CS 590 ETHICAL FALL 2024 HW4 ADVERSARIAL ATTACKS

DRASHTI SNEHALKUMAR PATEL

Homework Understanding :

The main goal of this homework is to understand how adversarial attacks can mislead machine learning models, especially text classification and how the attacks trick the model into making incorrect predictions. It also allows us to understand how vulnerable models can be to small changes and highlights the importance of robustness in machine learning models.

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

Part1:Prepare the Classification Model(def train_model(train_data_path, test_data_path))

My approach:

This was the most straightforward part of the entire homework as we had to simply train our logistic regression model on the Rotten Tomatoes training dataset.

To prepare the text data for the model, we used TF-IDF vectorization, which converts the text into numeric features that are accepted by the model. I like to work with TF-IDF because it's a good and simple way of text data representation that compensates the importance of crucial words without weakening their impact by frequent irrelevant words.

Following model training, we did a test run on the test dataset, which was also processed with the same TF-IDF vectorization technique.Further, we compare the model's accuracy in both the training set and test set to see its performance. The model had a training accuracy of 0.8949 and a test accuracy of 0.7833, which indicates that it performed well with the training data but suffered from some loss in performance when the model was applied to unseen data.

Results produced:

Model Name	**Training Accuracy**	**Testing Accuracy**
Logistic Regression	0.8949	0.7833

Difficulties faced during the implementation of the model

In Part 1, there were no significant issues that we faced in the ,model building of the Logistic Regression model. It was a relatively straightforward process, and the TF-IDF vectorization worked correctlyy iin converting the text data into numerical feature

—---

Part 2:Prepare Character Level Modification (def character_modify(word))

My approach:

Here, we have declared the character_modify(word) function, which randomly applies one of three character-level operations on a word: swap, substitute, or delete. The function randomly applies one of the three operations to all words it processes.

The swap operation swaps two adjacent letters in the word as explained to us as "great" to "graet".

The replace modification replaces some characters with predefined symbols, like replacing "a" with "@".

Further, the delete modification removes a random character from the word like great to grat. This approach introduces randomness and variability, simulating the way that small changes in the text may impact a model's predictions and we do this using the random library.

Our chosen 3 character level modification are as follows-

Swapping

The swap substitution randomly interchanges two adjacent characters in the word. And is useful to simulate typo errors or misprinted words, which might deceive a model.
We use the random.randint() function to select a random index, and then interchange the characters at the index and the next one. For example, "great" might become "graet" or "gerat" depending on which surrounding characters are interchanged.

The substitute modification replaces some of the word's characters using pre-defined symbols or numbers. It is regularly used to mimic normal text variations, such as replacing letters with numbers or special characters.

We created a dictionary that takes some characters and maps them to their substitutes like "a" changes to "@", "e" changes to "3", etc. The function goes through each character of the word, and if it is one of the characters that are saved in the dictionary, it substitutes the character. For instance, "great" could be converted to "gr3@t" or "gr3a+".

Deletion

The third modification method used is the delete modification which removes a random character from the word. This simulates errors where characters are accidentally omitted.

We again use the random.randint() method to select a random index and remove the character from this index. Like our examples given “great" could be shortened to "gret" or "geat" depending on what is removed.

We selected these three modifications since they closely resemble common text errors that can occur in real-world scenarios, such as typos, character substitution, or deletion. These modifications introduce controlled noise to the text, which is utilized to verify the robustness of the model to small but critical changes in the input.

These values were chosen randomly to make sure the changes to the text aren’t too extreme or too large. The goal was to create small, random changes that would challenge the model but not make the text unrecognizable.

We iterate 4 times to get our modified results for the 2 words - great and movie-

ITERATION	GREAT	MOVIE
1	gr3@t	m0v13
2	gr3@t	move
3	rgeat	mvoie
4	grea	omvie

From the 4 iterations of editing the words "great" and "movie," we can see that the edits increasingly transform the words. For "great," the edits are mostly character substitutions and deletions. For "movie," we see the same edits with substitutions, but the words remain identifiable, showing how small edits can still challenge the model without completely changing the text.

All the 4 outputs show variation and hence are correct in their nature showing randomness correctly

—---

Part 3:Untargeted Attack(untargeted_attack(text))

My approach:

I started by understanding what an untargeted attack is, which is a type of adversarial attack where one makes random modifications to the input data without any specific target. After I had grasped the idea, I proceeded to implement the attack by making random character-level transformations to the text words. The objective was to see how small, random adjustments to the text could be a challenge for the model to classify correctly.

By assigning a 40% chance of change to each word and then testing the model on the changed text, I was able to see how the model would do when these changes were made. I ran the attack 5 times to ensure that the results were not a fluke and to see how much the model's performance would vary when faced with these slight changes.

