AdvCodeMix: Adversarial Attack on Code-Mixed Data

Sourya Dipta Das
dipta.juetce@gmail.com
Jadavpur University
Kolkata, West Bengal, India

Soumil Mandal
soumil.mandal@utdallas.edu
University of Texas at Dallas
Texas, USA

Ayan Basak
ayanbasak13@gmail.com
Jadavpur University
Kolkata, West Bengal, India

Dipankar Das
dipankar.das@jadavpuruniversity.in
Jadavpur University
Kolkata, West Bengal, India

ABSTRACT
Research on adversarial attacks are becoming widely popular in the recent years. One of the unexplored areas where prior research is lacking is the effect of adversarial attacks on code-mixed data. Therefore, in the present work, we have explained the first generalized framework on text perturbation to attack code-mixed classification models in a black-box setting. We rely on various perturbation techniques that preserve the semantic structures of the sentences and also obscure the attacks from the perception of a human user. The present methodology leverages the importance of a token to decide where to attack by employing various perturbation strategies. We test our strategies on various sentiment classification models trained on Bengali-English and Hindi-English code-mixed datasets, and reduce their F1-scores by nearly 51% and 53% respectively, which can be further reduced if a larger number of tokens are perturbed in a given sentence.

CCS CONCEPTS
• Computing methodologies → Artificial intelligence; Natural language processing; Natural language generation; Lexical semantics;

KEYWORDS
adversarial attack, natural language processing, deep learning , code-mixed data, text classification

1 INTRODUCTION
In the age of globalization, code-mixed text inputs are a common phenomenon since it is quite natural for bilinguals to switch back and forth between two languages while communicating, both in verbal and textual forms [17]. Such textual instances are challenging to process as they often combine grammatical bases of both the languages in a single sentence, make use of colloquial terms and short-forms extensively, and standardized rules aren’t followed while transliterating when the pair of languages have different scripts. We introduce a three-step attack strategy that can be used for generating adversarial examples using minimal resources for any type of code-mixed data (with and without transliteration). We have used our framework to evaluate the success of adversarial attacks on a few sentiment classification models [14, 15, 21] that have been diagnosed effective on code-mixed data. Research on adversarial techniques has become an important aspect, especially for security-critical applications, as it helps us in both analyzing the fallacies of the models, and make them more robust. Some of the popular methods towards building robust pipelines include adversarial training [8] of the model and rejection of adversarial inputs [11]. In our approach, we do not make any replacements in the original sentence based on word synonyms as observed in previous adversarial attack approaches (Jin et al. [4], Li et al. [6]), or any such triggers at the word or sentence level [19], which ensures that the semantic similarity is automatically preserved. We only make phonetic perturbations at the sub-word and word-level, and replace words with their corresponding transliterated counterpart in the code-mixed sentence, as it can deceive the model in several cases despite keeping the overall structure intact. The main contributions of our paper are (a) preparation of a generalized model agnostic framework for generating adversarial examples of code-mixed data, (b) proposed novel language specific perturbation techniques that preserve the semantic structure of a sentence, (c) achieve a successful attack in a short time span per sentence.

2 RELATED WORK
Sun et al. [19] had systematically studied backdoors attacks on text data. They have shown the impact that different trigger mechanisms like sentence, character and word level triggers, non-natural triggers and special triggers can have on the attack framework. Jin et al. [4] had proposed the TEXTFOOLER framework that can generate adversarial text for binary text classification and text entailment.
We also had to identify language tags of each token correctly in where work to generate adversarial text in a black-box setting - De epWord- modules are given in following sub-sections.

