# **destroR**: Attacking Transfer Models with Obfuscous Examples to Discard Perplexity

by

Saadat Rafid Ahmed
20101425
Rubayet Shareen
20101434
Mansur Mahi
20101067
Nazia Hossain
20101258
Radoan Sharkar
20101263

A final thesis report submitted to the Department of Computer Science and Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
January 2024

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Students Full Name & Signature:**

Saadat Rafid Ahmed
20101425

Rubayet Shareen
20101434

Radoan Sharkar
20101263

Nazia Hossain
20101258

Mansur Mahi
20101067

# Approval

The thesis titled destroR: Attacking Transfer Models with Obfuscous Examples to Discard Perplexity submitted by

1. Saadat Rafid Ahmed (20101425)
2. Rubayet Shareen (20101434)
3. Radoan Sharkar (20101263)
4. Nazia Hossain (20101258)
5. Mansur Mahi (20101067)

Of Fall, 2023 has been accepted as satisfactory in fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 9, 2024.

**Examining Committee:**

Supervisor:
(Member)

Dr. Farig Yousuf Sadeque
Assistant Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson & Associate Professor
Department of Computer Science and Engineering
Brac University

# Ethics Statement

In our study, we have worked with several models and our aim was to find out their vulnerability by challenging them with augmented data. Our intention was only to find out the model's weaknesses in order to make them robust and better for public usage. Under no circumstances do we criticize or look down on the work of the researchers. We have the out-most respect for the work that has previously been done by them. Their work is admirable and deserves respect.

**Content Warning: This research paper includes explicit language as the datasets we used are from public comments from multiple sources. They are kept unmasked to keep the research transparent.**

# Abstract

Advancements in Machine Learning & Neural Networks in recent years have led to widespread implementations of Natural Language Processing across a variety of fields with remarkable success, solving a wide range of complicated problems. However, recent research has shown that machine learning models may be vulnerable in a number of ways, putting both the models and the systems theyre used in at risk. In this paper, we intend to analyze and experiment with the best of existing adversarial attack recipes and create new ones. We concentrated on developing a novel adversarial attack strategy on current state-of-the-art machine learning models by producing ambiguous inputs for the models to confound them and then constructing the path to the future development of the robustness of the models. We will develop adversarial instances with maximum perplexity, utilizing machine learning and deep learning approaches in order to trick the models. In our attack recipe, we will analyze several datasets and focus on creating obfuscous adversary examples to put the models in a state of perplexity, and by including the Bangla Language in the field of adversarial attacks. We strictly uphold utility usage reduction and efficiency throughout our work.

**Keywords:** Obfuscous Examples, Adversarial Example, Text Attack, Attack Recipe, Perplexity, NLP, NLI, Adversarial Training, Augmentation.

# Acknowledgement

First and foremost, we express our gratitude and acknowledgment to the Great Allah, whose divine guidance and blessings have been paramount in the successful completion of our thesis without encountering any significant disruptions.

Secondly, we express our gratitude to our supervisor, Dr. Farig Yousuf Sadeque, for his unwavering support and invaluable advice throughout our work. Regardless of the situation, he provided assistance and played a pivotal role in refining our research skills through his insightful guidance.

Finally, our heartfelt appreciation goes to our parents, whose unwavering support has been indispensable throughout this journey. It may not have been possible without their constant encouragement and prayers. We stand on the verge of graduation, grateful for their kind support that has been a guiding force in our achievements.

# Table of Contents

# List of Figures

# List of Table

# Chapter 1

# Introduction

“With great power comes great responsibility"- this quote from Spiderman always comes to our mind whenever we have something great in our hearts. Nowadays, Neural Networks and Transformers provide us with extreme power capable of handling almost any sort of NLP task thrown at it. So, our job is to detect its weaknesses and try to forge its characteristics like obsidian. However, seeing what is happening inside a Neural Network under the hood is almost impossible. Similarly, the traditional way of evaluating and assessing an NLP model needs to be more explicit and specific. Recent studies show that these somewhat state-of-the-art models are sometimes prone to adversaries. In other words, they lack robustness. In our research, we want to address this issue by solving it with a pipeline that can be used on any model to evaluate its robustness, create new adversarial examples, and train the models to make them more robust against these sorts of Obfuscus Examples; thus, discarding their perplexities.

## 1.1 Problem Statement

The concept of creating adversarial examples is prevalent in Image Processing. However, it is a new field in NLP. It is tough to introduce an adversarial attack in an NLP system as the input and outputs of the models are sequences and very hard to mask. We need text-attacking methods to assess the robustness of the NLP models and ensure their performance. Besides generating adversarial examples, we need to reduce the perplexities of the model and increase its performance. In our work, we look forward to creating a pipeline that attacks an NLP model in multiple ways and trains the model on those created adversaries, preferably including Bangla Language. In one sentence, the initial problem we are stating is:

“Are Bangla transfer models robust against adversarial attacks?”

## 1.2 Research Objectives

The aims and objectives of our research, as discussed previously:

1. Introducing new attack methods
2. Attack recipes to incorporate Bangla Language
3. Creating adversaries for a transfer model
4. Providing an adversarial attacking platform
5. Generating robustness report
6. Analyze attacking recipes

## 1.3 Challenges

We are faced some challenges while we try to create the pipeline:

1. Creating adversarial examples is somewhat complex, so developing a new attack recipe is challenging.
2. The method of creating adversarial examples is hard to optimize
3. Along with creating adversaries, we need to maintain the semantics and sentence structure properly in the verdict.
4. Furthermore, incorporating a new language in the field of adversaries is hard to achieve.

# Chapter 2

# Related Work

We are trying to create a pipeline that can attack a model, create adversaries, and at the same time, train the model on the adversaries. That is why we went through previous works where all these issues were addressed. The papers that we selected to study are all peer reviewed, and well cited. Before finalizing a paper for review, we first tried to understand the paper from a birds eye view and assessed the importance of that paper in our work. Thus we could get a proper background knowledge for the thesis.

The understanding of robustness relies on going beyond the standard evaluation metrics. This is where we find CheckList.[16] Generalization and classification tasks are the fundamental problems of NLP, which can be solved by using the train-validation-test framework. This renders the accuracy of a model. All the state-of-the-art models accrued their accuracy based on this evaluation. A general evaluation may overestimate the real-world performance as the primary basis of the model is still generalizing the endless possibilities of natural languages. If a model is tested by unbiased data, which is unusual to the test data, the model might fail. CheckList, a methodology, and tool for comprehensive behavioral testing are introduced to overcome the biased overestimation of general evaluation methods of NLP models in this paper [16]. This new evaluation methodology can detect severe bugs, which traditional benchmarking cannot do. Significant capabilities to consider while evaluating a model are Vocabulary and POS, Taxonomy, Robustness, Named Entity Recognition (NER), Fairness, Temporal, Negation, Coreference Semantic Role Labeling (SRL), and Logic. Based on these capability checks, test generation takes place, which can be done from scratch by the user, or it can be done by perturbing an existing dataset. Building a perturbation function is a challenging job, but it is able to generate many test cases at once. Focusing on the capability CheckList, abstractions from the list can lead to building a perturbation function in a more lucid and easier way. Testing the state-of-the-art models such as Microsofts Text Analytics, Google Clouds Natural Language, Amazons Comprehend, BERT-base, and RoBERTa-base, with MFT, INV, DIR test types, did not always give the same accuracy label as tested with general evaluation, while testing with abstractions such as Vocab+POS, Robust, NER, Temporal, Negation, SRL. A researcher can use this CheckList to investigate and fix their NLP models and can also contribute to this

open-source library. This makes evaluating regular NLP models significantly more accurate and creates a new baseline in this field.

Correspondingly, we tried another benchmarking method called DynaBench [18]. By encompassing different tasks, Dynabench collects data in rounds where placing SOTA models as the starting point. Initially, it puts humans and models in a loop and detects the shortcomings of a model by providing examples that the models get wrong but the humans dont, and afterward, the newly collected data are used to train the SOTA models again in a stronger way. Continuing the same process in cycles finds progress in benchmarking the state-of-the-art models. Dynabench starts working with four official tasks: Natural Language Inference (NLI), Question Answering (QA), Sentiment Analysis, and Hate Speech Detection. Firstly in NLI, NLP models were targeted into three rounds of adversarial attack with decreasing levels of accuracy. DynaBench adds a fourth round, ensembling two different architectural models as target adversaries to increase robustness. Second, in terms of QA, models will be considered as failed if the models prediction does not overlap enough with the annotator's answer. Third, a dichotomous analogy (positive or negative) of sentiment was used traditionally to examine a model. Lastly, hate speech detection was put in place as it is a really difficult thing to classify. After the initial tasks, vMER was around 33-44% in NLI, QA, Sentiment, and Hate speech increasingly, which is still very high across all tasks, concluding that NLP is far from solved. While facing caveats like unnatural distributions and distributional shift, the uncertainty of the not-yet-in-the-loop models, changing of benchmark, right and wrongs of generative tasks, as well as the high expense of manual annotation, Dynabench opens up to new research directions and optimistic goals like, Anyone can run a task, Multilingualism and Multimodality as well as, Live model evaluation.

Probing deeper into benchmarking methods, we get Eraser [9]. Evaluating Rationales And Simple English Reasoning (ERASER) is a benchmark for NLP models that are interpretable. State-of-the-art NLP models are built depending on deep neural networks, and it is not possible to see and understand how and why they make their predictions. Due to this limitation, there is an increase in interest in designing interpretable deep models for NLP where the reasoning behind the outputs can be explained and visualized. Works of ERASER have been inspired by GLUE and SuperGLUE. Interpretability is a vast topic. In ERASER, the focus is specifically on rationales. In ERASER, the datasets contain human-provided explanations or justifications for certain predictions. These explanations are clearly labeled and are meant to provide enough information to make a prediction, but they may not cover all aspects of the prediction. There are a total of seven datasets that are included in ERASER. ERASER provides datasets that include reference labels and explanations (rationales) marked by humans in a standardized format. The dataset includes Evidence inference, BoolQ, Movie Reviews, FEVER, MultiRC, Common Sense Explanations (CoS-E), e-SNLI, and Human Agreement. In ERASER, models are evaluated based on their ability to make accurate predictions and the explanations (rationales) they provide for those predictions. Comprehensiveness is calculated to better understand the model. The formula to calculate comprehensiveness is ($comprehensiveness = m(x_i)_j - m(x_i/r_i)_j$). A high score on

this measure means that the explanations or justifications given were important in making the prediction, while a low score indicates that they were not significant in the prediction process. On the other hand, if the score is negative, it means that the model's confidence in its prediction increased after the explanations or justifications were removed. Alongside comprehensiveness, sufficiency measures how well the included explanations or justifications within a dataset enable a model to make a prediction. The intention of Eraser is to play a role in facilitating progress on explainable models for NLP. Several metrics are proposed that can be used to measure the quality of rationales that have been extracted by the models, both in terms of agreement with human annotations and in terms of how closely the output adheres to the original input. The metrics provide a comparison of the specific aspects of interpretability. In simple terms, the goal of ERASER is to aid in designing more interpretable models.

After understanding the ways of evaluating different NLP models, we tried to understand ways of attacking a model and creating adversarial examples. TextAttack [15] is one of the best frameworks in this field, creating adversarial attacks. Influenced by the greatly popular adversarial learning research for Image classification models, interest was also growing in the NLP community to introduce adversarial robustness and adversarial attacks for NLP models. But formulating an attack from scratch and restricting them in the same benchmarking policy is quite challenging. To influence research and development of NLP robustness TextAttack, a Python framework, was created with a view to form new adversarial attacks, augment data, benchmark models with existing attacks, and adversarial training in NLP. Each attack is composed of four elements: search method, goal function, transformation, and set of constraints. All these elements serve a specific purpose, goal function tells us about the success of an attack, constraint forces us to keep the adversarial examples consistent, transformation creates the actual adversarial examples and search methods search and shortlists the candidate perturbation using the set of constraints. It is because of this python framework we can now measure the robustness of an NLP model, fortify our models against adversarial examples, increase performance, and to create new attack recipes. This opens up further scope in NLP adversarial research to upgrade existing models and create new ways to attack NLP models.

There are multiple types of attack recipes in NLP. Understanding the construction of such recipes is crucial in our study. To understand black box attacks and how Deep Learning-based Text Understanding(DLTU) models are attacked, we unrolled TEXTBUGGER [4]. Researchers of Zhejiang University propose this general black box attack framework. DLTUs perform incredibly well in machine translation, Q&A, and text classification tasks. As a result, we see the hype about using these models nowadays. To properly understand the ability and effectiveness of these models, we need to explore their robustness against adversarial examples. TextBugger, like other Adversarial example generators, generates adversaries in both white box and black box settings. Furthermore, it has three other features 1. effective, 2. evasive, and 3. efficient. In the black box settings, the TextBugger is only capable of querying the targeted model with output as the prediction. And for white box settings, the attacker has all knowledge of the model it intends to attack. If we break down

each attack can be broken down into three segments. 1. Finding important words, bug generation(adversarial word generation), and replacing the important word with the bug. In the black box setting, TextBugger first finds the important sentences and then, using a scoring function, searches for the important words that can be manipulated to perturb the classifier. Again, in a white box scenario, it finds the optimal points to attack by computing the jacobian matrix of the classifier. Then, it generated the bugs at both the word and character levels. In the character level, it can swap, delete, insert, and substitute and in the word level, it can only create a substitute. Lastly, it chooses the optimal bug based on the confidence level of the classifier and modifies the adversarial example or the bug accordingly. The experiments show that TextBugger's success rate ranges from 80.2% to 92.7% in white box settings and 65.7 to 100% in black box settings, which is significantly higher than any other alternative of TextBugger.

