SENTIMENT-AWARE LOCATION RECOMMENDATION USING MACHINE LEARNING

Nguyen Duc Thanh Vinh

1. Details of the project and baseline implementations

Details of project: Observing that LLM currently only searches websites and returns location links, I would like to improve this by incorporating a sentiment analysis model to automatically evaluate the number of positive and negative reviews. Locations with a lot of positive reviews will be prioritized because they are considered trustworthy and high quality.

Model: Logistic Regression (trained on hotel review dataset), Random Forest, Support Vector Machines (SVM), and XGboost. All models utilize TF-IDF as a word embedding method.

Dataset: https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews/data

Baseline Implementations: To establish a foundation for model evaluation, a baseline Logistic Regression model was implemented using default parameters, trained on a slightly imbalanced dataset containing 20,491 user feedback samples (15,093 positive and 5,398 negative). The model achieved 88.88% accuracy, 89% precision, 96% recall, 93% F1-score, and 82% AUC-ROC.

At this baseline stage:

- Class imbalance was not addressed
- No dimensionality reduction or feature selection techniques (e.g., min_df, max_df) were applied
- Models were trained with default settings (no hyperparameter adjustments)

Precision was especially emphasized due to the importance of minimizing false positives when recommending potentially low-quality locations.

2. Show the details of the dataset used

This study utilizes the TripAdvisor Hotel Reviews dataset, sourced from Kaggle, comprising 20,491 user reviews scraped from the TripAdvisor platform. Each review includes a star rating from 1 to 5. For the purpose of binary sentiment analysis, ratings of 1, 2, and 3 were grouped as negative, while ratings of 4 and 5 were considered positive.

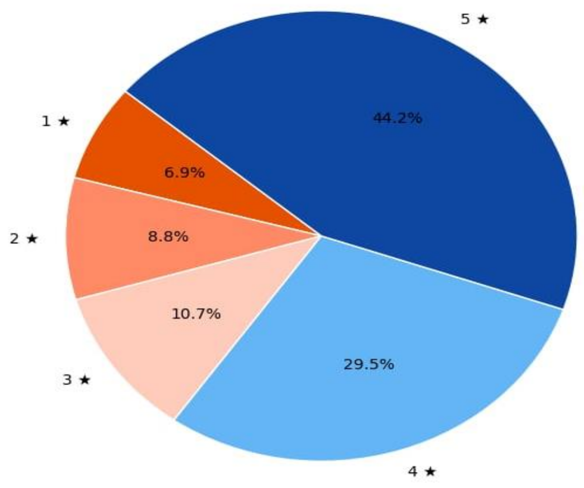

Figure 1. Distribution of ratings

Following this transformation, the dataset exhibited a moderate class imbalance: approximately 5,398 negative and 15,093 positive reviews—a ratio of about 1:3, as illustrated in the pie chart above. This imbalance presents a challenge for classification models, which may become biased toward the majority class. Without applying class weighting or using a weighted loss function, models are prone to defaulting to positive predictions, resulting in deceptively high accuracy while failing to capture negative sentiment—an issue that is particularly detrimental in recommendation systems where detecting low-quality experiences is critical. To address this, class weighting was incorporated in later stages of model optimization.

