4-Topic Modelling

November 15, 2024

1 Sentiment Analysis and Topic Modeling

In this section, we conduct sentiment analysis and topic modeling on Yelp reviews using VADER and LDA. This analysis helps identify major themes in the reviews and assess how customers feel about various aspects of their experience.

1.1 Goals

- Use VADER to determine the sentiment score for each review, focusing on service, food, and ambiance.
- Apply Latent Dirichlet Allocation (LDA) to uncover key topics discussed in customer reviews.

1.2 Steps

1. **Sentiment Analysis**: Utilize VADER to classify review sentiments into positive, negative, or neutral categories.
2. **Topic Modeling with LDA**: Identify common themes in reviews and associate them with sentiment scores.
3. **Topic Interpretation**: Summarize the key themes and explore how they relate to customer satisfaction.

[ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import pandas as pd

filtered_reviews = pd.read_csv('clean_filtered_reviews.csv')

filtered_reviews = filtered_reviews.dropna(subset=['all_text'])

# Use TF-IDF Vectorizer with bigrams for richer context
tfidf_vectorizer = TfidfVectorizer(max_df=0.9, min_df=2, stop_words='english',␣
  ↪ngram_range=(1, 2))
tfidf_matrix = tfidf_vectorizer.fit_transform(filtered_reviews['all_text'])

# Display feature names to verify the refined vocabulary
print("Sample feature names after TF-IDF vectorization:")
print(tfidf_vectorizer.get_feature_names_out()[:10])
```

```
# Fit LDA model
num_topics = 5
lda = LatentDirichletAllocation(n_components=num_topics, random_state=42)
lda.fit(tfidf_matrix)

# Display topics
def display_topics(model, feature_names, num_top_words):
    for idx, topic in enumerate(model.components_):
        print(f"Topic {idx + 1}:")
        print([feature_names[i] for i in topic.argsort()[-num_top_words:]])

# Display top 10 words for each topic
display_topics(lda, tfidf_vectorizer.get_feature_names_out(), 10)
```

[ ]:
```
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk

# Download VADER lexicon if not already done
#nltk.download('vader_lexicon')

# Instantiate Sentiment Intensity Analyzer
sia = SentimentIntensityAnalyzer()

# Function to calculate sentiment scores using VADER
def calculate_sentiment(text):
    score = sia.polarity_scores(text)
    return score['compound']  # Return the compound score which reflects the␣
 ↪overall sentiment

# Apply the function to calculate sentiment score for each review
filtered_reviews['sentiment_score'] = filtered_reviews['all_text'].
 ↪apply(calculate_sentiment)

# Verify that the sentiment scores have been added
print("Sample of sentiment scores added to DataFrame:")
print(filtered_reviews[['all_text', 'sentiment_score']].head())
```

[ ]:
```
# Assuming you have sentiment scores already computed for each review
for idx, topic in enumerate(lda.components_):
    # Identify relevant reviews based on words in the topic
    relevant_words = [tfidf_vectorizer.get_feature_names_out()[i] for i in␣
 ↪topic.argsort()[-10:]]
    relevant_reviews = filtered_reviews[filtered_reviews['all_text'].
 ↪apply(lambda x: any(word in x for word in relevant_words))]

    # Calculate the average sentiment score for the relevant reviews
    avg_sentiment = relevant_reviews['sentiment_score'].mean()
```

```python
    print(f"Average Sentiment for Topic {idx + 1}: {avg_sentiment}")
```