number of classes is truth label, In our proposed framework, we have developed a mechanism to order to perturb them into its complementary language. For a given pre-trained model, needs to identify a sub-word or character that can be perturbed within a to-

token importance calculation algorithm, and we select the top k tokens as the set of most-important tokens (C\text{CANDIDATES}) that can be attacked. The scoring approach is undertaken so that the number of perturbations to the original sentence can be minimized. We calculate the token importance, \( S_{w_i} \), using the equation below:

\[
S_{w_i} = \begin{cases} 
V_y(\hat{y}_{\text{label}}) - V_y(x_{-w_i}(\hat{y}_{\text{pred}})), & \text{if } \hat{y}_{\text{label}} = \hat{y}_{\text{pred}} \\
(V_y(\hat{y}_{\text{label}}) - V_y(x_{-w_i}(\hat{y}_{\text{label}})))
+ (V_y(x_{-w_i}(\hat{y}_{\text{pred}})) - V_y(\hat{y}_{\text{pred}})), & \text{if } \hat{y}_{\text{label}} \neq \hat{y}_{\text{pred}} 
\end{cases}
\]

where, \( V_y \) denotes the model prediction considering all the words in the original sentence, and, \( V_y(x_{-w_i}) \) denotes the model prediction with token \( w_i \) removed. \( \hat{y}_{\text{label}} \) is the original label class, \( \hat{y}_{\text{pred}} \) is the predicted class, and \( V_y(\hat{y}_{\text{label}}) \) denotes the probability value of the label class index in the model prediction.

### 4.2 Perturbation Techniques

Once we have our \( C_{\text{CANDIDATES}} \), from \( X_t \), we select tokens from this list in descending order of their importance in an iterative manner. Thereafter, we use a variety of perturbation techniques to alter this word in such a manner that it can adhere to the surrounding context and there is no significant change in the semantic structure of the sentence when this word is replaced. At the same time, we need to ensure that the perturbed word looks very much similar to a human-error and has a strong potential to force the target model to make a wrong prediction. We have used 3 perturbation techniques in our algorithm, in the order: (a) sub-word perturbation, (b) character-repetition, and (c) switch-word language. In order to identify a sub-word or character that can be perturbed within a token, we need to identify the language id of the token. We have different dictionaries defined for different languages, hence, language identification using the language id is really important so that we can load up the corresponding dictionary and replace the sub-word or character. We have used a character and phonetic based LSTM model [9] to obtain language ids for various tokens present in a sentence.

**Sub-Word Perturbation :** We have used a pre-existing dictionary of character groups that can be replaced by phonetically similar characters [9]. Essentially, these groups consists of character uni, bi and trigrams which are phonetically similar and are interchangeably used in social media based on user backgrounds (e.g. phu and f, au and ow). Whenever such a character-group is present in any particular word in the given sentence, we replace it with its corresponding value(s) from the dictionary. For example, in Bengali, word ’bhalo’→’valo’ (meaning good) and in Hindi, word ’gajab’→’gazb’ (meaning surprising). The sub-word perturbation technique ensures that both the meaning as well as the semantic structure is preserved.

**Character-Repetition Perturbation :** We also observed that character repetition was popular on social media, often to emphasize on something or for humour. Thus, we exploited this property

3 PROBLEM AND CHALLENGES

In this problem, we have a set of \( N \) sentences \( X = \{x_1, x_2, ..., x_N\} \) with an associated set of \( N \) labels \( Y = \{y_1, y_2, ..., y_N\} \) where total number of classes is \( M \). For a given pre-trained model, \( F \) needs to do a mapping \( F : X \rightarrow Y \) from an input sample, \( x \in X \) to a ground truth label, \( y_{\text{true}} \in Y \). Now, for an input sentence, \( x \in X \), a valid adversarial example, \( x_{\text{adv}} \) must meet the following criteria.

\[
F(x) \neq F(x_{\text{adv}}), \text{ and } S_{\text{Sim}}(x, x_{\text{adv}}) \geq \epsilon
\]

where \( S_{\text{Sim}}(.) \) is a semantic and syntactic similarity function and \( \epsilon \) is the minimum similarity between the input and adversarial samples. Here, \( x_{\text{adv}} = x + \Delta x \) where \( \Delta x \) is an imperceptible perturbation added into input sentence, \( x \). The main challenge has been to come up with novel perturbation techniques for code-mixed data. Previous frameworks for monolingual data, language model can not be used to perform perturbations as such a model is currently not available for code-mixed data. In code-mixed data domain, synonym-based replacement strategy will not work as it is very difficult to derive synonyms from bilingual tokens and also their synonyms may have the undesired effect of changing the contextual meaning. Also, simply perturbing characters in a code-mixed token renders it meaningless in most of the cases, as the meaning as well as the semantic structure of a word must be disturbed. Hence, we had to take into account phonetic similarity in order to make token perturbations. We also had to identify language tags of each token correctly in order to perturb them into its complementary language.

