Aspect-Based Sentiment Analysis Recommender System

Qasim Naveed

21L-5231

Rameez

21L-1775

Afnan Asif

21IL-1864

Taha Sheikh

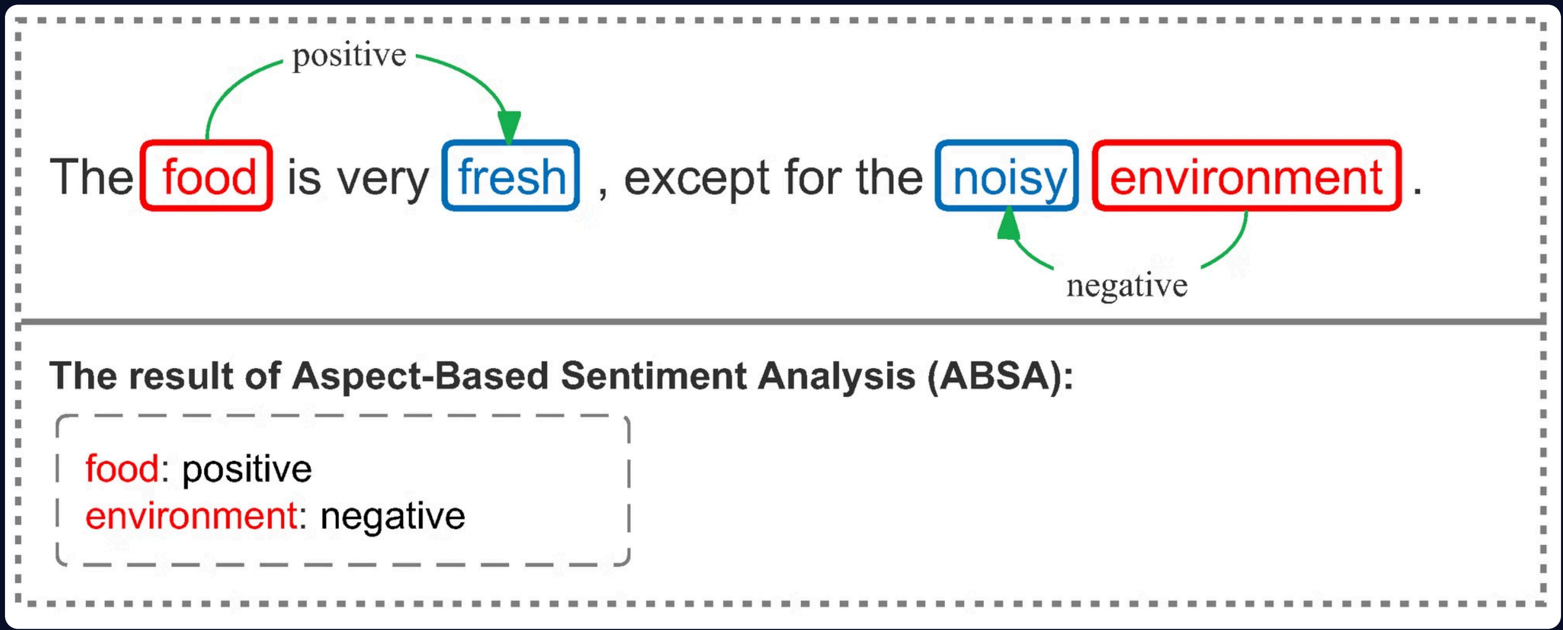
21L-5175

Agenda

1. Introduction
2. Data Exploration
3. Preprocessing
4. Aspect Classification
5. Sentiment Analysis
6. Recommendation Logic
7. Deployment
8. Conclusion

Introduction

- What is ABSA?
- Aspect-Based Sentiment Analysis breaks reviews into aspects and analyzes sentiments for each.
- Why is ABSA important?
- Helps businesses understand detailed customer feedback.
- Powers personalized product recommendations.



