Sentiment Classification using Sentence Embeddings: Exploiting Sentence Transformer Loss Functions Journal Submissions

Anonymous authors
Paper under double-blind review

Abstract

Evaluating customer sentiment plays a critical role in business success. By analyzing customer feedback, companies can swiftly identify expectations, areas for improvement, and pain points related to their products and services. Sentiment analysis, fueled by advances in natural language processing techniques, has become widely accepted for this purpose. In this study, we leverage the well-known "Twitter US Airline Sentiment" dataset to develop a sentence transformer architecture based on a pre-trained transformer model (mpnet-base). We fine-tune the model using appropriate loss functions to generate semantically rich sentence embeddings that are subsequently fed into gradient boosting-based machine learning algorithms. The resulting hybrid model achieves impressive sentiment prediction performance. Additionally, this study delves into the intricacies of various transformer loss functions that can be applied to fine-tune the sentence transformer model for enhanced sentiment classification performance. Our sentence transformer architecture, fine-tuned on CosineSimilarity loss function and combined with Light Gradient Boosting Machine Classifier, achieves an excellent accuracy of 86.5%, while demonstrating high recall rates even for minority sentiment classes (74.4% for neutral and 82.9% for positive sentiment) without any data augmentation. Our study emphasizes that fine-tuned sentence transformer models can outperform existing techniques for sentiment classification, particularly in tri-class sentiment scenarios and they come with the inherent advantages of lesser computational load and higher scalability opportunity.

1 Introduction

Traditionally, customer feedback surveys have served as a reliable method for companies to gauge customer sentiments. However, these surveys often suffer from low response rates, leading to the voices of the majority of customers going unheard. Additionally, the time-consuming nature of these surveys impedes swift responses from companies. With the advent of Natural Language Processing (NLP) and Big Data, companies can now gain insights into customer sentiment and identify trends by analyzing social media posts and review websites. In this context, sentiment analysis plays a crucial role. Companies can leverage artificial intelligence to implement real-time analytics of electronic communications from customers, such as comments on social media and online review sites, to understand the sentiments expressed from a consumer perspective. This approach is less intrusive and faster than traditional customer feedback surveys, and it tends to capture unbiased customer opinions more subtly. Consequently, companies can derive quick, actionable insights and make informed decisions based on the sentiment analysis of customer reviews and feedback. Numerous studies have been conducted to implement sentiment analysis on the Twitter US Airline Sentiment dataset, particularly utilizing transformer models such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), yielding excellent results. However, there has been limited focus on leveraging sentence transformer architectures to enhance prediction performance in sentiment analysis. This research aims to investigate whether the combination of sentence transformer models, fine-tuned with appropriate transformer loss functions and gradient boosting algorithms, can improve the model's prediction performance. This article explores the following research questions:

- How good are sentence embeddings for sentiment classification?
- What are the working theories behind the different transformer loss functions applicable for sentiment classification?
- What is the lift achieved in terms of different performance metrics when using fine-tuned sentence transformer models instead of base sentence transformer models?

This article has been structured as follows: Section 2 depicts related works carried out in this domain; Section3 underlines the data description and the comprehensive design approach followed to build the models; In Section4, the model performance is analyzed; finally Section5 talks about the broader implications and potential future direction of this research . Code is publicly available at https://github.com/***/Sentiment_Classification.

2 Understanding the sentiment landscape: SotA Analysis

Numerous research efforts have explored sentiment classification, particularly using the "Twitter US Airline Sentiment" dataset. Notable studies include:

- **Topic-Based Categorization Approach:** In their pioneering work, Pang et al. (2002) treated sentiment classification as a special case of topic-based categorization. This work can be considered as the stepping stone in the domain of sentiment classification. Here the authors explored the possibility of treating sentiment classification as a special case of topic-based categorization with the two "topics" being positive sentiment and negative sentiment. Though the accuracies achieved here for sentiment classification using standard machine learning techniques (Naive Bayes, maximum entropy classification, and support vector machines) outperformed human-produced baselines, they were worse compared to topic-based categorization probably due to lack of use of sophisticated text vectorization techniques capable of capturing the semantic relation among words.
- **Ensemble Classifiers:** Researchers in Sinha & Sharma (2020) used five different classifiers and compared their performances and concluded that ensemble classifiers yield better accuracy when compared to individual classifiers. The paper demonstrated that random forest classifier which is an ensemble classifier achieved the maximum accuracy of 76% on the US airline dataset.
- **Bag-of-Words and TF-IDF Techniques:** Tusar & Islam (2021) applied Bag-of-Words and TF-IDF (term frequency-inverse document frequency) techniques for numerical representation in combination with various ML classification algorithms (Support Vector Machine, Logistic Regression, Multinomial Naive Bayes and Random Forest) for sentiment classification. Support Vector Machine and Logistic Regression achieved the maximum accuracy of 77% on the US airline dataset with Bag-of-Words being used for numerical representation.
- **Capsule network:** In Demotte et al. (2021), authors proposed a novel approach based on capsule network architecture tailored for social media content analysis. Remarkably, the proposed approach achieved state-of-the-art performance even without relying on any linguistic resources, demonstrating their effectiveness in sentiment analysis. The proposed architectures achieved an accuracy of 82.04% on the US Airline dataset. This result highlighted significant accuracy enhancements in text processing for social media content analysis and at the same time offered a fresh perspective for sentiment analysis in the dynamic landscape of social media.
- **Affection Driven Neural Networks:** In Xiang et al. (2020), the authors focused on one of the key challenges being faced by the deep neural networks for sentiment analysis which is effectively incorporating external sentiment knowledge. In this work, the authors proposed an innovative approach called "affection-driven neural networks" that leverages affective knowledge. Affective knowledge is obtained in the form of a lexicon based on the Affect Control Theory . This lexicon represents affective attributes using three-dimensional vectors: Evaluation, Potency, and Activity (EPA). The EPA vectors are mapped to an affective influence value and integrated into LSTM (Long

