Customer Sentiment Analysis

1 Data Collection

The quality and quantity of data play a crucial role in the effective training of any Machine Learning models. Our main objective was to develop a sentiment analysis model that would predict customer satisfaction. The sources can be from support tickets, call transcriptions, survey responses, and social media data. The best source of data that was accessible to me was social media. However, scraping the data from Facebook comments was difficult and restricted by Meta. The collection of data from comments via the Facebook API required the admin access of the page itself. Thus, the data was collected from the Google Play Store reviews of the myWorldLink app and the comment section of all the videos published by Worldlink on YouTube. The collection of the data was facilitated by the API from Google. The data was collected and stored in a CSV file along with additional metadata.

2 Dataset Details

2.1 User Reviews

The dataset comprises about 10,000 user reviews for the app. It provides details of the review text, username, date of the review, and rating given to the app. This collection of reviews serves as a valuable resource for analyzing customer feedback and their sentiments.

2.2 Comment Data

It comprises about 3,000 comments on different videos uploaded by Worldlink. The sentiments of the customer can be portrayed by the comments from videos as well. As people tend to be vocal about their sentiments on other platforms, this gives insight into sentiments on the whole product lineup of Worldlink. It consists of the username, date, comment text, and video title.

3 Data Preprocessing

Data preprocessing is a crucial step in any data-driven project, ensuring that raw data is cleaned, transformed, and formatted for effective analysis or model training. The data we have collected is raw and needs to be preprocessed accordingly. Before performing any cleaning of data, we have to know the contents and type of the data. The reviews of the user in Nepal generally tend to be mixed with different languages such as English, Nepali, and even Romanized Nepali. This increases the complexity of this process hugely as Nepali is a low-resource language.

3.1 Language Detection and Filtering

The language in the data needs to be detected and handled accordingly. In this dataset, the languages are English, Nepali, and Romanized Nepali. Firstly, we checked whether the individual words in the reviews were valid English by checking the words against the dictionary. The Enchant library was used. If the entire text consists of more than 80% English words, this would likely be English reviews. The 20% leeway was added in consideration of various slang, shortcut words, complex words, or spelling mistakes in the review. For the remaining reviews, which were likely in Nepali or Romanized Nepali, we employed transliteration to convert Romanized Nepali (written in the Latin alphabet) into the Nepali script (Devanagari). Before transliterating, words like “Worldlink” were handled to ensure they remained unchanged. The process was applied to a dataset, separating English reviews into one file and the transliterated Nepali reviews into another. This approach efficiently filters and processes text data by language, enabling better analysis and application of language-specific operations.

3.2 Text Cleaning

Text cleaning is an important step to ensure the removal of inconsistencies, errors, or missing values, which need to be handled before analysis. The review text was converted to string format and checked for missing or NaN values. The rows with missing values were dropped. The entire text was converted from uppercase to lowercase. Punctuation marks, special characters, and symbols were also removed from the text. Because the model we tend to use was BERT, stopword removal, stemming, and lemmatization procedures to clean the text were not performed as they might change or remove the context from the sentences. Unlike traditional classification models, BERT can rather be harmed by it in some scenarios.

4 Data Labeling

Data labeling is the process of adding labels to raw data so that machine learning models can understand it. The goal of sentiment analysis is to classify text based on its emotional tone, typically categorizing it as positive, negative, or neutral. Data labeling for this task involves manually or automatically assigning these sentiment labels to text data.

4.1 Labeling of Playstore Reviews

The labeling of Playstore reviews was done based on the ratings provided by the users. The labels were assigned as follows:

Ratings of 1 and 2 were categorized as 0 (negative), indicating dissatisfaction with the app.

Ratings of 4 and 5 were labeled as 1 (positive), reflecting a positive user experience or approval of the app.

A rating of 3 was classified as 2 (neutral), representing a neutral stance where the user neither strongly liked nor disliked the app.