4 METHODOLOGY

In our proposed framework, we have developed a mechanism to attack text classification models under a black-box setting. The goal of our framework is to identify the \( k \) most important tokens \( w_1, w_2, ..., w_k \) and apply a set of perturbation techniques, \( P = \{p_1, p_2, ..., p_m\} \) on them iteratively until we get \( x_{\text{adv}} \) for the corresponding input sentence if attack is successful. More details of these different modules are given in following sub-sections.

### 4.1 Token Importance Calculation

We consider a sample sentence of \( n \) tokens, \( X_t = \{w_1, w_2, ..., w_n\} \). In order to calculate the importance of a particular token in a sentence, we replace that particular token by an UNKNOWN token, and obtain the prediction vector for the modified sentence. Each token is assigned a score \( S_{w_i} \) based on its impact on the sentence using a token importance calculation algorithm, and we select the top \( k \) tokens as the set of most-important tokens (\( C_{\text{CANDIDATES}} \)) that can be attacked. The scoring approach is undertaken so that the number of perturbations to the original sentence can be minimized. We calculate the token importance, \( S_{w_i} \), using the equation below:

\[
S_{w_i} = \begin{cases} 
V_y(\hat{y}_{\text{label}}) - V_y(x_{-w_i}(\hat{y}_{\text{pred}})), & \text{if } \hat{y}_{\text{label}} = \hat{y}_{\text{pred}} \\
(V_y(\hat{y}_{\text{label}}) - V_y(x_{-w_i}(\hat{y}_{\text{label}})))
+ (V_y(x_{-w_i}(\hat{y}_{\text{pred}})) - V_y(\hat{y}_{\text{pred}})), & \text{if } \hat{y}_{\text{label}} \neq \hat{y}_{\text{pred}} 
\end{cases}
\]

where, \( V_y \) denotes the model prediction considering all the words in the original sentence, and, \( V_y(x_{-w_i}) \) denotes the model prediction with token \( w_i \) removed. \( \hat{y}_{\text{label}} \) is the original label class, \( \hat{y}_{\text{pred}} \) is the predicted class, and \( V_y(\hat{y}_{\text{label}}) \) denotes the probability value of the label class index in the model prediction.
and created a dictionary of top characters which are frequently repeated. We select a character from the target word and repeat it one based on its value in the dictionary. Repeating certain characters do not alter the meaning of a word and also preserves its phonetic similarity, however, it might force a model to make a false prediction. For example, in Hindi, word ‘mafi’→‘mafii’ (meaning pardon) and in Bengali, word ‘paaoa’→‘paooa’ (meaning getting).

Switch-Word Language Perturbation: Given an input sentence, we have used a character and phonetic based LSTM model [9] to obtain language ids for various tokens present in it. Once we have the language id of a token, we back-transliterate and translate the word to its complementary language using contextual information and an LSTM-based seq2seq model; for example, ‘bacha’→‘baby’ and in Hindi, ‘byaah’→‘byaat’. This perturbation technique does not change the meaning of the word and also preserves contextual similarity to a great extent, however, it can force the model to make a false prediction.

