NLP Analysis of Meesho Product Reviews

Group 47

Team Members:
Rohit Raju Kamble (2411AI61)
Satyam Kumar (2411AI44)
Abhisar Anand (2411AI43)

Guided by:
Dr. Sourav Kumar Dandapat

April 21, 2025

Agenda

Problem Statement

- Analysis of Meesho product reviews using NLP techniques
- Objectives:
  - Perform sentiment analysis on customer reviews
  - Extract key aspects and opinions
  - Build classification models to predict sentiment
  - Visualize insights from the data
- Dataset: 24,999 product reviews from Meesho

Data Collection

- Scraped reviews using Apify client
- Combined multiple CSV files into one dataset
- Initial dataset columns:

```
Index(['productUrl', 'review/author/name',
'review/comments', 'review/product_name',
'review/rating', 'scrapedAt'], dtype='object')
```

Text Cleaning Pipeline

```python
def clean_text(text):
    text = text.lower()
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)
    text = re.sub(r'\\@\\w+|\\#', '', text)
    text = re.sub(r'<.*?>', '', text)
    text = re.sub(r'[^\\w\\s]', '', text)
    text = re.sub(r'\\d+', '', text)
    text = re.sub(r'\\s+', ' ', text).strip()
    return text
```

Additional steps:

- Tokenization and lemmatization
- Stopword removal
- Spelling correction

N-gram Extraction

```python
def get_top_ngrams(text_series, n=2, top_n=10):
    vec = CountVectorizer(ngram_range=(n, n)).fit(text_series)
    bag_of_words = vec.transform(text_series)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx])
                  for word, idx in vec.vocabulary_.items()]
    return sorted(words_freq, key=lambda x: x[1],
                  reverse=True)[:top_n]
```

Top N-grams

Top Bigrams:

- good price (795)
- low quality (769)
- suit need (757)
- poor quality (568)
- daily use (566)

Top Trigrams:

- fell apart cheap (501)
- highly functional great (341)
- excellent daily use (317)
- daily use love (317)
- poor performance worth (310)

Top Product Aspects

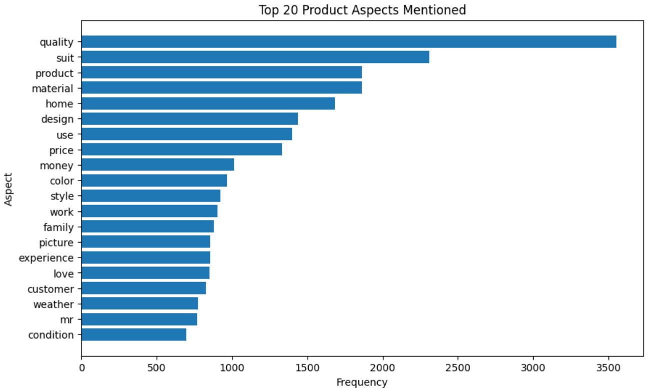

Figure: Top Product Aspects

Word Cloud

Figure: Word cloud of most frequent terms in reviews

Feature Extraction Methods

```
# Bag of Words
bow_vectorizer = CountVectorizer(max_features=5000)
X_train_bow = bow_vectorizer.fit_transform(X_train)

# TF-IDF
tfidf_vectorizer = TfidfVectorizer(
    max_features=5000,
    ngram_range=(1, 2))
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

Model Implementation

```python
def train_and_evaluate(model, X_train, X_test,
                       y_train, y_test, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    print(f"Accuracy: {accuracy_score(y_test, y_pred)
    :.4f}")
    return model

# Example: SVM with TF-IDF
svm_tfidf = SVC(kernel='linear',
                probability=True,
                random_state=42)
train_and_evaluate(svm_tfidf, X_train_tfidf,
                   X_test_tfidf, y_train, y_test,
                   "SVM with TF-IDF")
```

Model Performance Summary

Model	Features	Accuracy
Naive Bayes	BoW	0.6956
Naive Bayes	TF-IDF	0.6976
Random Forest	TF-IDF	0.6956
XGBoost	TF-IDF	0.6956
SVM	TF-IDF	0.7016

Detailed Model Performance

Best Model: SVM with TF-IDF

- Precision: 0.74 (class 0), 0.66 (class 1)
- Recall: 0.73 (class 0), 0.66 (class 1)
- F1-score: 0.73 (class 0), 0.66 (class 1)
- Accuracy: 0.7016

Model Parameters

- Kernel: linear
- Probability: True
- Random state: 42
- Features: TF-IDF (5000)
- N-gram range: (1,2)

Classification Report

```
SVM with TF-IDF Performance:
              precision    recall  f1-score   support

           0       0.74      0.73      0.73      2817
           1       0.66      0.66      0.66      2183

    accuracy                           0.70      5000
   macro avg       0.70      0.70      0.70      5000
weighted avg       0.70      0.70      0.70      5000
```

Key Findings

- SVM with TF-IDF performed best (70.16% accuracy)
- All models showed similar performance (70% accuracy)
- Key phrases reveal product quality and value concerns
- "Good price" and "low quality" most common bigrams
- Model performance consistent across metrics

Future Work

- Analyze temporal trends in reviews
- Build recommendation system based on aspects
- Hyperparameter tuning for better performance

Contributions

- Abhisar - Automatic Data Collection
- Satyam - Text prepossessing and visualization
- Rohit - feature engineering and Model Building

Thank You!