews*



## Results

### Sentiment Analysis

Tables B1-B3 in Appendix B below shows the results of all of the sentiment analysis models. Every model-feature pair performed similarly across all metrics, with the main exceptions being BoW-SVM & GloVe-SVM performing notably worse on correctly predicting True Negatives (low negative class recall). This is likely due to SVMs overall performing worse—something that could be attributed to limiting the number of iterations allowed to 1,000, resulting in the SVM models not reaching convergence.

The best performing models—determined primarily by the precision of each class—are BoW-MLP, followed by BoW-Logistic & TF-IDF-MLP, and TF-IDF-Logistic (in that order). A commonality among all the results is that class 0 (the negative class) had both lower precision (around 5 points lower on average) and lower recall (around 15 to 20 points lower on average). The most likely cause of this is the imbalance of the two classes, with the positive (1) class making up 85% of the data. Balancing these classes or changing the threshold for the classification are possible future implementations that could help fix this issue as we seek class to equal recall and precision across classes.

It is worth noting that sentiment is not being limited to the product itself in these reviews, but the overall review. This means that if there is a positive or negative brought up in the review not related to the product—for example: Shipping - it will still be classified as based on the sentiment of the review (not the product). As reviews related to products and reviews related to other factors like shipping are important to different stakeholders - it is worthwhile to make a distinction between the origins of these sentiments in the future. A post-hoc solution would be to utilize XAI techniques like LIME to identify keywords in the models' classification. This could

allow for the distinction of product-based sentiments and other sentiments by finding keywords related to either.

### **Latent Dirichlet Allocation (LDA)**

For each of the 26 positive topics and 29 negative topics created through LDA, the top 25 words were collected, along with their topic importances. These words were then interpreted by a human in an attempt to summarize what each topic is about. The results of this are shown below in Table 1. It is notable that the positive topics made much more sense as positive topics than the negative ones did as negative topics, with multiple negative topics easily passing for positive ones if you just look at the words. It has not been discovered why this is the case. These instances have been highlighted in red (see Table 1).

**Table 1**

*List of summaries for all topics*

Positive Topic Number	Positive Topic Summary	Negative Topic Number	Negative Topic Summary
0	Cheap/Low Prices	0	Numbers (just a bunch of numbers)
1	Easy to use/install	1	Not worth cost
2	Quick Shipping	2	Cheap quality
3	Brilliant or Mind blowing product	3	Difficult installation
4	Working perfectly	4	Useless product
5	Excellent or reasonable product	5	Awful product
6	A fabulous or lovely product	6	Bad Material
7	Numbers (just a bunch of numbers)	7	Received damage/broken

8	Superb product	8	Loved product
9	Wonderful product	9	Damaged Packaging
10	Small/good size	10	Unsatisfactory purchase
11	Hassle-free	11	Small size
12	Exceeded expectation	12	Disappointing product
13	Table set (specific type of product)	13	Poor condition
14	Long lasting.	14	Late Delivery
15	Recommended product	15	Do not buy
16	Must buy product	16	Bad quality product
17	Good quality/ delightful product	17	Good table
18	Good purchase	18	Expected good/ is worthless
19	Compact/lightweight	19	Horrible condition
20	Good assemble support/demo	20	Not working
21	Fast/good delivery	21	Expensive
22	Nice/good product	22	Slow delivery
23	Classy/valuable products	23	Short life span
24	Beautiful product	24	Bad quality for price
25	Overall good product	25	Wouldn't recomend
26	Great for price	26	Good product
		27	Worth every penny
		28	Waste of money

		29	Product not received
--	--	----	----------------------

*Note.* The red boxes in the “Negative Topic Summary” column represent topics that were made up of negative sentiment reviews but resulted in seemingly positive topics.

By analyzing all these topics together, we came up with 6 common categories across them. Not all of the topics fit into one of these categories but most of them do. They are as follows:

**Table 2**

*Common Categories found across LDA topics*

<u>Category</u>	<u>Description</u>	<u>Additional</u>
<b>Quality</b>	How good/bad is the product?	Below expectations came up commonly in negative topics
<b>“Vibe”</b>	How consumers respond to products (delightful, lovely, etc.)	Only appeared in positive topics
<b>Price</b>	How cheap or expensive	Often compared to expected worth in negative topics
<b>Shipping</b>	Was shipping good or bad? (fast, well packaged, etc.)	None
<b>Easy of Use</b>	Is it simple or hard to use?	None
<b>Condition</b>	Did it arrive in a good or bad condition?	None

This tells us that in addition to the product itself, many Flipkart users also leave reviews about the shipping and delivery processes. It is important to make this distinction as they are connected to different stakeholders. The platform of Flipkart is much more interested in knowing about how the shipping process is being received compared to sellers that only have control over

their product. Therefore, the sentiments of reviews could be related to different stakeholders and being able to separate them clearly is important.