4.3 Iterative Inference

In this step, we iteratively choose the next most-important word obtained using the token importance calculation algorithm, perturb it using the attack strategies one by one with sub-word-perturbation being used in the first trial, followed by character-repetition and switch-word language perturbations. The intuition behind the order of application of the above techniques is that the sub-word and letter-repetition perturbation techniques make changes only to a part of a word which tends to preserve its semantic structure the most. We obtain the model prediction vector, \(F(X_{adv})\) and the predicted class, \(y^\text{pred}_i\), by replacing each of these perturbed words in the original sentence. As soon as our prediction label becomes different from the original prediction of the unperturbed sentence, we claim that the attack is successful and terminate the process. If the system is unable to induce an attack even after trying out all possible perturbation techniques, we declare that an attack for the given sentence is unsuccessful. We also calculate the value of the maximum probability drop, \(P_{\text{drop}}\), that has been induced in the label class. We compute the label class probability value from the prediction vector of the adversarial sentence, \(y^\text{temp}_i\), that has produced the maximum drop among all perturbation techniques, for the current token, \(w_i\), and the label class probability value, \(y^\text{pred}_i\), of the prediction vector obtained using the current perturbation function, and compute their maximum difference, \(P_{\text{drop}} = \max(P_{\text{drop}}^{\text{max}})\). \(T_{\text{max}}\) is the perturbed token that produces the maximum probability drop in the label class of the prediction vector from \(X_{adv}\) obtained using \(w_i\). The details of our attack framework is explained in Algorithm 1.

5 EXPERIMENTS AND RESULTS

We have used a sentiment classification task to demonstrate the capability of our framework. Initially, we had trained deep learning models on the given code-mixed sentiment classification datasets and evaluated their performance on the validation and test sets. We have used the same models to perform inference on the adversarial samples. For models, we have taken into account several of the state-of-the-art models that have been used for sentiment classification over the years and have decided to finalize Bi-LSTM-CNN [3], BiGRU-CNN [3], Transformer [12], char-CNN [21] and mBERT [15] based architectures for demonstration of the model agnostic nature of our adversarial attack technique. The maximum input sequence length, vocabulary size, learning rate for these experiments were set at 25, 17k, and 0.001 respectively. We have summarized our results using F1-score, mean attack time per sentence, Mean Opinion Score (MOS) [18], and Adversarial Attack Success Rate (S\(_A\)d) which is, the proportion of test data points on which adversarial attack has been successful, in Tables 1, 2.

Evaluation Metric: Here, we used Mean Opinion Score (MOS) [18] to evaluate our system. Cosine similarity is not applicable in this case as there is no pre-existing model to obtain the embeddings of code-mixed sentences. We were supported by a group of volunteers who had been provided a list of 100 questions each, for each one of the models. The MOS for each model is calculated by taking the average MOS given by different human participants involved in the study. Each participant was asked to enter a score in
the range 0–4 for each perturbed sentence among a list of perturbed sentences produced by that particular model, for a different configuration of the number of perturbed words; a 0 indicates maximum similarity to the original sentence, while a 4 indicates the least.

Data Sets Details: We have used two code-mixed sentiment classification datasets on two different language pairs, “Bn-En” [10] and “Hin-En” [13] for our experiments to evaluate the effectiveness of our method. The “Bn-En” dataset contains 3206 and 943 samples in the training and test sets, while the “Hin-En” dataset contains 13845 and 1846 samples in training and test sets, respectively. The average length of sentences in “Bn-En”, “Hin-En” are 15, 12, respectively, and the mean Code-Mixing Index (CMI) [1] for “Bn-En” and “Hin-En” datasets are 22.1, 18.57 respectively.

Attacking Results and Ablation Study: From the results shown in Table 1, we can observe that the attack framework has been successful in forcing the models to make wrong predictions. Our approach is faster than other conventional approaches as those frameworks use another deep learning model or token similarity algorithm to bring about a successful perturbation. However, we use a hashing-based approach to partially or completely perturb tokens, which speeds up the entire operation to a huge extent. 

