



Low Resource Sentiment Analysis

MID FINAL YEAR PROJECT REPORT (2024-2025)

DEEKSHANT BANSAL
(21JE0285)

Supervisor : Dr. Annavarapu Chandra Sekhara Rao

INTRODUCTION

Sentiment analysis holds significant value for e-commerce companies, enabling them to gain insights from customer sentiment. By evaluating customer feedback, businesses can identify areas requiring improvement in their products or customer service. This empowers them to make informed decisions on how to better cater to customer needs, leading to enhanced sales and customer loyalty. Moreover, sentiment analysis assists in identifying positive customer feedback, which can be shared on company websites or social media platforms to influence potential customers' purchasing decisions.

Low resource sentiment analysis refers to the task of performing sentiment analysis in environments where resources are limited. This can include situations where:

1. **Data Availability:** There is a scarcity of labeled data for training machine learning models, especially in languages that are underrepresented in NLP research (i.e., non-English languages).
2. **Computational Power:** Limited access to powerful hardware like GPUs, which makes it challenging to train complex deep learning models.
3. **Storage and Memory Constraints:** When the available memory or storage is limited, it becomes difficult to deploy large models like BERT or GPT.

Under this report we will cover **sentiment analysis of hindi language** dataset.

In our work, we aim to address two primary tasks. First, we try to implement traditional machine learning algorithms, such as Support Vector Machine, Random Forest, Multinomial Regression etc, for sentiment analysis. In the next step, we advance to more sophisticated deep learning approaches by employing fine-tuned, general-purpose language representation models. These models are expected to capture the complex linguistic nuances present in code-mixed text more effectively, improving the overall performance of sentiment classification.

DATA AND EVALUATION TECHNIQUES

In our work, we use the Hindi dataset from **IIT Patna Movie and Product Reviews**. This dataset contains reviews annotated with both word-level language identification and sentence-level sentiment labels, which fall into three categories: Positive, Negative, and Neutral.

- **Positive:** Tweets in this category convey feelings of joy, express admiration or praise towards a product or movie. An example of a Hindi positive tweet is: "अच्छी क्वालिटी का कैमरा है |" (Translation: "A nice quality camera").
- **Negative:** These tweets express criticism, discontent, or negative opinions directed at an individual, group, product, or situation. For instance, a Hindi negative tweet reads: "डिंपल कपाड़िया लाउड परफार्मेंस के बावजूद फिल्म की जरूरत पूरी करने में असफल रही हैं।" (Translation: "Dimple Kapadia wasn't able to fulfill the film needs despite her loud performance").
- **Neutral:** This category includes tweets that provide factual information, news, or advertisements without showing any emotional bias. A typical Hindi example is: "इनमें क्रिस इवंस, माइकल मोनागन, एंथोनी मैकी जैसे नाम शामिल हैं।" (Translation: Chris Evans, Matthew McConaughey, Anthony Mackie are a part of this.)

The dataset is split into the train, valid and test in the ratio 8:1:1, and each part is stored separately in a CSV file. Each record in the CSV file has the following format |label ,review|.

The Movie review dataset consist of around 3000+ labeled data and Product reviews consist of 5000+ labeled data.

Type	Split	Total	Positive	Negative	Neutral
HINDI (MOVIE)	Train	2475	1039 (42%)	743 (30%)	693 (28%)
	Validation	309	121 (39%)	108 (35%)	80 (26%)
	Test	309	121 (39%)	96 (31%)	92 (29%)
	TOTAL	3093	1281	947	865

Type	Split	Total	Positive	Negative	Neutral
HINDI (PRODUCT)	Train	4177	1838 (44%)	1754 (42%)	585 (14%)
	Validation	522	225 (43%)	225(43%)	72 (14%)
	Test	522	240 (46%)	214 (41%)	68 (13%)
	TOTAL	5221	2303	2193	725

For each ML Model used in the study there will be a classification report for each category having the following -

- **Precision:** The proportion of correct positive predictions among all predictions made for a particular class.
- **Recall:** The proportion of correct positive predictions among all actual samples of that class.
- **F1-score:** The harmonic mean of precision and recall, providing a balance between them.
- **Support:** The number of true instances for each class in the dataset

Along with classification reports of each class there will also be Overall **Best Accuracy** for each ML Method used on the dataset. Also for all four metrics its macro average (average of three classes for each parameter) and a weighted average.

For the CNN Method we used the accuracy parameter till now and could further use other parameters too.