To further understand the construction of white-box attack recipes, we took a dive into HotFlip [2]. HotFlip provides an optimized white-box adversarial example generator to fool character-level neural network classifiers. In the white-box setting, the adversarial example generators have complete knowledge about the model it is trying to fool to generate worst-case attacks to explore larger vulnerabilities. HotFlip relies on the unit token flip. If we consider a unit flip like mood to mooP, overall, it should preserve the semantics of the sentence or sequence, but HotFlip takes advantage of this constraint and, based on gradients of one-hot input vectors, finds the best possible adversaries where the classifier will misclassify. More accurately spoken, using the beam search technique, it finds the manipulation with the maximum estimated loss for all possible flip operations. The atomic flips are a set of three operations flip, insert, delete. These operations may introduce some misspelled words, but the classifier model should interpret this. HotFlip adversarial example generator works very fast, and that is why it can produce many adversarial examples on which the model can be trained again. If a model is trained on these examples, the model will be robust against this kind of attack as well as unseen misspelled and unclean corpus. In the paper, it is also shown how the adversarys success rate is 69.32% over a clean dataset.

These attack recipes heavily depend on finding the appropriate point in the dataset where bugs can be fitted. To understand where text can be manipulated we came across a paper, "Universal Adversarial Triggers for Attacking and Analyzing NL" [19]. This paper searched the Universal adversarial triggers, a sequence of tokens that can trigger a model to generate a specific prediction with any input from a dataset. Then a short sequence was found to trigger a target during concatenated inputs from text classification conditions on text generation and reading comprehension. In addition, two text classifications, Sentiment Analysis and Natural Language Inference, were used. Then many models were attacked in sentiment analysis, like the SNLI model, ELMo- based model. Then the attack was successful as the model predicted the target span. Then triggers were created by considering white box access to particular models, which are transferable to other models for all dataset that was considered. Furthermore, Universal adversarial triggers and their algorithm with source code for attacks and experiments were introduced. Then the success

of triggers from data failures and models was shown, and it was assured that biases were exploited by models in the SNL dataset. For example, Triggers allow the GPT-2 language model to produce racist output even when trained in non-racial settings, the SNLI entailment accuracy to decrease from 89.94% to 0.55%, and 72% of "why" questions in SQuAD to be answered, "to murder American people. Again, it was shown that SQuAD models depend on type matching and this answer span surrounded tokens. These tokens successfully triggered the desired prediction, and a token replacement strategy that was HotFlip inspired was used. Therefore, Universal attacks are a context-independent model with a unique analysis tool as they highlight input-output patterns that a model can learn. Moreover, trigger search algorithms with more extended triggers were more utilized and effective. Triggers were prepended to paragraphs in this case, and other trigger tokens are optimized while the target span is predetermined and fixed. Then a decent guided search over tokens that finds short trigger sequences was proposed. Thus, the trigger gives an analysis of global model behavior, and the reason for being like this was input agnostic. By reading comprehension models, SNLI models help to identify heuristics learned and utilize dataset biases.

The second most crucial part of our study is fortifying the NLP models with adversarial training. In this field, Robustness Gym [17], is the first evaluation toolkit that comes to our mind. With the great achievements in NLP, we can see the transformation in data science and AI is taking place gradually. But in recent years, the researchers have faced challenges evaluating the NLP models. The practitioners mostly face challenges such as the Paradox of Choice: they are confused and get very little guidance on what evaluation should run next, Idiomatic Lock-in: Once it is set what to evaluate, one might get stuck in one of the evaluation idioms- Subpopulations, Transformation, Adversarial attacks, Evaluation sets. In most cases, one needs to glue together a combination of tools for evaluation that cuts across idioms, but it becomes difficult and lengthy to do so as each tool has its own abstractions and rules; workflow fragmentation: practitioners often face difficulty in consolidating evaluations and results using multiple tools and also they are not provided with the support for semantic versioning and sharing findings; hence it becomes difficult to report evaluation information easily. Robustness Gym, a unified toolkit, is a solution to these challenges, which is an open-interface design to support all the evaluation idioms. Robustness Gym enables the practitioners to get the versioning evaluations in test benches and also allows them to report the findings with robustness reports. For performing a continual evaluation, the Contemplate Create Consolidate loop has been proposed, which uses Robustness Gym to address these challenges. In this loop, we start with contemplating, where we need to consider the practitioner's tasks- existing research works, available datasets, evaluation goals, prior evaluations, constraints, etc., before moving forward with the evaluation. In the abstraction of Robustness Gym, we can see that it breaks robustness testing into two stages- cached operations and slice builders. It also helps users to create robustness reports for any model on a TestBench easily. Robustness Gym has made it easier for researchers and practitioners to perform novel analyses of existing tasks and models by investigating their performances. First, fine-grained analysis of name entity linking(NEL) across three widely used commercial APIs and three state-of-the-art academic systems, and the result has shown that the state-of-the-art academic outperformed the

other one. Most of the models struggle on some tropical(NFL Slices), lagging by more than 10% in the academic part. On the other hand, the second analysis is of state-of-the-art text summarization. Here, the models struggle with the examples that required the most abstraction, degrading by more than 9%. The researchers are still working on it. Despite this lagging, Robustness Gym is quite a promising toolkit for researchers and practitioners to use for evaluations.

Furthermore, we took a dive into another paper that defines debugging of NLP models [22]. State-of-the-art NLP models deliver incredible superhuman performances. But one thing that is very worrisome is that they have serious bugs in them. Some of the bugs may look really easy to spot, but in reality, they are very easy to miss. The reason is that they are buried in an exponential explosion of possible language phrasings. Fixing bugs is actually much harder than finding them. One complicacy is that if we go on to fix a bug, we accidentally create new bugs. Bugs are needed to be found and fixed effectively. There are two approaches to fixing bugs. The first approach is user-driven, and the second approach is automated. There are several upsides and downsides to a user-driven method. Those are i)The proper behavior of the model is known to the user, ii) it Can break down the problem with ease, iii)Slow and costly iv) Limited by user creativity. On the other hand, the upsides and downsides of using automated methods are i)Behavior is only known in a restricted domain, ii)The right abstraction is not always known iii) Fast and cheap iv)The creativity is effortless, and it is based on trial and error. The advantages of both approaches can be combined to get a better result. This can be done by pairing a user with a generative language model. The framework is called Adaptive Testing and Debugging (Ada testing). The language model tries to generate tests so that the model fails. The role of the user is validating and at times, managing and organizing these tests. The user here guides the model as it tries to find the weak points where the model fails. To actually understand the efficiency and improvement provided by the Ada test, it was paired with CheckList, and the user studied with NLP experts. Even though CheckList is a state-of-the-art user-based testing approach, Ada Testing outperforms CheckList in all the tasks and models that were considered. It was noticed that users found five times more bugs with Ada Test. One interesting thing to note is that Ada Test does not require any programming background. It is very useful and important that the Ada Test is accessible to people from a wide range of backgrounds. Through loopings, bugs are found and organized into a test tree. After that, the model is fixed and has been updated, and it is checked for new bugs. This process is repeated many times until no serious bugs are found. This is a tight debugging iterative cycle which is similar to software development. NLP models need to be trained with wild data beyond their cross-validation. When Ada testing is used, it is seen that there is a significant improvement in performance when the model is tested with data without cross-validation. Ada test combines the benefits of both manual testing by humans and automated testing using language models. It involves a process of testing, identifying issues, and retesting to ensure high quality. Additionally, it helps prevent the introduction of new bugs and is suitable for a wide range of models and users due to its flexibility.

Again, we take a look into another adversarial training method proposed in Learning

The Difference That Makes A Difference With Counterfactually-Augmented Data [10] paper. Here, the authors are taking a standard dataset and creating a counterfactually revised dataset based on the original dataset, putting it on a label by human-in-the-loop process, and storing the counterparts in the database. Then the authors combined the data, and remarkable improvements were noticed. Kaushik and his research group found that the classifiers such as SVMs, NB, Bi-LSTMs, ELMo models with LSTM, and BERT fail to achieve the general dataset accuracy scores while testing with the counterfactual dataset. This failure occurs due to the inability to identify spurious data patterns by the models. Here, counterfactually augmented data added to the base data improves the accuracy. The authors created a new dataset from the original one through a human-supervised process. They took the original dataset, and the expert annotators counterfactually augmented the data. Here, while the documents are counterfactually manipulated, their goal is to separate spurious and non-spurious connections by acting exclusively on the component of interest, resulting in classifiers that perform better when spurious associations do not travel outside of their domain. Finally, the original data and the counterpart were combined, creating a new dataset. Here, (i) reaching counterfactual target labels, (ii) maintaining the coherence of the data, and (iii) avoiding unnecessary changes, were the primary goal for annotators. After the changes in the dataset, the resulting factors reach a new level of accuracy in classification model testing and open up new research opportunities.

In order to create a robust model, we also need to understand why a model is making certain decisions. So we study Right for the Wrong Reasons [11]. This research paper discusses a new evaluation set, HANS Heuristic Analysis, for the NLI system. Here three Heuristics were mainly focused on with few examples. Firstly, the Lexical overlap heuristic ignores the ordering of words in the sentence and recognizes its identity that is to be taken by NLI models. Secondly, the Subsequence heuristic works with a group of words which is expected that standard RNN will embrace it. Lastly, tree-based NlI models take appended parts of the parse tree of the Constituent heuristic. In addition, their template discussed where five templates were in favor of the heuristic. And five templates were against the heuristic. Ensuring the plausibility of creation from the template is one of the advantages of generating examples from a template. Here core vocabulary was drawn, and all nouns were plausible subjects of all verbs or apparent objects of all transitive verbs. Here verbs were chosen so that they appeared at least 50 times in all relevant frames of the MNLI training set. Then, three models were discussed, which explained the strategies for representing input sentences DA, ESIM, and SPINN and also mentioned BERT. This transformer model uses attention to compute sentences instead of recurrence. DA model utilizes no word information order, whereas a sequential encoder works ESIM instead of a tree-based hybrid inference model. Moreover, the SPINN parser gave accurate parses for the HANS example. Implementations from AlenNLP were used for the DA and ESIM models. Furthermore, code from GitHub repositories was used for SPINN and BERT models, and all the models were trained on MNLI. In addition, all models scored highest on the MNLI test. Again on the HANS data set, all the models performed severely, with less than 10% accuracy in many cases. Then all the models were compared based on the heuristics, and all models were unsuccessful in illustrating all three heuristics. Furthermore, an ex-

ternal dataset was also used, which differed from the HANS dataset. The data set was collected through human judgments on a subset of HANS. Also, there were 95 participants on Amazon Mechanical Turk, including three experts from annotators. The accuracy was 76% from mechanical Turk participants and 97% from expert annotators. The result is said to utilize a training set that does not support heuristics so that a model can be prevented from learning a heuristic. Thus in this research paper, the HANS dataset was introduced to know if NLI models behave consistently or not with the heuristic. Lastly, it was suggested that the models perform well in HANS still there is a lot of room for progress.

Similarly, we explore models Local Decision Boundaries and a new annotation paradigm [12]. A more stringent NLP annotation paradigm closes systemic gaps in test data. After constructing a dataset, it was proposed that the authors manually tamper the test cases in modest but relevant ways to create contrast sets by changing the gold label. Again, contrast sets show a model's decision border locally, allowing for more precise linguistic evaluation.To help identify the optimal decision boundary close to the test instances, it was suggested that dataset authors actively perturb examples from their test set to create contrast sets. The original dataset authors simply need perturbed test sets big enough to derive verified conclusions regarding model behavior. Though the amount of labor varies, a person-week may provide high-quality perturbed test sets of roughly 1000 examples for several extensively researched NLP benchmarks. Contrast sets for ten distinct NLP datasets illustrate their effectiveness, and contrast sets are not adversarial. Still, model performance is poorer than on the original test setsup to 25% in certain circumstances. Our comparison sets are new evaluation standards, and we urge dataset development initiatives to follow similar annotation techniques.A novel annotation paradigm for NLP test sets based on contrastive instances was given. The technique preserves most of the previous dataset production procedures but fills in specific systematic dataset gaps. By moving assessments from accuracy on i.i.d. test sets to consistency on contrast sets, it was assessed if models had learnt the needed skills or just recorded a dataset's quirks. Ten NLP datasets were contrasted and presented as new assessment benchmarks. Future data collection should include contrast sets to evaluate both new and old NLP datasets.

Furthermore, we tried to understand how black-box predictions can be explained by Influence functions [13]. Modern deep learning NLP models are opaque and need to give a more clear understanding of how the model works. This led to the rise of several interpreting methods that explain how the model works and the reasoning behind its output. This study explores the application of influence functions in natural language processing (NLP) as a means of interpreting the decisions made by neural text classifiers. Influence functions are used to determine which training examples had the greatest impact on the model's output by identifying their influence on the model's decisions. It is seen that the influence function is useful for language inference. Saliency maps are also used to interpret models. In many cases, it is seen that saliency maps fail to provide a clear interpretation. That is when we see that influence function is useful. Saliency maps fail at complex semantic tasks. Saliency maps are like debugging tools. It can only tell which inputs are important

but cannot tell why. In this paper, many experiments are done to evaluate the potential utility of influence functions, which is going to help understand NLP models. The influence of examples in training data. What influence functions does is that it provides a method for tracing model predictions back to training examples. In the tasks of NLI, it is seen that influence functions are better suited to interpret models because it is more semantically complicated. The goal is to establish trust in model prediction by taking its aid.

