"Something Something Hota Hai!" An Explainable Approach towards Sentiment Analysis on Indian Code-Mixed Data

Aman Priyanshu* [1], **Sudarshan Sivakumar*** [2], **Supriti Vijay*** [2], **Nipuna Chhabra*** [2], **Aleti Vardhan*** [2]
[1]Department of Information and Communication Technology
[2]Department of Computer Science and Engineering
Manipal Institute of Technology,
Manipal Academy of Higher Education,
Manipal, India-576104.
{aman.priyanshu, sudarshan.sivakumar, supriti.vijay,
nipuna.chhabra, aleti.vardhan}@learner.manipal.edu

Abstract

The increasing use of social media sites in countries like India has given rise to large volumes of code-mixed data. Sentiment analysis of this data can provide integral insights into people's perspectives and opinions. Code-mixed data is often noisy in nature due to multiple spellings for the same word, lack of definite order of words in a sentence, and random abbreviations. Thus, working with code-mixed data is more challenging than monolingual data. Interpreting a model's predictions allows us to determine the robustness of the model against different forms of noise. In this paper, we propose a methodology to integrate explainable approaches into code-mixed sentiment analysis. By interpreting the predictions of sentiment analysis models we evaluate how well the model is able to adapt to the implicit noises present in code-mixed data.

1 Introduction

In the Hindi movie Jab We Met, the lead actress Kareena Kapoor has a famous line in which she says, "Main apni favourite hoon," which translates to "I am my favourite." While this dialogue remains iconic in Indian pop culture even 14 years after the film's release, it also represents the phenomenon of code-mixing that is very common in multilingual countries like India. Code-Mixing is a phenomenon of embedding linguistic units such as phrases or words of one language into another language (Poplack and Walker, 2003). English words, for instance, are often mixed with Hindi sentences to form what is colloquially known as Hinglish. Over the past decade, social media sites like Twitter and Facebook have seen an exponential rise in users from countries like India. This has led to an accumulation of a large volume of code-mixed data on these platforms. Therefore, analysing and processing this data has become imperative to provide users with the best possible experience on these sites. An important technique used to analyse such large amounts of data is sentiment analysis, which labels a given text as positive, negative, or neutral. Its applications include product management, customer satisfaction review, hate speech detection and censoring of abusive and derogatory speech.

The rise in the use of deep learning models for sentiment analysis has led to more accurate predictions (Peters et al., 2018; Vaswani et al., 2017; Wang et al., 2018).These algorithms behave like black boxes, making it difficult for practitioners to understand model predictions. Explainable AI provides a set of methods to help us understand and interpret a model's decisions. Specifically for the task of sentiment analysis of textual data, these techniques allow us to understand how a word or phrase influences the sentiment of the text.

This paper discusses the implications and importance of model-agnostic explainable methods for code-mixed data. We observe that due to the unavailability of NLP tools for Hindi-English Code-Mixed text and the noisy nature of such data, several popular methods for Sentiment Analysis are not applicable (Prabhu et al., 2016). Therefore we first conduct a survey of models on the SAIL code-mixed dataset (Sarkar, 2018) and interpret their decisions with LIME and SHAP.

2 Related Work

2.1 Sentiment Analysis of Code-Mixed Data

While sentiment analysis has been extensively studied for English text (Joshi et al., 2010; Socher et al., 2013; Zhang et al., 2018), it is only recently that its applications to code-mixed data have been explored. Traditional machine learning approaches such as Support Vector Machines, logistic regression, and random forests have shown a reasonable level of accuracy in this task(Mishra et al., 2018).

* All authors have contributed equally to the work.

Even shallow parsing (Sharma et al., 2016) and word-level identification of code-mixed data (Chittaranjan et al., 2014) have proven to boost classification performance.

However, deep learning models such as LSTMs and CNNs perform comparatively better as they efficiently process sequential data.(Niu et al., 2018; Wang et al., 2016; Kim, 2014). (Kumar et al., 2018) used phonemic sub-word units with a hierarchical Bi-directional LSTM (BiLSTM) model to detect sentiment in Hi-En code-mixed texts. (Choudhary et al., 2018) employs Bidirectional LSTM networks with shared parameters to capture a sentiment-based representation of the sentences. On the other hand, (Kumar et al., 2020) presents an ensemble of both CNN and self-attention based LSTM, using XLM-R embeddings (Conneau et al., 2020).