Short-term Memory) models. This integration allows the neural network to highlight affective terms during sentiment analysis. The proposed approach consistently outperformed conventional LSTM models by 1.0% to 1.5% in terms of accuracy across three large benchmark datasets. For the US airline dataset, the proposed neural network architecture comprising of LSTM-AT (LSTM with attention mechanism to re-weight important words before the fully connected layer) with Evaluation achieved the highest accuracy of 82.3%.

- **Deep Neural Networks:** Deep learning models have emerged as promising solutions for NLP challenges. In Dang et al. (2020), the authors reviewed recent studies that employ deep learning to tackle sentiment analysis problems, particularly sentiment polarity. They explored models using TF-IDF and word embeddings, applying them to various datasets. In this paper, for the US airline dataset, Word2vec in combination with RNN (Recurrent neural network) achieved the highest accuracy of 90% However, this study was limited to binary classification and is expected to yield lesser accuracy when applied to tri-class sentiment classification.
- **Hybrid BERT Models:** In Talaat (2023), the author proposed a hybrid approach that combines BERT with Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Gated Recurrent Unit (BiGRU) algorithms. The author created hybrid deep learning models by stacking two versions of BERT (RoBERTa & DistilBERT) with BiLSTM and BiGRU layers. RoBERTa (Robustly Optimized BERT Approach) (Liu et al., 2019) is a variant of BERT optimized for sequence-to-sequence modeling. Like BERT, RoBERTa is a transformer-based language model that uses self-attention to process input sequences and generate contextualized representations of words in a sentence. It generates word embeddings effectively. One crucial difference between RoBERTa and BERT is that RoBERTa was trained on a much larger dataset. Moreover, RoBERTa uses a dynamic masking technique during training that helps the model learn more robust and generalizable representations of words. The proposed architecture of RoBERTa with BiGRU layers yielded the highest accuracy of 86% for the US airlines dataset.

As evident from above, techniques behind sentiment analysis have evolved gradually starting from lexicon based approach, traditional machine learning algorithms and simple text vectorization techniques like Bag-of-Words and TF-IDF to complex neural networks and language models. Table 1 displays performances of the existing techniques vis-à-vis our technique on the Twitter US Airline Sentiment data.

Table 1: Comparison of performances of existing SotA techniques with our technique

Model used	**Accuracy**
Random Forest Classifier (Sinha & Sharma, 2020)	76%
Bag of Words + Support Vector Machine/ Logistic Regression (Tusar & Islam, 2021)	77%
Capsule Network (Demotte et al., 2021)	82.04%
LSTM-AT (Xiang et al., 2020)	82.3%
Word2vec+ RNN (two classes only) (Dang et al., 2020)	90%
RoBERTa with BiGRU layers (Talaat, 2023)	86%
Our Research - Fine-tuned Sentence Transformer + LGBMC	**86.5**%

3 Methodology

3.1 Data

In this study, the dataset used is a very popular one, taken from Kaggle's Twitter US Airline Sentiment which consists of scraped tweets from 2015 Twitter for 6 major airlines operating in US with 'positive', 'negative' and 'neutral' labels. This dataset was originally released by CrowdFlower in 2015 and comprises of 14,640 tweets spread across three sentiment classes, positive, negative, and neutral and was labelled manually. However,

Table 2: Data distribution for Twitter US airline dataset

#Records	#Positive Tweets	#Neutral Tweets	#Negative Tweets
14,640	2363 (16%)	3099 (21%)	9178 (63%)

this dataset is imbalanced as evident from table 2 with maximum records having negative sentiment. The tweets in the dataset belong to six American airlines which are American, Delta, Southwest, United, US Airways, and Virgin America. There are 15 columns in the dataset, however, for this study, only the text and airline sentiment columns have been used because the research focuses solely on textual data and its corresponding sentiment label. This is to ensure that the model only ingests the relevant data, avoiding the inclusion of any unrelated information.

3.2 Design Approach

- Transformer Architecture: A sentence transformer was created from a pretrained transformer model (mpnet-base) by performing pooling on the token embeddings. This novel concept of generating sentence embeddings using siamese and triplet network structures was introduced in Reimers & Gurevych (2019).
- Fine Tuning: The Sentence Transformer was further finetuned on the training dataset using applicable loss functions.
- Text Vectorization: Converted the field 'text' of the training data to numerical representation using sentence transformer embeddings.
- Model Building and Prediction: Embeddings were used as attributes to create a hybrid machine learning model based on gradient boosting based machine learning algorithms. The hybrid model predicted the sentiment on unseen text.