## **Conclusion**

### **Key Findings**

As seen in the above results section, through the use of NLP techniques, our team was able to analyze customer reviews using classification models and LDA successfully. We found that to predict a user review with the best accuracy, utilizing a BoW-MLP model worked best compared to other training methods. Whereas for LDA, our team extracted 25 keywords per topic within a respective sentiment category which we then used to analyze the topic. Six common themes were identified: Quality, Price, Shipping, Ease of Use/Installation, Condition, and, interestingly, the “vibe” of said product.

### **Limitations**

The first limitation we encountered began in the early stages of our project. We began our work in Google Colab for easier cooperation between group members. Google Colab uses Python 3.10. For different parts of our project, we needed to use the Gensim and the Sentence Transformers modules. However, in Python 3.10, Gensim and Sentence Transformers both require a different version of NumPy to function (Gensim needs an older version; Sentence Transformers need current). This was originally resolved by ordering the code so that the part that required Sentence Transformers came after we no longer needed Gensim so we could uninstall Gensim, upgrade NumPy, and then install Sentence Transformers. This issue is also possible to resolve using Python 3.11. Later in testing, we also realized our dataset was unbalanced which resulted in a loss of accuracy for the negative class in all of our classification

models. While we did not balance the data at the time, this issue can be considered in future work.

We also noticed limitations within the individual classification models. Logistic Regression is not able to handle complex, non-linear patterns or wording differences in comparison to other advanced models. SVM, on the other hand, is more suited to handle high-dimensional data; however, this model is difficult to measure as it does not provide direct probability scores and is slow to train on larger datasets. Finally, using MLP neural networks also had its limitations. The data imbalance mentioned previously caused several inconsistencies in the results compared to other models. Due to its nature, MLP also took a longer time to train.

Individually, the vectorizers also had their limitations. Vectorizers relying on the count of words (CountVectorizers and TF-IDF Vectorizers) failed to capture the overarching context of our data as they consider each token independently. This issue could also potentially lead to overfitting. However, the main issue was that such vectorizers would misinterpret phrases and thus incorrectly classify reviews by sentiment, particularly reviews containing sarcasm. GloVe's vector assignment by word, regardless of usage, was also ambiguous. Data like reviews can be noisy and contain domain-specific terms, brand names, abbreviations, or user misspellings, which are not within GloVe's pretrained vocabulary. This defaults the token to a zero vector that weakens the final representation. Lastly, Sentence Transformers effectively capture context and sentiment but can be slow and less accurate on slang-heavy e-commerce reviews due to limited domain training. Nonetheless, each vectorizer's effectiveness varies when paired with different models.

## **Future Work and Applications**

Our future work will focus on improving the recall score for the positive class (1) by exploring class rebalancing techniques or adjusting the classification threshold. To improve scores overall, we will also focus on improving data quality by addressing potential noise introduced by reviews of a specific product by various sellers, as well as attempting to review our dataset to feature a more balanced mix of positive/negative training reviews.

Overall, future applications of a fully functioning, refined sentiment analysis and topic modeling pipeline can be used by a multitude of e-commerce platforms to flag underperforming or low-quality products, distinguish between seller and product issues, and provide feasible feedback for sellers. With more work, this system could also be modified for other domains reliant on providing a positive experience for users. With this in mind, our project not only fulfills its initial goal of bridging the gap between consumers and sellers but also lays a foundation for other applications and meaningful impact.

### ***Code***

The code and dataset used for this project can be found in this GitHub Repository:

<https://github.com/aeftyb/NLP-finalProject>



## References

Vaghani, N., & Thummar, M. (2023). *Flipkart Product reviews with sentiment Dataset* [Dataset]. Kaggle.

<https://www.kaggle.com/datasets/niraliivaghani/flipkart-product-customer-reviews-dataset>

Varma, A. (2022, August 26). Topic Modeling with Gensim (Python). Machine Learning Plus.

<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/#17howtofindtheoptimalnumberoftopicsforlda>

Zvornicanin, E. (2025, February 28). When coherence score is good or bad in topic modeling?