METHODOLOGY

Till now, I have tried 2 approaches on these datasets:

- **ML based approach:** Applied various ML algorithms and checked for the best and the most consistent result.
- **CNN approach:** Used bert based model to tokenize the data and passing the output through a CNN Model.

1. ML BASED APPROACH

1.1. DATA CLEANING AND PREPROCESSING:

- **Removing Special Characters:** The code removes punctuation and newlines from the Hindi text to prepare it for translation. Example - "यह एक अच्छा दिन है।" is converted to "यह एक अच्छा दिन है" .
- **Translating text:** The [googletrans](#) library is used to translate the cleaned Hindi text to English. Example: Original Hindi Text: "यह एक अच्छा दिन है।".

After removing special characters: "यह एक अच्छा दिन है"

After translation: "It is a good day".

- **Stopwords Removal:** Stopwords (e.g., "is", "the", "and") are words that do not add significant meaning to the text. The code removes stopwords from the translated text using the NLTK library. Example: Translated text: "It is a good day"

After removing stopwords: "good day".

- **Text Cleaning:** Converts the text to lowercase. Removes any character that is not a letter (removes numbers, special characters, etc.). Ensures single spacing between words. Example: Text: "Good Day! The temperature is 25 degrees."

After cleaning: "good day the temperature is degrees"

- **Stemming:** It reduces words to their root form to unify variations of the same word. Example: "playing", "plays", and "played" are reduced to "play".

1.2. MODELING APPROACHES:

This section outlines various machine learning models used for sentiment classification. After preprocessing the text data, different models are trained and evaluated to find the best one for predicting sentiment.

1.2.1. Text Vectorization

To transform the preprocessed text data into a format suitable for machine learning models, we used a technique called **Bag-of-Words (BoW)**. This involves converting the text into numerical vectors by counting the frequency of each word in the dataset. Each review is represented as a vector, where each element indicates the presence or absence of a word.

- **Why Vectorization is Important:** Machine learning models cannot directly process text data. Converting the text into numerical features allows us to use algorithms like Naive Bayes, Support Vector Machines, and others for classification.
- **Real-world Example:** If the dataset contains sentences like "Great product" and "Not a good product", the vector representation would count the occurrences of words like "great", "good", and "product" across all the reviews.

1.2.2. Model Selection and Training

I experimented with various machine learning models to find the best one for sentiment classification. Here's a breakdown of the approaches used:

- **Multinomial Naive Bayes:**
 - This model is effective for text classification, particularly when dealing with categorical data like word counts.
 - It assumes that the features (words) are conditionally independent given the class (sentiment).
 - **Pros:** Efficient and performs well with high-dimensional data.
 - **Cons:** Assumes independence between features, which might not always hold in natural language.
- **Support Vector Machine (SVM):**
 - The SVM algorithm finds the optimal hyperplane that separates the data points of different classes.
 - I tested both linear and radial basis function (RBF) kernels to see which performs better.
 - For SVM, we tuned parameters like the regularization strength C.
 - **Pros:** Works well with high-dimensional data and can handle non-linear relationships.
 - **Cons:** Can be computationally intensive, especially for large datasets.
- **Logistic Regression:**
 - A simple yet powerful classifier that uses a logistic function to model the probability of a given input belonging to a particular class.

- We tuned hyperparameters such as regularization strength to optimize performance and prevent overfitting.
- **Pros:** Easy to interpret and implement, performs well on linearly separable data.
- **Cons:** May struggle with non-linear data unless combined with feature engineering techniques.
- **Random Forest:**
 - An ensemble learning method that builds multiple decision trees and combines their outputs to improve accuracy.
 - We varied the number of trees to find the optimal model.
 - **Pros:** Robust to overfitting and handles missing values well.
 - **Cons:** Can be slower to train, especially with a large number of trees.
- **Gradient Boosting Classifier:**
 - Another ensemble technique that builds trees sequentially, with each new tree correcting errors made by the previous ones.
 - **Pros:** High accuracy, particularly when tuned properly.
 - **Cons:** Requires careful tuning of hyperparameters like the learning rate and number of trees, which can be time-consuming.
- **K-Nearest Neighbors (KNN):**
 - A non-parametric classifier that assigns a class based on the majority vote of the nearest neighbors.
 - We experimented with different values of k (number of neighbors).
 - **Pros:** Simple to understand and implement.
 - **Cons:** Can be inefficient for large datasets due to its high computation cost.