Dataset Overview

- Dataset: Women Clothing E-Commerce Reviews (Kaggle).
- Key Features:
- Review Text: Detailed customer reviews.
- Rating: Numerical scores.
- Recommended IND: Binary indicator for product recommendation.
- Total Records: ~23,000

	Clothing ID	Age	Title	Review	Rating	Sentiment	Positive Fe	Division Na	Departme	Class Name
0	767	33		Absolutely wonderful - silky and sexy and comfortable	4	1	0	Initmates	Intimate	Intimates
1	1080	34		Love this dress! it's sooo pretty. i happened to find it in a store, and i'm glad i	5	1	4	General	Dresses	Dresses
2	1077	60	Some ma	I had such high hopes for this dress and really wanted it to work for me. i initia	3	0	0	General	Dresses	Dresses
3	1049	50	My favorit	I love, love, love this jumpsuit. it's fun, flirty, and fabulous! every time i wear it	5	1	0	General Pe	Bottoms	Pants
4	847	47	Flattering	This shirt is very flattering to all due to the adjustable front tie. it is the perfec	5	1	6	General	Tops	Blouses
5	1080	49	Not for th	I love tracy reese dresses, but this one is not for the very petite. i am just und	2	0	4	General	Dresses	Dresses
6	858	39	Cagrcoal	I aded this in my basket at hte last mintue to see what it would look like in per	5	1	1	General Pe	Tops	Knits
7	858	39	Shimmer,	I ordered this in carbon for store pick up, and had a ton of stuff (as always) to	4	1	4	General Pe	Tops	Knits
8	1077	24	Flattering	I love this dress. i usually get an xs but it runs a little snug in bust so i ordered	5	1	0	General	Dresses	Dresses
9	1077	34	Such a fu	I'm 5"5' and 125 lbs. i ordered the s petite to make sure the length wasn't too	5	1	0	General	Dresses	Dresses
10	1077	53	Dress loo	Dress runs small esp where the zipper area runs. i ordered the sp which typic	3	0	14	General	Dresses	Dresses
11	1095	39		This dress is perfection! so pretty and flattering.	5	1	2	General Pe	Dresses	Dresses
12	1095	53	Perfect!!!	More and more i find myself reliant on the reviews written by savvy shoppers l	5	1	2	General Pe	Dresses	Dresses
13	767	44	Runs big	Bought the black xs to go under the larkspur midi dress because they didn't	5	1	0	Initmates	Intimate	Intimates
14	1077	50	Pretty par	This is a nice choice for holiday gatherings. i like that the length grazes the kr	3	1	1	General	Dresses	Dresses
15	1065	47	Nice, but	I took these out of the package and wanted them to fit so badly, but i could te	4	1	3	General	Bottoms	Pants
16	1065	34	You need	Material and color is nice. the leg opening is very large. i am 5'1 (100#) and t	3	1	2	General	Bottoms	Pants

Data Preprocessing

- Why Preprocessing?
- Remove noise for better model performance.
- Standardize input for consistent analysis.
- Steps Taken:
- Lowercasing, punctuation removal, stopword removal.
- Balancing the positive and negative sentiments.
- Lemmatization to normalize words.

```python
def read_data(path: str) -> pd.DataFrame:
    try:
        df = pd.read_csv(path, header=0, index_col=0)
        return df
    except Exception as e:
        print(f"Error loading data: {str(e)}")
        return pd.DataFrame()


def clean_text(words: str) -> str:
    words = re.sub("[^a-zA-Z]", " ", words)
    text = words.lower().split()
    return " ".join(text)


def remove_numbers(text: str) -> str:
    new_text = []
    for word in text.split():
        if not re.search('\\d', word):
            new_text.append(word)
    return ' '.join(new_text)


def remove_stopwords(review: str) -> str:
    stop_words = stopwords.words('english')
    clothes = ['dress', 'color', 'wear', 'top', 'sweater', 'material', 'shirt',
               'jeans', 'pant', 'skirt', 'order', 'white', 'black', 'fabric',
               'blouse', 'sleeve', 'even', 'jacket']
    text = [word.lower() for word in review.split() if word.lower() not in
            stop_words and word.lower() not in clothes]
    return " ".join(text)


def get_lemmatize(text: str) -> str:
    lem = WordNetLemmatizer()
    lem_text = [lem.lemmatize(word) for word in text.split()]
    return " ".join(lem_text)


def preprocess_data(data: str) -> str:
    data['Review'] = data['Review'].astype(str)
    data['Review'] = data['Review'].apply(clean_text)
    data['Review'] = data['Review'].apply(remove_numbers)
    data['Review'] = data['Review'].apply(remove_stopwords)
    data['Review'] = data['Review'].apply(get_lemmatize)
    return data
```