[Online Image] Baeldung <https://www.baeldung.com/cs/topic-modeling-coherence-score>

## Appendix A

### Figure A1

*Vectorization with CountVectorizer*

```
# Initialize count vectorizer
count_vectorizer = CountVectorizer(max_features=5000)
# Fit vectorizer on train data
X_train_count_vec = count_vectorizer.fit_transform(X_train)
# Used fitted vectorizer to transform test and val data
X_val_count_vec = count_vectorizer.transform(X_val)
X_test_count_vec = count_vectorizer.transform(X_test)
```

### Figure A2

*Vectorization with TF-IDF*

```
# Initialize count vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
# Fit vectorizer on train data
X_train_tfidf_vec = tfidf_vectorizer.fit_transform(X_train)
# Used fitted vectorizer to transform test and val data
X_val_tfidf_vec = tfidf_vectorizer.transform(X_val)
X_test_tfidf_vec = tfidf_vectorizer.transform(X_test)
```

**Figure A3***Vectorization with GloVe Embedding*

```

# Load GloVe embeddings
# Only take the time to install GloVe if it was not been installed yet
try:
    print(glove_model)
except:
    glove_model = api.load("glove-wiki-gigaword-100")

# Create a function for assigning embeddings
def glove_embeddings(review,glove_model=glove_model):
    """
    Input: a single review in the form of one continuous string.
    Output: Mean of all token embeddings in review.

    Steps:
    1.) Tokenize review by words.
    2.) Assign embedding to each token (word) from glove model.
    3.) Return mean of all embeddings.

    """
    tokens = review.split()
    embeddings = [glove_model[token] for token in tokens if token in glove_model]
    # Handle the case where no embeddings are found for the review
    if not embeddings:
        # Return a zero vector of the correct size
        return np.zeros(glove_model.vector_size)
    else:
        # Return mean of embeddings, reshaped to a 1D array
        return np.mean(embeddings, axis=0).reshape(1, -1) # Reshape ensures consistent dimensions

# Embed reviews
# Use pandas apply for parallel processing
# Stack the embeddings into a 2D NumPy array
X_train_glove = np.vstack(X_train.progress_apply(glove_embeddings).to_numpy()) # Stack embeddings into a 2D array
X_val_glove = np.vstack(X_val.progress_apply(glove_embeddings).to_numpy()) # Stack embeddings into a 2D array
X_test_glove = np.vstack(X_test.progress_apply(glove_embeddings).to_numpy()) # Stack embeddings into a 2D array

```

**Figure A4***Vectorization with Sentence Transformers*

```

model_name = 'sentence-transformers/all-MiniLM-L6-v2'
model = SentenceTransformer(model_name)

X_train_embeddings = np.vstack(model.encode(X_train.tolist()))
X_val_embeddings = np.vstack(model.encode(X_val.tolist()))
X_test_embeddings = np.vstack(model.encode(X_test.tolist()))

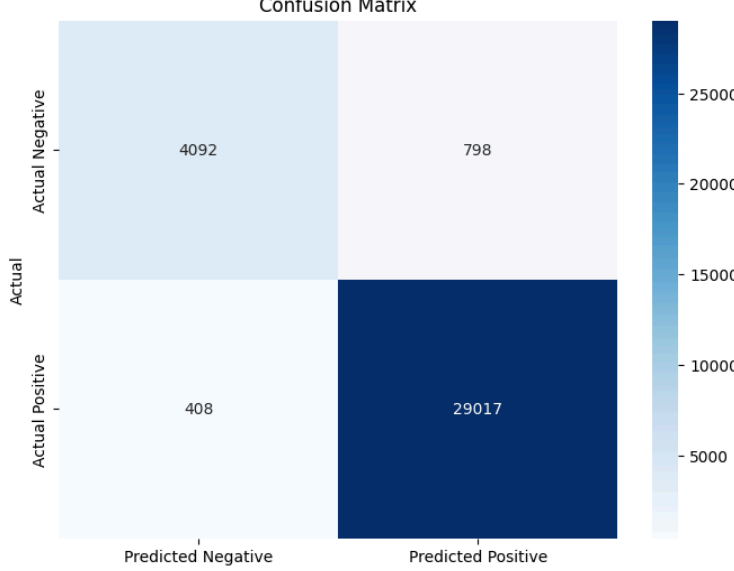
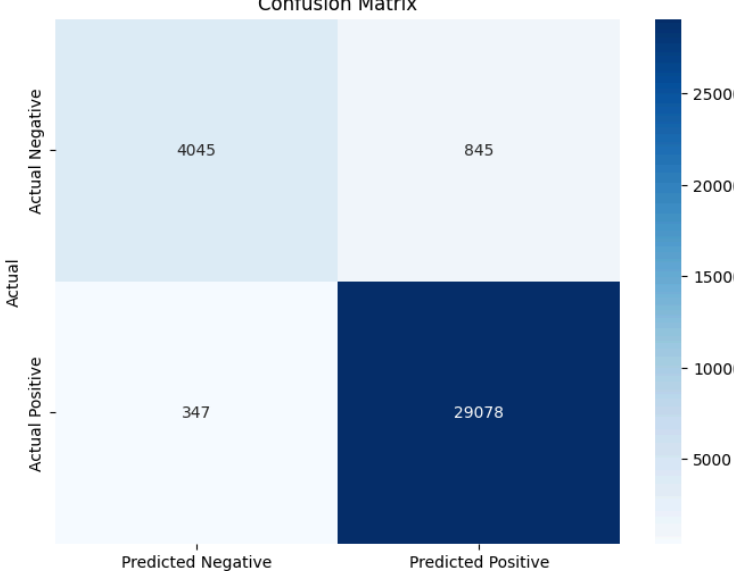
```