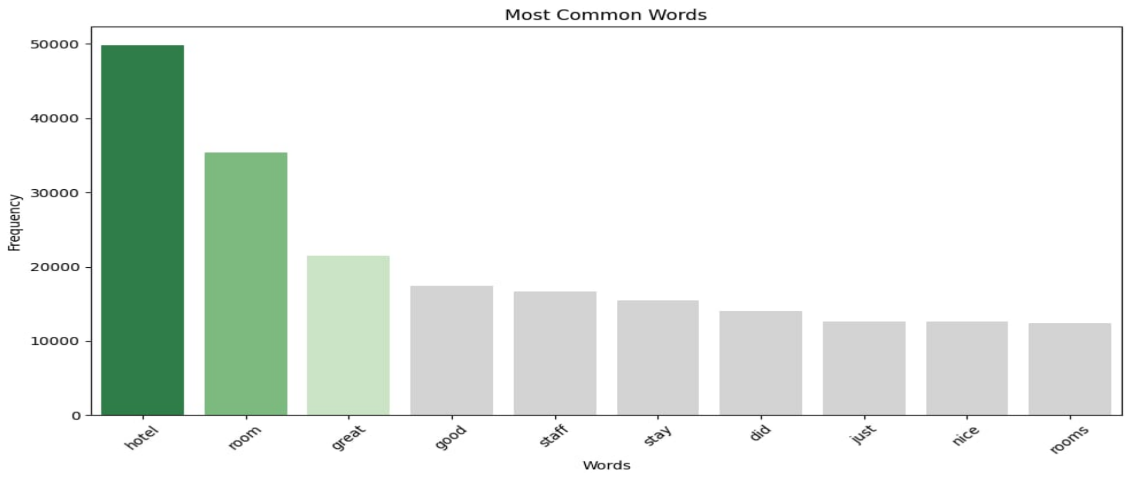

Figure 2. Most common words in dataset

An initial exploratory data analysis was conducted to understand the review content better. As shown in the bar chart of most frequent words (Figure 2), terms such as *hotel*, *room*, and *great* appeared most often, indicating that user sentiment heavily revolves around these core aspects of hotel service.

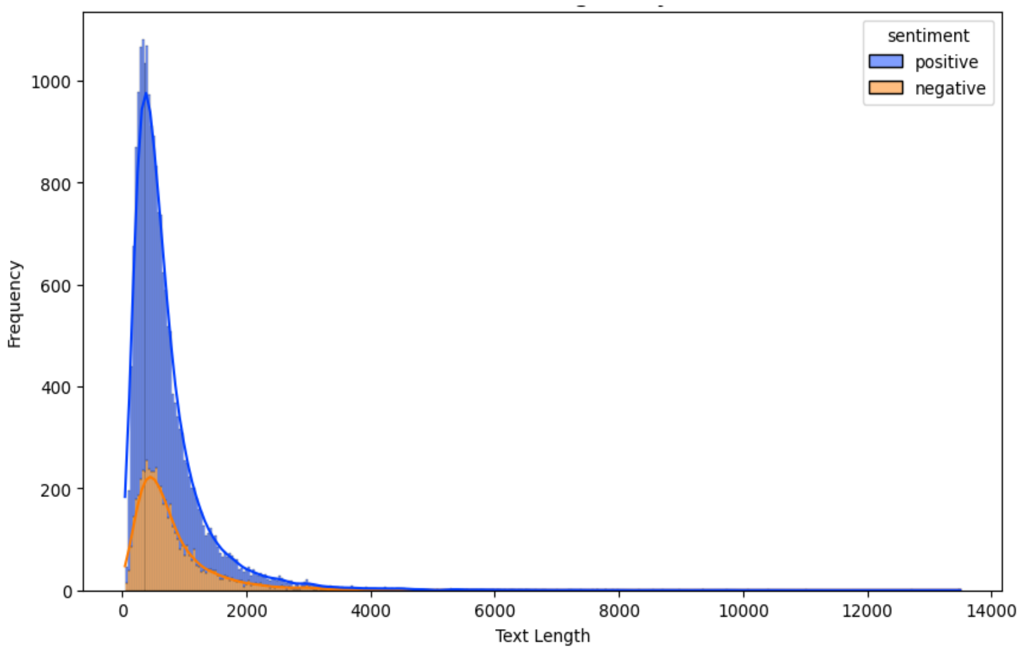

Figure 3. Distribution of text lengths by sentiment

Moreover, the analysis of text length distribution by sentiment (Figure 3) revealed a notable trend: negative reviews tend to be shorter and more emotionally charged, whereas positive reviews are generally longer and more descriptive. This suggests that dissatisfied users often leave brief, direct complaints, while satisfied customers are more inclined to provide detailed feedback.