Figure 1 describes the entire process flow briefly.

3.3 Proposed Model Architecture and Methodology

3.3.1 Underlying Sentence Transformer Model

To begin with, the underlying base transformer model that has been used is MPNet as it combines the strength of masked language modeling adopted in BERT and permuted language modeling adopted in XLNet (Yang et al., 2019). This MPNet model was proposed in Song et al. (2020) where the authors identified drawbacks for both BERT and XLNet. Though BERT has been enormously successful for the most common natural language processing tasks like sentiment classification and named entity recognition, yet it overlooks the dependency among predicted tokens during pre-training. XLNet introduced permuted language modeling to address this issue but suffers from position discrepancy between pre-training and fine-tuning. MPNet not only leverages the dependency among predicted tokens like XLNet but it also takes auxiliary position information as input, allowing the model to see the full sentence and reducing position discrepancy. By combining these features, MPNet delivers superior performance when compared to its peers like BERT, XLNet and RoBERTa.

As a text is passed through this transformer model, the model generates contextualized embeddings for each of the tokens of the text. Next, the embeddings are passed through a mean pooling layer to get a fixed length embedding for each of the input texts. Mean pooling averages the token embeddings generated by the transformer model to create the sentence embeddings which are nothing but compressed information along a sequence of token embeddings with lower level of granularity. This creates a sentence transformer model which is now capable of converting each input text to an embedding of fixed dimension. Since each input text, irrespective of its length now gets converted to an embedding of a fixed dimension, finding proximity among

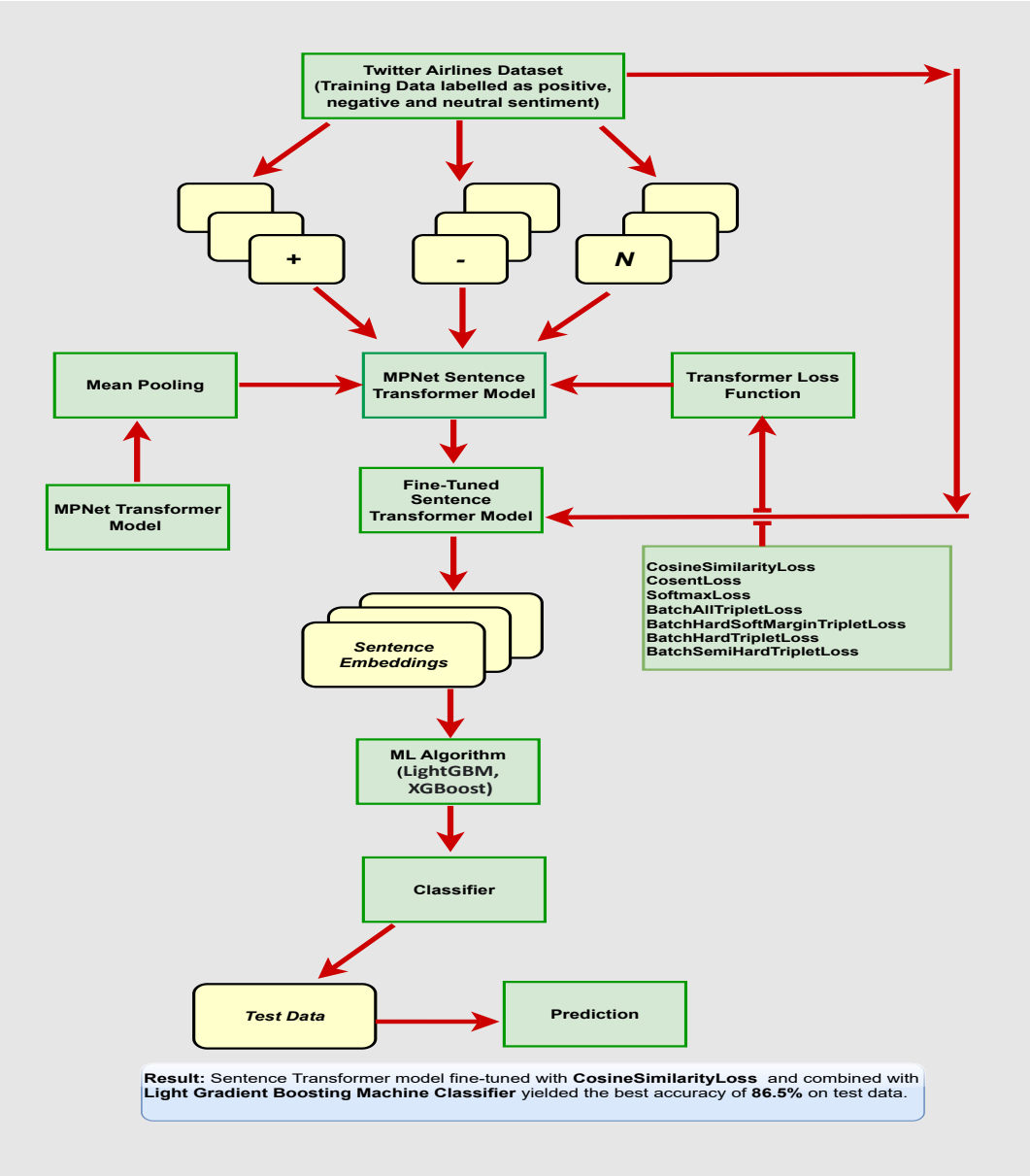

Figure 1: Process flow diagram

those embeddings is now computationally much less intensive and achieves similar results when compared to transformer models like BERT or RoBERTa.