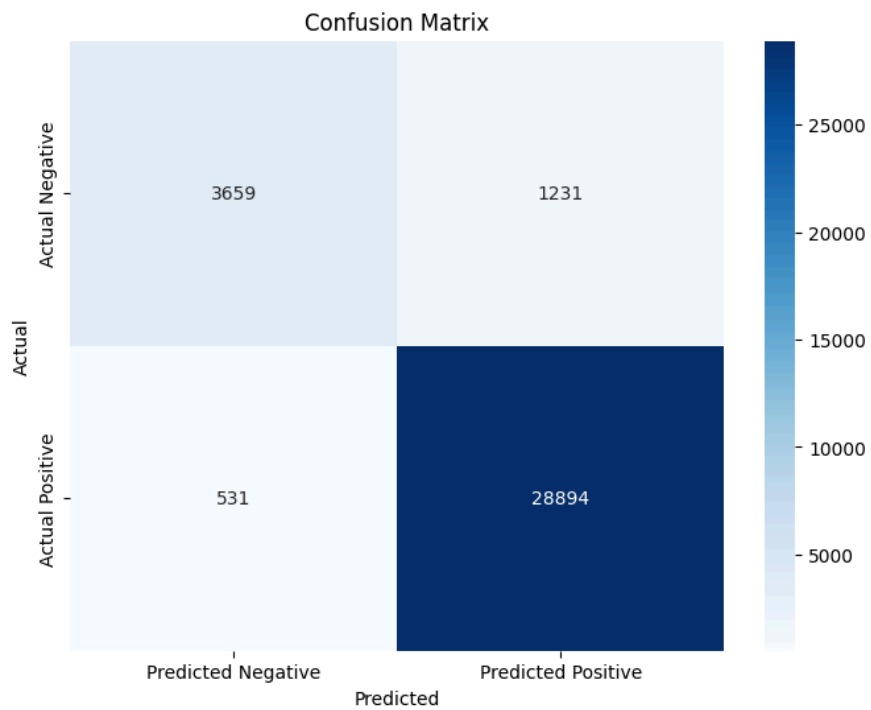
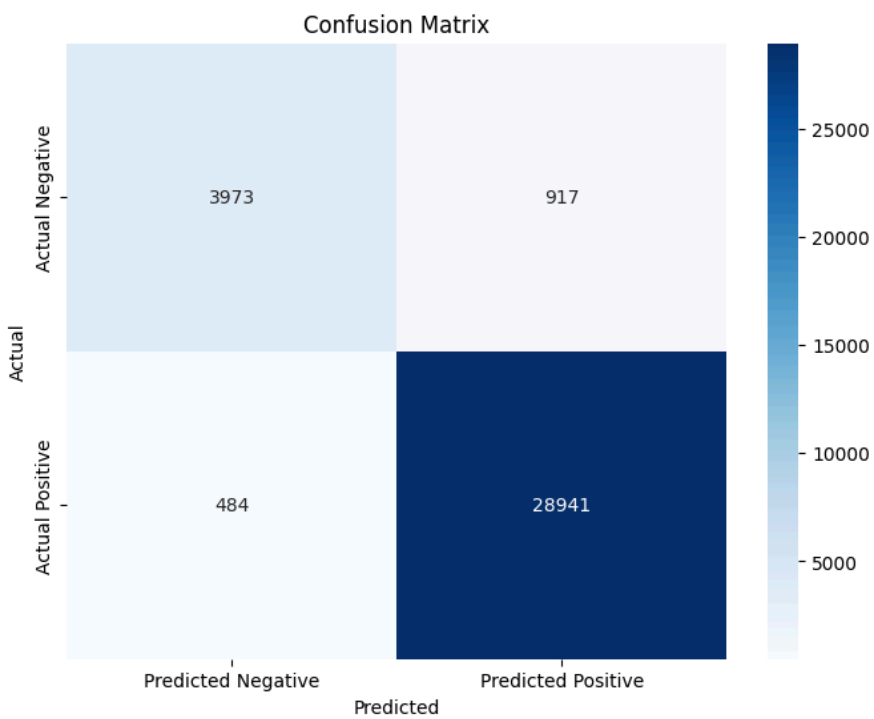
## Appendix B