Table 1: Adversarial Attack Results On Different Models - Hindi-English Code-Mixed Data

| Model      | Before Attack | Top 2 Words | After Attack | Top 4 Words | Top 8 Words |
|------------|---------------|-------------|--------------|-------------|-------------|
|            | F1            | Time(s) | MOS | SRAdv | F1 | Time(s) | MOS | SRAdv | F1 | Time(s) | MOS | SRAdv |
| Bi-LSTM-CNN | 0.8966        | 0.7147 | 0.5052 | 0.2083 | 0.2778 | 0.4828 | 0.5038 | 0.2917 | 0.5122 | 0.3296 | 0.6097 | 0.4096 | 0.667 |
| Bi-CRU-CNN | 0.8927        | 0.7255 | 0.5374 | 0.0833 | 0.263 | 0.5078 | 0.6410 | 0.4150 | 0.4836 | 0.3496 | 0.7628 | 0.7067 | 0.6819 |
| Transformer | 0.8904        | 0.6852 | 0.3643 | 0.3750 | 0.3001 | 0.4726 | 0.4473 | 0.7083 | 0.5037 | 0.3149 | 0.5240 | 0.8333 | 0.6532 |
| char-CNN   | 0.8600        | 0.6185 | 0.4217 | 0.2917 | 0.4401 | 0.4957 | 0.4664 | 0.3750 | 0.4889 | 0.4191 | 0.5335 | 0.5000 | 0.562 |
| mBERT      | 0.9132        | 0.8365 | 0.6843 | 0.1428 | 0.158 | 0.7008 | 0.8821 | 0.4266 | 0.2768 | 0.5155 | 1.1967 | 0.5714 | 0.4369 |

Table 2: Adversarial Attack Results On Different Models - Bengali-English Code-Mixed Data

| Model      | Before Attack | Top 2 Words | After Attack | Top 4 Words | Top 8 Words |
|------------|---------------|-------------|--------------|-------------|-------------|
|            | F1            | Time(s) | MOS | SRAdv | F1 | Time(s) | MOS | SRAdv | F1 | Time(s) | MOS | SRAdv |
| Bi-LSTM-CNN | 0.8904        | 0.7147 | 0.5052 | 0.2083 | 0.2778 | 0.4828 | 0.5038 | 0.2917 | 0.5122 | 0.3296 | 0.6097 | 0.4096 | 0.667 |
| Bi-CRU-CNN | 0.8927        | 0.7255 | 0.5374 | 0.0833 | 0.263 | 0.5078 | 0.6410 | 0.4150 | 0.4836 | 0.3496 | 0.7628 | 0.7067 | 0.6819 |
| Transformer | 0.8904        | 0.6852 | 0.3643 | 0.3750 | 0.3001 | 0.4726 | 0.4473 | 0.7083 | 0.5037 | 0.3149 | 0.5240 | 0.8333 | 0.6532 |
| char-CNN   | 0.8600        | 0.6185 | 0.4217 | 0.2917 | 0.4401 | 0.4957 | 0.4664 | 0.3750 | 0.4889 | 0.4191 | 0.5335 | 0.5000 | 0.562 |
| mBERT      | 0.9132        | 0.8365 | 0.6843 | 0.1428 | 0.158 | 0.7008 | 0.8821 | 0.4266 | 0.2768 | 0.5155 | 1.1967 | 0.5714 | 0.4369 |

Table 3: Code-mixed Perturbation Performance Study

| Dataset | Perturbation Methods | SR_Perturb |
|---------|----------------------|------------|
| Bn-En   | Sub-Word             | 85.9       |
|         | Character-Repetition  | 93.08      |
|         | Switch-Word Language  | 79.94      |
| Hin-En  | Sub-Word             | 88.5       |
|         | Character-Repetition  | 90.52      |
|         | Switch-Word Language  | 87         |

resilient to adversarial attacks than the other models. The mBERT model remains the most robust model in this case as well. We have also performed an experiment as ablation study to estimate the effectiveness of each of the 3 perturbation techniques by enumerating the corresponding perturbation success rate (SR_perturb), which is the percentage of the vocabulary words that could be successfully perturbed using the given technique. From Table 3, we infer that the letter-repetition technique is the most successful in Bengali-English data due to a possibility of fewer changes in the word semantic structure, and sub-word perturbation turns out to be the most effective in the Hindi-English data, which can be attributed to a greater sense of the semantic structure of the code-mixed tokens to changes at the sub-word level.