Pre-processing

- read data
- clean text
- remove numbers
- remove stopwords
- get lemmatize

```
wordcloud = WordCloud(background_color="white", max_words=len(positive_words))

wordcloud.generate(positive_words)

plt.figure(figsize=(13,13))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

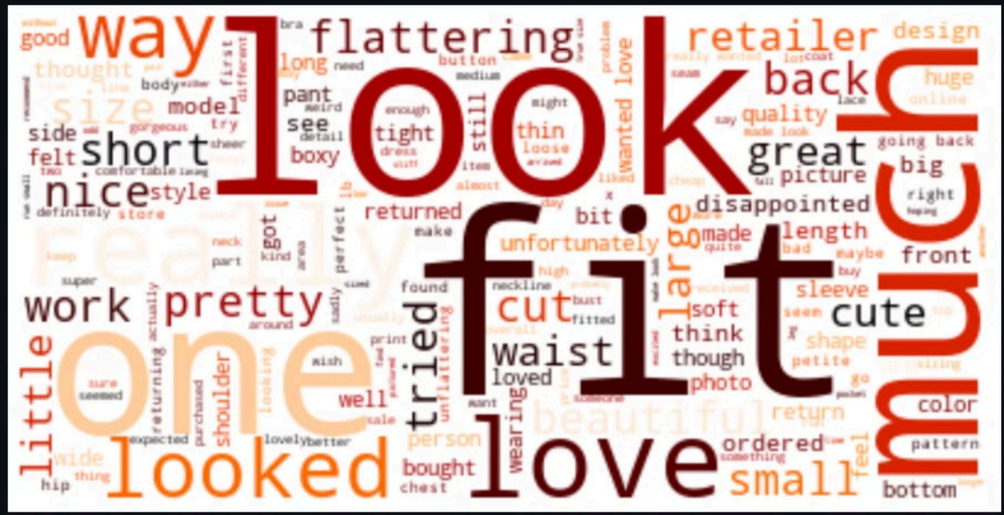
Positive Words

Sentiment Imbalance

1→19314

0→4172

```
wordcloud = WordCloud(background_color="white", max_words=len(negative_words), colormap='gist_heat')

wordcloud.generate(negative_words)

plt.figure(figsize=(13,13))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



Negative Words

Remove one half of + reviews

1→8000

0→4172

```python
from wtpsplit import SaT

sat_sm = SaT("sat-12l-sm")
sat_sm.half().to("cuda")
sat_sm.split("The material is soft the design is stylish. However, I feel it lacks durability for regular use.")
```

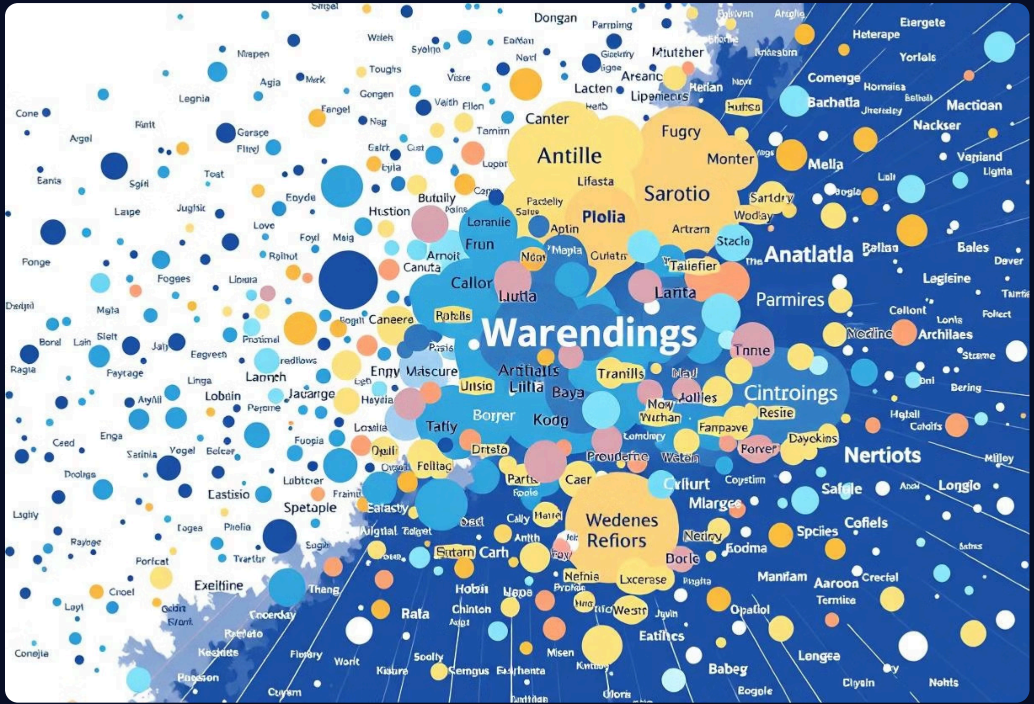
```
['The material is soft ',
 'the design is stylish. ',
 'However, I feel it lacks durability for regular use.']
```

