**CS 590 ETHICAL FALL 2024 EC1 ADVERSARIAL ATTACKS**

**DRASHTI SNEHALKUMAR PATEL**

**Homework Understanding :**

This report builds on our previous homework by taking the basic text classification model and adding adversarial training. We use the same untargeted attack methods from earlier to generate extra training examples and improve the model's robustness.

---

For the dataset , we use the rotten_tomatoes dataset (train and test ) which contain movie reviews .

We are reusing functions from our hw 4 and adding on other functions also as asked .

**1. Train_model()**

I use this method to load my training and testing data, apply TF-IDF vectorization, and train a baseline Logistic Regression model. It is my baseline, establishing the performance of the original model on pristine data. It provides me with the baseline against which I can compare the improvement after adversarial training.We took this from the hw4.

**2. character_modify()**

From hw4 I implemented this function to add random character-level changes—swap, replace, or remove characters, within a target word.

It is an essential function since it simulates the kind of minimal text perturbations (like typos) that may mislead a classifier. Its randomness helps to simulate real iNput error simulation.

**3.untargeted_attack()**

From changes originating from character_modify, this function applies the attack to every word in a piece of text with a fixed 40% probability.

I use untargeted_attack to generate adversarial examples from the clean text. Adversarial examples are a necessity while testing the vulnerability of the model and when employing them to enhance the training set for adversarial training.

**4.apply_untargeted_attack_to_dataset()**

This function uses the untargeted attack on the whole test dataset and stores the altered texts in a CSV file. I create the adversarial test data once to maintain consistent evaluation across all experiments. This gives a fixed test set to evaluate the model's performance on perturbed data.

**5. adversarial_training()**

At the core of my approach, this function constructs a new training CSV that contains the original training instances and k adversarial instances generated for each instance with untargeted_attack.

I ensure that only training data is used here to maintain things clean and distinct from the test set. This augmented dataset is intended to retrain the model so that it is more resilient against the kind of perturbations the attack introduces.

**6. train_adversarial_model()**

This function trains a fresh Logistic Regression model on the adversarial training data formed in the earlier step.

I also create a fresh TF-IDF vectorizer here so that the model learns from the augmented dataset. This provides us with an adversarially trained model which I hope will perform better on perturbed inputs.

**7.evaluate_model()**

This is the aggregate estimation function where I run experiments on a set of k values ranging from 1 to 5 and take average results over multiple runs. First, I try the original model on the original and adversarial test data as a baseline.

Next, I generate adversarial training data, train a fresh model, and test its accuracy for each value of k. By averaging results across multiple runs, I can preserve the randomness that comes with generating adversarial examples.

This function makes it easy to directly compare how varying the number of adversarial examples per instance (k) affects model robustness.

**Results produced:**

| Model | Original Test accuracy | Adversarial Test accuracy |
|---|---|---|
| **Original (HW4)** | 0.7833 | 0.7092 |
| **Adversarial (k = 1)** | 0.7807 | 0.7182 |
| **Adversarial (k = 2)** | 0.7833 | 0.7263 |
| **Adversarial (k = 3)** | 0.7824 | 0.7349 |
| **Adversarial (k = 4)** | 0.7807 | 0.7370 |
| **Adversarial (k = 5)** | 0.7867 | 0.7351 |

The accuracy values are averaged over 5 runs per k value.

**Inferences**

**1. Overall best model**

Based on the result of the experiments with various k values (1 to 5), the best model seems to be the one that was trained with k = 5.

The average accuracy on the original test set was 0.7867, while on the adversarial test set, it was 0.7351. The model with k = 5 consistently outperformed models with smaller values of k.

The main takeaway is that training with higher numbers of adversarial examples (higher k) improves the model's generalization and resilience to perturbations.

As k is higher, the model is increasingly exposed to more variations in training data, and this seems to make it more resilient to adversarial attacks, producing a better overall performance on original and adversarial test sets both.

This shows that higher adversarial exposure through adversarial training improves model performance on adversarial examples without compromising competitive accuracy on the source dataset.

**2. Is adversarial training seeming to work, why or why not?**

Adversarial training is indeed working, as can be seen from the increase in adversarial test data performance with increased k values.

The adversarially trained models always perform better than the original model on the adversarial test set. For example, when k = 5, the adversarial model had an accuracy of 0.7351 on the adversarial test data, which is greater than the original model's accuracy of 0.7092.

**It is helpful for the following reasons:**

The adversarial training process enhances the dataset by introducing perturbations, simulating real-world input mistakes. Training on original and adversarial examples makes the model stronger, identifying patterns even when the data is slightly altered. This exposure allows the model to learn generalizable features that are not overly sensitive to small perturbations in the text, resulting in improved performance when given adversarial inputs.

But the model's accuracy on adversarial test data is still lower than its accuracy on clean data, i.e., while adversarial training makes things better, it does not completely eliminate us from being vulnerable to adversarial attacks.

**There are a number of reasons why adversarial training might not solve the problem completely:**

Even though we're training the model with adversarial examples, a small number of changes (k) might not be enough to account for all the possible ways an attack can happen. Increasing k helps, but there might still be situations or attacks that aren't represented by the examples that we generate.

Our untargeted attack only modifies the text in certain limited ways (swapping, substituting, and removing characters). This does not account for all possible attacks that may take place.

Logistic Regression is a simple model that can work well for many tasks, but it might not be sufficiently robust to withstand adversarial attacks in NLP. More complex models like deep neural networks might work better so they canbe explored.

If the model is exposed to a large number of adversarial examples that are significantly different from real-world data, then the model can end up learning too much from the examples leading to bad performance .

**Improvement possibilities:**

Use more advanced attack strategies, like word substitutions or others might better simulate real-world attacks.

Employ techniques like back-translation, paraphrasing, or synonym substitution to increase the training data and condition the model to generalize better .

Train multiple models on different subsets of adversarial data or perturbation methods, and combine their predictions might cause improvements.

Use more powerful models like RNNs, CNNs, or Transformers, which comprehend text context better and are more resistant to simple character-level attacks.

Use regularization techniques for training.

**Challenges and issues**

1.Generating adversarial examples for each instance and training the model on augmented data can be computationally expensive and time-consuming. This could be particularly challenging when working with large datasets

2.Coding challenges were significant, but I managed to address them and complete the task effectively.
3.Understanding and working with k values so as to check and track its impact.

**Conclusion:**

We applied adversarial training in this project to enhance the robustness of a Logistic Regression model against text-based adversarial attacks. Including adversarial examples in the training set resulted in improved performance on adversarial test sets, especially with higher k values. Although the adversarially trained models were more resilient than the base model, they remained worse than the base on the original test set. This implies that while

adversarial training is helpful, it does not eliminate vulnerabilities altogether. Future developments can include more diverse adversarial examples, more advanced models, and regularization to prevent overfitting.

Note: Use of Ai made for coding and concept clearance help.