4.2 Labeling of YouTube Comments

The labeling of YouTube comments needed to be done manually. However, due to the time constraints of this project, manual labeling was not possible. This was done with the help of the Llama3 model running locally, which was used to classify these reviews into three sentiment categories: positive (1), negative (0), and neutral (2). Each review is read from a CSV file, and a prompt is generated for the model to evaluate the sentiment of the review. The model analyzes the text and returns a sentiment label, which is then added as a new column in the dataset. The labeled dataset, with the original reviews and their corresponding sentiment labels, is saved to a new CSV file. This automated process streamlines the labeling of large datasets while ensuring consistency and efficiency.

5 Tokenization

Tokenization is the process of converting raw text into a format that a machine learning model can process. It involves breaking down the input text into smaller components, typically tokens, which can represent words, subwords, or characters. The AutoTokenizer class from the transformers library is used to load the pre-trained BERT tokenizer for the "bert-base-multilingual-uncased" model. The text is first truncated or padded based on the specified max length to ensure all input sequences are of equal length. The truncation prevents longer inputs from exceeding the model's capacity, while padding ensures shorter inputs are padded with special tokens. The tokenizer encodes the text into token IDs, which are the numerical representations corresponding to subword units in the model's vocabulary. The tokenized data is returned for model input as tensors for efficient computation. The tokenized form of text is how the BERT model can capture context and semantic relationships.

6 Model Selection and Training

The selection of the model is heavily based on the type of task and the data available for the task. The dataset for this project consists of mixed-language data. While we can use only the English data for training, this may convey alternating results as the scope of the analysis is the Nepalese market, so the inclusion of Nepali is also crucial. Thus, the model that has been selected is the multilingual-uncased variant of BERT. This is a pre-trained model on multilingual language.

The selection of bert-base-multilingual-uncased is due to its ability to generalize across languages, as it was pre-trained on text from 104 languages. This multilingual foundation allows the model to capture shared linguistic patterns and semantic relationships, making it uniquely suited for sentiment analysis in multiple linguistic contexts, which is the case in our dataset. Unlike alternatives like LSTM or XGBoost, which require language-specific architectures or extensive labeled datasets, BERT leverages cross-lingual transfer learning, enabling robust performance even in low-resource languages. For instance, fine-tuning the model on English sentiment data can yield reasonable accuracy in languages like Spanish or Hindi without additional training, a capability critical for global applications. Benchmark studies underscore its superiority: on tasks like the XLM-T dataset (Twitter data in 8 languages), this BERT variant achieves 85% accuracy in zero-shot cross-lingual transfer, outperforming LSTM and XGBoost by significant margins. Its state-of-the-art performance stems from its ability to handle code-switching, dialectal variations, and morphologically rich languages, challenges where traditional models falter due to their reliance on manual feature engineering or sequential processing limitations.

A key strength of bert-base-multilingual-uncased lies in its bidirectional Transformer architecture, which generates context-aware embeddings by analyzing text in both directions simultaneously. This bidirectional context is indispensable for sentiment analysis, where meaning often hinges on nuanced relationships between words (e.g., negation in "not great"). This highlights its superiority over unidirectional models like LSTM, which struggle with long-range dependencies, or XGBoost, which cannot infer context without handcrafted features. Additionally, while training BERT from scratch is resource-intensive, fine-tuning the pre-trained multilingual model is computationally efficient. By adding a simple classification layer to BERT's pooled output, the model adapts rapidly to sentiment labels with minimal hardware requirements and preprocessing due to its uncased design and WordPiece tokenizer. In contrast, scaling LSTM or XGBoost for multilingual tasks demands significant computational overhead, hyperparameter tuning, and language-specific pipelines, making BERT the pragmatic choice for scalable, accurate, and adaptable sentiment analysis.

7 Performance and Evaluation

The model showed good results with performance metrics revealing both stability and notable fluctuations before converging to robust results. Initial epochs showed promising progress across all the metrics. However, epochs 4-5 showed instability and a dip in metric score across both training and validation. The model recovered gradually from middle epochs onward and peaked at 84.7% accuracy in the final epoch. The model achieved its highest performance: 84.7% accuracy, 0.847 precision, 0.847 recall, and an F1 score of 0.847. The dip in middle epochs may reflect the challenges in adapting to the multilingual aspect or noisy data, but the recovery reaffirms the effectiveness of the fine-tuning strategy. Overall, the final metrics validate the model's suitability for multilingual sentiment analysis.