Sentence Boundary Segmentation

Glove Embeddings

Glove Embeddings

GloVe embeddings are a popular technique for representing words as dense vectors. These vectors capture semantic relationships between words, meaning words with similar meanings will have vectors close to each other in vector space.



Build the model

We create a model using embedding layer and Bidirectional LSTM layers. Bidirectional LSTMs are supported in Keras via the E

```
model = Sequential([
    embedding_layer,
    Bidirectional(LSTM(embedding_dim, return_sequences=True)),
    Bidirectional(LSTM(embedding_dim)),
    Dense(6, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          850600

 bidirectional (Bidirection  (None, 100, 32)           14976
 al)

 bidirectional_1 (Bidirecti  (None, 32)                6272
 onal)

 dense (Dense)               (None, 6)                 198

 dense_1 (Dense)             (None, 1)                 7

=================================================================
Total params: 872053 (3.33 MB)
Trainable params: 21453 (83.80 KB)
Non-trainable params: 850600 (3.24 MB)
_________________________________________________________________
```

BiDirectional LSTM

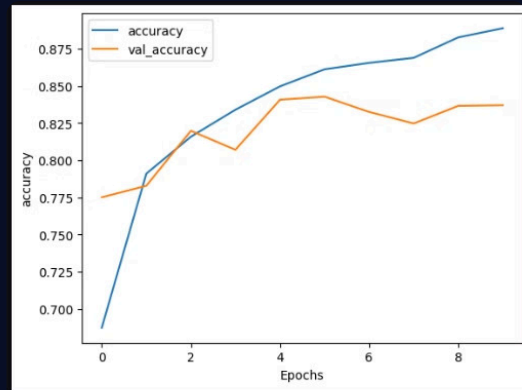
```
history = model.fit(X_train_pad, y_train,
                    batch_size=32,
                    epochs=10,
                    validation_data=(X_test_pad, y_test),
                    verbose=1)
```

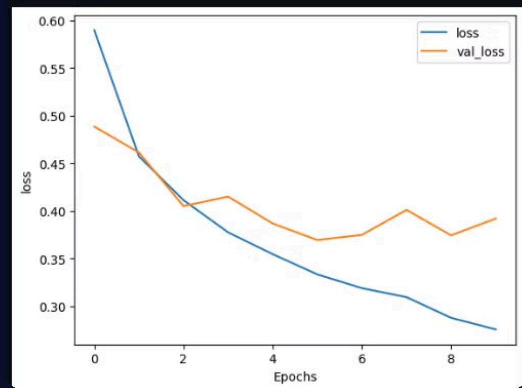
```
Epoch 1/10
305/305 [==============================] - 20s 50ms/step - loss: 0.5895 - accuracy: 0.6872 - val_loss: 0.4885 - val_accuracy: 0.7749
Epoch 2/10
305/305 [==============================] - 14s 47ms/step - loss: 0.4574 - accuracy: 0.7909 - val_loss: 0.4614 - val_accuracy: 0.7828
Epoch 3/10
305/305 [==============================] - 14s 46ms/step - loss: 0.4114 - accuracy: 0.8158 - val_loss: 0.4051 - val_accuracy: 0.8197
Epoch 4/10
305/305 [==============================] - 14s 46ms/step - loss: 0.3776 - accuracy: 0.8339 - val_loss: 0.4151 - val_accuracy: 0.8070
Epoch 5/10
305/305 [==============================] - 14s 46ms/step - loss: 0.3548 - accuracy: 0.8496 - val_loss: 0.3869 - val_accuracy: 0.8407
Epoch 6/10
305/305 [==============================] - 14s 46ms/step - loss: 0.3336 - accuracy: 0.8611 - val_loss: 0.3696 - val_accuracy: 0.8427
Epoch 7/10
305/305 [==============================] - 14s 47ms/step - loss: 0.3191 - accuracy: 0.8655 - val_loss: 0.3750 - val_accuracy: 0.8324
Epoch 8/10
305/305 [==============================] - 14s 47ms/step - loss: 0.3097 - accuracy: 0.8689 - val_loss: 0.4012 - val_accuracy: 0.8246
Epoch 9/10
305/305 [==============================] - 16s 52ms/step - loss: 0.2880 - accuracy: 0.8826 - val_loss: 0.3745 - val_accuracy: 0.8366
Epoch 10/10
305/305 [==============================] - 14s 47ms/step - loss: 0.2759 - accuracy: 0.8888 - val_loss: 0.3920 - val_accuracy: 0.8370
```