Lastly, we needed a common platform to evaluate and understand the decision of a machine learning model in the perspective of input sequence. That is why we dived into a paper named "Why Should I Trust You?" [1]. Even though we can see the widespread use of machine learning models, they are primarily like black boxes. How and why a model makes a prediction is not always fully known. That is why it is essential to understand why a model makes a particular prediction in order to trust the prediction it made. Here, in this paper, to address this issue, LIME is introduced. It is a novel explanation technique that aids in understanding how and why the model makes a prediction. The explanations are proposed in a non-redundant way. This is very helpful in choosing an appropriate model for a particular task, and also it can be used to improve a model that is performing poorly. One extension of LIME is SP-LIME. It performs via submodular optimization. In layperson's terms, this paper suggests a way where a model's predictions are explained by showing the significant part of the input sequence. If we can get a direct connection between the input and output sequence, we can increase the efficiency of any attack method by a significant factor.

In our pursuit of finding something new, we might need some tools. Allen NLP [3] is one such tool for us. AllenNLP Interpret is an add-on to AllenNLP that lets one flexibly interpret NLP models. This toolkit helps to interpret results and explain why a model made such a prediction. AllenNLP Interpret comes with a set of interpretation methods and a library of visualization components for the front end. The toolkit is flexible, and interpretation methods like saliency maps and adversarial attacks can be applied to various models and tasks to interpret the models' predictions. Even though most state-of-the-art NLP models work very well, they are prone to ambiguity or confusion. These NLP models have some particular problems. For example, they latch on to superficial patterns, reflect unwanted social biases, and significantly underperform humans on a myriad of tasks. Researchers and practitioners try to find out why a model made a particular prediction, but there is no correct explanation. Instance-level interpretation methods provide explanations for specific model predictions. The interpretations are useful for finding out the strengths and weaknesses of models. There are several use cases for AllenNLP Interpret, and some of the major ones are uncovering Model Biases, finding decision rules, and diagnosing errors. In AllenNLP Interpret, the focus is on gradient-based saliency maps and adversarial attacks. Here, three saliency methods are considered. They are Vanilla Gradient, Integrated Gradients, and SmoothGrad. The goal of saliency maps is to find out why the model made a certain prediction. On the other hand, in adversarial attacks, words are replaced or flipped to see how the models' predictions change. Two types of adversarial attacks are considered here.

The first one is Untargeted and Targeted HotFlip and the second one is Input Reduction. In HotFlip, words are changed to see if it changes the model's prediction. In input reduction, words are removed as much as possible, and it is checked if the model's prediction changes or stays the same. The toolkit can currently interpret six tasks: reading comprehension, text classification, named entity recognition, etc. AllenNLP Interpret is a toolkit that is flexible and makes it possible to develop and test methods for interpreting a wide range of NLP models and tasks. This toolkit is open source, and it is continuously evolving. New interpretation methods are being developed to improve this toolkit.

Lastly in our work we might need to encounter Bangla WordNet[5]. Thats why we came across a paper that claims to create a Bangla Wordnet. Due to the lack of a good Bangla wordnet, we could not progress in NLP-related research very well. Hence, in this paper, The authors are developing BanglaNet using a semi-automatic cross-lingual sense mapping approach to link the wordnet with Princeton WordNet. Global WordNet Association (GWA) has enlisted almost all wordnets at several levels, including EuroWordNet, WOLF, Balkan WordNet, IndoWordNet, etc. These wordnets included over a thousand languages but only two of them were for the Bengali language in the IndoWordNet. This wordnet was developed using existing WordNets to incorporate language used in the Indian subcontinents. One of them was created using the popular expand approach, and it has semantic relations between synsets. This is the most contextual Bengali WordNet so far. In this research work, an expanded approach was followed to construct BanglaNet, and both automatic and manual methods were applied in the process. For automatic concept mapping, cross-lingual ambiguity is concerning. Among its various forms, uni-directional ambiguity in one-to-one and one-to-many has been addressed in this research. BanglaNet was developed by a specific methodology. First, it extracts literals from the Bangla lexicon and translates them into English. For each English literal, extract concepts available in Princeton WordNet. Then runs Wu & Palmers similarity score calculation algorithm. And finally, the map Bengali concept with the PWN concept. With this approach, base lexicons and concepts can be generated. But it comes with the gloss in Bengali for words, along with its synonymy. As the aim of this research is to make a relation with the PWN concept, the focus had to shift to cross-lingual mapping. In this process of BnaglaNet Development, there are two steps. The first one is Word Word Translation. A word in one language can be represented by multiple words in another language. So, it collects all possible English translations for Bengali words in the lexicon, including multiple POS. And the second step is Linking with Princeton WordNet using Probabilistic Model. In this step, the score of the probable concept from PWN for a BanglaNet concept gets calculated using a specific algorithm for calculating the probability score. After that, the Bengali synset is linked with the PWN synset using an algorithm for linking the concept first pass. To ensure the correctness of these linked concepts, multiple procedures are used. Also, if in a synset, any synonym has got only one concept tagged to it, then the concept from PWN which scored high probability in the probability calculation algorithm, would be chosen. The statistic of the initial data can be found in the table 2.1

From the table, it can be implied that to link 4665 concepts with PWN, 3729 PWN concepts have been used, which means there are cross-multiple concepts within

| | | Noun | Adj | Verb | Adv | Total |
|---|---|---|---|---|---|---|
| Initial | synsets | 18311 | 5713 | 2777 | 438 | 27239 |
| | words | 28311 | 8136 | 2923 | 788 | 40158 |
| Linked | synsets | 3174 | 1352 | 73 | 66 | 4665 |
| | words | 7477 | 2971 | 130 | 170 | 10748 |

Table 2.1: Status of linked Synset and Words from initial dataset

two WordNets. A Bengali word represents a particular concept in PWN which can be used to verify and add confidence to concept linking more. Our approach for BanglaNet does not consider gloss though the initial synset contains it. So, the accuracy of the algorithm can be improved by working on it and expanding BanglaNet by incorporating gloss. In this research work, a baseline of the system has been laid down. From this research analysis, around 5000 words from initially collected data are automatically linked up with Princeton WordNet. This is just the initial stage for further development of the Bengali wordnet.

# Chapter 3

# Exploratory Data Analysis

In the course of our research, we have identified four datasets serving distinct purposes. One of these datasets are directly associated with the fine-tuned models that form the basis of our research. The remaining three datasets have been curated to introduce additional variations in the dynamics of the data, allowing for a comprehensive examination of the robustness of the models. The names of the selected datasets are as follows: blp23[1], youtube_sentiment, CogniSenti, and BASA_cricket[2].

## 3.1 Dataset Overview

### 3.1.1 BASA_cricket

The dataset "BASA_cricket" was collected from the research paper "Datasets for Aspect-Based Sentiment Analysis in Bangla and Its Baseline Evaluation" [6]. This dataset was designed for sentiment analysis tasks in Bangla. The dataset comprises around 2900 comments gathered from Facebook, Prothom Alo, BBC Bangla, and a few other platforms. The comments are categorized into five different aspect categories, each associated with three polarity labels: positive, negative, and neutral. The aspect categories include diverse dimensions related to cricket. For our research, we utilized only the sentiment information (positive, negative, and neutral) from the dataset.

### 3.1.2 CogniSenti

The dataset under consideration has been developed by the authors of the research paper titled "Sentiment Classification in Bangla Textual Content: A Comparative Study" [14] This dataset is composed of comments made from various social media platforms, mostly comments were extracted from posts extracted from Twitter and Facebook. The annotation process involved manual categorization by individuals

[1]https://github.com/blp-workshop/blp_task2

[2]https://github.com/banglanlp/bangla-sentiment-classification/tree/main

fluent in the Bangla language, resulting in the categorization of posts into three distinct classes.

In totality, the dataset contains 6,570 annotated posts, with 5,628 originating from Twitter and 942 from Facebook statuses. The data was further divided into three subsets: the training set comprising 4,599 posts, the development set containing 985 posts, and the test set comprising 986 posts.

### 3.1.3 youtube_sentiment

The dataset utilized in this study was collected from the research paper titled "Detecting Multilabel Sentiment and Emotions from Bangla YouTube Comments" [7]. The data collected from comments from various YouTube videos, facilitated by the use of YouTube API version 3. After obtaining the dataset, it was systematically categorized into three categories. These are, *3 class*, *5 class*, and *Emotion*.

For the specific purposes of our research, we selected the data associated with the three-class label, while at the same time converted the five-class labels to conform to this three-class categorization. The primary focus of our investigation revolves around the class labels of our data which are positive, negative, or neutral. The dataset comprises a total of 2,796 comments, further divided into test, train, and development segments.

The dataset has been annotated by multiple expert native Bengali speakers belonging from diverse backgrounds, ensuring a nuanced and culturally grounded perspective in the sentiment classification task.

### 3.1.4 blp23

The dataset utilized in this study was collected from the Bangla Language Processing (BLP) workshop held at EMNLP 2023 on December 7th, 2023, in Singapore[23]. This dataset was chosen due to its credibility as a trusted source and its comprehensive coverage across 13 distinct domains, encompassing areas such as politics, education, agriculture etc. The dataset is a composite of the **MU**ltiplatform **BA**ngla **SE**ntiment (MUBASE) and SentNob datasets.

The SentNob dataset specifically comprises public comments sourced from social media platforms in response to posted content. This dataset exhibits a moderate agreement score of 0.53, indicating a moderate level of consensus among annotators. It is a multiclass dataset where the text is classified into Positive, Negative, and Neutral sentiment categories.

Conversely, the MUBASE dataset is an extensive collection that includes comments from both Tweets and Facebook posts, with each comment labeled according to its polarity. The combined dataset offers a rich and diverse set of data suitable for conducting sentiment analysis in the Bangla language. Its multiclass nature and varied sources contribute to its robustness while exploring sentiments across different domains within the Bangla linguistic context.

## 3.2 Label Distribution

The datasets selected for our study are categorized into three labels: negative, neutral, and positive. Among the incorporated datasets, blp23 boasts the highest number of datapoints, while the youtube_sentiment dataset contains the fewest. Notably, among the four datasets, "BASA_cricket" exhibits the shortest average text length, approximately 49 words per data point. On the contrary, "CogniSenti" displays the longest average text length, amounting to 96 words per data point. These variations in dataset size and text length provide diverse dimensions for our analysis, contributing to a comprehensive evaluation of sentiment analysis models.

| Dataset/Label | Negative | Neutral | Positive | Total |
|---|---|---|---|---|
| youtube_sentiment | 865 | 539 | 553 | 1957 |
| BASA_Cricket | 1515 | 194 | 376 | 2085 |
| CogniSenti | 919 | 2633 | 1047 | 4599 |
| blp23 | 15767 | 7135 | 12364 | 35266 |

Table 3.1: Data count in details of each dataset

To illustrate the distribution of data across all datasets, a horizontal column chart is presented, depicting the ratios of labels within each dataset. This visualization serves to convey the proportional representation of negative, neutral, and positive labels across the selected datasets.


Figure 3.1: Class label distribution of each dataset (in percentage ratio)

## 3.3 Comprehensive Analysis

A model's performance heavily relies on the dataset it's trained on. Therefore, to better understand the model's performance, we need to analyze the datasets thoroughly.

### 3.3.1 youtube_sentiment

The 'youtube_sentiment' dataset consists of a total of 1957 data points, which may not be a large number, but the data itself is quite interesting. This dataset specifically contains comments from YouTube videos. Comments on YouTube videos are typically influenced by factors such as the video's informativeness, popularity, upload date, etc. Unlike other social media platforms, YouTube videos continue to receive comments even years after their upload date, making these comments valuable for sentiment analysis. In our exploration of various social media datasets, we observed that YouTube comments exhibit two distinct properties: a lower usage of emojis and a higher frequency of links to blogs, articles, etc. Notably, we found no null values in this dataset, and after preprocessing the data, we were left with one of the most diverse datasets, despite its relatively modest size.

This dataset has 865 Negative, 539 Neutral, 553 Positive labeled data. Here is a bar chart to see the ratio better:

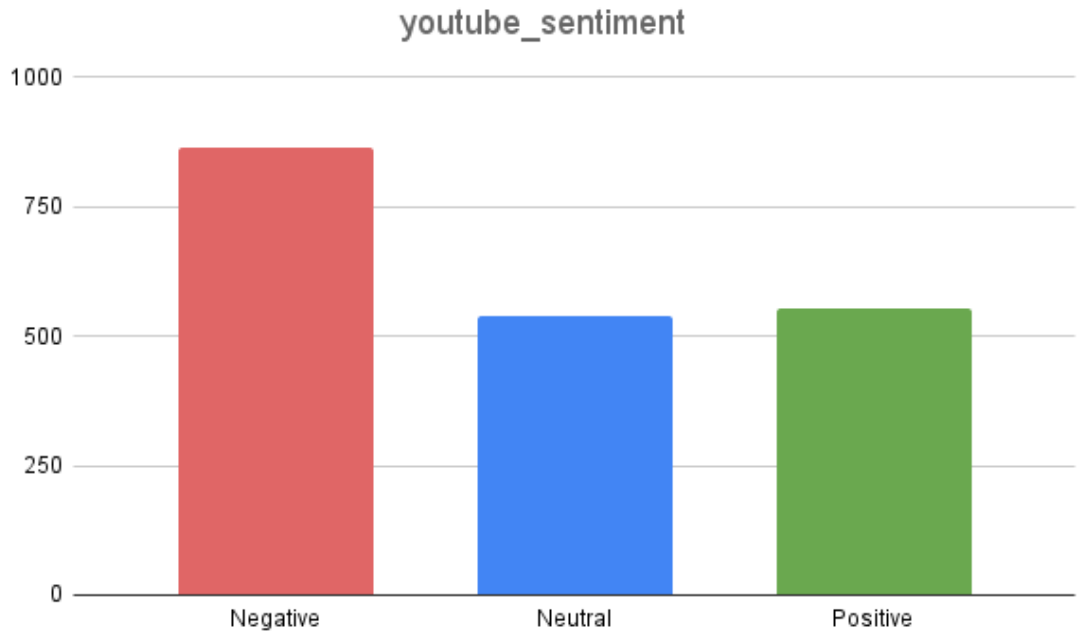


Figure 3.2: Class label distribution of *youtube_sentiment*

Provided are statistical measures for the 'word_count' of data points, including the mean, standard deviation, minimum, and quartiles (25%, 50%, 75%), along with the maximum. These metrics offer insights into the central tendency and variability of the data:

| data_points | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|
| 1957 | 11.14 | 23.91 | 1.0 | 4.0 | 6.0 | 11.0 | 560.0 |

To further understand the distribution and identify potential outliers, we used a box plot visualization technique. This visualization allows us to observe the spread of the data without explicitly stating the values in the middle.