### Confusion Matrices of Each Model & Vectorizer Pair

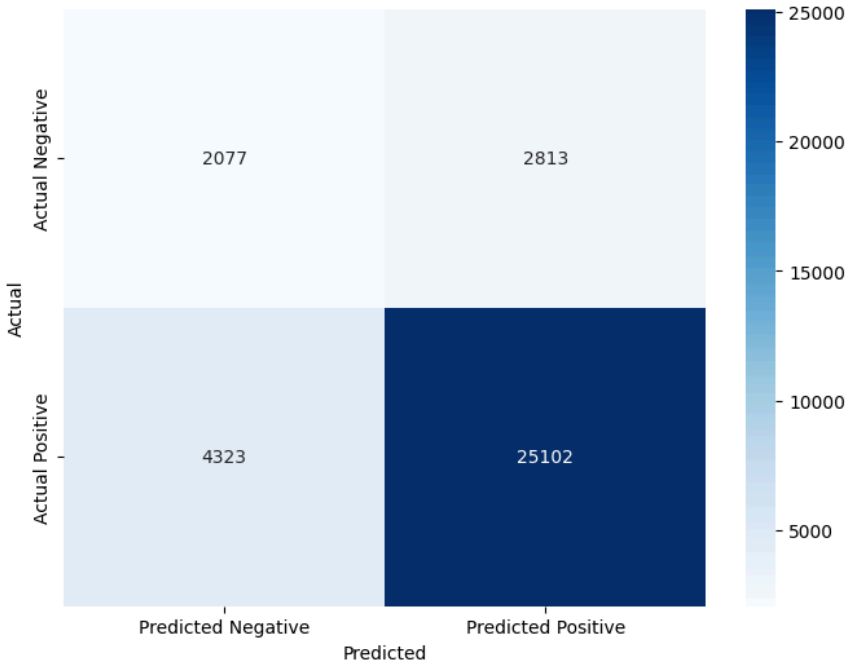
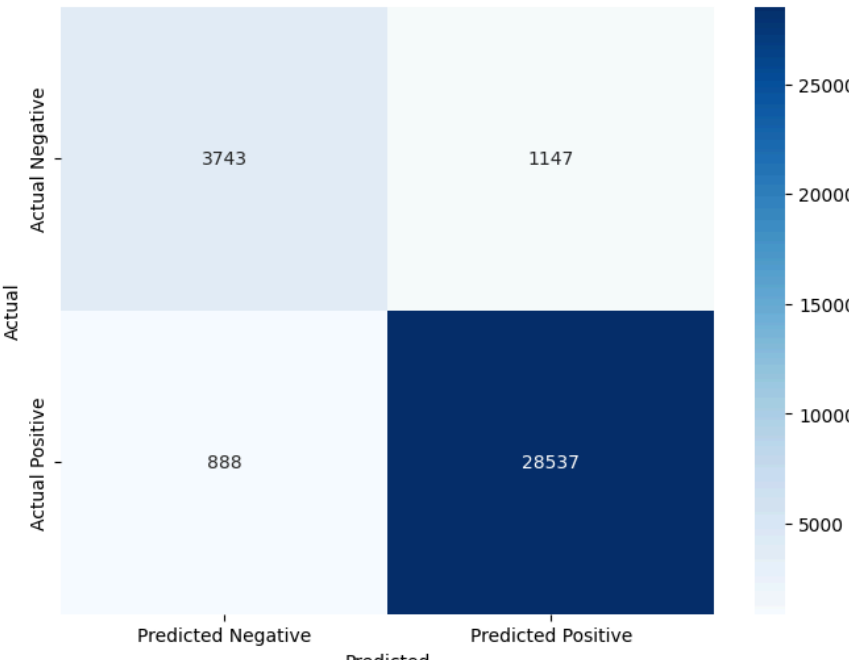
**Table B1**