Figure 2 depicts the creation of sentence transformer architecture briefly.

3.3.2 Transformer Loss Functions

Now the above sentence transformer model will be fine-tuned on the training dataset with the help of suitable loss functions for an enhanced performance on the target task. Selection of appropriate loss function is critical towards model's performance. Unfortunately, there is no fixed recipe to determine the most fitting loss function. However, it largely depends on the structure of the data and the target task. Since the dataset used here is labelled, the loss functions relevant for supervised learning are discussed here. Details of all the loss functions can be found in Reimers & Gurevych. Principally, the objective of the loss function is to

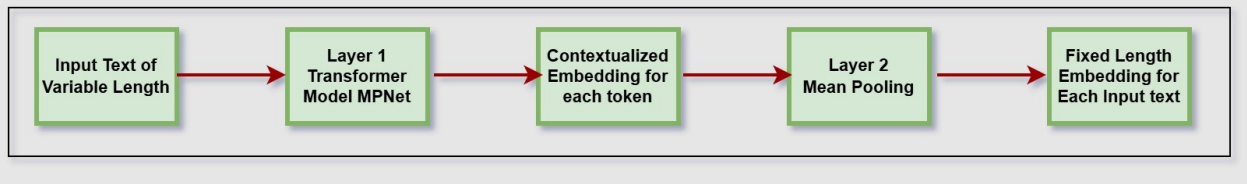

Figure 2: Sentence transformer creation process

optimize such that texts with same labels are as close as possible in the vector space whereas the texts with different labels lie farthest from each other. The loss functions that have been used in the architecture are as follows:

1. **CosineSimilarityLoss**
2. **Cosine Sentence Loss (CosentLoss)**
3. **SoftmaxLoss**
4. **BatchAllTripletLoss**
5. **BatchHardSoftMarginTripletLoss**
6. **BatchHardTripletLoss**
7. **BatchSemiHardTripletLoss**

CosineSimilarityLoss: In order to finetune a sentence transformer model on CosineSimilarityLoss, the model needs to be trained that a pair of texts from the training data has a predefined degree of cosine similarity. Therefore, each training example should consist of a pair of texts from the data along with a label indicating their similarity score that allows the model to understand how similar the two texts are. So, the next step involves converting the dataset into a format that can be ingested by the sentence transformer model. The model cannot accept raw text. Hence each example must be converted to a sentence_transformers.InputExample class and then to a torch.utils.data.DataLoader class to train the model.

Below is a toy example for demonstration purpose showing how to fine tune the model on cosine similarity loss.

```
train_examples = [
    InputExample(texts=[ postext1, postext2], input_label=1),
    InputExample(texts=[ postext1, negtext1], input_label =0),
    InputExample(texts=[ postext1, neutext1], input_label =0.5),
]
```

Here the model is being taught that the similarity score between the texts in the first pair is 1 since they belong to the same class of positive sentiments. Similarity score between the texts in the second pair is set to 0 so that the model understands that a positive sentiment text has no similarity with a negative sentiment text. Finally, the third pair has the input label set to 0.5 to inform the model that a positive sentiment text is somehow related to a neutral sentiment text but not closely enough. Then the model encodes each pair of texts, using the underlying sentence transformer model into fixed length embeddings, say $vect_1$ and $vect_2$. Then it computes the cosine similarity between the two, say sim($vect_1$, $vect_2$). Subsequently, it uses mean squared error (MSE) as the loss function to compare sim($vect_1$, $vect_2$) with the input label and while minimizing MSE, it aims to minimize the L2 norm of the error given as:

$$Loss = \|input\_label - sim(vect_1, vect_2)\|_2 \tag{1}$$

Here input label is the target similarity score in the range [0,1] for each pair of sentences. So similar pair of sentences belonging to the same class should be set for a similarity score closer to 1 whereas dissimilar pair of sentences belonging to different classes should be set for a lower similarity score. One interesting aspect here is although the cosine similarity ranges between [-1,1], the label or the target similarity score has been deliberately set in the range [0,1] since for every normalized vector (p), there exists exactly one normalized vector (q) such that the cosine similarity between (p) and (q) is -1. However, there are infinite vectors for which the cosine similarity between (p) and each of those vectors is 0. Consequently, using a loss function that only becomes 0 when the cosine similarity between two vectors is -1 is undesirable.

Cosine Sentence Loss (CoSENT): CoSENT loss function, introduced in Jianlin (2022) expects each of the input examples consists of a pair of texts and a target label, representing the expected pairwise similarity score between the texts in the pair. The generic loss function for CoSENT loss is defined as

$$Loss = log(1 + \sum_{sim(i,j)>sim(k,l)} e^{\lambda(s(k,l)-s(i,j))}) \tag{2}$$

Where (i,j) and (k,l) are any random input pairs of texts from the training data set such that pairwise cosine similarity between(i,j) , say s(i,j) is always greater than the pairwise cosine similarity between(k,l), say s(k,l). The summation is over all possible pairs of input pairs in the batch that match this condition and hence this loss function is also applicable for multi-class classification problems as long as there is an ordinal relation among the different classes. As evident, in order to minimize the loss, the expression $\sum_{sim(i,j)>sim(k,l)} e^{\lambda(s(k,l)-s(i,j))}$ needs to be minimized which can be achieved only by pushing s(i,j) up and pushing s(k,l) down. Here λ is a hyperparameter for scaling the pairwise cosine similarity scores.