Figure 3.3: Box-plot of each classes (label vs word_count) - *youtube_sentiment*

To visualize the most frequent words from this dataset, we can plot word clouds for each label:

Negative Neutral Positive

Figure 3.4: Word clouds of *youtube_sentiment* dataset.

### 3.3.2 BASA_cricket

The passion for the sport of cricket is immense among Bengali-speaking individuals worldwide, leading to numerous discussions about the game. In order to leverage this popularity, we selected a dataset that not only aligns with the widespread interest in cricket but also exhibits the shortest average length of data points among the four datasets, approximately 8 words per data point.

Here are the label distribution for this dataset:

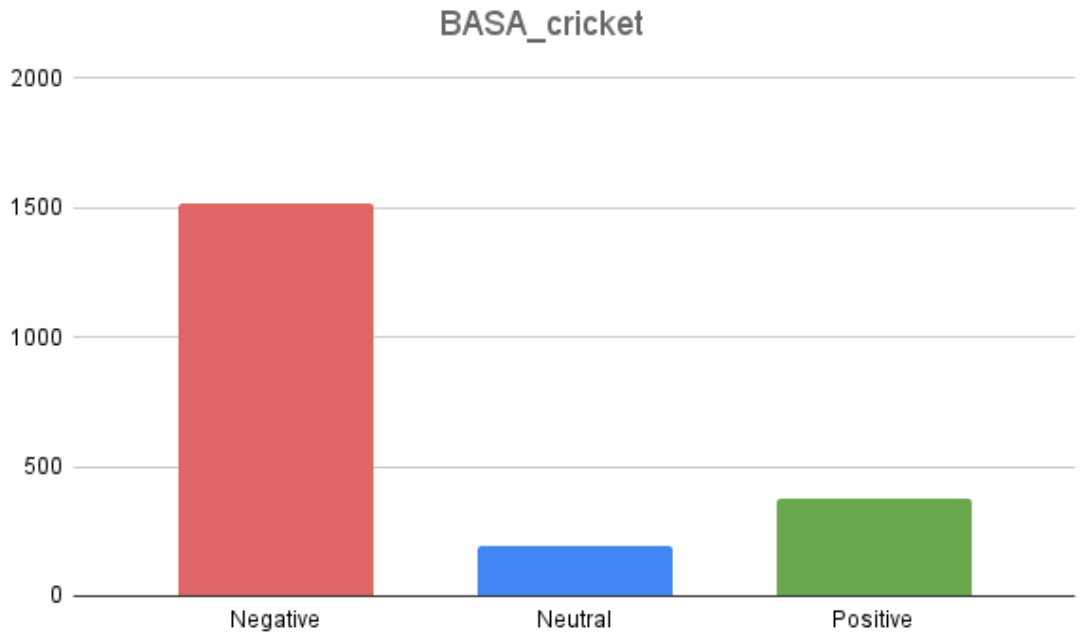


Figure 3.5: Class label distribution of *BASA_cricket*

It is evident that the ratio of negatively labeled data is considerably higher than that of the other two classes. Given that a significant portion of individuals who speak the Bengali language are from Bangladesh, the inconsistent performance of the Bangladesh Cricket Team may have significantly contributed to this disproportionate ratio.

To get some other aspects of the data we can see the data description as well.

| **data_points** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
|---|---|---|---|---|---|---|---|
| 2085 | 7.71 | 3.66 | 1.0 | 5.0 | 7.0 | 10.0 | 26.0 |

Provided are statistical measures for the 'word_count' of data points, including the mean, standard deviation, minimum, and quartiles (25%, 50%, 75%), along with the maximum.

If we employ a box plot for visualization, we can gain an overview of certain numerical values separately in each class.

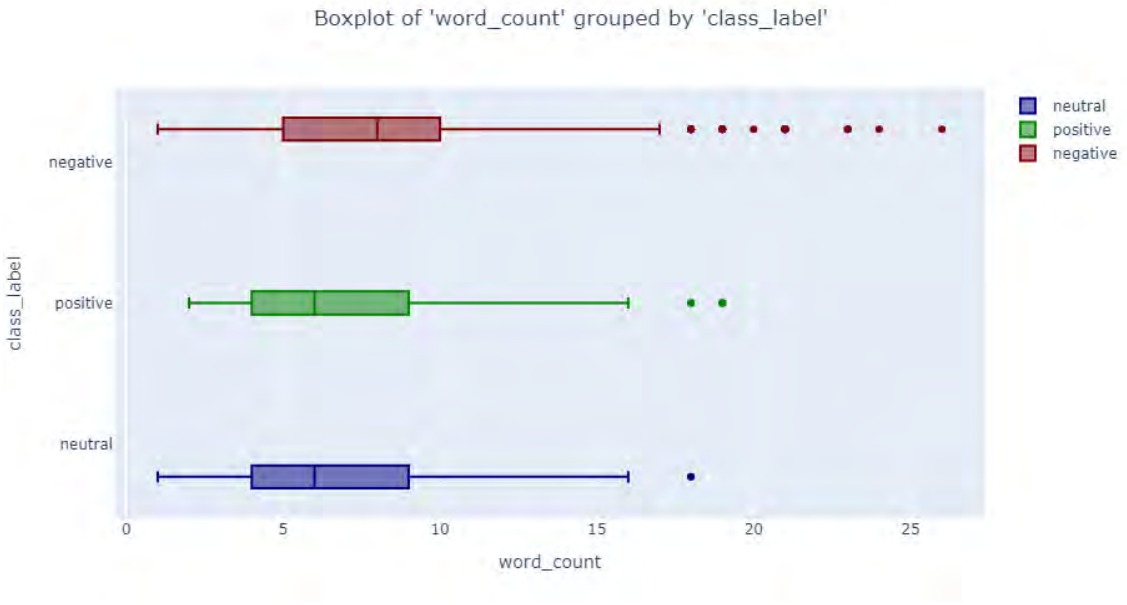


Figure 3.6: Box-plot of each classes (label vs word_count) - *BASA_cricket*

To visualize the most frequent words from this dataset, we can plot word clouds for each label:

Negative Neutral Positive

Figure 3.7: Word clouds of *BASA_cricket* dataset.

### 3.3.3 CogniSenti

The *CogniSenti* dataset is derived from the most widely used social media platforms globally, namely Facebook and Twitter. At any given moment, individuals are actively composing and sharing their life stories on these social media platforms. In contemporary times, expressing one's emotions and sentiments is best achieved through these digital mediums. Therefore, a dataset comprising such invaluable information has the potential to offer insights that may not be available in other datasets.

This dataset happens to possess the largest average length of data points among the four datasets we have examined, containing an average of around 16 words per data point. It comprises a total of 4599 data points, with 919 classified as negative, 2633 as neutral, and 1047 as positive.

Below is the class label distribution for the *CogniSenti* dataset:

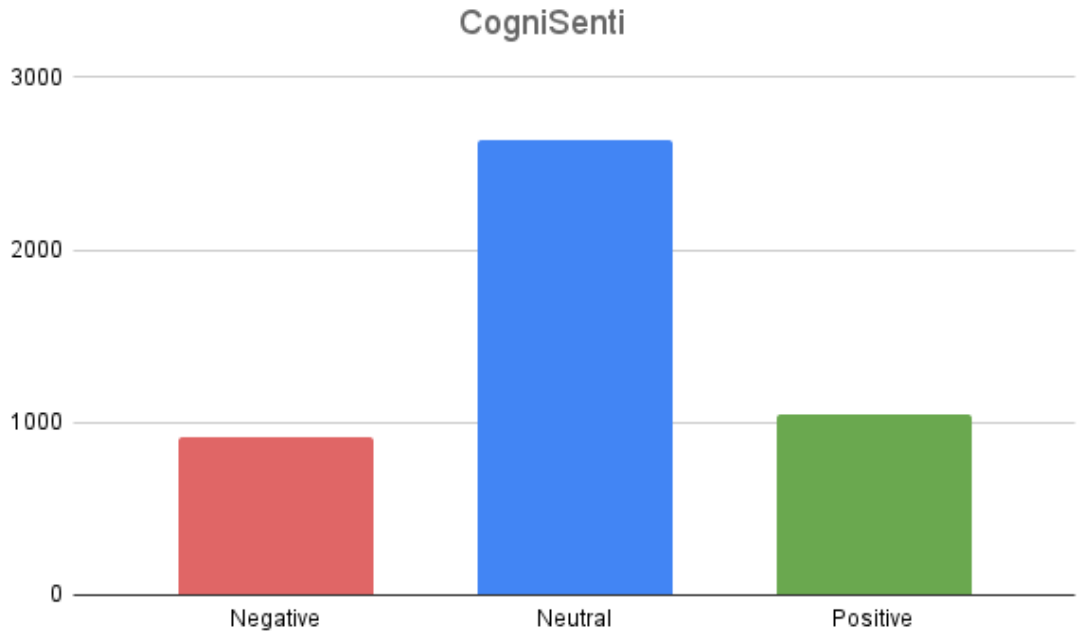


Figure 3.8: Class label distribution of *CogniSenti*

To get more understanding of the dataset, we can analyze the data description:

| **data_points** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
|---|---|---|---|---|---|---|---|
| 4599 | 16.32 | 9.64 | 1 | 13 | 16 | 18 | 136 |

Provided are statistical measures for the 'word_count' of data points, including the mean, standard deviation, minimum, and quartiles (25%, 50%, 75%), along with the maximum.

According to the data description, it is apparent that this dataset consists of data points that are highly expressive and likely carry significant sentimental value. To

provide a visual representation of the description for each class, the box-plot method can be employed.

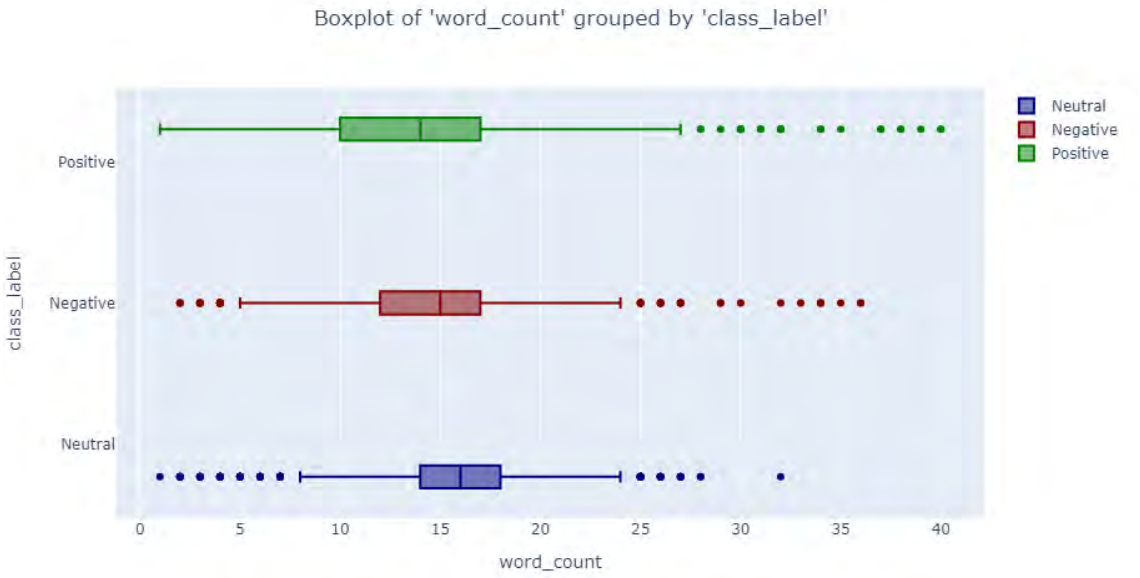


Figure 3.9: Box-plot of each classes (label vs word_count) - *CogniSenti*

To visualize the most frequent words from this dataset, we can plot word clouds for each label:

Negative Neutral Positive

Figure 3.10: Word clouds of *CogniSenti* dataset.

### 3.3.4 blp23

The *blp23* dataset was meticulously curated from the Bangla Language Processing workshop, and its credibility stands as a testament to its reliability. We opted for this dataset due to its sourcing from reputable platforms and its comprehensive coverage of a vast array of contexts across 13 distinct domains, including politics, education, agriculture, and many others. As we have done our study on youtube, social media, and cricket, we thought of a dataset that covers some other major topic that people convey their sentiment towards. The dataset is a compilation of the **MU**ltiplatform **BA**ngla **SE**ntiment (MUBASE) and SentNob datasets, which contributes to its diversity. Notably, the SentNob dataset primarily consists of public comments on various social media platforms, offering a spectrum of distinctive opinions on diverse content.