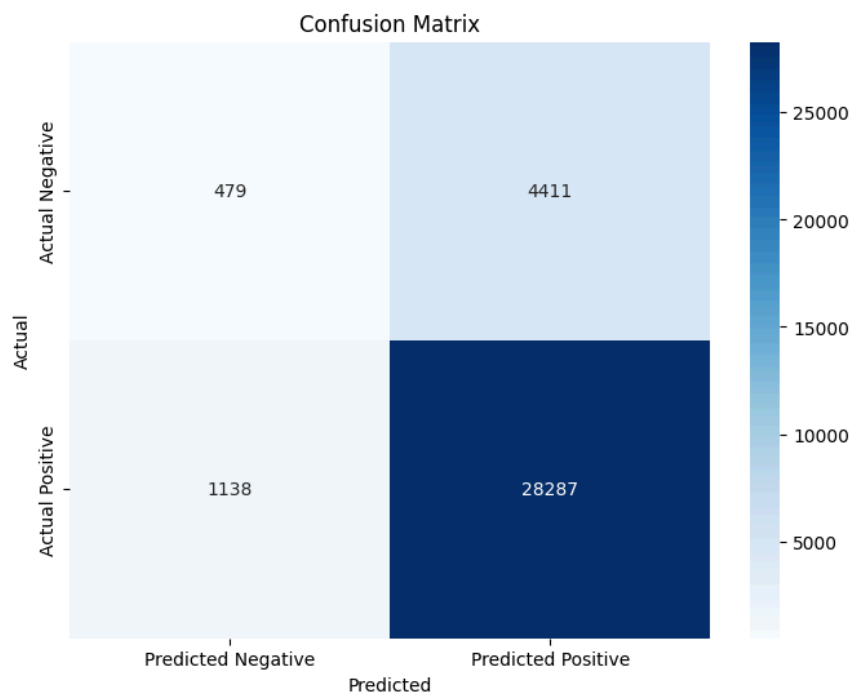
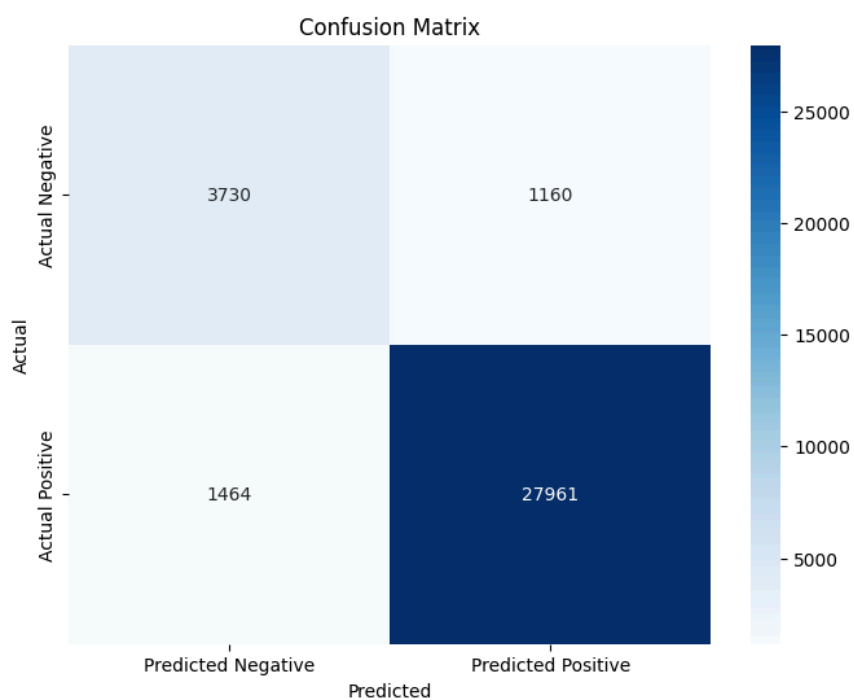
#### *Logistic Regression Classification Results*

	<u>Logistic Regression</u>									
<u>Count Vector (BoW)</u>	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Negative</th><th>Predicted Positive</th></tr><tr><th>Actual Negative</th><td>4092</td><td>798</td></tr><tr><th>Actual Positive</th><td>408</td><td>29017</td></tr></table></div>		Predicted Negative	Predicted Positive	Actual Negative	4092	798	Actual Positive	408	29017
	Predicted Negative	Predicted Positive								
Actual Negative	4092	798								
Actual Positive	408	29017								
<u>TF-IDF</u>	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Negative</th><th>Predicted Positive</th></tr><tr><th>Actual Negative</th><td>4045</td><td>845</td></tr><tr><th>Actual Positive</th><td>347</td><td>29078</td></tr></table></div>		Predicted Negative	Predicted Positive	Actual Negative	4045	845	Actual Positive	347	29078
	Predicted Negative	Predicted Positive								
Actual Negative	4045	845								
Actual Positive	347	29078								