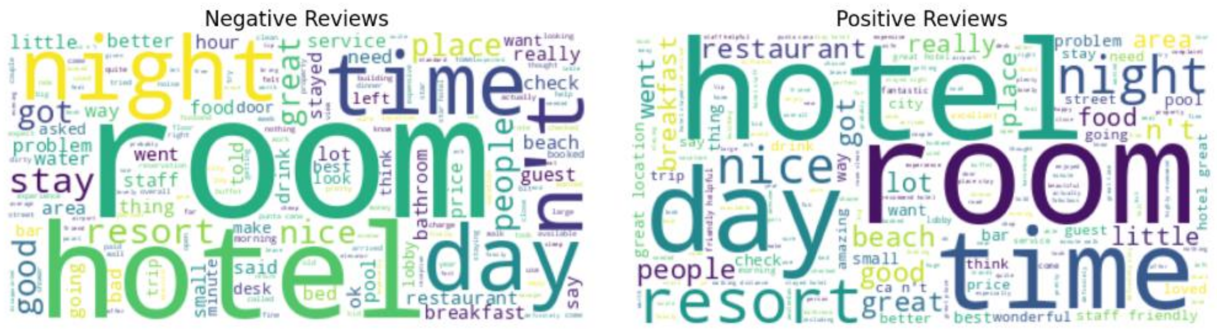

Figure 4. Word clouds of negative and positive reviews.

Additionally, word clouds were generated for both sentiment classes to visualize dominant themes (Figures A and B). In positive reviews, users frequently mentioned facilities, location, and service quality. Conversely, negative reviews often focused on issues such as staff behavior, room conditions, and overall service dissatisfaction. These patterns not only guided the feature engineering process but also shaped the evaluation strategy, particularly in emphasizing precision, which aimed to minimize false positives in downstream recommendation systems.

3. Detail of evaluation criteria

Evaluate Criteria: If we only rely on the ratio of positive feedback to the total number of feedback to evaluate the quality of a hotel, we are likely to encounter bias in the evaluation. For example:

- Hotel A: only 2 reviews, both positive → 100% positive, but the data is too small to be reliable.
- Hotel B has 500 reviews, of which 85% are positive. Although the ratio is lower, the large number of reviews makes the reliability much higher.

Solution: Bayesian Average Rating

Bayesian Average Rating is a method that helps adjust the rating score to avoid objects with little data but unreasonably high scores.

$$Weighted\ Score = \frac{v}{v+m} \cdot R + \frac{m}{v+m} \cdot C$$

With:

v: the number of feedback for the item

R: average rating of the item

m: minimum number of reviews required for trustworthiness

C: mean rating across all items in the system

4. Results of your algorithm and comparison with baselines

To evaluate the performance of various algorithms for sentiment classification of feedback (positive vs. negative), we compared different model variants against their baseline implementations. Our baseline models used default settings without addressing class imbalance or feature reduction. The following table summarizes the performance:

Table 1. Model performance comparison using various preprocessing and balancing techniques.

Model Variant	Accuracy	Precision	Recall	F1_score	AUC_ROC
Logistic Regression (baseline)	88.88%	89%	96%	93%	82%
Logistic Regression (balanced, TF-IDF)	87.34%	94%	88%	91%	86%
XGBoost (chi2 k=500 + TF-IDF reduce)	85.66%	86%	97%	91%	76%
Random Forest (balanced chi2 k=500 TF-IDF reduce)	85.66	86%	97%	91%	76%
SVC (class_weight=balanced)	**89.70%**	**92%**	**94%**	**93%**	**86%**

Precision is the most critical metric for this project since it directly affects the quality of location recommendations.

While Logistic Regression with class_weight=balanced achieved the highest precision (94%), SVC with balancing reached the highest overall accuracy and F1-score, showing strong generalization between precision and recall.

XGBoost and Random Forest provided high recall but lower precision, which may lead to over-recommending places with mixed feedback.

Best model to choose:

→ To evaluate the effectiveness of various machine learning models for sentiment-based location recommendation, we conducted extensive experiments using multiple classifiers under different settings. Among all the models evaluated, the Support Vector Classifier (SVC) with class_weight=balanced achieved the most favorable balance between precision and recall—two metrics crucial to our application. While high precision ensures that only genuinely well-reviewed locations are recommended, high recall enables the system to discover more potentially good locations that may otherwise be overlooked.

These results demonstrate that the SVC not only reduces false positives but also captures a broader range of positive user feedback. Therefore, we selected SVC as the final model for deployment in our automated sentiment-aware location recommendation system.