There are a total of 35266 data points in the dataset and 15767 are labeled as negative, 7135 are labeled as neutral and 12364 of them are labeled as positive.

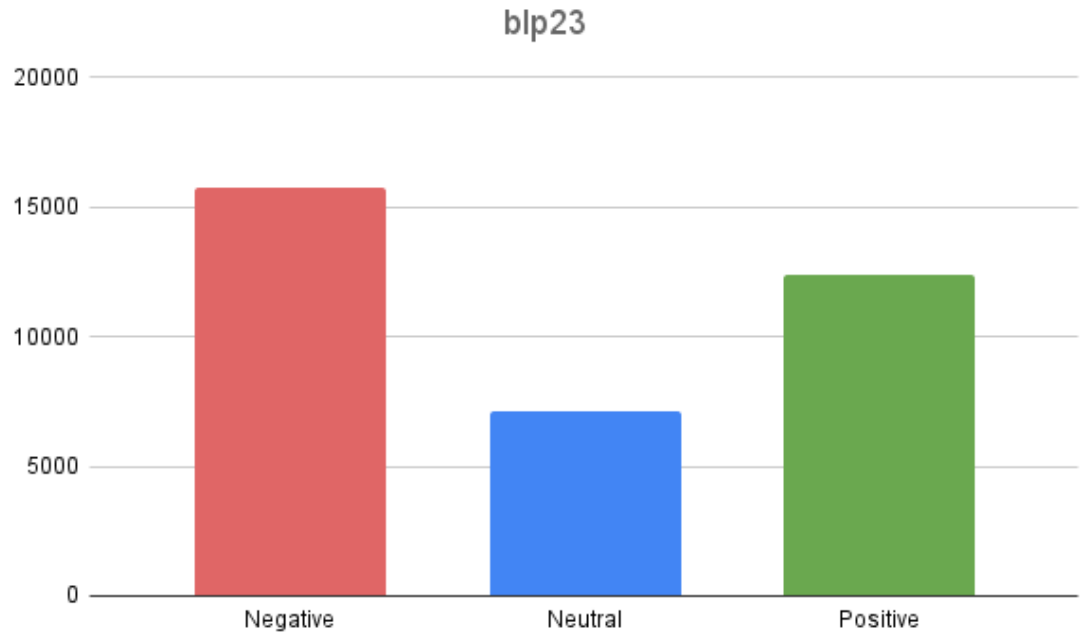


Figure 3.11: Class label distribution of *blp23*

To get more understanding of the dataset, we can analyze the data description:

| **data_points** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
|---|---|---|---|---|---|---|---|
| 35266 | 10.25 | 16.26 | 1 | 5 | 8 | 12 | 1687 |

Provided are statistical measures for the 'word_count' of data points, including the mean, standard deviation, minimum, and quartiles (25%, 50%, 75%), along with the maximum.

For individual labels, we can visualize through box-plot:

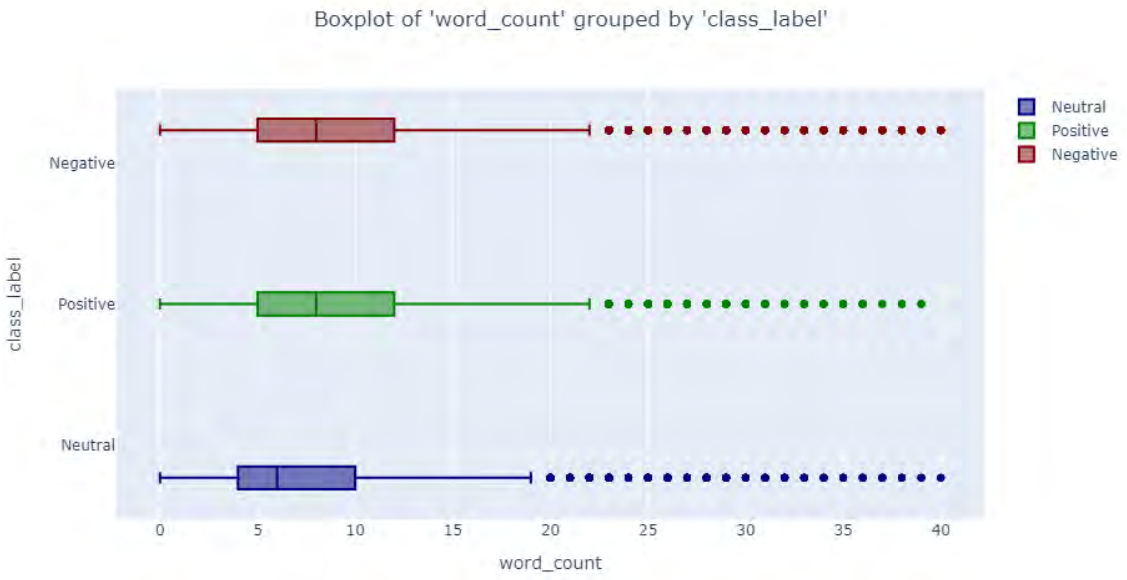


Figure 3.12: Box-plot of each classes (label vs word_count) - *blp23*

To visualize the most frequent words from this dataset, we can plot word clouds for each label:

Negative Neutral Positive

Figure 3.13: Word clouds of *blp23* dataset.

# Chapter 4

# Attack Methodology

Our research focuses on proposing sophisticated algorithms and techniques that can be utilized to augment datasets and attack uninterpretable models. The key to achieving these algorithms lies in comprehending the context and developing well-crafted adversaries that will contribute to the task. Due to the complexity of this task, we formulated multi-dimensional attack methodologies to address this issue.

## 4.1 Theoretical Framework

Generating context and semantically-aware adversaries with reasonable orthographic and lexical exploitation is critical for generative algorithms to create adversarial examples. Given the limited availability of tools for the Bangla Language, a low-resource language, we leverage existing tools developed by previous researchers to facilitate our adversarial attack algorithms. Our contribution involves introducing three distinct methods of adversarial attacks, which includes one at the word level and two at the sentence level, all achieved through a profound understanding of language models (LLMs) and their contextual meaning.

### 4.1.1 Bangla Paraphrase Tool

To come up with a sentence-level adversarial methodology, we employed the capabilities of a paraphrasing tool. We came across multiple possible methods of creating a paraphrase augmentation. However, we chose the csebuetnlp/banglat5_banglaparaphrase, a finetuned version of their own BanglaT5 model for providing diverse and context-aware paraphrases of Bangla sentences [20]. The high PINC score (0.65, 0.76, 0.80) ensured us of diverse adversaries, and the BERTScore(lower 0.91 - 0.93, upper 0.98) provided trustworthiness in preserving the coherence with the original text while being aware of the context in bigram to quad-gram awareness [20]. When paraphrased, the structure of a sentence usually changes along with the lexical state. By generating paraphrased adversaries, we expect to see some mispredictions or a decreased performance of LLMs by attacking certain models and understanding their capability of handling different sentence structures without changing the semantics.

### 4.1.2 Bangla Back Translation

Back Translation has always been one of the popular tasks in NLP. That is why we can come across many resources on Bengali translation. Translation creates a semantic bridge between two languages; however, it often fails to maintain the same sentence structure. We used this phenomenon to our advantage and got inspired by the adversarial attack, TransFool [24]. We tried to find a tool to provide us with the capabilities to achieve this methodology. So, the pipeline should look like this,

**Theoritical Idea of Bangla Back Translation**

Initial Language$_1$ ⇒ Intermediate Language ⇒ Initial Language$_2$

---

Here, Initial Language$_1$ and Initial Language$_2$ are in the same language but theoretically different in Structure/Lexicon

We chose English to be our intermediate language, as it has the most resources on Bangla to English translation. For this reason, we used 'csebuetnlp/banglat5_nmt-_en_bn and 'csebuetnlpbanglat5_nmt_bn_en. This pipeline of Back Translation should provide us with semantically close adversaries with different structures.

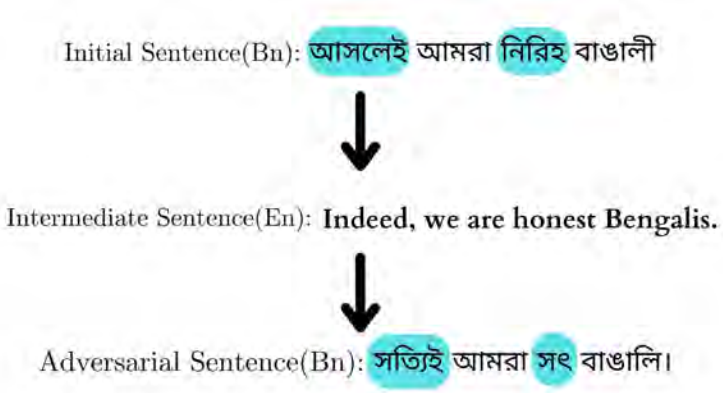


Figure 4.1: Back-translation illustrative idea

### 4.1.3 BERT Unmask

Regarding sentence classification, the classifier may disproportionately assign weight to a specific token, influenced by its frequent repetition in the Language Model (LLM) training dataset. This phenomenon introduces bias in LLMs, favoring the emphasized token and potentially affecting the model's overall performance and generalization. Our hypothesis for this attack development is to harness this bias and perturb it with another augmented text to obscure the model we are attacking. The masked language modeling capability of Bidirectional Encoder Representations from Transformers (BERT) provides us with the weapon to attack these LLMs. We identify these important aka. Biased tokens and replace them with the unmasked tokens we get from the BERT model. We experimented with multiple BERTs but

finalized with the xlm-roberta-base, pre-trained on 2.5TB of filtered CommonCrawl data containing 100 languages. It contains 525 Bengali tokens and 77 Bengali Romanized tokens, making a size of 8.4 GiB and 0.5 GiB [8]. As this dataset contains multiple languages, it provides us with a large domain of tokens and helps the attacking method to create much more profound unmasked tokens. Thus, contributing to the core of this word-level attack methodology.

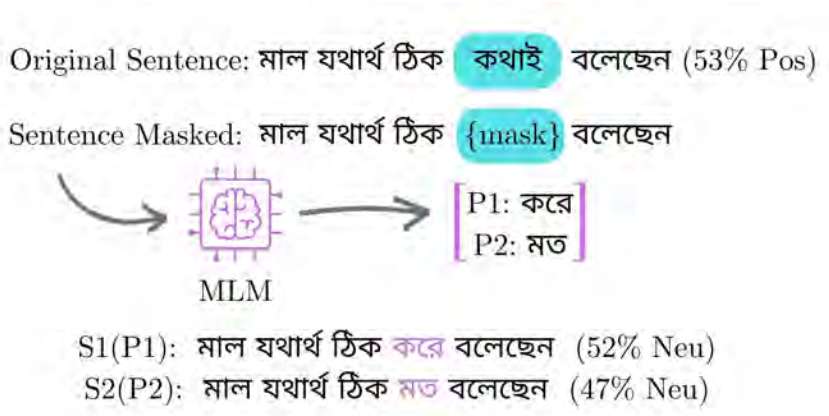


Figure 4.2: One-Hot BERT Perturb illustrative idea

## 4.2 Augmentation Recipe Development

All three attacks we propose in this paper are completely black-box attacks. These attacks take a model in the form of an object which takes a sentence and returns a (label, confidence score). None of the internal architectures of the models are traversed in these algorithms. This gives us a deep understanding of the models and their semantic and lexical understanding capabilities.