The untargeted attack had three major stages which we workedon

Text Transformation in which text was broken down into individual words, and in every word, there were 40% probabilities that a transformation would be performed. The transformations were swapping two characters side by side, replacing characters with alike-looking symbols (e.g., 'a' to '@'), or deleting a random character. This made it look like real-world text mistakes such as typos or character substitutions that might occur in actual texts.

Dataset Alteration- Once the method of altering individual words was defined, the entire dataset was altered. All the text in the dataset underwent the untargeted attack, wherein words were changed according to the given rules. The changed texts were stored in a new file to maintain the original data for further comparison.

Model Evaluation-The modified dataset was employed to evaluate the trained Logistic Regression model in Part 1. The accuracy of the model was evaluated by running the attack five times, and its average accuracy was calculated to measure the effect of the attack on the model's performance.

Brief description of our functions which we created for this part-

untargeted_attack function()

There was a function that was tasked with altering every word in an input text by a random 40% chance. The changes applied were either a swap of characters, replacement, or erasure. This function had a direct influence on the input text since it changed the words in a manner that replicates natural, insignificant text errors.

apply_untargeted_attack_to_dataset function()

The function utilized the original test dataset, executed the untargeted_attack function for every text, and saved the attacked texts in a new file. It was the purpose to create a new dataset of perturbed texts to be used in testing the robustness of the model.

evaluate_model_on_modified_texts function()

This function evaluated the performance of the model on the manipulated texts. The model was tested for five runs, and its accuracy for each run was recorded. The average accuracy was calculated to find the overall impact of the untargeted attack on the performance of the model. This function provided the metric needed to confirm the robustness of the model against adversarial attacks.

These functions together examine how random changes in characters impact the performance of the model to correctly classify text, giving insight into how well the model withstands.

Result:

Our 5 run accuracy table is as follows:

Run	**Accuracy**
Run1	0.6604
Run2	0.6763
Run3	0.6744
Run4	0.6547
Run5	0.6716
Average Accuracy	0.6675

Table for unattacked text and attacked text accuracies -

Dataset	**Training Accuracy**	**Testing Accuracy**
Original (Unmodified)	0.8949	0.7833
Modified (Attacked)		0.6675 (from 5 runs)

Inferences

The model's accuracy dropped from 0.7833 (original test accuracy) to 0.6675 (average after untargeted attack). This shows that random changes in the text heavily influence the performance of the model, reducing its accuracy by approximately 14%.
The untargeted corruption, which randomly changes 40% of the words, introduces noise to the input text. That created a substantial accuracy decrease, demonstrating the sensitivity of the model to small changes in text.

Although the decline, the model is still superior to random chance, but it highlights that the Logistic Regression model with TF-IDF features is not highly robust against adversarial inputs like random text modification.

The results show that Logistic Regression is quite sensitive to perturbed inputs and fails to generalize when the text is perturbed. The attack demonstrates that the model may be over-relying on specific words or phrases and hence vulnerable to minor perturbations.
The untargeted attack effectively challenged the performance of the model, illustrating areas for improvement in adversarial input management. Future studies may involve strengthening robustness through advanced methods and more robust models.

Modified data samples

Text	Label
weighty @nd ponderous ubt every ibt @$ filling as he treat of he title .	1

1f not a home run , then at least a solid b@$3 hit .	1
th3 film truly does erscue [the unk br0th3r$] from motown’s shadows . it’ abot tie .	1
weighty @nd ponderous ubt every ibt @$ filling as he treat of he title .	1
everyone’s insecure in lovely and amazing , a poignant nad rwyly amusing film about mothers , daughters and their relationships .	1

Analysis

The modification of the text to include the untargeted attack has a significant impact on the readability of the text and on whether it is correctly classified or not. We can see a lot of changes in the text in the new untargeted-attack_rotten-tomatoes.csv dataset .The attack incorporates random character-level manipulation in approximately 40% of the words of each sentence, including substitutions, deletions, and character transposition. These changes, while minimal, disrupt the text's continuity and bring about random errors that make the text less readable, as in words like "ubt" for "but", or "rwyly" for "really".

From a readability perspective, the changes cause faint distortions that affect the coherence of the sentences. The key words that play a crucial role in understanding the meaning or emotion of the sentence are being changed, hence creating it challenging for the model to read the text accurately. For instance, within the sentence, "everyone's insecure in lovely and amazing, a poignant nad rwyly amusing film…," the modifications of "insecure", "amazing", and "rwyly" render the text semantically indeterminate, hence more challenging for the model to identify the implied meaning of the text.

These alterations also change the model's reliance on special keywords. The words "amazing", "insecure", and "filling" are essential to the model for classification based on sentiment, but through altering these words, the random attack disrupts the model's ability for an accurate classification. The randomness of the attack suggests that key words are altered in a way so as to introduce uncertainty into the model's decision.

