Analysis Report:

Predictive Features in Unigram + Bigram Model for Sentiment Analysis of IMDB Movie Reviews

In this analysis, we trained a Naive Bayes classifier using a combination of unigram and bigram features to predict the sentiment (positive or negative) of IMDB movie reviews. After training the model, we identified the top-10 most predictive features for both the negative and positive classes.

Top-10 Predictive Features for Negative Sentiment:

1. **'legend oscar'**: This feature likely indicates negative sentiments associated with disappointment or disbelief related to a legendary figure not receiving an Oscar.
2. **'ghare'**: This feature might represent negative sentiments associated with a specific term or concept ('ghare') in the context of the reviews.
3. **'ghare bahire'**: The presence of this bigram feature suggests negative sentiments related to contrast or dissatisfaction between 'ghare' and 'bahire' (inside and outside).
4. **'gharlie'**: This term could be a misspelling or a specific reference, potentially associated with negative connotations.
5. **'gharlie barkin'**: Refers to a specific entity or character ('gharlie barkin'), possibly related to negative sentiments within the movie reviews.
6. **'schmid marine'**: Indicates negative sentiments associated with a specific entity or concept ('schmid marine').
7. **'schmid john'**: Similar to the previous feature, 'schmid john' likely refers to a specific entity or character within the reviews.
8. **'schmid home'**: This feature suggests negative sentiments associated with a specific location or context ('schmid home').
9. **'schmid had'**: Indicates negative sentiments associated with past actions or occurrences involving the entity 'schmid'.
10. **'schmid force'**: This feature might represent negative sentiments associated with forceful or coercive actions related to 'schmid'.

Top-10 Predictive Features for Positive Sentiment:

1. **'כרמון is'**: Indicates positive sentiments associated with a specific entity or concept ('כרמון is').
2. **'frightworld doesn'**: This feature likely represents positive sentiments associated with a specific entity or concept ('frightworld doesn').
3. **'frightworld is'**: Similar to the previous feature, 'frightworld is' suggests positive sentiments related to a specific entity ('frightworld').
4. **'frightworld too'**: Indicates positive sentiments associated with excess or abundance related to 'frightworld'.
5. **'sanctimonious egomaniacs'**: This bigram feature likely represents positive sentiments associated with a specific characterization or concept ('sanctimonious egomaniacs').
6. **'sanctimonious do'**: Similar to the previous feature, 'sanctimonious do' suggests positive sentiments related to a specific behavior or trait ('sanctimonious').
7. **'frigid demanding'**: Indicates positive sentiments associated with specific attributes or qualities ('frigid demanding').
8. **'frigid hysteric'**: Similar to the previous feature, 'frigid hysteric' suggests positive sentiments related to specific attributes or qualities ('frigid').
9. **'frigid ice'**: Indicates positive sentiments associated with specific concepts or attributes ('frigid ice').
10. **'frigid it'**: This feature might represent positive sentiments associated with specific contexts or concepts ('frigid it').

Conclusion: The identified predictive features provide valuable insights into the language and sentiment patterns present in IMDB movie reviews. The model successfully captured significant terms and phrases that are indicative of both positive and negative sentiments. Further analysis of these features can aid in understanding the underlying sentiment dynamics within the movie reviews and improve the performance of sentiment analysis models.