6 CONCLUSION

In this paper, we have presented a generic framework that can attack code-mixed classification models by identifying and perturbing important tokens. Our word-importance calculation algorithm ensures that an attack is successful with a very low percentage of word perturbations in the original sentence, and the entire process is completed within a very short duration. Also, the low values of MOS indicate that the perturbed sentences are very similar to the original ones. We have been able to reduce the F1-scores of both Bengali-English and Hindi-English code-mixed datasets, which shows that our attack framework can be successful with a variety of language pair.
REFERENCES

[1] Amitava Das and Björn Gambäck. 2014. Identifying Languages at the Word Level in Code-Mixed Indian Social Media Text. In Proceedings of the 11th International Conference on Natural Language Processing. NLP Association of India. Goa, India. 378–387. https://www.aclweb.org/anthology/W14-5152

[2] Ji Gao, Jack Lanchantin, Mary Lou Sozza, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In 2018 IEEE Security and Privacy Workshops (SPW). IEEE, 50–56.

[3] Anupam Jamatia, Steve Swamy, Björn Gambäck, Amitava Das, and Swapam Debbhara. 2020. Deep Learning Based Sentiment Analysis in a Code-Mixed English-Hindi and English-Bengali Social Media Corpus. International journal on artificial intelligence tools 29, 5 (2020).

[4] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In Proceedings of the AAAI conference on artificial intelligence. Vol. 34. 8018–8025.

[5] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. arXiv preprint arXiv:1812.05327 (2018).

[6] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xiangyang Xue. 2018. Textbugger: Generating adversarial text against real-world applications. arXiv preprint arXiv:2004.09984 (2020).

[7] Luxiang Liu, Xueyi Chen, Shihkun Feng, Shuojuan Wang, Xuan Ouyang, Yu Sun, Zhengjie Huang, and Weiyue Su. 2020. K2018 at SemEval-2020 task 9: Adversarial training for code-mixing sentiment classification. arXiv preprint arXiv:2009.03673 (2020).

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017).

[9] Soumil Mandal, Sourya Datta, and Dipankar Das. 2018. Language identification of bengali-english code-mixed data using character & phonetic based lstm models. arXiv preprint arXiv:1803.03859 (2018).

[10] Soumil Mandal, Sambit Kumar Mahata, and Dipankar Das. 2018. Preparing bengali-english code-mixed corpus for sentiment analysis of indian languages. arXiv preprint arXiv:1803.04000 (2018).

[11] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 135–147.

[12] Daniel Palomino and Jose Ochoa-Luna. 2020. Palomino-Ochoa at SemEval-2020 Task 9: Robust System based on Transformer for Code-Mixed Sentiment Classification. arXiv:2011.09948 [cs.CL].

[13] Braja Gopal Patra, Dipankar Das, and Amitava Das. 2018. Sentiment Analysis of Code-Mixed Indian Languages: An Overview of SAIL, Code-Mixed Shared Task@ ICON-2017. arXiv preprint arXiv:1803.06745 (2018).

[14] Parth Patwa, Gustavo Aguilar, Sudipta Kar, Sunar Pandey, Srinivas PYKL, Björn Gambäck, Tammoz Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. arXiv e-prints (2020), arXiv–2008.

[15] Telmo Pires, Eva Schlüngel, and Dan Garrette. 2019. How multilingual is multilingual bert? arXiv preprint arXiv:1906.01502 (2019).

[16] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating adversarial text against real-world applications. arXiv preprint arXiv:2006.16176 (2020).

[17] Samson T an and Shafiq Joty. 2021. Code-Mixing on Sesame Street: Dawn of the Adversarial Polyglots. arXiv preprint arXiv:2103.09593 (2021).

A APPENDIX

B QUALITATIVE RESULTS

In Tables 5 and 4, we show the qualitative results of the effect of our perturbation algorithm on both the Bengali-English and Hindi-English code-mixed datasets. We observe that the perturbations that have been introduced in the sentences are negligible when the number of perturbed tokens are lesser, which can be attributed to an unintentional human error or a typing error. However, as the number of tokens perturbed increases, the attack success rate increases while the quality of sentences degrade. The perturbations do not alter the overall meaning of a sentence, neither does it effect the semantic structure. Hence, our perturbation algorithm can attack deep learning models in a very subtle manner.