2. CNN APPROACH

2.1. DATA CLEANING AND PREPROCESSING:

- **Label Encoding:** The sentiment labels were categorical (e.g., positive, negative, neutral) but needed to be converted into numerical format for training. This was achieved using Pandas. This transformation allowed the model to treat sentiments as numerical classes, simplifying the classification task.
- **Tokenization:** The BERT tokenizer was used to convert text into tokens that BERT can understand. This process also involved adding special tokens ([CLS] and [SEP]), converting tokens into numerical IDs, and generating attention masks.
- **Padding and Truncation:** Since BERT requires inputs to have a fixed length, each text was padded or truncated to a maximum length of 128 tokens. This ensured uniformity across inputs, preventing variable sequence lengths from affecting the model.

2.2. MODEL APPROACHES:

2.2.1. Text Representation using BERT Embeddings

- BERT (**bert-base-multilingual-cased**) was used to extract rich contextual embeddings from the text data. This model is pre-trained on a large corpus of multilingual text and can capture semantic nuances effectively.
- **BERT Tokenizer:**
 - The text was tokenized into subword units, ensuring that even rare or misspelled words were represented meaningfully.
 - The output of BERT included a **768-dimensional vector** for each token, encapsulating its contextual meaning.
- **Embedding Extraction:**
 - For each input text, the last hidden state of BERT was used to generate embeddings. These embeddings were reshaped to fit the input shape required by the CNN model.

2.2.2. Model Architecture: CNN on Top of BERT Embeddings

A custom Convolutional Neural Network (CNN) was designed to classify sentiment using the embeddings generated by BERT:

- **CNN Layers:**
 - **Conv1 Layer:** The first convolutional layer applied 128 filters with a kernel size of 3. The output was passed through a ReLU activation function and max-pooling to reduce dimensionality.
 - **Conv2 Layer:** The second layer used 64 filters with a kernel size of 3, followed by ReLU activation and max-pooling.
 - **Global Average Pooling:** The pooled features were further reduced by averaging across the sequence length, making the model robust to variations in text length.
 - **Fully Connected Layer:** The final dense layer mapped the output to sentiment classes (positive, negative, neutral).
- **Benefits of CNN on BERT Embeddings:**
 - The convolutional layers captured local patterns in the embeddings (e.g., phrases that indicate sentiment).
 - Pooling layers reduced the computational load while retaining key information, improving the model's efficiency.

2.2.3. Training and Optimization

The training process involved several steps to ensure that the model learned effectively:

- **Freezing BERT Parameters:** To save on computational resources, the BERT model's parameters were kept frozen. Only the CNN layers were trained.
- **Loss Function: Cross-entropy loss** was used, which is standard for multi-class classification tasks.
- **Optimizer:** The **Adam optimizer** was chosen due to its adaptive learning rate capabilities, which help with faster convergence.
- **Training Loop:**
 - For each batch, the BERT model generated embeddings, which were passed through the CNN model for classification.
 - The loss was computed, and gradients were backpropagated to update the CNN layer weights.
 - Metrics such as accuracy were tracked to monitor performance during training.

OBSERVATIONS AND RESULTS

In this section, we present a detailed analysis of the results obtained from applying various models to our sentiment analysis task. The observations are based on key performance metrics which help evaluate the effectiveness of each model in classifying the text data.

By comparing the strengths and weaknesses of different approaches, we aim to provide insights into their suitability for text classification and highlight the trade-offs involved in selecting a particular model. This analysis will serve as a guide for interpreting the results and understanding the practical implications of the chosen methodologies.

1. ML APPROACH

1.1. Performance on the Movie Review Dataset

The evaluation of machine learning models on the movie review dataset, consisting of training, validation , and testing subsets, highlights distinct performance trends.

Summary Table: Model Performance on Testing Set (Movie Review Dataset)

Model	Accuracy	Precision (Weighted)	Recall (Weighted)	F1-Score (Weighted)
Multinomial Naive Bayes	0.40	0.40	0.40	0.39
SVC (Support Vector Classifier)	0.45	0.41	0.45	0.33
Random Forest	0.41	0.41	0.42	0.41
Gradient Boosting	0.40	0.40	0.40	0.38
K-Nearest Neighbors	0.43	0.38	0.44	0.30
Logistic Regression	0.45	0.45	0.45	0.44

Key Insights:

- **Logistic Regression** achieved the best accuracy (0.45) and F1-score, demonstrating its ability to distinguish sentiment more effectively.
- **SVC** had a good accuracy but lacked in F1-score.
- **Neutral class** consistently shows poor metrics across models, indicating difficulties in classifying this sentiment.
- **Positive class** achieves higher recall across most models, particularly in SVC (0.96) and KNeighbors (0.96).
- Models such as **RandomForest** deliver balanced performance across all metrics.