Below is a toy example for demonstration purpose showing how to fine tune the model on cosent loss.

```
train_examples = [
    InputExample(texts=[ postext1, postext2], input_label=1),
    InputExample(texts=[ postext1, negtext1], input_label =-1),
    InputExample(texts=[ postext1, neutext1], input_label =0),
]
```

Here the model is being taught that the expected pairwise similarity score between the texts in the first pair is 1 since they belong to the same class of positive sentiments. Expected pairwise similarity score between the texts in the second pair is set to -1 so that the model understands that a positive sentiment text has opposite meaning with respect to a negative sentiment text. Finally, the third pair has the input label set to 0 to inform the model that a positive sentiment text is not related to a neutral sentiment text and their embeddings are orthogonal to each other. Unlike CosineSimilarityLoss, here the target similarity score is in the range [-1,1] for each pair of sentences since pushing the similarity score towards 1 for the pair with similar texts (positive and positive) and towards -1 for the pair with dissimilar texts (positive and negative) will ensure minimum loss.

SoftmaxLoss:This loss was introduced by the authors in Reimers & Gurevych (2019). For each pair of texts under consideration, it concatenates the corresponding sentence embeddings, say $vect_a$ and $vect_b$ with the element-wise difference $|vect_a - vect_b|$ and multiplies it with the trainable weight $W_t \in R^{3n*k}$. Then it adds a softmax classifier to it. So, the final output is given by:

$$Output = softmax(W_t(vect_a, vect_b, |vect_a - vect_b|)) \tag{3}$$

Here n denotes the dimension of the sentence embeddings, k denotes the number of labels or classes and the loss optimized is the cross-entropy loss.

Triplet Loss: This concept was introduced in Schroff et al. (2015). Here triplet refers to three entities – one anchor text, one positive text belonging to the same class of anchor text and one negative text belonging to a different class than that of the anchor text. Let's say the corresponding sentence embeddings are a, p and n respectively. The objective of the loss function is to push down the distance between a and p, say d(a,p) towards 0 and simultaneously push up the distance between a and n, say d(a,n) such that $d(a, n) > d(a, p)+\alpha$

where α is a hyperparameter. Hence the loss function can be defined as:

$$Loss = max(d(a,p) - d(a,n) + \alpha, 0) \tag{4}$$

As evident, $d(a,p) - d(a,n) + \alpha$ must always be negative to ensure we get 0 loss. The goal of the triplet loss is to make sure that:

- Two texts from the same class have their embeddings close together in the embedding space
- Two texts from different class have their embeddings far apart by some margin.

There are three types of triplets as follows:

- **Easy Triplets**: triplets with 0 loss, because $d(a,p) + \alpha < d(a,n)$
- **Hard Triplets**: triplets where the negative entity is closer to the anchor than the positive entity, i.e. $d(a,n) < d(a,p)$
- **Semi Hard Triplets**: triplets where the negative entity is not closer to the anchor than the positive entity, but have positive loss, i.e. $d(a,p) < d(a,n) < d(a,p) + \alpha$

α is more commonly referred as the margin. The distance function used here is the euclidean distance.

BatchAllTripletLoss tries to minimize the loss for all valid triplets from the training data.

BatchHardTripletLoss uses only the hardest positive and negative samples, rather than all possible, valid triplets from the training data. So, for each anchor, it gets the hardest positive and hardest negative to form a triplet.

BatchHardSoftMarginTripletLoss also uses only the hardest positive and negative samples from the training data. However, it does not require a margin.

BatchSemiHardTripletLoss uses only semi-hard triplets out of all valid triplets from the training data.

Figure 3 graphically demonstrates the differences among hard, semi-hard and easy triplets along the embedding space.

A triplet (a, p, n) is valid if a, p, n are distinct and label[a] = label[p] and label[a] $\neq$ label[n]. Here labels are represented as integers, where the same label corresponds to sentences from the same class. Additionally, the training dataset should include a minimum of two examples per label class.

3.3.3 ML Algorithms

For design experiment modeling, each of the fine-tuned models was used in combination with following popular ML algorithms for classification:

- **LightBGM (LBGM):** Authors in Ke et al. (2017) efficiently addresses the challenges posed by high feature dimension and large data size in Gradient Boosting Decision Tree by proposing LightGBM which can significantly improve efficiency and scalability. LightGBM stands for Light Gradient-Boosting Machine which is a fast, distributed, gradient boosting framework based on decision tree algorithm and is used for classification and other machine learning problems. It leverages gradient boosting to construct a strong learner by sequentially adding weak learners in a gradient descent manner. Unlike other boosting algorithms, LightGBM splits the tree leaf wise with the best fit i.e. it chooses the leaf that yields the largest decrease in loss, resulting in efficient tree construction. It also uses histogram-based decision tree learning algorithm which buckets continuous attribute values into discrete bins resulting in faster training and lower memory usage. Hence the word "Light" comes into play.