5. Code snippets for the important part

The code below comes from the best model.

```
#1. Dataset Loading and Preprocessing
import kagglehub
import os
import pandas as pd

# Download and load dataset
path = kagglehub.dataset_download("andrewmvd/trip-advisor-hotel-reviews")
csv_path = os.path.join(path, "tripadvisor_hotel_reviews.csv")
df = pd.read_csv(csv_path, encoding="utf-8", encoding_errors="replace")

# Convert ratings to binary sentiment
def label_sentiment(rating):
    return 'negative' if rating <= 3 else 'positive'

df['sentiment'] = df['Rating'].apply(label_sentiment)
```

Load dataset from Kaggle (*trip-advisor-hotel reviews*), convert ratings to binary sentiment (1, 2, 3 as negative; 4, 5 as positive)

```
#2. Text Cleaning and Lemmatization
import string
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

clean_line = []
for line in df['Review']:
    cl_line = line.strip().lower()
    cl_line = ''.join(c for c in cl_line if c not in string.punctuation and
c.isascii() and not c.isdigit())
    cl_line = ' '.join(cl_line.split())
    words = cl_line.split()
    cl_line = ' '.join(word for word in words if word not in stop_words)
    cl_line = ' '.join(lemmatizer.lemmatize(word) for word in cl_line.split())
    clean_line.append(cl_line)

df['Review'] = clean_line
```

The raw text data is cleaned to remove noise such as punctuation and special characters. After that, lemmatization is applied to reduce words to their base forms, improving consistency in the text.

```
# 3. Label Encoding and Data Splitting
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
df['sentiment'] = encoder.fit_transform(df['sentiment'])  # 1 = positive, 0 =
negative

x = df['Review']
y = df['sentiment']

x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42, stratify=y)
# 4. Text Vectorization (TF-IDF)

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words="english")
x_train = vectorizer.fit_transform(x_train)
x_test = vectorizer.transform(x_test)
# 5. Training SVC with Class Balancing
from sklearn.svm import SVC

svc = SVC(class_weight='balanced', random_state=42)
svc.fit(x_train, y_train)
y_pred = svc.predict(x_test)
 # 6. Evaluation Metrics and Visualization
from sklearn.metrics import (
    accuracy_score, f1_score, precision_score, recall_score,
    roc_auc_score, classification_report, confusion_matrix,
    ConfusionMatrixDisplay
)
import matplotlib.pyplot as plt

print(f"Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f} %")
print(f"Precision: {precision_score(y_test, y_pred):.2f}")
print(f"Recall: {recall_score(y_test, y_pred):.2f}")
print(f"F1 Score: {f1_score(y_test, y_pred):.2f}")
print(f"AUC ROC Score: {roc_auc_score(y_test, y_pred):.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred,
zero_division=0))

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
plt.show()
```

Label encoding (convert negative to 0, positive to 1), stratified data splitting to maintain y distribution in train and test sets, TF-IDF word embedding, train SVC (handle imbalanced data using class_weight=balanced), and plot confusion matrix

6. Possible Improvements

While the current model delivers promising results, several areas remain open for future improvement:

- **Deep Learning Models**: Exploring LSTM or transformer-based models (e.g., BERT) may enhance the model's ability to understand contextual meaning, especially in longer and more complex reviews.
- **Multi-class Sentiment**: Moving beyond binary classification to a multi-class approach (e.g., negative, neutral, positive) could provide more fine-grained insights into customer sentiment.
- **Explainability**: Integrating interpretability techniques such as SHAP or LIME could make the model's predictions more transparent and trustworthy for stakeholders.
- **Language Enhancement**: Incorporating sentiment lexicons or fine-tuned embeddings tailored to the hospitality domain might help capture subtle expressions in hotel reviews.
- **Handling Imbalance with Data Augmentation**: While class weights were used to address imbalance, data augmentation techniques (e.g., paraphrasing underrepresented reviews) could further enrich the training data for minority classes.

7. Conclusion and Findings

In this project, we performed sentiment classification on hotel reviews using various machine learning models. Among both baseline and improved models tested, the Support Vector Classifier (SVC) with class_weight='balanced' delivered the best overall performance, achieving an accuracy of 89.70%, precision of 92%, recall of 94%, and an F1-score of 93%. These results reflect not only a high precision (correctness of positive predictions) but also excellent recall, indicating the model's strong ability to identify most relevant sentiments, both positive and negative.

Compared to the baseline Logistic Regression model, the SVC significantly improves recall while maintaining high precision. This makes it particularly suitable for hotel recommendation systems, where identifying both satisfied and dissatisfied customers is essential for reliable suggestions.