Predictions on a test set:

```
loss, accuracy = model.evaluate(X_test_pad,y_test)
print('Test accuracy :', accuracy)
```

```
77/77 [==============================] - 1s 13ms/step - loss: 0.3920 - accuracy: 0.8370
Test accuracy : 0.8369609713554382
```





Model 2

We also test a second model with a Dropout layer for decrease overfitting and we increase number of epochs to 20.

```
model2 = Sequential([
    embedding_layer,
    Bidirectional(LSTM(64, return_sequences=True)),
    Bidirectional(LSTM(32)),
    Dense(16, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

optimizer = tf.keras.optimizers.Adam(lr=1e-4)
model2.compile(loss='binary_crossentropy',
               optimizer=optimizer,
               metrics=['accuracy'])

model2.summary()
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          850600

 bidirectional_4 (Bidirecti  (None, 100, 128)          84480
 onal)

 bidirectional_5 (Bidirecti  (None, 64)                41216
 onal)

 dense_4 (Dense)             (None, 16)                1040

 dropout_1 (Dropout)         (None, 16)                0

 dense_5 (Dense)             (None, 1)                 17

=================================================================
Total params: 977353 (3.73 MB)
Trainable params: 126753 (495.13 KB)
Non-trainable params: 850600 (3.24 MB)
_________________________________________________________________
```

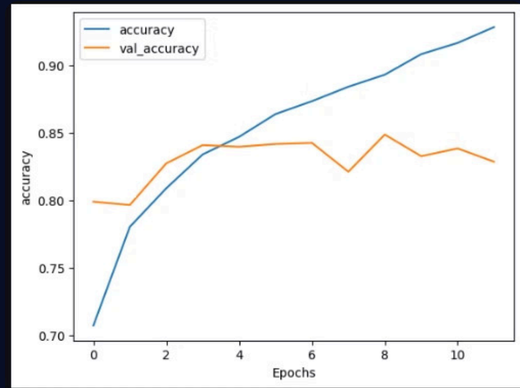
With Dropout

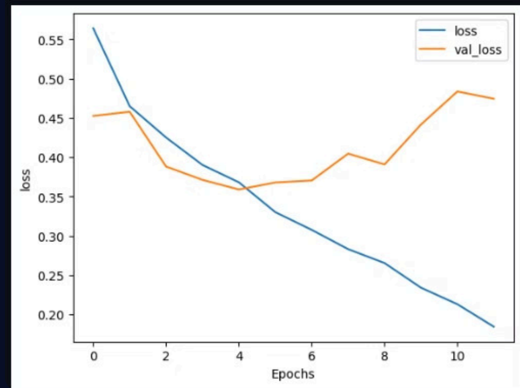
```
history2 = model2.fit(X_train_pad, y_train,
                      batch_size=32,
                      epochs=20,
                      validation_data=(X_test_pad, y_test),
                      verbose=1,
                      callbacks=callbacks)
```

```
Epoch 1/20
305/305 [==============================] - 26s 67ms/step - loss: 0.5641 - accuracy: 0.7071 - val_loss: 0.4525 - val_accuracy: 0.7988
Epoch 2/20
305/305 [==============================] - 19s 62ms/step - loss: 0.4650 - accuracy: 0.7801 - val_loss: 0.4579 - val_accuracy: 0.7963
Epoch 3/20
305/305 [==============================] - 19s 61ms/step - loss: 0.4251 - accuracy: 0.8087 - val_loss: 0.3882 - val_accuracy: 0.8271
Epoch 4/20
305/305 [==============================] - 18s 60ms/step - loss: 0.3902 - accuracy: 0.8337 - val_loss: 0.3712 - val_accuracy: 0.8407
Epoch 5/20
305/305 [==============================] - 18s 61ms/step - loss: 0.3680 - accuracy: 0.8468 - val_loss: 0.3590 - val_accuracy: 0.8394
Epoch 6/20
305/305 [==============================] - 19s 61ms/step - loss: 0.3301 - accuracy: 0.8636 - val_loss: 0.3679 - val_accuracy: 0.8415
Epoch 7/20
305/305 [==============================] - 19s 61ms/step - loss: 0.3076 - accuracy: 0.8732 - val_loss: 0.3704 - val_accuracy: 0.8423
Epoch 8/20
305/305 [==============================] - 18s 60ms/step - loss: 0.2830 - accuracy: 0.8838 - val_loss: 0.4046 - val_accuracy: 0.8209
Epoch 9/20
305/305 [==============================] - 18s 60ms/step - loss: 0.2653 - accuracy: 0.8928 - val_loss: 0.3908 - val_accuracy: 0.8485
Epoch 10/20
305/305 [==============================] - 19s 61ms/step - loss: 0.2340 - accuracy: 0.9080 - val_loss: 0.4416 - val_accuracy: 0.8324
Epoch 11/20
305/305 [==============================] - 19s 61ms/step - loss: 0.2130 - accuracy: 0.9163 - val_loss: 0.4837 - val_accuracy: 0.8382
Epoch 12/20
305/305 [==============================] - 19s 61ms/step - loss: 0.1844 - accuracy: 0.9280 - val_loss: 0.4744 - val_accuracy: 0.8283
Epoch 12: early stopping
```

```
loss, accuracy = model2.evaluate(X_test_pad,y_test)
print('Test accuracy :', accuracy)
```

```
77/77 [==============================] - 2s 21ms/step - loss: 0.4744 - accuracy: 0.8283
Test accuracy : 0.828336775302887
```





