Machine Design Document (MDD)

Project Title: Hotel Reviews Sentiment Analysis
Domain: Natural Language Processing (NLP) & Machine Learning

Student Name: Shreya Chittranshi

Course: Capstone Project Submission
Date: 20th August 2025

1. Executive Summary

Customer reviews are one of the most valuable assets in the hospitality industry. They provide insights into guest satisfaction, identify strengths, and highlight problem areas in hotel services. However, manually analyzing thousands of reviews is time-consuming and subjective. This project leverages Natural Language Processing (NLP) and Machine Learning (ML) techniques to build an automated sentiment analysis system that classifies hotel reviews as positive or negative.

The proposed solution uses TF-IDF vectorization to transform textual reviews into numerical features, followed by a Logistic Regression classifier to predict sentiment. The system achieves high accuracy and interpretability, making it suitable for real-world deployment in hotel management systems or customer feedback dashboards.

This Model Design Document (MDD) outlines the motivation, dataset, preprocessing, model design, implementation, evaluation, deployment, risks, and future directions.

2. Introduction

2.1 Background

In today's competitive hospitality industry, online reviews influence customer booking decisions more than advertisements or pricing. A single negative review can discourage potential customers, while positive reviews can boost reputation. Hotels often receive thousands of reviews across multiple platforms such as TripAdvisor, Booking.com, and Expedia.

Manually reading and categorizing these reviews is inefficient. Therefore, sentiment analysis — an automated approach to classifying text into positive or negative polarity — becomes critical.

2.2 Problem Statement

The challenge is to design a system that can accurately and automatically classify hotel reviews as either positive or negative, with minimal computational cost and high reliability.

2.3 Objectives

- Build a machine learning pipeline that classifies hotel reviews into positive/negative sentiment.
- Preprocess textual reviews by cleaning, normalizing, and tokenizing text.
- Train and evaluate models using robust metrics such as F1-score.
- Deploy the model as a user-friendly demo application for live predictions.

2.4 Scope

- English-language reviews only.
- Binary classification (positive vs. negative).
- Dataset from Kaggle’s hotel review corpus.
- Focus on classical ML methods for efficiency.

3. Literature Review

Sentiment analysis has been widely studied in NLP. Key methods include:

- **Rule-based approaches**: Use predefined sentiment lexicons (e.g., SentiWordNet). Limited in handling context.
- **Machine learning approaches**: Use classifiers like Logistic Regression, SVM, and Random Forest trained on labeled text datasets.
- **Deep learning approaches**: Leverage word embeddings (Word2Vec, GloVe) with LSTM, CNN, or Transformers for contextual understanding.

For this project, we adopt a **machine learning approach** using TF-IDF features due to its simplicity and interpretability.

Sentiment analysis has evolved significantly over the last two decades. Early approaches relied on **lexicon-based methods**, where dictionaries of positive and negative words were used to assign polarity. Although interpretable, these approaches often failed in handling complex sentences and sarcasm.

With the rise of machine learning, algorithms such as **Naïve Bayes, Logistic Regression, and SVM** were widely adopted for sentiment analysis. These models, combined with **TF-IDF**

vectorization, proved effective for large text datasets. For example, Pang et al. (2002) demonstrated the usefulness of SVMs for movie review classification.

In recent years, deep learning models such as **LSTM networks** and **transformer-based models (BERT, RoBERTa)** have achieved state-of-the-art accuracy. However, these require GPUs, longer training times, and more computational resources. For practical, lightweight deployment, classical ML models remain highly competitive, especially when interpretability and speed are important.

Given the constraints of this project, **Logistic Regression with TF-IDF features** was chosen as the primary modeling approach, balancing accuracy, interpretability, and resource efficiency.

4. System Requirements & Tools

4.1 Hardware Requirements

- Processor: Intel i5 or above (or Google Colab CPU runtime).
- Memory: Minimum 8 GB RAM.
- Storage: ~2 GB free disk space for dataset, libraries, and artifacts.

4.2 Software Requirements

- Programming Language: Python 3.9+
- Libraries:
    - Scikit-learn (ML algorithms, pipeline, metrics)
    - Pandas, NumPy (data manipulation)
    - Matplotlib, WordCloud (visualizations)
    - NLTK (stopwords, tokenization)
    - Joblib (model saving)
    - Streamlit (deployment app)

4.3 Dataset Requirements

- **Dataset:** Kaggle hotel review dataset.
- Columns: Review, Rating/Sentiment.
- Size: ~20,000–50,000 rows.

---

5. Data Understanding & Preprocessing

The dataset consists of textual hotel reviews paired with sentiment labels (positive/negative). Some datasets provide ratings (1–5 stars), which are mapped to binary polarity:

- Ratings ≥ 4 → Positive
- Ratings ≤ 2 → Negative
- Ratings = 3 → Neutral (dropped for simplicity).

5.1 Preprocessing Steps

1. **Lowercasing:** Converts all words to lowercase.
2. **Removing Noise:** Eliminates URLs, mentions, hashtags, punctuation, and numbers.
3. **Stopword Removal:** Removes common English stopwords (“the”, “is”, “at”).
4. **Tokenization:** Splits text into words.
5. **TF-IDF Vectorization:** Converts processed text into numerical features.

5.2 Example

Raw review: *“The room was spacious, but the staff were rude and unhelpful.”*
Cleaned text: *“room spacious staff rude unhelpful”*

---

6. Model Design

The overall pipeline is as follows:

Input Review → Text Cleaning → TF-IDF Vectorization → Logistic Regression → Sentiment Output

6.1 Baseline Model

- Dummy classifier (predict majority class).
- Establishes minimum benchmark.