Sub-word and bi-gram based models also perform well due to their ability to capture dialect variations in Hindi and the inconsistencies of code-mixing (Prabhu et al., 2016). However, due to the noisy nature of the dataset, models still perform poorly. (Ghosh et al., 2017) uses extensive pre-processing to remove noise from raw text, allowing better performance to be achieved.

2.2 Explainable AI approaches to sentiment analysis

In order to trust a model's predictions, one must be able to interpret the reasons behind its decisions. Several attempts have been made to infer the text classification and sentiment analysis predictions of language models. State of the art model-agnostic explanations like LIME (Local Interpretable Model-Agnostic Explanations) (Ribeiro et al., 2016) and SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017) enable better visualisation and analysis of AI models.

(Kokalj et al., 2021) adapts SHAP to transformer models such as BERT-based classifiers. (Chen et al., 2020) proposes a model-agnostic approach named HEDGE to build hierarchical explanations by detecting the interactions between features of a model. One of the drawbacks of these model-agnostic techniques is the computational complexity and the processing time required to execute them. (Bodria et al., 2020) addresses this by using attention-based techniques to extract meaningful sentiment scores with a lower computational cost than existing XAI methods.

3 Preliminaries

Various techniques have been used to achieve accurate results in sentiment analysis. A simplistic and computationally inexpensive approach is the random forest. A random forest is generated by bagging multiple uncorrelated decision trees but may fail to capture context and complexities. We look at Convolutional Neural Networks and Long Short Term Memory(LSTM) to overcome this problem. Word embeddings, which are numeric representations for each word within a given sentence, are used to enable contextual learning over these networks.

Convolutional Neural Networks are a class of Neural networks commonly used in computer vision. They also show promising results in NLP tasks such as Sentiment Analysis. When applied to the input embedding matrix, the convolution operation captures the local features and relationships between the words of the sentence, outperforming feed-forward neural networks and machine learning algorithms such as Random Forests (Kim, 2014).

Although CNNs are great for capturing local features, they lose information about the order of words in a given sentence. The loss of sequential information significantly hampers performance, and recurrent neural networks such as LSTMs and GRUs are better suited for this. **LSTMs** are designed to infer sequential data, making them some of the most viable neural network architectures for sentiment analysis. Their ability to deal with vanishing gradients and their relative insensitivity to gap length gives them an advantage over other sequential models.

However, sentence structure goes beyond just sequential comprehension, and therefore a structured model that would allow word association becomes necessary for performance improvement (Hochreiter and Schmidhuber, 1997).

Bidirectional Encoder Representations from Transformers (BERT) utilizes the bidirectional training of Transformers, a popular attention model for learning. The architecture is built for the objective of masked language modelling and achieves considerably high performance compared to other word embedding techniques. The masked language model enables a fusion of both left-to-right and right-to-left contexts and makes the Transformer bidirectional. BERT also achieves Next Sentence Prediction (NSP), where the model receives a pair

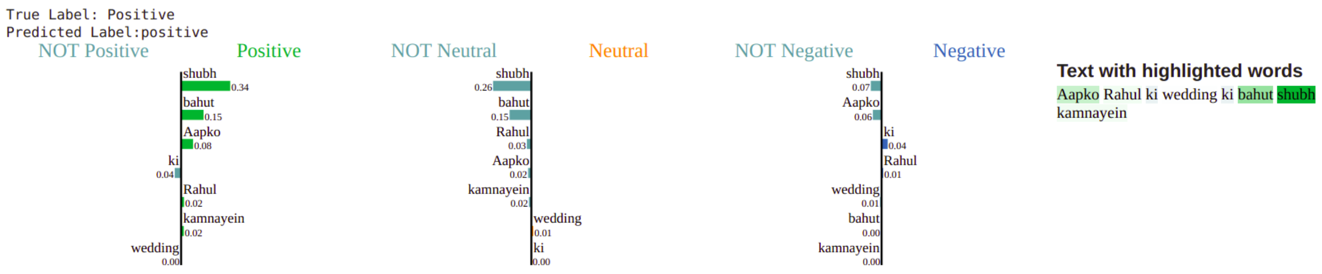

Figure 1: LIME gives numeric explainations to each and every word in a given sentence. This allows one to infer how important a word is to the classifier function, allowing the user to understand the predicted sentiment better.

of sentences as inputs and predicts whether the second sentence is in conjunction with the first (Devlin et al., 2018). XLM-RoBERTa, a variant of BERT is a transformer-based cross-lingual language model trained on 100 different languages (Conneau et al., 2020).