  LightGBM utilizes two novel techniques:

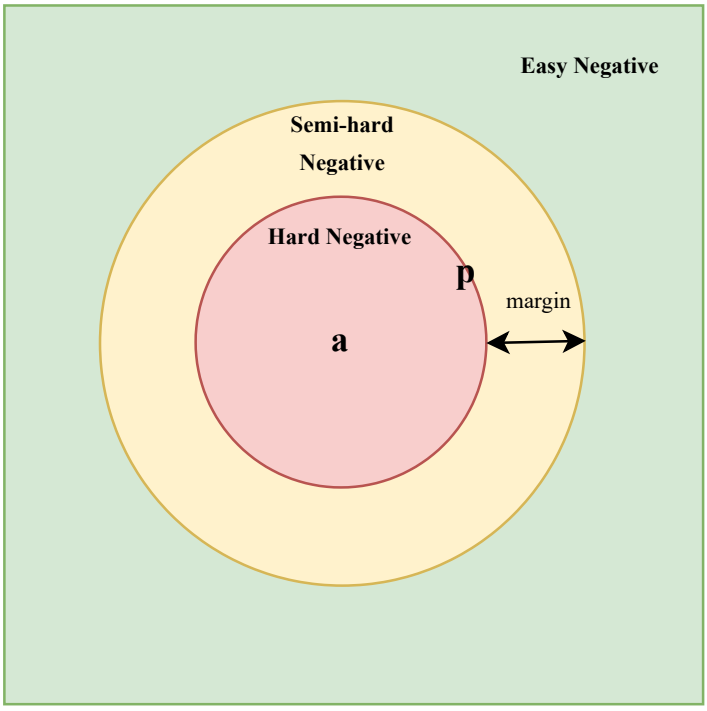

Figure 3: Triplets along embedding space.

Gradient-based One-Side Sampling (GOSS): This technique enables LightGBM to put emphasis on data instances with larger gradients, which are more significant for computing information gain, leading to accurate gain estimation with reduced data size.

Exclusive Feature Bundling (EFB): EFB bundles mutually exclusive features to reduce the number of features without significantly affecting the accuracy of split point determination, thus enhancing efficiency.

- **XGBoost:** XGBoost which stands for eXtreme Gradient Boosting, is a powerful machine learning tool known for its efficiency, speed, and accuracy. This technique was introduced in Chen & Guestrin (2016). Like LGBM, XGBoost is based on based on decision tree algorithms and leverages gradient boosting to construct a strong learner by sequentially adding weak learners in a gradient descent manner. The paper proposes a novel sparsity-aware algorithm for sparse data. It also introduces a weighted quantile sketch for approximate tree learning. Unlike LGBM, XGBoost builds trees level-wise (depth-wise), expanding the tree layer by layer, adding new models to correct errors made by previous ones, which leads to higher memory usage compared to LGBM.

4 Results

4.1 Model Evaluation Parameters

The training data for the model comprises of 80% of the original data and the model predicts on the test data comprising of the remaining 20%.

Performance evaluation parameters used in this study are Precision, Recall, F1 and Accuracy. Precision, Recall and F1 have been calculated for each class. Hence True Positive, True Negative, False Positive and False Negative also need to be understood with context to each individual class. For example, when calculating the metrics for positive sentiment class, any review having label other than positive sentiment will be considered as a negative instance. Similarly, when calculating the metrics for negative sentiment class, any review having label other than negative sentiment will be considered as a negative instance and so on.

True Positive(TP): The number of instances where the classification system correctly predicts a positive sentiment as a positive one or a negative sentiment as a negative one or a neutral sentiment as a neutral one.

True Negative(TN): The number of instances correctly predicted as negative instances.

False Positive(FP): The number of instances incorrectly predicted as positive instances.

False Negative(FN): The number of instances incorrectly predicted as negative instances.

Precision : Precision is defined as the ratio of correctly classified positive instances to the total number of samples predicted as positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

Recall: Recall, also known as sensitivity measures how many of the actual positive instances were correctly predicted. This metric is particularly important in determining how well the model is predicting the minority class in an imbalanced dataset

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

F1 score: F1 score is the harmonic mean of Recall and Precision, combining both metrics into a single value. It is the most used metric after accuracy. It balances precision and recall. F1 score manages the trade-off between recall and precision.

$$\text{F1} = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{7}$$

Accuracy: It is the ratio of correct classifications to total predictions given by the model. Accuracy is a good metric to use for sentiment classification when the sentiment classes are balanced.

$$\text{Accuracy} = \frac{CorrectPredictions}{TotalPredictions} \tag{8}$$

Table 3 provides a detailed breakdown of the results. As evident from Table 3, the base sentence transformer model in conjunction with XGBoost does a pretty decent job with an overall accuracy of 83.5% and high precison, recall and F1 scores for the negative sentiment class. This indicates that the sentence embeddings generated by the base sentence transformer model are semantically rich. However for the minority classes of positive and neutral sentiments, the performance metrics, specially recall and F1 scores are not that promising. Subsequently, sentence transformer model fine-tuned on CosineSimilarityLoss in conjunction with LGBM classifier achieves the highest accuracy of 86.5%. It is also observed that finetuning the sentence transformer model on CosineSimilarityLoss provides a lift of 3 percentage points in terms of accuracy, a lift of > 10 percentage points in terms of recall and a lift of > 6 percentage points in terms of F1 score for the minority classes (positive and neutral sentiment classes) when compared to the performance of base sentence transformer model. This demonstrates the contribution of the fine-tuning process for sentence transformer models in enhancing the models' classification performance, particularly for the minority classes. The performance metrics for the negative sentiment class are always better compared to the other two classes as it is the majority class in the dataset. However, the minority classes also consistently achieve acceptable precision, recall and F1 scores with the fined-tuned sentence transformer models without the need of any data augmentation.