GloVeSentence Transformer

**Table B2***SVM Classification Results*

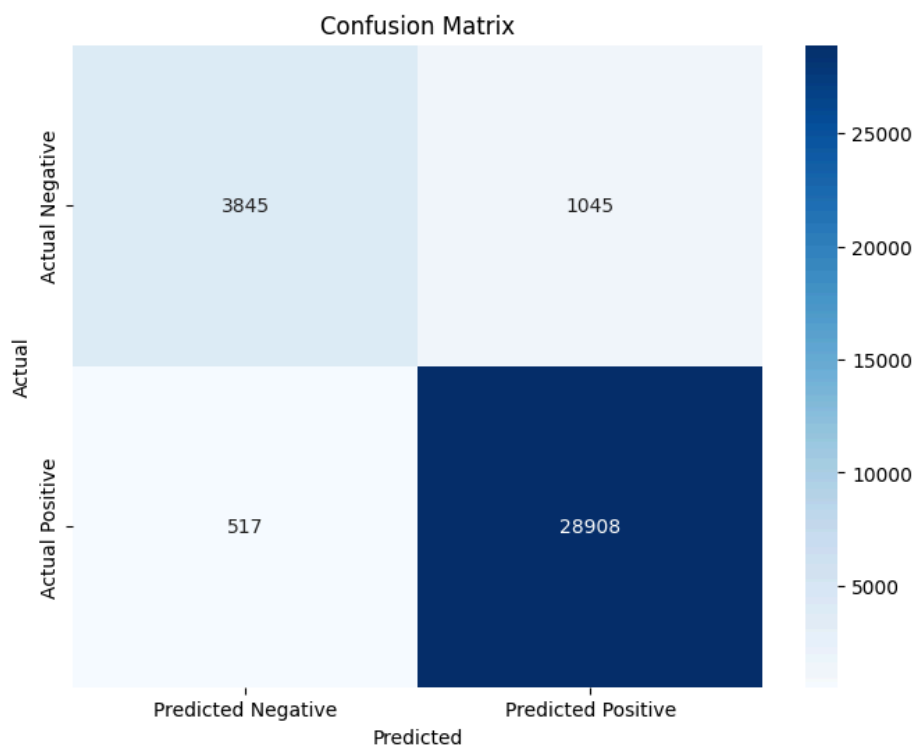
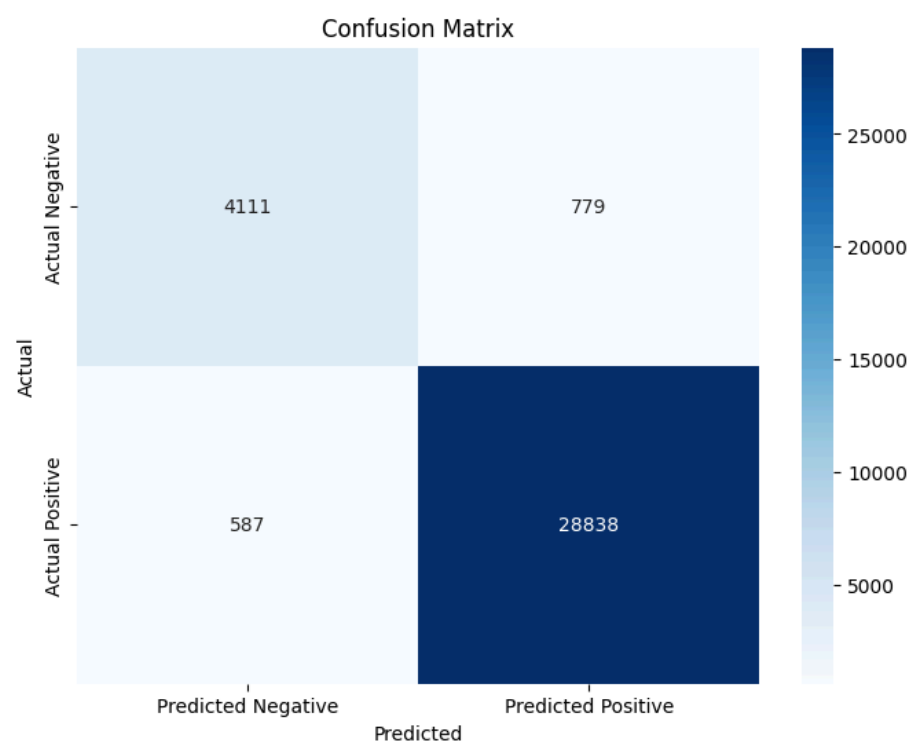
	<u>SVM</u>									
<u>Count Vector (BoW)</u>	<div><p>Confusion Matrix</p><table><tr><th>Actual \ Predicted</th><th>Predicted Negative</th><th>Predicted Positive</th></tr><tr><th>Actual Negative</th><td>2077</td><td>2813</td></tr><tr><th>Actual Positive</th><td>4323</td><td>25102</td></tr></table></div>	Actual \ Predicted	Predicted Negative	Predicted Positive	Actual Negative	2077	2813	Actual Positive	4323	25102
Actual \ Predicted	Predicted Negative	Predicted Positive								
Actual Negative	2077	2813								
Actual Positive	4323	25102								
<u>TF-IDF</u>	<div><p>Confusion Matrix</p><table><tr><th>Actual \ Predicted</th><th>Predicted Negative</th><th>Predicted Positive</th></tr><tr><th>Actual Negative</th><td>3743</td><td>1147</td></tr><tr><th>Actual Positive</th><td>888</td><td>28537</td></tr></table></div>	Actual \ Predicted	Predicted Negative	Predicted Positive	Actual Negative	3743	1147	Actual Positive	888	28537
Actual \ Predicted	Predicted Negative	Predicted Positive								
Actual Negative	3743	1147								
Actual Positive	888	28537								