3.1 Explainable AI

In order to increase transparency and accountability in model architectures, we utilize model-agnostic methods, like LIME and SHAP.

LIME (Ribeiro et al., 2016) stands for Local Interpretable Model-Agnostic Explanations (LIME). Local here refers to examining the model's behaviour around an instance of the dataset rather than the entire dataset. The explanations are in the form of numerical measurements of significant features of the input data. LIME is also model-agnostic, which means that it can be applied to any machine learning model. LIME works based on input perturbations and their respective changes on the model output.

As LIME works on the assumption that every complex model is linear on a local scale, it creates a simple linear model to approximate how the global model behaves at a particular local data point. A proximity-based weighted set of perturbed inputs is used to train the local interpretable model by learning their associated outputs from the target or global model. The outcomes of the locally linear model give us illustratable numeric values to interpret model decisions.

SHAP or SHapley Additive exPlanations (Lundberg and Lee, 2017) leverages the idea of Shapley values of game theory for model feature influence scoring. These values can be defined as the average marginal contribution of a feature value over all possible coalitions. Unlike LIME, SHAP can be used to interpret feature dependency on the entire model. This methodology can summarize model dependency metrics, allowing us to better leverage interpretation for the entire model and the dataset. Thus, it provides us with better global explanations than local explanations.

4 Experimental Results

4.1 Dataset

The SAIL 2017 dataset (Sarkar, 2018) is a sentiment classification dataset on code-mixed text, with each instance labelled as positive, negative, or neutral in sentiment. This data was collected using the Twitter4j10 API to extract Hindi-English code-mixed data from Twitter. Common Hindi words were collected in Romanized format and then searched using the above API. The data was then annotated manually. It consists of a total of 12,601 instances of text, with corresponding sentiment labels. The dataset is pre-split into three subsets: training set, validation set, and test set; their distribution has been presented in Table 1. There are a total of 199071 words across all the instances with 99965 Hindi words and 99106 English words. As seen below, the dataset is relatively balanced, making accuracy an appropriate evaluation metric.

	Positive	**Neutral**	**Negative**	**Total**
Train	3202	4558	2319	10079
Val	399	578	283	1260
Test	385	586	290	1261

Table 1: The Table describes the data distribution for the SAIL 2017 dataset.

We provide some examples from the dataset below along with the sentiment of the examples

- *jabardust post hai buddy.*
  Translation : it's an amazing post, buddy
  Sentiment : Positive

Models	SVM	Random Forest	CNN	CNN Glove Embeddings	LSTM	LSTM Glove Embeddings	XLM-RoBERTa
Train Accuracy	87.46%	96.93%	53.17%	58.02%	61.00%	66.08%	85.30%
Test Accuracy	56.22%	54.16%	54.08%	56.98%	49.48%	59.09%	72.21%

Table 2: The Table details the model accuracy results upon training different machine learning models. We use this as a benchmarking reference to evaluate model utility, and choose to infer the model which performs the best, for further inspection using LIME and SHAP.

- *Aur puucha ke fourthg memberg kaha hai??*
  Translation : and asked where is the fourth member
  Sentiment : Neutral
- *bachpan me inn advertisement ne bahut confuse kiya hai*
  Translation : in my childhood, this advertisement confused me a lot
  Sentiment : Negative

Noise in code mixed data extracted from social media platforms includes abbreviations, variation in spellings for the same word, no standard grammar typos, emoji's etc. As shown in the examples, words like *puucha*, *me* etc don't have one standard spelling that is agreed upon by different users. *Puucha* for instance can also be written as *pucha* or *poocha*. On the other hand, typos refer to misspelled words like *fourthg*, *memberg* etc whose standard spellings would be *fourth* and *member* respectively.

4.2 Classification Results

In our venture to integrate explainable AI on code-mixed data, we evaluated multiple models for performance comparison. We provide the resultant performances of each of these in Table 2. Our comparison spanned tree-based random forest, SVMs, LSTMs, CNNs, and a RoBERTa model pre-trained on codemixed data. For the random forest and SVM model, we used a TF-IDF vectorizer as our word representation. While for the LSTM and CNN architecture, we compare randomly initialized word embeddings and pre-trained GloVe embeddings. These GloVe embeddings were pre-trained on the Twitter crawl, which consisted of about two billion tweets. As shown in Table 2, these embeddings improved LSTM and CNN performance significantly and allowed it to generalize better. Whereas the use of random forest gave us a highly overfitted classifier, as shown in Table 2.