Figure 4 displays the normalized confusion matrix for the finetuned model with the highest accuracy, thus indicating the recall rates diagonally in blue boxes for each of the classes.

Table 3: Results summary

Model	**Accuracy**	**Negative**			**Neutral**			**Positive**		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Base Model + LBGM	82.9%	86.8%	93.0%	89.8%	68.3%	61.9%	65.0%	84.4%	70.8%	77.0%
Base Model + XGB	83.5%	86.8%	94.0%	90.2%	70.7%	61.6%	65.9%	84.2%	71.0%	77.0%
FT Model on CosineSimilarityLoss + LBGM	**86.5%**	92.0%	91.5%	91.7%	72.0%	74.4%	73.2%	84.9%	82.9%	83.9%
FT Model on CosineSimilarityLoss + XGB	86.2%	91.4%	91.8%	91.6%	72.0%	72.1%	72.0%	84.5%	83.0%	83.8%
FT Model on CoSentLoss + LBGM	85.4%	91.7%	90.7%	91.2%	68.7%	72.9%	70.7%	84.2%	81.2%	82.7%
FT Model on CoSentLoss + XGB	85.6%	91.4%	91.0%	91.2%	69.7%	72.3%	70.9%	84.5%	81.8%	83.1%
FT Model on SoftmaxLoss + LBGM	86.1%	90.3%	92.6%	91.4%	73.2%	71.7%	72.4%	86.1%	79.9%	82.9%
FT Model on SoftmaxLoss + XGB	86.3%	90.5%	92.6%	91.6%	73.9%	72.1%	73.0%	85.6%	80.3%	82.9%
FT Model on BatchAllTripletLoss + LBGM	85.4%	90.4%	92.2%	91.3%	69.9%	71.1%	70.5%	86.4%	77.8%	81.9%
FT Model on BatchAllTripletLoss + XGB	85.1%	90.2%	92.1%	91.2%	69.6%	71.0%	70.3%	86.0%	76.7%	81.1%
FT Model on BatchHardSoftMarginTriplet-Loss + LBGM	85.7%	90.3%	92.9%	91.6%	70.7%	71.0%	70.9%	87.5%	77.2%	82.0%
FT Model on BatchHardSoftMarginTriplet-Loss + XGB	85.7%	90.0%	93.0%	91.5%	71.2%	71.5%	71.3%	87.8%	76.1%	81.5%
FT Model on BatchHardTripletLoss + LBGM	85.3%	88.8%	92.9%	90.8%	72.3%	68.2%	70.2%	87.7%	78.4%	82.80%
FT Model on BatchHardTripletLoss + XGB	85.4%	88.9%	93.1%	91.0%	72.4%	68.2%	70.3%	87.2%	77.8%	82.2%
FT Model on BatchSemiHardTripletLoss + LBGM	85.6%	90.0%	92.6%	91.3%	70.7%	70.8%	70.8%	87.4%	77.8%	82.3%
FT Model on BatchSemiHardTripletLoss + XGB	85.8%	90.0%	92.8%	91.4%	71.5%	70.8%	71.2%	87.4%	78.0%	82.5%

5 Broader Impact Statement

Sentiment analysis is a powerful technique in artificial intelligence that has important business applications. For example, organizations can use sentiment analysis to gauge public opinion about their products and services. E-commerce companies can monitor the overall sentiment of reviews of their different products to identify any suspicious vendors that use false advertising or scamming tactics. This research can help businesses develop inexpensive, easy to implement models for sentiment analysis with state-of-the-art performance. Below are some perceived advantages of our approach over existing techniques for sentiment classification.

- **No text cleaning**

  One of the key challenges faced by existing methods using the conventional text vectorization methods like Bag-of-Words and TF-IDF is the need for extensive text preprocessing (such as removing stopwords, special characters, and emojis). This not only makes the entire process computationally more expensive but also leads to loss of valuable information. On the other hand, sentence transformers are designed to handle raw text effectively. They learn contextual embeddings directly from the input text. Unlike the traditional text vectorization methods, Sentence Transformers capture semantic meaning, making them robust to minor noise (e.g., typos, punctuation, and special characters).