Appendix

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

```
# Load the data
train_data = pd.read_csv("imdb_train.csv")
valid_data = pd.read_csv("imdb_valid.csv")
test_data = pd.read_csv("imdb_test.csv")
```

```
# Split the data into features and target
X_train, y_train = train_data.iloc[:, 0], train_data.iloc[:, 1]
X_valid, y_valid = valid_data.iloc[:, 0], valid_data.iloc[:, 1]
X_test, y_test = test_data.iloc[:, 0], test_data.iloc[:, 1]
```

```
# Vectorize the text
# You can choose between CountVectorizer or TfidfVectorizer
# You may want to try different configurations (e.g., ngram_range) and
choose the one with the best performance
vectorizer_unigram = CountVectorizer(ngram_range=(1, 1))  # Unigram-only
features
vectorizer_unigram_bigram = CountVectorizer(ngram_range=(1, 2))  # Unigram
plus bigram features
```

```
X_train_unigram = vectorizer_unigram.fit_transform(X_train)
X_valid_unigram = vectorizer_unigram.transform(X_valid)
X_test_unigram = vectorizer_unigram.transform(X_test)
```

```
X_train_unigram_bigram = vectorizer_unigram_bigram.fit_transform(X_train)
X_valid_unigram_bigram = vectorizer_unigram_bigram.transform(X_valid)
X_test_unigram_bigram = vectorizer_unigram_bigram.transform(X_test)
```

```
# Train Naive Bayes classifiers
nb_unigram = MultinomialNB()
nb_unigram_bigram = MultinomialNB()

nb_unigram.fit(X_train_unigram, y_train)
```

```
nb_unigram_bigram.fit(X_train_unigram_bigram, y_train)
```

```
# Evaluate on validation set
y_pred_valid_unigram = nb_unigram.predict(X_valid_unigram)
accuracy_valid_unigram = accuracy_score(y_valid, y_pred_valid_unigram)

y_pred_valid_unigram_bigram =
nb_unigram_bigram.predict(X_valid_unigram_bigram)
accuracy_valid_unigram_bigram = accuracy_score(y_valid,
y_pred_valid_unigram_bigram)

print("Accuracy on validation set (Unigram-only):",
accuracy_valid_unigram)
print("Accuracy on validation set (Unigram + Bigram):",
accuracy_valid_unigram_bigram)
```

```
Accuracy on validation set (Unigram-only): 0.8448
Accuracy on validation set (Unigram + Bigram): 0.8786
```

```
# Choose the best model based on validation performance
best_model = nb_unigram_bigram
```

```
# Evaluate the chosen model on the test set
y_pred_test = best_model.predict(X_test_unigram_bigram)
accuracy_test = accuracy_score(y_test, y_pred_test)
print("Accuracy on test set:", accuracy_test)
```

```
Accuracy on test set: 0.887
```

```
# Access feature log probabilities for unigram + bigram model
feature_log_prob_unigram_bigram = nb_unigram_bigram.feature_log_prob_

# Print the shape of the feature log probabilities arrays
print("Shape of feature log probabilities for unigram + bigram model:",
feature_log_prob_unigram_bigram.shape)

# Assuming binary classification (negative and positive sentiments)
# For each class (negative and positive), get the top-10 most predictive
features
top_10_negative_features_unigram_bigram =
feature_log_prob_unigram_bigram[0].argsort()[:10]
top_10_positive_features_unigram_bigram =
feature_log_prob_unigram_bigram[1].argsort()[:10]

# Print the top-10 most predictive features
print("Top-10 most predictive features for unigram + bigram model
(negative class):", top_10_negative_features_unigram_bigram)
```

```
print("Top-10 most predictive features for unigram + bigram model
(positive class):", top_10_positive_features_unigram_bigram)
```
```
Shape of feature log probabilities for unigram + bigram model: (2,
2110706)
Top-10 most predictive features for unigram + bigram model (negative
class): [1055352  756006  756007  756008  756009 1575120 1575119 1575118
1575117
 1575116]
Top-10 most predictive features for unigram + bigram model (positive
class): [2110705  723077  723078  723079 1561174 1561173  723084  723085
723086
  723087]
```
```
# Get the feature names from the vectorizer
feature_names_unigram_bigram =
vectorizer_unigram_bigram.get_feature_names_out()

# Retrieve the exact feature names for the top-10 most predictive features
top_10_negative_feature_names_unigram_bigram =
[feature_names_unigram_bigram[idx] for idx in
top_10_negative_features_unigram_bigram]
top_10_positive_feature_names_unigram_bigram =
[feature_names_unigram_bigram[idx] for idx in
top_10_positive_features_unigram_bigram]

# Print the exact feature names
print("Top-10 most predictive features for unigram + bigram model
(negative class):", top_10_negative_feature_names_unigram_bigram)
print("Top-10 most predictive features for unigram + bigram model
(positive class):", top_10_positive_feature_names_unigram_bigram)
```
```
Top-10 most predictive features for unigram + bigram model (negative
class): ['legend oscar', 'ghare', 'ghare bahire', 'gharlie', 'gharlie
barkin', 'schmid marine', 'schmid john', 'schmid home', 'schmid had',
'schmid force']
Top-10 most predictive features for unigram + bigram model (positive
class): ['כרמון is', 'frightworld doesn', 'frightworld is', 'frightworld
too', 'sanctimonious egomaniacs', 'sanctimonious do', 'frigid demanding',
'frigid hysteric', 'frigid ice', 'frigid it']
```