Inferring the performance presented in Table 2, we can see that the XLM-RoBERTa model is the best performing model and acts as the appropriate candidate for further exploration using explainable AI. This can be attributed to the fact that XLM-RoBERTa was pretrained on code-mixed data and better captures the contextual meaning of words. As the model gives significantly higher performance than its other counterparts, we used LIME and SHAP to infer this classification aptitude.

4.3 Inferring LIME for Global Explainations

Negative	Neutral	Positive
laat (kick)	thk (ok)	shadi (wedding)
kutta (dog)	kuch (a bit)	mubarak (congrats)
theek (ok)	abhi (now)	karein (do)
karega (will do)	are (exclamation)	jai (hail)
sala (abuse)	mai (I)	maja (fun)
maar (to beat)	haan (yes)	lage (toward)
phle (first)	kaash (wish)	aage (ahead)
kyu (why)	ghar (home)	wah (wow)
bohot (very)	_	paise (money)
nahi (no)	agar (if)	ache (good)

Table 3: The Table describes globally significant words using LIME explainations. We mention their English translations alongside them.

LIME provides independent explanations for each instance of the data. However, to compare and evaluate it against SHAP, we compute the global significance of each feature using its local explanations. To better represent their data-wide relevance, we calculate the mean of each explanation weight provided by every word; this allows us to pull important words for model predictions. We specifically look at Hindi words and measure their importance in sentiment predictions. Here, Hindi words were sampled from the corpus if they did not belong to the Google 10,000 words vocabulary[1].

[1] https://github.com/first20hours/google-10000-english

This allows us to determine how well the model can learn from code-mixed sentences. We describe the computation in the following equation,

$$\sum_{word_i \in vocab} \sum_{sentence_i \in 0}^{n} \frac{(word\_weight)}{n} \quad (1)$$

We provide results from this computation in Table 3. As we can see, the table works as a great summarizing tool for each sentiment, capable of pulling offensive texts for negative samples and respectful or congratulatory words for positive sentences. For instance, the term "*mubarak*" is a Hindi word meaning congratulations and can be seen significantly contributing towards the positive sentiment. We use these new computations to provide for LIME's global explanations for various future experiments.

4.4 Inferring the XLM-RoBERTa model

This section discusses the results of both LIME and SHAP explainable approaches on the code-mixed data. We present a comparison between the two methodologies by evaluating them for both global and local explanations. As SHAP can define sample-based and global explanations, we used the statistical mean of word importance extracted by LIME to infer their global significance. This allows us to compare both models on a fundamental level, thus enabling us to evaluate better.

(Ancona et al., 2018) proposed an approach called sensitivity-n to test the gradient-based attribution methods in deep neural networks. However, one of the main drawbacks of this methodology is that two distinct explainable models can not be compared with each other due to their gradient-based attribution. Another metric that we explored was perturbations to compare different methodologies. RemOve And Retrain—ROAR is another monumental advancement in this area, which uses accuracy as a metric to estimate feature importance in deep neural networks (Hooker et al., 2019). It removes data points estimated to be most important and retrains it to measure the change in performance. This allows it to compare between different explainable models. However, as the SAIL dataset consists of only a limited number of samples, making the accuracy more discrete, we choose to evaluate it on the log-odds scores. While developing these metrics, we also construct an additional metric specific to code-mixed data.

We present a comparison between the two methodologies using three metrics, described by Mean Absolute Error of Log-Odds Scores on Deletion (MAELOSD) which is defined as,

$$= \sum_{j=0}^{M} \frac{(|log\_odds_{ini}^{(j)} - log\_odds_{fin}^{(j)}|)}{M} \quad (2)$$

Where ini refers to initial sentences without word deletions, fin refers to the final sentences after the deletion of polarizing words, and M is the total number of sentences. We define polarizing words as those which have been given the highest weights by the explanations of the LIME and SHAP models.

1. MAELOSD of Sentence-Interpreted Polarizing Words (*MAELOSD-Sentence*): We delete the n most polarizing words as returned by our explainable model from the sentence and recompute the final Log-Odds Scores. We then calculate the Mean Absolute Error (MAE) for all samples as shown in Equation 2. This process is repeated for different values of n.