Aspect Classification

Goal

- Categorize sentences into predefined aspects (e.g., Fit, Fabric).

Approach

- Generate a synthetic dataset using LLM.
- Train a transformer for doing aspect classification
- Handles multiple aspects in a single review.

Challenges

- Ambiguity in aspect relevance.
- Filtering irrelevant sentences.

```
import ollama
import pandas as pd

aspects = ["Fit", "Fabric", "Color", "Design", "Durability", "Price", "Comfort", "Category", "None of these"]
sentiments = ["Positive", "Negative", "Mixed"]

Qodo Gen: Options | Test this function
def generate_review(model_name, prompt):

    try:
        response = ollama.chat(model=model_name, messages=[{'role': 'user', 'content': prompt}])
        return response['message']['content'].strip()
    except Exception as e:
        print(f"Error for prompt '{prompt}': {e}")
        return None

rows = []
model_name = "llama3.1:8b"

for aspect in aspects:
        for _ in range(500):

            prompt = f"Write a small review whether positive, negative or mixed of a piece of cloth u bought (could be anything shirt, trouser etc) focusing on the aspect: {aspect}/ write at most 2 sentences and dont write anything except review, use casual and
            simple english. Make it as unique as possible. dont use punctuations"
            print(f"\nGenerating review for Aspect: {aspect}")

            response = generate_review(model_name, prompt)


            if response:
                print(f"Response: {response}\n")
                rows.append({"Review": response, "Aspect": aspect})

df = pd.DataFrame(rows)
df.to_csv("synthetic_reviews_dataset2.csv", index=False)
print("\nDataset saved to synthetic_reviews_dataset.csv!")
```

Use llama3:8b For Data Generation

"I'm loving this new shirt I got but the fit is a bit off - its too loose in the sleeves and too tight around the waist giving me an awkward silhouette

or

I was hyped about my new joggers but they're totally off the mark with the fit - way too baggy and sloppy for my liking",Fit

```
import pandas as pd
import re


file_path = "synthetic_reviews_dataset2.csv"
df = pd.read_csv(file_path)
Qodo Gen: Options | Test this function


def clean_reviews(review):

    if re.search(r"\s[Oo][rR]\s", review):

        parts = re.split(r"\s[Oo][rR]\s", review, maxsplit=1)
        return parts[0].strip()
    return review


df["Review"] = df["Review"].apply(clean_reviews)

output_file = "cleaned_dataset2.csv"
df.to_csv(output_file, index=False)

print(f"Cleaned dataset saved to {output_file}!")
```