6.2 Feature Extraction

- TF-IDF with n-gram range (1,2).
- Min document frequency = 5.
- Max document frequency = 0.9.

6.3 Classifier

- Logistic Regression (with class balancing).

- Hyperparameters tuned using GridSearchCV.

6.4 Flow Diagram

(Insert diagram showing input → preprocessing → feature extraction → classifier → prediction)

---

7. Implementation & Training

The project was implemented in Google Colab.

- **Step 1:** Import libraries.
- **Step 2:** Load dataset.
- **Step 3:** Clean and preprocess text.
- **Step 4:** Train/test split (80/10/10).
- **Step 5:** Train baseline and tuned models.
- **Step 6:** Evaluate using classification metrics.
- **Step 7:** Save model with Joblib.
- **Step 8:** Build Streamlit app for demo.

Libraries Required

- import pandas as pd
- import numpy as np
- import re
- import string
- from sklearn.model_selection import train_test_split
- from sklearn.feature_extraction.text import TfidfVectorizer
- from sklearn.linear_model import LogisticRegression
- from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
- import pickle

Data Preprocessing Code

```
def clean_text(text):

    text = text.lower()

    text = re.sub(r'\[.*?\]', '', text)

    text = re.sub(r'https?://\S+|www\.\S+', '', text)

    text = re.sub(r'<.*?>+', '', text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)

    text = re.sub(r'\n', '', text)

    text = re.sub(r'\w*\d\w*', '', text)

    return text


data['cleaned'] = data['Review'].apply(lambda x: clean_text(str(x)))
```

Feature Extraction

```
X = data['cleaned']

y = data['Sentiment']


tfidf = TfidfVectorizer(max_features=5000)

X_vectorized = tfidf.fit_transform(X)


X_train, X_test, y_train, y_test = train_test_split(X_vectorized, y, test_size=0.2,
random_state=42)
```

Model Training & Evaluation

```
model = LogisticRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)


print("Accuracy:", accuracy_score(y_test, y_pred))

print(classification_report(y_test, y_pred))
```

Save Model

```
pickle.dump(model, open("sentiment_model.pkl", "wb"))

pickle.dump(tfidf, open("tfidf.pkl", "wb"))
```

---

8. Testing Plan

- **Unit Testing**: Verify preprocessing functions remove unwanted characters.

- **Integration Testing**: Check vectorization + model prediction pipeline.
- **System Testing**: End-to-end testing from user input → output sentiment.
- **User Testing**: Validate predictions with actual hotel reviews.

Deployment Plan

- **Backend**: Python (Flask or Streamlit).
- **Frontend**: Simple text input field for review entry.
- **Model Loading**: Load sentiment_model.pkl and tfidf.pkl.
- **Output**: Display predicted sentiment with probability score.

Example (Streamlit UI):

```
import streamlit as st


st.title("Hotel Review Sentiment Analysis")
user_input = st.text_area("Enter a hotel review:")


if st.button("Predict"):
  transformed = tfidf.transform([user_input])
  prediction = model.predict(transformed)
  st.write("Predicted Sentiment:", prediction[0])
```

---

8. Evaluation

8.1 Metrics

- Accuracy
- Precision, Recall, F1-score
- Confusion Matrix

8.2 Results

- Logistic Regression achieved **Accuracy: ~90%**, **F1-score: 0.89**.

- Confusion matrix showed balanced classification across both classes.

8.3 Error Analysis

- Reviews with mixed sentiments (positive + negative comments) caused misclassifications.
- Example: *"The food was delicious but the room was dirty."* → predicted *positive*.

8.4 Feature Importance

- Top positive words: "clean", "amazing", "friendly", "spacious".
- Top negative words: "dirty", "rude", "broken", "noisy".

---

9. Deployment

The trained pipeline was saved using Joblib and integrated into a **Streamlit web application**.

App Features:

- Textbox for review input.
- Predict button.
- Displays prediction (positive/negative) + probabilities.

This lightweight app allows hotels to quickly test reviews without needing advanced infrastructure.

---

10. Risks & Limitations

10.1 Risks

- **Imbalanced Dataset:** If too many positives vs. negatives, accuracy may be misleading.
- **Sarcasm/Irony:** Hard to detect with simple models.
- **Domain Shift:** Model may fail if tested on non-hotel reviews.

10.2 Mitigation

- Used class weights in Logistic Regression.
- Documented sarcasm as limitation.
- Recommend periodic retraining on new reviews.

---

11. Future Enhancements

- Multi-class sentiment (very positive, neutral, very negative).
- Multilingual support (Hindi, Spanish, French reviews).
- Deep learning methods (BERT, DistilBERT).
- Dashboard integration for live monitoring of reviews.

---

12. Conclusion

This project successfully demonstrated the design and implementation of a sentiment analysis system for hotel reviews. By leveraging TF-IDF vectorization and Logistic Regression, the model achieved strong performance with minimal resources. The system can assist hotels in quickly analyzing customer feedback, identifying service gaps, and improving guest satisfaction.

Future enhancements, such as deep learning and multilingual support, can further improve the model's robustness and applicability across domains.

---

13. References

- Pang, B., & Lee, L. (2002). *Thumbs up? Sentiment Classification using Machine Learning Techniques*. ACL.
- Kaggle Dataset: Hotel Reviews Sentiment Analysis.
- Scikit-learn documentation: https://scikit-learn.org
- Streamlit documentation: https://streamlit.io
- NLTK documentation: https://www.nltk.org