2. MAELOSD of Model-Interpreted Polarizing Words (*MAELOSD-Model*): We repeat the exact computation with summarized weights for the entire vocabulary(English and Hindi). As discussed above, we computed the global significance of words extracted by LIME to evaluate on the same metric. Whereas we simply used SHAP to extract its global explanations for the entire corpus.

3. MAELOSD of Code-Mixed Words (*MAELOSD-CodeMixed*): This metric calculates sentence-wise MAE upon deletion of top n polarizing Hindi words. While in MAELOSD-Sentence we delete both English and Hindi words, here we delete only the Hindi words. We consider all words which are not part of the GloVe (Wikipedia 2014 and Gigaword 5 crawl) vocabulary as Hindi words. We choose this specific embedding as there is a low chance of finding Hindi words in the Wikipedia and Gigaword 5 crawl. For instance, the word *accha* does not appear in the GloVe vocabulary, and is considered a Hindi word.

With these metrics, we can extract the significant features for the two methodologies. We expect the

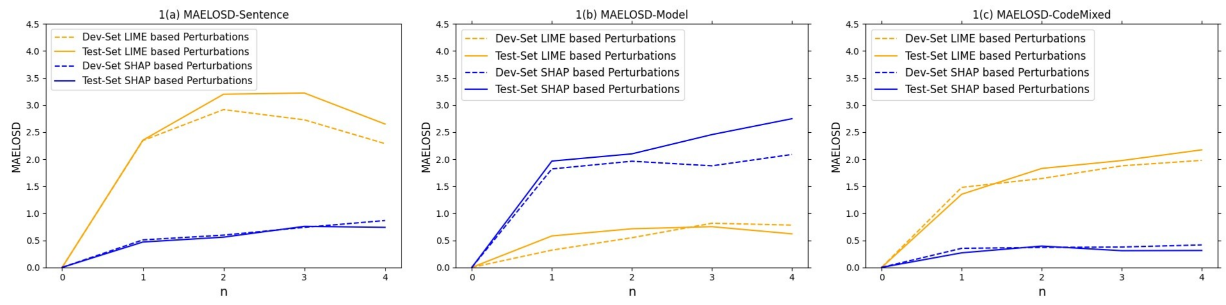

Figure 2: Metric MAELOSD for Sentence, Model, and Code-Mixed is presented in this figure.

mean absolute error to increase on the deletion of polarizing words since the model would make poor predictions.

The results of all three metrics are shown in Figure 2 where n on the x-axis refers to the number of words deleted. We can clearly observe from Figure 2(a) that LIME causes a higher variation in the MAELOSD-Sentence than SHAP. This is due to the local inferential nature of LIME, which does not incorporate the entire training corpus into its evaluation. The explainable advantage LIME displays over SHAP can be observed in the figure.

We also observe that MAELOSD-Model, which takes into account the whole dataset, returns expected results. LIME performs significantly worse due to its local nature. Furthermore, when we consider the average feature importance, certain words which may not be as significant to the local instance, are removed during metric evaluations. Figure 2(b) displays this clearly, where we can see SHAP having a better MAELOSD variation than LIME.

We can also see the impact of Hindi words from our results in Figure 2(c), where the error increases in proportion to the number of words deleted. We observe that the MAELOSD error in Figure 2(c) and 2(a), while different, follow a similar trajectory for $n \leq 3$. This establishes the significance of Hindi words in model prediction. SHAP's local explanations are interdependent on other sentences in the corpus, due to which it can perform poorly on data coming from a diverse vocabulary. As we can see SHAP's MAELOSD is close to zero for all values of n, which may imply that the Hinglish vocabulary is much more diverse than the English vocabulary. This could also be due to irregular spellings and the diverse vocabulary of code-mixed data.

An important observation one can make in Figure 2(a) is the drop in MAELOSD upon deleting $n \geq 3$ words. Although MAELOSD allows us to compare the methodologies, if a large subset of important features is removed, the remaining features may form a different distribution than the training data. Thus, it becomes unclear whether the degradation in MAELOSD comes from the distribution shift or due to the poor performance of the explainable model (Zhou et al., 2021).

Our results align with the application of both LIME and SHAP, demonstrating their local and global natures respectively. We can see the application and easy integration of model-agnostic interpretability pipelines on code-mixed data.