1.2. Performance on the Product Review Dataset


The product review dataset, analyzed across training, validation and testing demonstrated different model dynamics.

Summary Table: Model Performance on Testing Set (Product Review Dataset)

Model	Accuracy	Precision (Weighted)	Recall (Weighted)	F1-Score (Weighted)
Multinomial Naive Bayes	0.58	0.50	0.58	0.53
SVC (Support Vector Classifier)	0.61	0.52	0.61	0.56
Random Forest	0.55	0.47	0.55	0.50
Gradient Boosting	0.61	0.60	0.61	0.59
K-Nearest Neighbors	0.43	0.50	0.43	0.43
Logistic Regression	0.54	0.51	0.54	0.51

Key Insights:

- I. **Gradient Boosting** proves to be the most balanced model, excelling in all metrics and providing the best overall performance.

- 
- II. **SVC** also performs well, offering high accuracy and recall, making it a competitive choice.
 - III. **K-Nearest Neighbors** struggles significantly in all metrics, making it unsuitable for this dataset.
 - IV. **Random Forest** and **Logistic Regression** perform adequately but do not match the precision and recall of Gradient Boosting or SVC.

1.3. Summary -

- **Multinomial Naive Bayes**
 - It performed well on product dataset but not Movie dataset.
 - Therefore it is not generalized.
- **SVC (Support Vector Classifier)**
 - Accuracy wise it was best in both the sets but was average in f1 score
 - It could be used as a good technique.
- **Random Forest**
 - It performed evenly on all metrics in both the datasets
 - Therefore it is a generalized approach and can be used.
- **Gradient Boosting**
 - It performed best in the product dataset but not as good on the movie dataset.
 - Therefore it is not a generalized approach.
- **K-Nearest Neighbors**
 - It didn't perform upto the mark in both the dataset like other models.
 - Therefore it is not a great approach to use.
- **Simple Logistic Regression**
 - It performed best on movie dataset and decently on product dataset.
 - Therefore it can be a good classification technique.

2. CNN APPROACH-

The accuracy obtained for the CNN based approach is better than any of the above approaches solely as it is around 0.54 for Movie dataset and 0.69 for Product Dataset.

So DL based approaches are seeming to be better in the sentiment analysis task.

CONCLUSION

In this project, we explored various approaches to perform low-resource sentiment analysis on Hindi datasets, specifically focusing on movie and product reviews. The results demonstrated that deep learning models, particularly a CNN built on top of BERT embeddings, achieved superior accuracy compared to traditional machine learning models. While approaches like Support Vector Machines and Logistic Regression performed consistently, the CNN-based method showed better scalability and adaptability to complex language nuances.

The project successfully highlights the potential of leveraging modern NLP techniques in resource-constrained environments, proving that even with limited data, efficient sentiment classification can be achieved. This work lays the foundation for further research into optimizing model performance for low-resource languages, helping to bridge the gap in NLP coverage across different linguistic contexts.

FUTURE SCOPE

In this section, we briefly describe the future steps for the implementation of our Hindi Sentiment Analysis project. So far, we have implemented traditional machine learning algorithms and the sole CNN model on a small ranging dataset.

For the future of the project, I am thinking of some of the following points:

- **Exploring Advanced Deep Learning Architectures:** Future work can explore more sophisticated models beyond CNN and traditional ML approaches, such as Long Short-Term Memory (LSTM) networks, Transformers, and attention-based models. These architectures can capture longer dependencies in text and provide better contextual understanding, which could significantly improve performance on complex sentiment data.
- **Leveraging Pre-trained Language Models:** Incorporating state-of-the-art pre-trained models like BERT variations (e.g., RoBERTa, XLM-R) can enhance sentiment classification accuracy, especially for low-resource languages. These models can be fine-tuned on sentiment datasets to leverage their vast pre-existing knowledge for better generalization and sentiment capture.
- **Using Larger and More Diverse Datasets:** To achieve higher accuracy and robustness, future research can focus on utilizing much larger datasets sourced from various domains, such as product reviews, social media posts, and news articles.
- **Exploring Ensemble Approaches:** Combining traditional and deep learning models in ensemble methods could lead to a more balanced and robust sentiment analysis model, especially for diverse datasets like movie and product reviews.

These future approaches will help us advance beyond traditional machine learning techniques, and by leveraging the power of pre-trained language models, we hope to achieve more accurate and efficient sentiment classification.

REFERENCES

- [Dataset from IIT Patna](#)
- [CNN Approach paper](#)
- [ML Approach paper used](#)