To construct the attack algorithm, we need multiple parameters that will be used to tweak the data point in such a way that the model can be perturbed. These are:

- model: An object, that takes a sentence and returns (prediction, prediction score)
- dataset: A pandas dataframe, with columns 'label' and 'text'
- labelMap: In case the model labels are different from the dataset, a dictionary to map the model labels to the dataset label to generalize the attack
- csvFileName: Path to the file the output will be stored in
- generate_report: will generate the report or not
- max_pass: The attack will attack at most the number of datapoints in the initial dataset (only for One-Hot Word Swap Attack)

The attacks are constructed below:

- **Bangla Paraphrase Attack**: Algorithm 01
- **Bangla Back Translation Attack**: Algorithm 2
- **One-Hot Word Swap Attack**: Algorithm 3 & 4 combined

### 4.2.1 Bangla Paraphrase Attack

**Algorithm 1:** Bangla Paraphrase Attack

**Input:** model, dataset, labelMap=None, csvFileName=None, generate_report=False
**Output:** dataset_attacked

```
initialize_columns(dataset)
for data point to dataset do
    init_pred, init_score := model(data_point['text'], labelMap)
    if data_point['label'] ≠ data_point['initPred'] then
        data_point['atckSuccess'] := 'Misprediction'
        continue
    augmented_text := paraphrase_augment(data_point['text'])
    atck_pred, atck_score := model(augmented_text, labelMap)
    data_point['atckPred'] := atck_pred
    data_point['atckScore'] := atck_score
    data_point['atckSent'] := augmented_text
    if data_point['atckPred'] == data_point['initPred'] then
        data_point['atckSuccess'] := 'Fail'
    else
        data_point['atckSuccess'] := 'Pass'
if generate_report then
    print_report(initial_model_performance(dataset))
    print_report(performance_after_attack(dataset))
save_to_csv(dataset, csvFileName)
return filtered_dataset(dataset)
```

### 4.2.2 Bangla Back Translation Attack

**Algorithm 2:** Bangla Back Translation Attack

**Input:** model, dataset, labelMap=None, csvFileName=None, generate_report=False
**Output:** [status, attacks]

```
initialize_columns(dataset)
/* Attacking                                                    */
for data point to dataset do
    init_pred, init_score := model(data_point['text'], labelMap)
    if data_point['label'] ≠ data_point['initPred'] then
        data_point['atckSuccess'] := 'Misprediction'
        continue
    augmented_text :=
     raw_back_translation_augment(data_point['text'])
    atck_pred, atck_score := model(augmented_text, labelMap)
    data_point['atckPred'] := atck_pred
    data_point['atckScore'] := atck_score
    data_point['atckSent'] := augmented_text
    if data_point['atckPred'] == data_point['initPred'] then
        data_point['atckSuccess'] := 'Fail'
    else
        data_point['atckSuccess'] := 'Pass'
/* Generating Report                                            */
if generate_report then
    print_report(initial_model_performance(dataset))
    print_report(performance_after_attack(dataset))
save_to_csv(dataset, csvFileName)
return filtered_dataset(dataset)
```

### 4.2.3 One-Hot Word Swap Attack

**Algorithm 3:** One-Hot Word Swap for a Single Sentence

```
   Input: model, tok, init_pred, init_score, label_map=None
   Output: dataset_attacked
   /* Prioritizing each candidate token                                  */
 1 for i ← 0 to length(tok) - 1 do
 2     new := join_tokens_except_i(tok, i)
 3     p, s := model(new, label_map)
 4     if p ≠ init_pred then
 5         score.append([i, replace_i_with_mask(tok, i), p, ∞])
                                  // Most significant perturbation candidate
 6     else
 7         if s < init_score then
 8             score.append([i, replace_i_with_mask(tok, i), p, s,
                round((init_score - s) × 100)])
   /* Sort the score list                                                */
 9 score := sort_score_by_last_element_descending(score)
   /* Initialize empty lists                                             */
10 attacks := empty list
11 status := 'Fail'
12 unmasker := create_pipeline('fill-mask', model='xlm-roberta-base')
   /* Iterate over scores                                                */
13 for i ← 0 to length(score) - 1 do
14     u := unmasker(score[i][1])
       /* Iterate over unmasked sequences                                */
15     for a in u do
16         p, s := model(a['sequence'], label_map)
17         if p ≠ init_pred then
18             status := 'Pass'
19             attacks.append((a['sequence'], p, s, 'Pass'))
                                  // Flip of Initial Prediction = Passed Attack
20         else
21             attacks.append((a['sequence'], p, s, 'Fail'))
   /* Sort and select the top 10 successful attacks                      */
22 attacks := sort_successful_attack(attacks)
23 attacks := take_top_10_successful_attacks(attacks)
24 return [status, attacks]
```

**Algorithm 4:** One-Hot Word Swap Attack

**Input:** model, dataset, max_pass, labelMap=None, csvFileName=None, generate_report=False
**Output:** dataset_attacked

```
                                                    // holding all the performed attacks
tokenizer := BasicTokenizer()
dataset['tok'] := dataset['text'].apply(tokenizer)
/* Initializing the DataFrame                                                    */
df := create_empty_dataframe(columns=['id', 'text', 'label', 'initPred',
 'initScore', 'atckPred', 'atckScore', 'atckSuccess', 'atckSent'])
/* Attacking                                                                     */
number_success := 0
for row to range(len(dataset)) do
    iP, iS := model(dataset.text[row], labelMap)
    if iP ≠ dataset.label[row] then
        df.loc[len(df)] := 'id': dataset['id'][row], 'text': dataset['text'][row],
         'label': dataset['label'][row], 'initPred': iP, 'initScore': iS,
         'atckPred': '-', 'atckScore': '-', 'atckSuccess': 'Misprediction',
         'atckSent': '-'
    else
        status, attacks := atck_sentence(model, dataset['tok'][row], iP, iS,
         label_map=labelMap)
        if status == 'Pass' then
            number_success := number_success + 1
            Total number of Successful Augmentation of Datapoints:
            number_success
        if number_success == max_pass then

        for a in attacks do
            df.loc[len(df)] := 'id': dataset['id'][row], 'text':
             dataset['text'][row], 'label': dataset['label'][row], 'initPred': iP,
             'initScore': iS, 'atckPred': a[1], 'atckScore': a[2], 'atckSuccess':
             a[3], 'atckSent': a[0]
; /* Generating Result                                                           */
if csvFileName ≠ None then
    dataset.to_csv(csvFileName)
if generate_report then
    Generate_Report()
return dataset
```

## 4.3 Attacked Model Description

### BERTs General Overview

ka05ar/banglabert-sentiment is a transformers-based classifier model, which is a fine-tuned version of "csebuetnlp/banglabert" for sentiment classification on the blp23 Task 2 dataset, as mentioned earlier. Banglabert is a large language model created by pretraining the base ELECTRA architecture, with 768 embedding sizes, 768 hidden sizes, 12 attention heads, 3072 feed-forward sizes, generator-to-discriminator ratio of 1/3, 110M parameters with 256 batch size for 2.5M steps on a v3-8 TPU instance on GCP[21]. The pretraining of Bangla gave the model a profound understanding of the Language itself. That is why it could be further finetuned for specific purposes with relatively small datasets like sentiment classification. So, the model by itself does not know the task of sentiment classification. Rather, it understands the whole Language, which gives the model an upper hand in creating the classifiers. That is why it can be challenging to attack such models. We used the sentiment classification pipeline of the huggingface library to construct the models.

Similarly, the Arunavaonly/Bangla_multiclass_sentiment_analysis_mode is a fine-tuned version of xlm-roberta-base [8]for sentiment classification. Unlike the ELECTRA it is different in the training process. On the one hand, ROBERTA is trained using the masked language modeling technique and Next Sentence Prediction(NSP). And, ELECTRA is trained to distinguish between the original and replaced tokens. This helps in more efficient use of the training data and leads to better model performance. These two model architectures give our recipes a challenge in fooling them.

The diagram below (4.3) shows the general process of Classification with BERT.

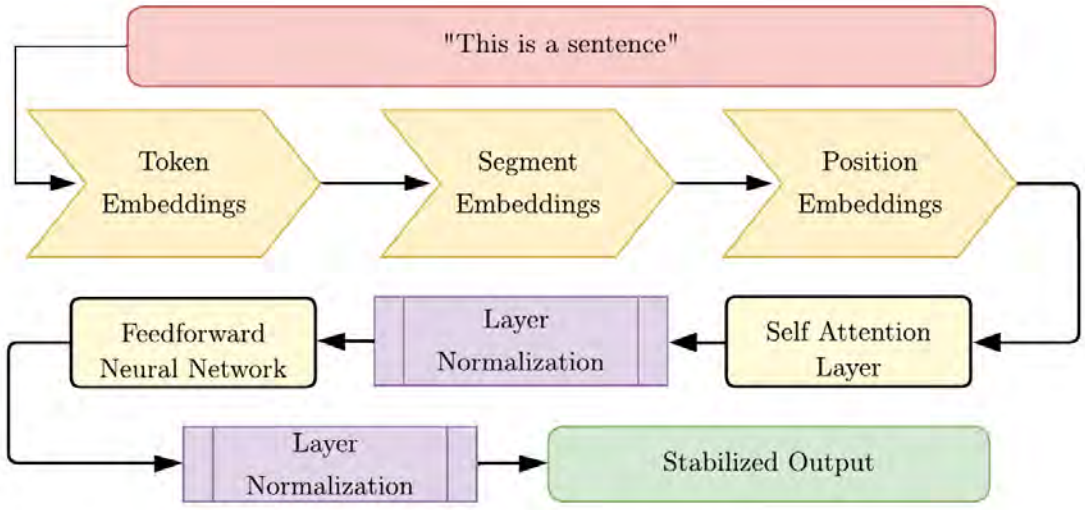


Figure 4.3: The Whole Architecture of Sentiment Classification

### Self-Attention and Feed-forward Neural Network

The attention layer computes the Attention Score. It is a scaled dot product that first takes the input sequence and transforms that input into three types of vectors:

1. Query Vector, Q
2. Key Vector, K
3. Value Vector, V

Then, the attention score is calculated by the following equation:

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Here, $d_k$ is the dimension of vector K.

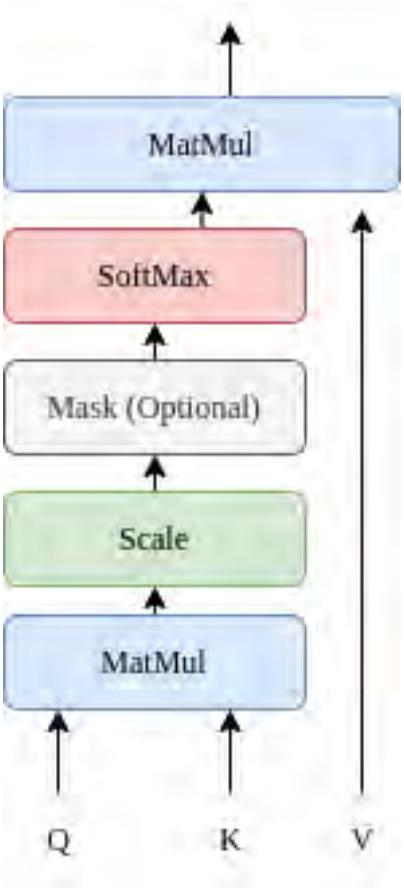


Figure 4.4: Workings of Attention Layer

After this process, the data are passed onto the FNN. The FNN consists of two linear layers with an activation function (ReLU) in between. The purpose of the FFN is to transform the token-level representations captured by the self-attention layer. It adds a non-linear element to the transformer encoder layer, allowing the model to learn more complex interactions between the tokens in the sequence.

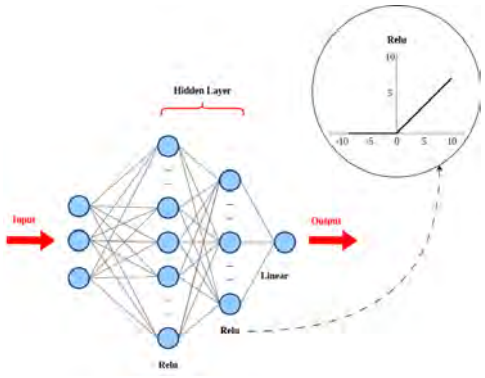


Figure 4.5: Neural Network & Relu Activation Function

In conclusion, the whole Transformer architecture can be described as 4.6.

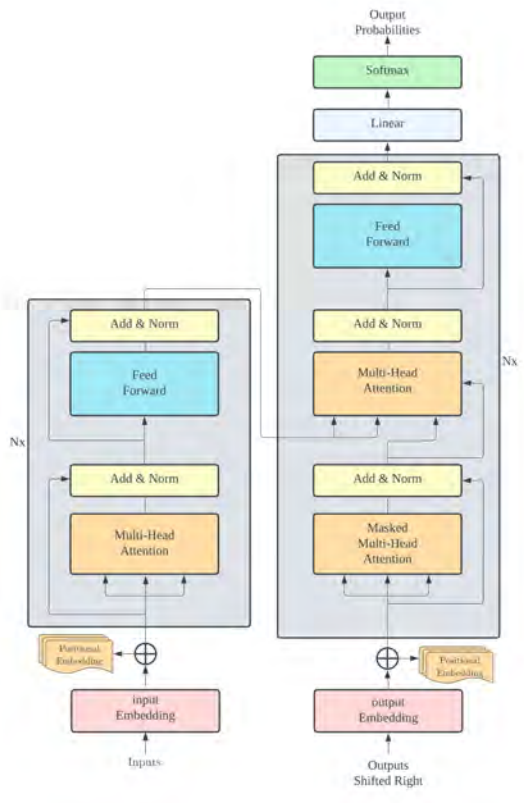


Figure 4.6: BERT Architecture

# Chapter 5

# Attack Result Analysis

## 5.1 Quantitative Analysis of Attacks

Following the development of the attacks, we proceeded to initiate an adversarial attack against some LLM models on sentiment classification, thereby reducing their effectiveness. Our first target was the BUET NLP Bangla Bert, which had undergone fine-tuning on a dataset known as blp23, primarily designed for ternary polarity sentiment classification. This dataset was created by Kawsar Ahmed (ka05ar). In each instance, we observed a noteworthy decrease in the initial score. We attacked the model with both an external dataset, a YouTube-sentiment dataset, a dataset, the model was not trained on, and an internal dataset, the dataset, the model was trained on. So, we could judge the attacks performance with both the domains the training dataset covered and the domains covered by the external dataset. A fascinating insight we found was that, when attacking the internal datasets, we could put the model with greater perplexity rather than the external dataset.