5 Conclusion and Future Work

Code-mixed data is an integral part of communication in multilingual communities and their culture. The application of state-of-the-art interpretable methods on this assortment of data will pave a path towards the adoption of the same during real-world implementation. Our use of LIME and SHAP, which quantify local and global model explanations, allows us to display their importance and relevance in sentiment analysis of code-mixed data. Our work demonstrates that locally sensitive explanations provided by LIME, give higher mean absolute error when the number of deletions increases. Here, essential deletions refers to the deletion of polarizing words which influence the classifier's output. We also observe the model's ability to pick up polarizing Hindi words, even in a code-mixed context. SHAP can be applied to produce high-quality global interpretations and extract model sensitivity. On the other hand, LIME can be used for locally sensitive explanations, allowing developers to focus on individual users' sentiments.

For future explorations, we would like to explore these techniques on more datasets and compare their performance. We also look to include other explainable methods and develop a specific metric to measure the global impact of Hindi words in a code-mixed corpus. We would also like to investigate other techniques such as lexical normalization to reduce noise and measure its impact on our predictions. The trends of biases in code-mixed data can also be observed through explainable methods.

We believe that our work serves as a valuable resource for code-mixed ML practitioners, developers, and researchers. The integration with explainable methods paves a new path towards future research and development.

Acknowledgements

The authors of the paper are grateful to the reviewers in reviewing the manuscript and their valuable inputs are appreciated. We would also like to thank the Research Society MIT for supporting the project.

References

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2018. Towards better understanding of gradient-based attribution methods for deep neural networks.

Francesco Bodria, A. Panisson, A. Perotti, and Simone Piaggesi. 2020. Explainability methods for natural language processing: Applications to sentiment analysis. In *SEBD*.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating hierarchical explanations on text classification via feature interaction detection.

Gokul Chittaranjan, Yogarshi Vyas, Kalika Bali, and Monojit Choudhury. 2014. Word-level language identification using crf: Code-switching shared task report of msr india system. In *Proceedings of The First Workshop on Computational Approaches to Code Switching*, pages 73–79.

Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018. Sentiment analysis of code-mixed languages leveraging resource rich languages.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Souvick Ghosh, Satanu Ghosh, and Dipankar Das. 2017. Sentiment identification in code-mixed social media text.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019. A benchmark for interpretability methods in deep neural networks.

Aditya Joshi, AR Balamurali, Pushpak Bhattacharyya, et al. 2010. A fall-back strategy for sentiment analysis in hindi: a case study. *Proceedings of the 8th ICON*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification.

Enja Kokalj, Blaž Škrlj, Nada Lavrač, Senja Pollak, and Marko Robnik-Šikonja. 2021. BERT meets shapley: Extending SHAP explanations to transformer-based classifiers. In *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*, pages 16–21, Online. Association for Computational Linguistics.

Ayush Kumar, Harsh Agarwal, Keshav Bansal, and Ashutosh Modi. 2020. Baksa at semeval-2020 task 9: Bolstering cnn with self-attention for sentiment analysis of code mixed text.

Upendra Kumar, Vishal Singh, Chris Andrew, Santhoshini Reddy, and Amitava Das. 2018. Consonant-vowel sequences as subword units for code-mixed languages. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions.

Pruthwik Mishra, Prathyusha Danda, and Pranav Dhakras. 2018. Code-mixed sentiment analysis using machine learning and neural network approaches.

Dejiao Niu, Zheng Xia, Yawen Liu, Tao Cai, Tianquan Liu, and Yongzhao Zhan. 2018. Alstm: adaptive lstm for durative sequential data. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 151–157. IEEE.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Shana Poplack and James A Walker. 2003. Pieter muysken, bilingual speech: a typology of code-mixing. cambridge: Cambridge university press, 2000. pp. xvi+ 306. *Journal of Linguistics*, 39(3):678–683.

Ameya Prabhu, Aditya Joshi, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier.

Kamal Sarkar. 2018. Ju_ks@sail_codemixed-2017: Sentiment analysis for indian code mixed social media texts.

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Srivastava, Radhika Mamidi, and Dipti M Sharma. 2016. Shallow parsing pipeline for hindi-english code-mixed social media text. *arXiv preprint arXiv:1604.03136*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jin Wang, Liang-Chih Yu, K Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional cnn-lstm model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 225–230.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Jianlong Zhou, Amir H. Gandomi, Fang Chen, and Andreas Holzinger. 2021. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5).