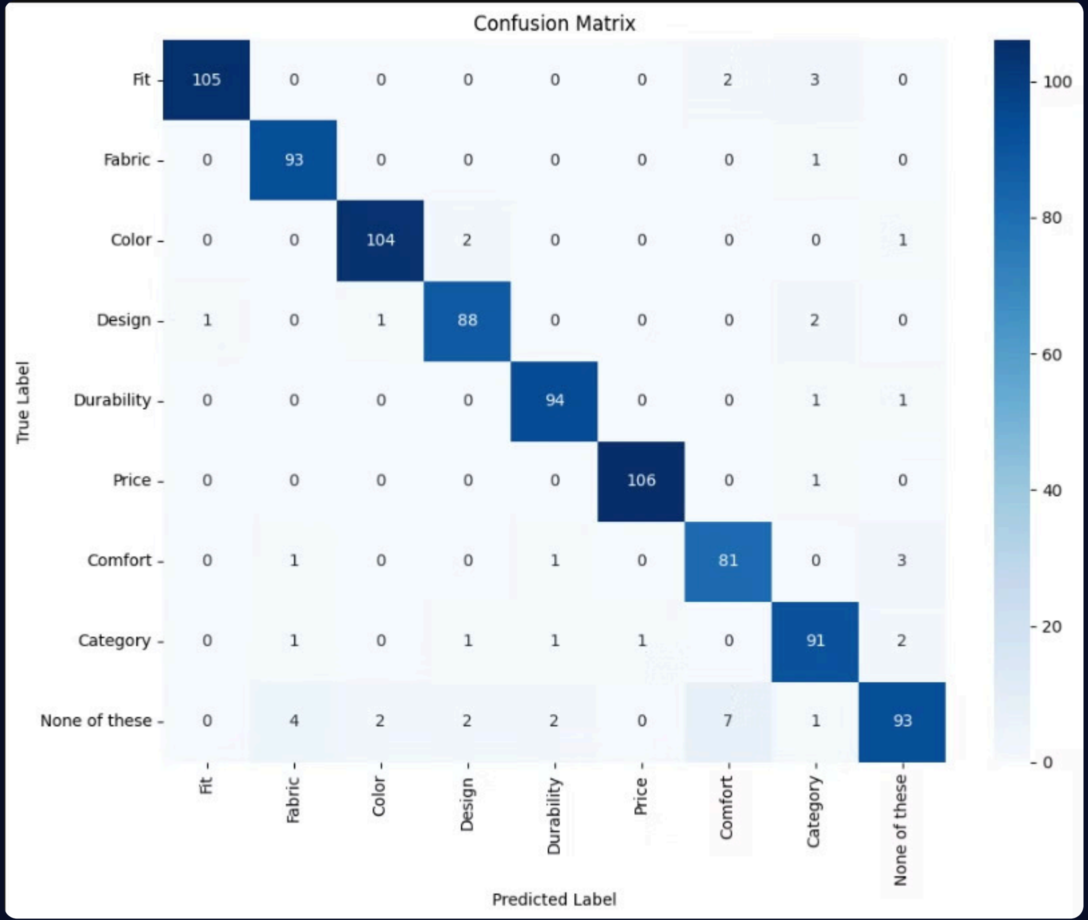
```
Cleaned dataset saved to cleaned_dataset2.csv!
```