C ERROR ANALYSIS

We observe that it is relatively easy to produce an adversarial attack in the case of shorter sentences. It can be partially explained by the fact that in many cases, a particular word might have a much higher contribution to the overall context of a short sentence compared to a longer sentence. Hence, we are easily able to switch the predicted label for a shorter sentence after perturbing less than four tokens only, while we might need to further a larger number of tokens in the case of a longer sentence. Also, some perturbations like the letter repetition perturbation might add a larger number of redundant characters to a word which might be quite conspicuous, however, they do not alter the meaning of the sentence and is unable to “fool” the model because of its phonetic similarity. We have provided some unsuccessful adversarial samples on mBERT model for both Bengali-English and Hindi-English datasets in Tables 6 and 7.
Table 4: Hindi-English Adversarial Attack Samples

| Model | Perturbation Level (K) | Examples | Ground Truth | Predicted Label | Perturbed Label | MOS |
|-------|------------------------|----------|--------------|----------------|----------------|-----|
| Bi-LSTM CNN | 2 | piik up da cal damn dumb gal he neds u dhaniya pyar kruti h tu use bat kr use mar rha hoga wo andar se | neutral | neutral | neutral | 0.125 |
| | 4 | pick up da call damn dumb gal he neds u dhaniya pyar kruti h tu use bat kr use mar rha hoga wo andar se | neutral | neutral | neutral | 0 |
| | 8 | pick up da cal damn dumb gal he neds u dhaniya pyar kruti h tu use bat kr use mar rha haaga wo within se | neutral | neutral | neutral | 0.125 |
| Bi-GRU CNN | 2 | pick up da cal damn dumb gaal he neds u dhaniya pyar kruti h tu use talk kr use mar rha hoga wo andar se | neutral | neutral | neutral | 0.125 |
| | 4 | pick up da cal damn dumb gaal he neds u dhniya pyar kruti h tu use talk kr use mar rha hoga wo andar se | neutral | neutral | negative | 0.125 |
| | 8 | pick up da cal damn dumb gill he neds u dhniya pyar kruti h tu use talk kr use mar rha hoga wo andar se | neutral | neutral | negative | 0.150 |
| charCNN | 2 | pick up da cal damn dumb gal he neees u dhaniya pyari kruti h tu use bat kr use mar rha hoga wo andr se | neutral | neutral | negative | 0.125 |
| | 4 | pick up da cal dmn dumb gal he neds u dhaniya pyar kruti h tu use bat kr use mar rha hoga wo andr se | neutral | neutral | negative | 0.125 |
| | 8 | pick up daa clll dmn dumb gal he neds u dhaniya pyar kruti h tu use bat kr use mar rha hoga wo andaarrr se | neutral | neutral | negative | 0.250 |
| Transformer | 2 | pick up da cal damn dumb ladki he neds pyar kriti tu use bat kr use mar rha hoga wo andar se | neutral | neutral | neutral | 0.125 |
| | 4 | pick up da cal damn duumb ladki he neds pyar kriti tu use bat kr use mar rha hoga wo andar se | neutral | neutral | positive | 0.125 |
| | 8 | pick up da cal damn goonga ladki he neds u dhaniya pyar kruti h tu use bat kr use mar rha hoga wo andar se | neutral | neutral | positive | 0.125 |
| mBERT | 2 | pick up da cal damn dumb gal hee neds dhaniya pyaarrr krkti tu use bat kr use mar rha hoga wo andar se | neutral | neutral | neutral | 0.125 |
| | 4 | pick up da cal damn dumb gal hee neds dhaniya pyaarrr krkti tu use bat kr use mar rha hoga wo andar se | neutral | neutral | neutral | 0.125 |
| | 8 | pick up da cal damn dumb gal Hee nedeec dhaniya paaarrr krtili tu useeeec bat kr useeeec mar rha hoga wo insiderr se | neutral | neutral | negative | 0.375 |
Table 5: Bengali-English Adversarial Attack Samples