The details of this attack ka05ar/banglabert-sentiment[1] can be found in 5.2. In the case of One-Hot BERT Perturb, we could observe that it creates the most amount of adversarial examples; that is, for each attackable datapoint, at most ten adversaries were created, and reduced the performance of the model(F1 Score macro) by 37% on the External dataset and 40% on the internal dataset. Similarly, for the Bangla Paraphrase attack, we observed 18%(External Dataset) and 22%(Internal Dataset), and In Back translation, we observed a 22%(External) and 23%(Internal) reduction in the F1 Score. However, the relation between attachable data points

[1]https://huggingface.co/ka05ar/banglabert-sentiment

| Attack Methodology | Dataset | Attacks | | Initial Score | Attack Score | DIfference |
|---|---|---|---|---|---|---|
| | | Datapoints | % Successful | F1 Score Macro | F1 Score Macro | F1 Score |
| Bangla Paraphrase Attack | External | 304 | 12.83 | 100% | 82% | 18% |
| | Internal | 344 | 13.37 | 100% | 78% | 22% |
| Back Translation | External | 304 | 17.76 | 100% | 78% | 22% |
| | Internal | 72 | 13.89 | 100% | 77% | 23% |
| One-Hot BERT Perterb | External | 160 | 25.63 | 100% | 63% | 37% |
| | Internal | 190 | 30.53 | 100% | 60% | 40% |

Table 5.1: Attack on *k05ar/banglabert-sentiment*

| Attack Number and Attack | | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | Previous confidence of failed attacks | After Attack Average |
|---|---|---|---|---|---|---|---|---|---|
| Passed Attack Prediction Confidence | Mean | 51.6 | 50.75 | 52.49 | 52.89 | 50.36 | 45.16 | | 50.54 |
| | Median | 48.71 | 45.65 | 51.27 | 49.39 | 47.38 | 43.3 | | 47.62 |
| | Mode | 44.34 | 34.58 | 43.68 | 49.86 | 50.59 | 42.87 | | 44.32 |
| Failed Attack Prediction Confidence | Mean | 73.56 | 71.43 | 82.68 | 78.43 | 71.99 | 70.19 | **77.29** | **74.71** |
| | Median | 76.75 | 73.92 | 86.65 | 84.03 | 77.51 | 76.69 | **82.60** | **79.26** |
| | Mode | 54.58 | 65.4 | 76.65 | 85.91 | 52.89 | 79.3 | **74.49** | **69.12** |
| Misprediction Score | Mean | 58.85 | 59.74 | 63.2 | 57.63 | 58.85 | 57.81 | 59.65 | |
| | Median | 53.59 | 56.23 | 55.6 | 54.51 | 53.59 | 57 | 54.70 | |
| | Mean | 83.01 | 44.16 | 83.01 | 44.16 | 83.01 | 44.16 | 67.47 | |

Table 5.2: Before vs After Attack Confidence *ka05ar/banglabert-sentiment*

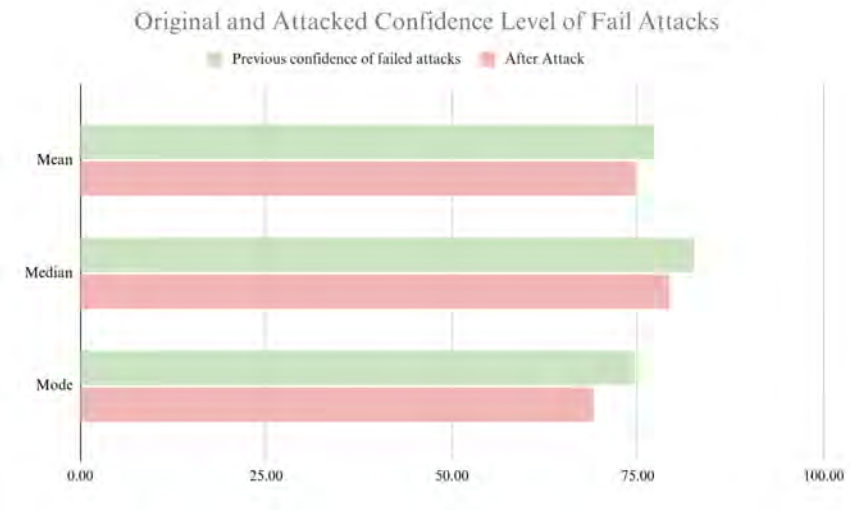


Figure 5.1: Confidence Reduction on Failed Attacks

and the adversarial examples generated is 1:1. From this analysis; we can consider that the world-level One-Hot BERT perturb attack is the most sophisticated attack as the mean reduction of performance is the greatest in this case among all the attacking methodology. This performance reduction is much expected, as the One-Hot Bert Perturb may sometimes be delusional and not always create a sound and non-inverting adversary, and the other sentence-level attacks attack the model of lexical and structural modification.

Three cases can emerge when attacking a model based on a data point. The attack may cause three cases for each data point: Pass, Fail, or Misprediction. For the passed attacks, the models prediction is inverted. However, the mean confidence level of the models prediction is around 50.54%, which can be questionable. This can be considered a state of perplexity or confusion for the model. Here, the actual goal of the attacks is achieved. Furthermore, we may not see a prediction label inversion when we look into the failed attacks. However, we still observe a reduction in confidence level, which could also be a subtle achievement of our attack algorithms. We observed that the mean performance of the model was reduced from 77.29% to 74.71%. When a model already makes a misprediction for a particular data point, we do not attack it. It can already be considered a passed attack.

To further observe the performance of our attack recipes, we initialized some more

attacks. The Performance generated in the table shows us that in all the cases the recipes were able to significantly decrease the performance of the models.

- Model Attacked
  - M1: Arunavaonly/Bangla_multiclass_sentiment_analysis_mode
  - M2: ka05ar/banglabert-sentiment
- Dataset
  - D1: Basa_Cricket
  - D2: cogniSenti

| Attack | Attack Number | Model | Dataset | Previous F1 | Attack F1 | % Successful Attacks |
|---|---|---|---|---|---|---|
| **Paraphrase** | 2.1 | M1 | D1 | 100 | 76 | 20.59 |
| | 2.4 | M1 | D2 | 100 | 74 | 24.07 |
| | 3.1 | M2 | D1 | 100 | 66 | 15.52 |
| | 3.4 | M2 | D2 | 100 | 74 | 21.62 |
| **Back Translation** | 2.2 | M1 | D1 | 100 | 77 | 23.53 |
| | 2.5 | M1 | D2 | 100 | 78 | 20.37 |
| | 3.2 | M2 | D1 | 100 | 66 | 20.69 |
| | 3.5 | M2 | D2 | 100 | 83 | 10.81 |
| **One-Hot Swap** | 2.3 | M1 | D1 | 100 | 68 | 29.38 |
| | 2.6 | M1 | D2 | 100 | 71 | 27.39 |
| | 3.3 | M2 | D1 | 100 | 53 | 30 |
| | 3.6 | M2 | D2 | 100 | 78 | 13.78 |

Table 5.3: Attack Performance on different models

## 5.2 Illustrative analysis

The examples we present for our attacks will include the following properties:

1. Original Sentence
2. Initial Prediction
3. Initial Score
4. Augmented Sentence
5. Prediction after Augmentation
6. Score after Augmentation
7. Percentage Delta
8. Verdict (Result)

The verdict for our attacks is binary, indicating whether the attack passes or fails. An attack is considered successful if the label after augmentation differs from the initial label, and it is deemed unsuccessful if the labels remain the same. However, relying solely on a binary verdict may lead to missing valuable insights. To extract more nuanced information from our research, we introduce a weight for failed attacks to assess their potential success. This is achieved through the introduction of a value **delta**.

The delta, $\delta$, is computed as follows:

$$delta, \delta = (initialScore - augmentedScore) \times 100$$

It's important to note that the delta value is relevant only for failed attacks, where the sentiments differ after augmentation.

Interpreting the equation, a positive delta value indicates that the augmented score is lower than the initial score. For instance, if a data point has an initial positive score of 0.89 and, after augmentation, the score becomes 0.54, the delta value would be 35. This implies that the model lost 35% confidence in the sentence being positive. Conversely, a negative delta value suggests that the model gained confidence, rather than losing it, after augmentation.

### 5.2.1 Bangla Paraphrase Attack

"BANGLA PARAPHRASE ATTACK" is an algorithm that performs data augmentation at the phrasal level. This algorithm will only change the structure of the sentence while retaining the original semantic meaning of the sentence. The data augmentation will only alter or insert expressions while having the least impact on the meaning of the original sentence. As stated previously, we used the "BUET NLP paraphrasing T5 Model" to paraphrase data in our selected datasets. The goal of this data augmentation is to confuse the model, put it into a state of perplexity, and study how the change in structure after augmentation affects the prediction of the model. This will provide some interesting insights and expose the models vulnerability. This will aid in improving the robustness of the model further. This data augmentation was done on two datasets, which are the *youtube_sentiment* dataset and the *blp23* dataset. Table 5.4 shows some augmentation results, which will demonstrate and help us understand this attack better.

From Table 5.4, example 1, we can see that the initial sentence is predicted to be negative with a confidence score of 0.90, which can be considered to be very high. However, after data augmentation, the model mispredicts the class label with a confidence score of 0.46. Here, the attack successfully confused the model because the sentence before and after the augmentation conveyed the same meaning, and both should have been negative sentiments. The two changes added to the augmented sentence are the word "একজন" before the word "সৌদি" and "পাগল" is transcribed in bengali instead of "Pagol" in the original text, which does not change the semantics rather makes the structure more profound and formal. This is a great example of the model's understanding of the Romanized text of Bengali. Similarly, in Example 2, the only change the augmenter introduces is orthographic variation, which fools

| | | Sentences | Prediction | Score | % $\delta$ \| Result |
|---|---|---|---|---|---|
| 1 | **Original** | শাহীন আখাউডা সৌদি প্রবাসী pagol | Negative | 0.90 | N/A \| P |
| | **Augmented** | শাহিন আখাউডা একজন সৌদি প্রবাসী পাগল। | Neutral | 0.46 | |
| 2 | **Original** | এই বটম জেস্চার মে বি প্রথম আনে ব্ল্যাকবেরি! | Neutral | 0.46 | N/A \| P |
| | **Augmented** | এই বটম জেসচারই প্রথম ব্ল্যাকবেরি নিয়ে আসে! | Positive | 0.66 | |
| 3 | **Original** | হ গেছে পুরাই | Negative | 0.41 | N/A \| P |
| | **Augmented** | সে পুরোপুরি ঠিক আছে। | Positive | 0.60 | |
| 4 | **Original** | আমার দেশ বাংলাদেশ । আমি গরবিত | Positive | 0.38 | -49.22 \| F |
| | **Augmented** | আমার দেশ বাংলাদেশ, আমি গর্বিত। | Positive | 0.88 | |
| 5 | **Original** | রাশিয়ায় বড় জয়ের পথে পুতিনের দল | Positive | 0.89 | 51.51 \| F |
| | **Augmented** | পুতিনের দল রাশিয়ার বিশাল বিজয়ের পথে। | Positive | 0.37 | |
| 6 | **Original** | সালাম স্যার | Positive | 0.82 | 18.79 \| F |
| | **Augmented** | সালাম স্যার। | Positive | 0.64 | |

Table 5.4: Bangla Paraphrase Attack Examples

the model into having increased confidence in the faulty prediction of the augmented sentences. This slight paraphrasing in the text should not have changed the initial prediction of the model. Thus, we can conclude that the model failed to identify the change in the structure of a sentence.

However, there are some complications in the augmented text. In certain cases, it can be observed that the augmentation does not hold the initial semantics consistency but rather can be found to be delusional. This is caused by the nature of the tools(LLMs) we used to create the augmentation attacks. This case can be observed in Table 1, example 3. Moreover, in examples 4 and 5, we can see that only after slight augmentation did the change in confidence score change by -49.22 and 51.51, respectively increasing both increasing the performance of the model. This is a huge change in score for only a slight change in sentence structure. This can be further noticed in Table 1, example 6, where only a "" at the end of the sentence changes model confidence drastically. Here, the model demonstrates a biased capacity for punctuation awareness, displaying an inability to sense or appropriately respond to the subtle difference conveyed through punctuation marks in the input text. These are great examples of how a model can be in a state of perplexity, and they help us to understand the internal biases and errors of Uninterpretable Models.

### 5.2.2 Bangla Back Translation Attack

After experimenting with this sentence-level back-translation attack with several data points, we observed that the sentences that have been retranslated back to the language of the original text will have slightly different sentence structures and minor altercations in meaning. This augmentation aims to confuse the model and exploit its vulnerability by making it mispredict the synthetic sentences. Exposing the vulnerability of models will help to understand the model's weakness better and increase its robustness in the future.

From Table 5.5, example 1, we can see that the model predicts the original sentence to be a negative sentiment with a confidence score of 0.78. After the original sentence was augmented, the model predicted the augmented sentence to be of positive sentiment with a confidence score of 0.51. If we look at the sentences before and after

| | | Sentences | Prediction | Score | % $\delta$\| Result |
|---|---|---|---|---|---|
| 1 | **Original** | জয় জনগন ঠেলা সামলা | Negative | 0.78 | N/A \| P |
| | **Augmented** | জয় জনতা, ধাক্কা সামলাও। | Positive | 0.51 | |
| 2 | **Original** | এক কথাই অসাধারন | Positive | 0.82 | N/A \| P |
| | **Augmented** | এটা অবিশ্বাস্য। | Neutral | 0.47 | |
| 3 | **Original** | ঘেউ! :P | Negative | 0.77 | N/A \| P |
| | **Augmented** | বার্ক। | Neutral | 0.48 | |
| 4 | **Original** | খুব ভালো লাগল | Negative | 0.91 | 44.9 \| F |
| | **Augmented** | এটা সুন্দর। | Negative | 0.46 | |
| 5 | **Original** | ভাই মাইয়্যা গুলোরে দেখাইলেন আর বীরপুরুষ ৩ জন কে কেন দেখালেন না ! তুইও মজা লস ভাই | Negative | 0.70 | 21.87 \| F |
| | **Augmented** | ভাই, কেন আপনি তাদের এবং তিন নায়ক প্রদর্শন না? | Negative | 0.48 | |
| 6 | **Original** | প্রকৃত মুক্তিযোদ্ধা | Neutral | 0.49 | 0 \| F |
| | **Augmented** | প্রকৃত মুক্তিযোদ্ধা | Neutral | 0.49 | |

Table 5.5: Bangla Back Translation Attack Examples

augmentation, we see that the augmented sentence still conveys the same meaning but, with a slightly different sentence structure and diversity in the lexicon. The sentence after augmentation is still of negative sentiment. Here, we can see that the model mispredicted the sentiment. Therefore, the model has failed. Example 2 again shows that the model fails similarly and changes its prediction from positive to neutral sentiment after augmentation. In both examples 1 and 2, our attack was successful, and we managed to put the baffle in the model. In the case of example 3, we can see that the model lacks an understanding of English in Bengali Transliteration, where "ঘেউ"(meaning barking sound) changes to "বার্ক"(Bark written in Bengali Script) and forcing it to mispredict. More examples demonstrate the same result, but only two have been picked for showcasing.