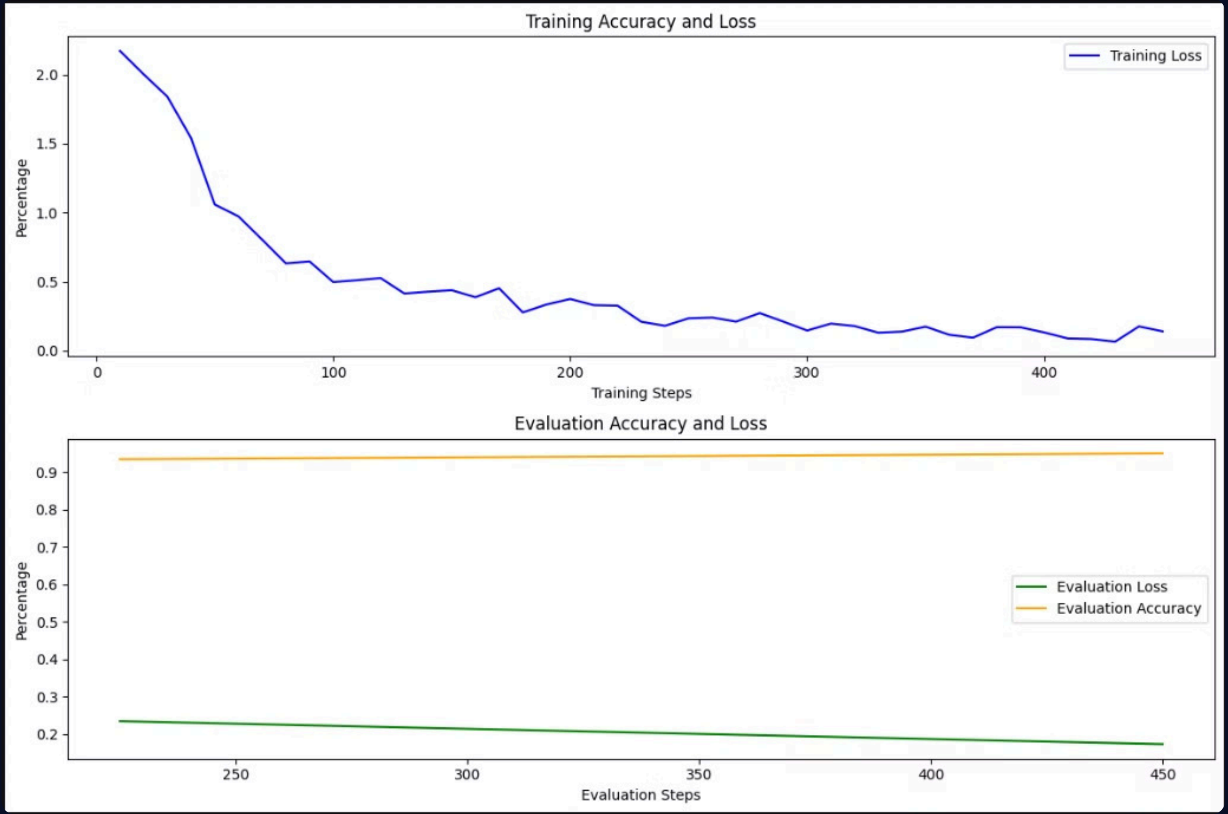
```python
from torch.utils.data import Dataset

class AspectDataset(Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        return {
            "input_ids": self.encodings["input_ids"][idx],
            "attention_mask": self.encodings["attention_mask"][idx],
            "labels": self.labels[idx]
        }


train_dataset = AspectDataset(train_encodings, train_labels)
test_dataset = AspectDataset(test_encodings, test_labels)
```

```python
from transformers import BertForSequenceClassification


num_labels = len(label_mapping)
model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels=num_labels
)
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

BERT (Pre Trained)


Confusion Matrix
Fit 105 0 0 0 0 0 2 3 0
Fabric 0 93 0 0 0 0 0 1 0
Color 0 0 104 2 0 0 0 0 1
Design 1 0 1 88 0 0 0 2 0
Durability 0 0 0 0 94 0 0 1 1
Price 0 0 0 0 0 106 0 1 0
Comfort 0 1 0 0 1 0 81 0 3
Category 0 1 0 1 1 1 0 91 2
None of these 0 4 2 2 2 0 7 1 93
True Label
Fit
Fabric
Color
Design
Durability
Price
Comfort
Category
None of these
Predicted Label
100
80
60
40
20
0


Training Accuracy and Loss
Training Loss
Percentage
Training Steps
Evaluation Accuracy and Loss
Evaluation Loss
Evaluation Accuracy
Percentage
Evaluation Steps

Recommendation Logic

- Personalization:
- Users specify preferences for aspects (e.g., prioritize Fit).
- Products scored based on aspect alignment.
- Computation:
- Weighted dot product for similarity scoring.
- Top n recommendations sorted by score.

Deployment

- Platform: Flask for backend APIs.
- Frontend:
- HTML/CSS/JavaScript for dynamic product displays.
- Database: SQLite for managing users, products, and reviews.

Results and Insights

- System Performance:
- Accuracy: ~85% for sentiment classification.
- Real-time updates for new reviews.
- Impact:
- Improved recommendation relevance.
- Enhanced user satisfaction through personalization.

Challenges and Limitations

- Data Challenges:
- Ambiguity in reviews (e.g., sarcastic comments).
- Imbalanced datasets affecting model performance.
- Real-time Reviews:
- Real-time reviews can be challenging as the data for aspect classification is synthetic and cannot accurately capture reality.

Conclusion

Summary:

ABSA recommender system bridges customer feedback and personalized shopping experiences.

Impact:

Drives user engagement and business outcomes.