GloVeSentence Transformer

**Table B3**  
*MLP Classification Results*

	<u>MLP</u>									
<u>Count Vector (BoW)</u>	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Negative</th><th>Predicted Positive</th></tr><tr><th>Actual Negative</th><td>4202</td><td>688</td></tr><tr><th>Actual Positive</th><td>465</td><td>28960</td></tr></table></div>		Predicted Negative	Predicted Positive	Actual Negative	4202	688	Actual Positive	465	28960
	Predicted Negative	Predicted Positive								
Actual Negative	4202	688								
Actual Positive	465	28960								
<u>TF-IDF</u>	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Negative</th><th>Predicted Positive</th></tr><tr><th>Actual Negative</th><td>4159</td><td>731</td></tr><tr><th>Actual Positive</th><td>465</td><td>28960</td></tr></table></div>		Predicted Negative	Predicted Positive	Actual Negative	4159	731	Actual Positive	465	28960
	Predicted Negative	Predicted Positive								
Actual Negative	4159	731								
Actual Positive	465	28960								



GloVeSentence Transformer



### Figure C2

### Top 50 Keywords in Negative Reviews



*Note.* A word cloud was generated to display the top 50 keywords in negative reviews. The largest words are the most common words, while the smaller ones were seen less commonly. The dominant keywords in this category are product, bad, quality, waste, dont, buy, work, money, disappoint. The keywords indicate that the focus is around the low product quality and wasted money, emphasizing the disappointment from customers.

## Appendix D

### Questions & Answers From the Presentation

**Q1: Why is recall for positive class consistently low?**

A: To begin, we used 0 to represent a negative class for our purposes and it was the 0 class that consistently had a low recall. As for why, the most likely explanation is the lack of class balance. 85% of reviews had a positive sentiment to begin with and there were some sarcastic negative sentiment reviews in the dataset that used similar words to positive sentiment reviews. It is likely that by balancing the classes, we would at the very least see more equal metrics across classes. As for sarcasm, Sentence Transformers represent the best way to address them due to embeddings containing semantic meaning but, our current text cleaning process has us remove stopwords and lemmatize our text - which usually leads to worse performance with transformer models that expected raw text.

**Q2: What does D represent in your coherence score calculations?**

A: D represents a document or set of documents from which a word is drawn. For example, D could represent the review: “super! great cooler excellent air flow.” from which word i - “great”, and word j - “cooler” would be taken, to calculate how often “cooler” occurs with “great” for everytime “great” appears in the document D.

**Q3: How did you handle cases where different websites (or shipping experiences) affected product reviews?**

A: We currently are making no distinction between the two. However, we acknowledge that in future work, it will be important to separate them as they represent different stakeholders (Product = Seller; Shipping = Flipkart). A post-hoc solution would be to utilize explainable AI - such as LIME - to find which words have the biggest impact on the sentiment classification of

shipping when compared to the product and utilize that to separate the sentiments. Future work will likely need to be conducted to properly separate reviews by stakeholders involved and have independent models classify the sentiments.

**Q4: Did you check if topics drifted when moving across product categories (e.g., electronics vs. clothing)?**

A: While there were niche differences (how you describe a cheap electronic may be different from the clothing which is then reflected in the topic), the topics could generally be applied across products. There was one notable anomaly being tables/table sets. It appears that tables both came up so frequently and were commonly mentioned in reviews regarding them, which led to an entire topic (in both positive and negative sentiments) being dedicated to tables/table sets.