- **Lesser computational cost**

  Techniques leveraging transformer architectures like BERT may provide comparable performances for sentiment classification but they usually come with a high computational price tag. Sentence

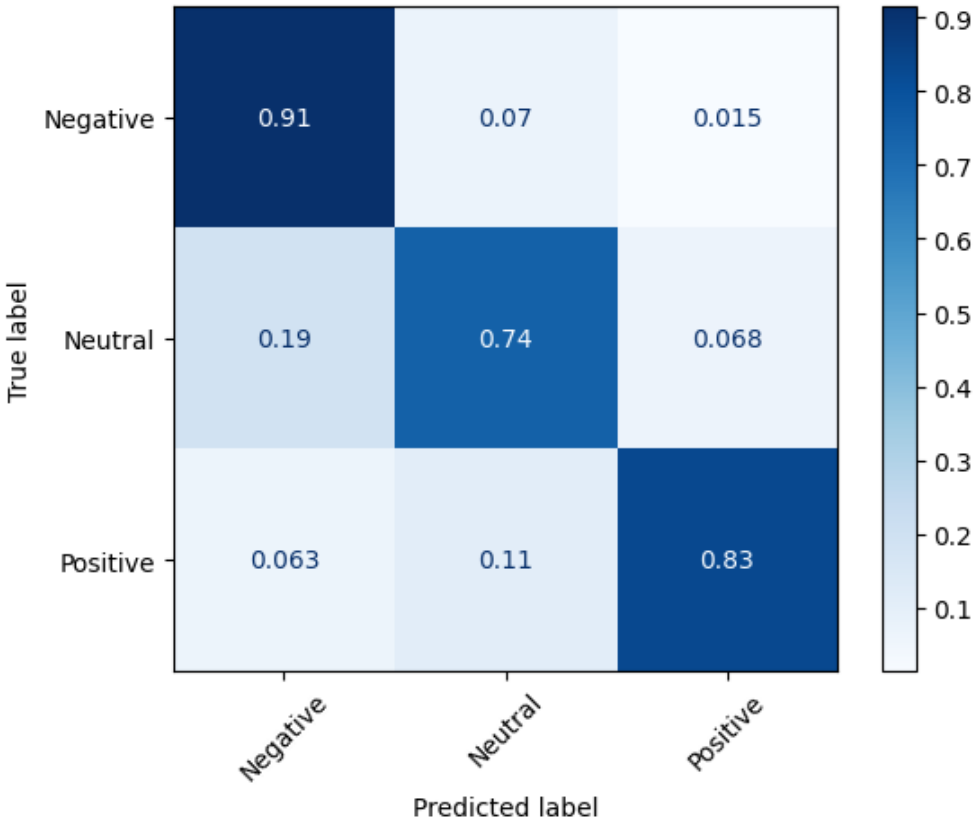

Figure 4: Normalized confusion matrix

transformers directly produce embeddings for entire sentences, enabling them to handle variable-length input (sentences) without increasing computational complexity which is why their processing is more efficient than BERT's token-by-token processing. As a result while using pre-trained models, sentence transformers are generally less computationally expensive than BERT.

- **Scalability**

  Sentence transformers are designed to handle large datasets efficiently and can generate embeddings without compromising performance. This scalability is crucial for businesses experiencing rapid growth or dealing with ever-expanding text corpora. Scalability also extends to fine-tuning sentence transformers as pre-trained models can be fine-tuned on domain-specific data without significant computational overhead. This adaptability will allow the users to tailor sentence embeddings to specific tasks while maintaining scalability.

The future plan is to extend this study further to incorporate text summarization and topic modeling for the review texts whose sentiments will be predicted by the model so as to identify the major keywords responsible for such predicted sentiments, especially for the negatively predicted sentiments. This will enable organizations to recognize their own shortcomings on a real time basis just by ingesting the customer reviews.

References

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Advances in Neural Information Processing Systems*, 30:3146–3154, 2016. URL `https://arxiv.org/abs/1603.02754`.

Nhan Cach Dang, María N. Moreno-García, and Fernando De la Prieta. Sentiment analysis based on deep learning: A comparative study. *Electronics*, 9(3):483, 2020.

P. Demotte, K. Wijegunarathna, D. Meedeniya, and I. Perera. Enhanced sentiment extraction architecture for social media content analysis using capsule networks. *Multimedia Tools and Applications*, 82:8665–8690, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Su Jianlin. Cosent (1): A more effective sentence vector scheme than sentence-bert, Jan 2022. URL https://kexue.fm/archives/8847.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30:3146–3154, 2017. URL http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pp. 79–86, Philadelphia, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118704. URL https://aclanthology.org/W02-1011.

Nils Reimers and Iryna Gurevych. Sentence transformer losses. https://sbert.net/docs/package_reference/sentence_transformer/losses.html [Accessed: (17th September 2024)].

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*, 2015.

Avishek Sinha and Pooja Sharma. Comparative analysis of machine learning classifiers on us airline twitter dataset. *International Research Journal of Engineering and Technology (IRJET)*, 7(8):798–803, 2020.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*, 2020.

Amira Samy Talaat. Sentiment analysis classification system using hybrid bert models. *Journal of Big Data*, 10(1):110, 2023.

Md. Taufiqul Haque Khan Tusar and Md. Touhidul Islam. A comparative study of sentiment analysis using nlp and different machine learning techniques on us airline twitter data. *arXiv preprint arXiv:2110.00859*, 2021.

Rong Xiang, Yunfei Long, Mingyu Wan, Jinghang Gu, Qin Lu, and Chu-Ren Huang. Affection driven neural networks for sentiment analysis. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 112–119, Marseille, France, May 2020. European Language Resources Association. URL https://aclanthology.org/2020.lrec-1.14.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.