| Model       | Perturbation Level (K) | Examples                                                                 | Ground Truth | Predicted Label | Perturbed Label | MOS  |
|-------------|------------------------|--------------------------------------------------------------------------|--------------|-----------------|-----------------|------|
| Bi-LSTM     | 2                      | vai eita hobe na vai tmio jao vai plz vai vipoode pore jabo              | negative     | negative        | neutral         | 0.125|
|             | 4                      | vai eita hobe na vai tmio jao vai plz vai vipoode pore jabo              | negative     | negative        | neutral         | 0.125|
|             | 8                      | vai eeta hobe na vai tmio jao vai plz vai vipode pore jabo               | neutral      | neutral         | neutral         | 0.150|
| Bi-GRU      | 2                      | vai eeta hobe na vai voutoo jao vai plz vai vipoode pore jabo             | negative     | negative        | neutral         | 0.150|
|             | 4                      | vai eeta hobe na vai tmeeo jao vai plz vai vipoode pore jabo              | negative     | neutral         | neutral         | 0.150|
|             | 8                      | vai eit hobe na vai tmeeo jao vai plz vai vipoode pore jabo               | neutral      | neutral         | neutral         | 0.150|
| charCNN     | 2                      | voi eita hobe na vai tmio jao voi plz voi vipoode pore jbo                | negative     | negative        | neutral         | 0.125|
|             | 4                      | voi eita hobe na vai tmio jao voi plz voi vipoode pore jbo                | negative     | neutral         | neutral         | 0.125|
|             | 8                      | vaee eita hobe na vaee tmio jao vaee plzz vaee vipoode pore jaaboo        | neutral      | neutral         | neutral         | 0.125|
| Transformer | 2                      | vai eita hobe na vai tmio jao vai plz vai vipoode pore jabo               | negative     | negative        | neutral         | 0.125|
|             | 4                      | vai eit hobe na vai tmio jao vai plz vai vipoode pore jabo               | negative     | negative        | neutral         | 0.125|
|             | 8                      | vai eita hobe na vai tmio jao vai plz vai vipoode pore jabo               | neutral      | neutral         | neutral         | 0.125|
| mBERT       | 2                      | vai eit hobe na vai tmio jao vai plz vai vipoode pore jabo               | negative     | negative        | positive        | 0.150|
|             | 4                      | vai eita hobe na vai tmio jao vai plz vai vipoode pore jabo               | negative     | neutral         | neutral         | 0.125|
|             | 8                      | vai eita hobe na vai tmio jao vai plz vai vipoode pore jabo               | neutral      | neutral         | neutral         | 0.125|

Table 6: Bengali-English Adversarial Attack Error Samples

| Perturbation Level (k) | Examples                                                                 | Ground Truth | Predicted Label | Perturbed Label | MOS  |
|------------------------|--------------------------------------------------------------------------|--------------|-----------------|-----------------|------|
| 2                      | Denmark err movie Festen conventional film theke ekdom alada Eta bohu festival prize peyechilo | negative     | negative        | negative        | 0.125|
| 4                      | Denmark err movie Festen conventional film theke ekdom alad Eta bohu festival prize peyechilo | negative     | negative        | negative        | 0.125|
| 8                      | Denmark err movie Festen conventional film theke ekdom alad Eaaa bohw festival prije piyechiloo | negative     | negative        | positive        | 3.125|

Table 7: Hindi-English Adversarial Attack Error Samples

| Perturbation Level (k) | Examples                                                                 | Ground Truth | Predicted Label | Perturbed Label | MOS  |
|------------------------|--------------------------------------------------------------------------|--------------|-----------------|-----------------|------|
| 2                      | Hlo salmon sir mai aur mere mami ppa ake bhut bde fan h                  | neutral      | neutral         | neutral         | 0.125|
| 4                      | Hlo salmon sir mai aur mere mami ppa ake bhut bige fan h                 | neutral      | neutral         | neutral         | 0.125|
| 8                      | Hlo salmon sir mai aur mere mami ppa ake bhut bige fan ffinh h           | neutral      | neutral         | neutral         | 0.675|