However, we faced some complications while experimenting with this attack algorithm. We noticed some sentences completely lost their meaning after augmentation. Yet, it is much expected, as we know, that the translation models(Bn to En and En to Bn) are T5, or Text-to-Text Transfer Transformers, are Transformer based encoder-decoder architecture that is utilized to create the algorithm, are not the best in terms of quality and sometimes gets dilutional spitting noisy translations(Table 5.5 example 5). Furthermore, in the case of short sentences, the attack algorithm seems to perform no(Table 5.5 Example 6) or a small amount of change. Nevertheless, the fewer modifications can also create a large impact, as observed in Rable 5.5 Example 4. Thus, we can conclude that the augmentations created in this attack algorithm hold semantics in a much more profound manner than the others.

### 5.2.3 BERT Base Masked Attack

The masked attack process stands out as one of the most intricate methods among the three developed attacks, intending to create larger synthetic data points. As outlined in the methodology, this process involves replacing a word in a sentence with a mask ("<mask>"). The mask is subsequently filled with another word, and the sentiment is evaluated by the models. The complexity of the attack increases significantly due to the intricate nature of the masking and unmasking processes. For a successful attack, the masked word must carry meaningful significance, and the unmasked replacement should possess a comparable level of significance. To mitigate the inherent probabilistic nature, we executed the mask and unmask process approximately 10 times per data point, adding an additional layer of complexity to the attack methodology.

Here are some examples to better understand the attack and how it gained success.

| | | Sentences | Prediction | Score | % $\delta$ \| Result |
|---|---|---|---|---|---|
| 1 | **Original** | আসলেই <আমরা> নিরিহ বাঙালী | Neutral | 0.37 | N/A \| P |
| | **Augmented** | আসলেই <একজন> নিরিহ বাঙালী | Positive | 0.62 | |
| 2 | **Original** | <বিদেশে> পড়ালেখা করছে বাংলাদেশের প্রচুর ছেলেমেয়েরা, তারা নিজেরাই নিজেদের কর্মসংস্থান করে নিচ্ছে। এতে করে দেশের অনেক উপকার হচ্ছে। | Positive | 0.53 | -33.89 \| F |
| | **Augmented** | <এখানে> পড়ালেখা করছে বাংলাদেশের প্রচুর ছেলেমেয়েরা, তারা নিজেরাই নিজেদের কর্মসংস্থান করে নিচ্ছে। এতে করে দেশের অনেক উপকার হচ্ছে। | Positive | 0.87 | |
| 3 | **Original** | মাল যথার্থ ঠিক <কথাই> বলেছেন। | Positive | 0.54 | N/A \| P |
| | **Augmented** | মাল যথার্থ ঠিক <করে> বলেছেন। | Neutral | 0.54 | |

Table 5.6: One-Hot Word Swap Attack

Here, two distinct types of examples are evident within the attacks, including a few ad-hoc instances. Let's delve into the analysis of these two types.

In Example 1, the mask was originally placed on the subject of the sentence, and post-augmentation, the mask was substituted with another subject. This type of masking example generally follows the Subject-Verb-Object (SVO) structure after mask replacement. However, in this specific case, our model exhibited a confidence of 0.62 in positive labels, mislabeling a neutral sentence as positive. This instance underscores how the model can misconstrue sentence meaning even when the structure remains consistent after augmentation. A similar case is observed in Example 2, where the subject was once again replaced with another subject. Although the model predicted the correct label in this instance, the confidence delta is notably high, with the model gaining an additional 33.89 percent confidence in positive due to a mere change in the sentence's subject. Notably, the subject, in this case, is not even the most significant component of the sentence. This pattern of delusional outcomes is observed consistently throughout the dataset with examples of this nature.

Moving to the second type of example, we encounter interesting scenarios where the sentence mask wasn't replaced by an identical one. In Example 3, the question "what did he say?" was originally answered by the sentence, but after augmentation, the sentence shifted to answering "how did he say?" Despite this alteration maintaining the original sentence's meaning, the model inaccurately predicted it to be neutral, while the original sentence was positive. This pattern of delusional errors by the model is recurrent for such examples across the dataset.

# Chapter 6

# Limitations & Future Work

While we have successfully constructed the attack recipes to assess the robustness of these models, the process was far from straightforward, and we encountered several limitations along the way. Some limitations are as follows:

Throughout our research, we struggled with the limited availability of resources and tools. The main reason for this is that Bangla is a low-resource language, hence Bangla tools are not readily available compared to a high-resource language like English. Finding high-quality datasets in Bengali with sufficient data points was challenging because of limited availability. Along with that, we failed to find a high-quality Bangla paraphrasing tool. As a result, many of the augmented sentences were incorrect. Furthermore, we used deep-learning LLMs to create our adversarial attack methodology. But, we noticed, due to some noise that might be present in the training datasets of the initial tools we used, the attack becomes delusional while creating the adversaries.

Our research was not straightforward like developing a model and finding its accuracy. It was quite a challenge to assign a numerical weight to our findings. Thus, to evaluate the data we tried our best to present the insight by introducing a delta value, but still it might be possible to show much more potential details that can reveal more about the working patterns of BERT models. Due to a lack of proper evaluation techniques, we faced limitations while fully expressing our research findings.

While augmenting data using multiple algorithms, we found it problematic to assess if the sentence was semantically correct after augmentation. While researching about it, we found the need for a dependency parse tree for the Bangla language, but unfortunately, there wasnt any usable dependency parse tree for the Bangla language.

As technology is evolving exponentially, we thrive to conduct more research to contribute to the evaluation. While we get rid of the limitations over time, we would like to delve more into our research in the future. Our plans for future study into this topic are as follows:

Adversarial analysis and attacks on LLMs are comparatively contemporary domains in Bengali language processing. Our current study focused on attacking transfer models to create a state of perplexion with augmented data and analyze the capabilities of the attacking algorithms. This study exposed the vulnerability of LLMs (Large Language Models), which have demonstrated superhuman performance. However, in our future research, we plan to emphasize creating a pipeline along with some new adversarial attack recipes that harness the specialties of the Bengali language, enhance the model's resilience, and make it more robust by providing adversarial training. We intend to work with Bengali linguists and the general public to create augmentation algorithms, usable tools (like parse trees, word cloud, etc.), and datasets that may help us create more profound adversarial examples. This research can help Bangla bypass the state of low-resource language and help the continual advancement of models' robustness, security, and comprehensibility.

# Chapter 7

# Conclusion

The rising prominence of Neural Networks, transformers, and related technologies in NLP has driven their exponential growth due to their supreme performance. Consequently, ensuring robustness and developing adversarial recipes have become crucial research challenges in NLP. To tackle this issue, we present a comprehensive pipeline with three distinct Bangla attack recipes that empower users to understand the robustness of their models against these adversarial attacks. These adversarial attack pipelines help us understand the internal state of a model in a descriptive manner which could be used to perform further research on increasing the robustness of a particular model. Throughout our research, we presented a new way of evaluating a model, in future, these attack pipelines will create the road to make better models with profound robustness.

# Bibliography


[1] M. T. Ribeiro, S. Singh **and** C. Guestrin, *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*, 2016. arXiv: 1602.04938 [cs.LG].

[2] J. Ebrahimi, A. Rao, D. Lowd **and** D. Dou, *HotFlip: White-Box Adversarial Examples for Text Classification*, 2018. arXiv: 1712.06751 [cs.CL].

[3] M. Gardner, J. Grus, M. Neumann **andothers**, *AllenNLP: A Deep Semantic Natural Language Processing Platform*, 2018. arXiv: 1803.07640 [cs.CL].

[4] J. Li, S. Ji, T. Du, B. Li **and** T. Wang, **?**TextBugger: Generating Adversarial Text Against Real-world Applications,**?** *CoRR*, **jourvol** abs/1812.05271, 2018. arXiv: 1812.05271. **url**: http://arxiv.org/abs/1812.05271.

[5] K. M. T. H. Rahit, K. T. Hasan, M. Al-Amin **and** Z. Ahmed, **?**BanglaNet: Towards a WordNet for Bengali Language,**?** **in***Global WordNet Conference* 2018. **url**: https://api.semanticscholar.org/CorpusID:20421263.

[6] M. A. Rahman **and** E. Kumar Dey, **?**Datasets for Aspect-Based Sentiment Analysis in Bangla and Its Baseline Evaluation,**?** *Data*, **jourvol** 3, **number** 2, 2018, ISSN: 2306-5729. DOI: 10.3390/data3020015. **url**: https://www.mdpi.com/2306-5729/3/2/15.

[7] N. Tripto **and** M. E. Ali, **?**Detecting Multilabel Sentiment and Emotions from Bangla YouTube Comments,**?** **september** 2018, **pages** 1–6. DOI: 10.1109/ICBSLP.2018.8554875.

[8] A. Conneau, K. Khandelwal, N. Goyal **andothers**, **?**Unsupervised Cross-lingual Representation Learning at Scale,**?** *CoRR*, **jourvol** abs/1911.02116, 2019. arXiv: 1911.02116. **url**: http://arxiv.org/abs/1911.02116.

[9] J. DeYoung, S. Jain, N. F. Rajani **andothers**, **?**ERASER: A benchmark to evaluate rationalized NLP models,**?** *arXiv preprint arXiv:1911.03429*, 2019.

[10] D. Kaushik, E. Hovy **and** Z. C. Lipton, **?**Learning the difference that makes a difference with counterfactually-augmented data,**?** *arXiv preprint arXiv:1909.12434*, 2019.

[11] R. T. McCoy, E. Pavlick **and** T. Linzen, *Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference*, 2019. arXiv: 1902.01007 [cs.CL].

[12] M. Gardner, Y. Artzi, V. Basmova **andothers**, *Evaluating Models' Local Decision Boundaries via Contrast Sets*, 2020. arXiv: 2004.02709 [cs.CL].

[13] X. Han, B. C. Wallace **and** Y. Tsvetkov, **?**Explaining black box predictions and unveiling data artifacts through influence functions,**?** *arXiv preprint arXiv:2005.06676*, 2020.

[14] M. A. Hasan, J. Tajrin, S. A. Chowdhury **and** F. Alam, **?**Sentiment Classification in Bangla Textual Content: A Comparative Study,**?** *CoRR*, **journvol** abs/2011.10106, 2020. arXiv: 2011.10106. **url**: https://arxiv.org/abs/2011.10106.

[15] J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin **and** Y. Qi, **?**Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp,**?** *arXiv preprint arXiv:2005.05909*, 2020.

[16] M. T. Ribeiro, T. Wu, C. Guestrin **and** S. Singh, **?**Beyond accuracy: Behavioral testing of NLP models with CheckList,**?** *arXiv preprint arXiv:2005.04118*, 2020.

[17] K. Goel, N. Rajani, J. Vig **andothers**, **?**Robustness gym: Unifying the nlp evaluation landscape,**?** *arXiv preprint arXiv:2101.04840*, 2021.

[18] D. Kiela, M. Bartolo, Y. Nie **andothers**, *Dynabench: Rethinking Benchmarking in NLP*, 2021. arXiv: 2104.14337 [cs.CL].

[19] E. Wallace, S. Feng, N. Kandpal, M. Gardner **and** S. Singh, *Universal Adversarial Triggers for Attacking and Analyzing NLP*, 2021. arXiv: 1908.07125 [cs.CL].

[20] A. Akil, N. Sultana, A. Bhattacharjee **and** R. Shahriyar, *BanglaParaphrase: A High-Quality Bangla Paraphrase Dataset*, 2022. arXiv: 2210.05109 [cs.CL].

[21] A. Bhattacharjee, T. Hasan, W. Ahmad **andothers**, **?**BanglaBERT: Language Model Pretraining and Benchmarks for Low-Resource Language Understanding Evaluation in Bangla,**?** **in***Findings of the Association for Computational Linguistics: NAACL 2022* M. Carpuat, M.-C. de Marneffe **and** I. V. Meza Ruiz, **editors**, Seattle, United States: Association for Computational Linguistics, **july** 2022, **pages** 1318–1327. DOI: 10.18653/v1/2022.findings-naacl.98. **url**: https://aclanthology.org/2022.findings-naacl.98.

[22] M. T. Ribeiro **and** S. Lundberg, **?**Adaptive Testing and Debugging of NLP Models,**?** **in***Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* Dublin, Ireland: Association for Computational Linguistics, **may** 2022, **pages** 3253–3267. DOI: 10.18653/v1/2022.acl-long.230. **url**: https://aclanthology.org/2022.acl-long.230.

[23] M. A. Hasan, F. Alam, A. Anjum, S. Das **and** A. Anjum, *BLP 2023 Task 2: Sentiment Analysis*, 2023. arXiv: 2310.16183 [cs.CL].

[24] S. Sadrizadeh, L. Dolamic **and** P. Frossard, *TransFool: An Adversarial Attack against Neural Machine Translation Models*, 2023. arXiv: 2302.00944 [cs.CL].