In general, the untargeted attack shows that small, random alterations of the input text leads to affecting our model. While the sentence structure is not violated, the random alterations affect readability and the model's ability to classify the texts correctly. This discussion illustrates how the model can be misled by simple adversarial manipulations, and the need for more robust models that can effectively process noisy or manipulated text inputs.

—---

Part 4: Targeted Attack

Approach

In Part 4, our goal is to prepare a more specific attack that targets the words that the model relies on for prediction specifically. This is how we proceeded-

def greedy_select()

The first step in the targeted attack was identifying the most important words that the model relies on for classification. This was done by observing the extent to which the model's probability for the correct label is affected upon removing a word.
We iterated through each word in the text, removed it one at a time, and checked how the model’s prediction changed. Words causing the largest drop in probability were considered most impactful and were prioritized for modification.

def targeted_attack()

After identifying the most important words, the next step was to modify these words using character-level modifications previously used. The modification continued until the model’s prediction changed significantly.
If the model’s prediction was still correct, we continued to modify the next most important word based on the greedy selection.

Def apply_targeted_attack_to_dataset()

The targeted attack was applied to the whole dataset. For each text, we first checked if the model was classifying it the way it was supposed to. If not, the text did not change. If the text was classified as it should, we applied the targeted attack.

The updated text was saved to a new file, keeping the same structure as the original dataset ,i.e., the text column was updated with the updated text and the label was not altered.

def evaluate_model_on_modified_texts()

We evaluated the Logistic Regression model on the updated texts after modification. The intention was to assess the performance of the model after targeted attack.
We conducted the evaluation 5 times and computed the average accuracy to see how much the targeted attack impacted the model's performance.

This approach helped me identify the words the model was highly reliant on and showed how vulnerable the model is to adversarial attacks on those key words.

Dataset	**Training Accuracy**	**Testing Accuracy**
Original (Unmodified)	0.8949	0.7833
Modified Untargeted Attack		0.6675 (from 5 runs)
Modified Targeted Attack		0.2674(from 5 runs)

The outcome indicates that while the baseline model performed well on the raw dataset without modification, with high training accuracy at 0.8949 and testing accuracy at 0.7833, its performance greatly suffered under attack. The untargeted attack resulted in a moderate loss in accuracy (0.6675), but the targeted attack dropped accuracy further to 0.2674, which shows it effectively reduced the robustness of the model by intelligently modifying the most useful words.

Observations
The result of the attacks demonstrates that the untargeted attack resulted in a moderate decrease in accuracy, decreasing the model's performance from 0.7833 on the initial data to 0.6675. This shows that the untargeted attack, which randomly modifies words in the text,

inserts noise that weakens the model but keeps it partially resilient due to the randomness and less intense nature of the changes. Conversely, the targeted attack resulted in a greater drop in accuracy to 0.2674. This drastic drop indicates that the targeted attack was better at exploiting the weaknesses of the model by strategically changing the most effective words the model employed for classification. By focusing on the most influential words in the model's decision-making, the attack is of much greater effect, demonstrating its validity and efficiency in lowering the performance of the model.

Modified data samples

Text	Label
consistently clever @nd suspensful .	1
throws in enough clever @nd un3xp3ct3d twists to make the formula feel fres .	1
in addition to sporting one of the worts titles in recent cinematic history , b@ll1$t1c : ecks vs . sever also features terrible , abnal d1@l0gu3 ; convenient , h0l3-r1dd3n poltting ; superficail characters and a rather dul , unimagintive cr chase .	0
a cellophane-pop remake of the punk classic ladies and gentlemen , the fabulous stains . . . crossroads is n3v3r much w0r$3 th@n bl@nd ro b3tt3r tahn inconsequentila .	0
he thin l00k$ l1k3 a mad-efor-home-video qu1ck13 .	0

Inferences drawn:

The readability of targeted texts has significantly decreased due to the nature of the attacks. The untargeted attack, which changes words randomly (through character swapping, replacement, or deletion), embedded different typographical errors, such as substituting letters with symbols or character deletions. For example, words like "clever" become "cl3v3r" and "unexpected" becomes "un3xp3ct3d". These alterations, although still understandable, rendered the sentences less readable and harder to comprehend.

The attack that specifically targeted key words also introduced distortions, particularly to words considered important to the classifier's decision, which made the texts more difficult to read overall. Despite this, the sentences still maintained their basic structure, although readability was considerably degraded by overuse of substitutions and changes in characters.

The attacks were effective to attack the words that had the biggest impact on the model's predictions, but the effectiveness of this differed by attack method. In the untargeted attack, the changes were introduced randomly across the text independent of the importance of the words in classification.

This randomness also implied that some of the most important words, which likely carried significant weight in the model's decision-making, were either not altered at all or too subtly to affect the classifier's output in any significant way.

The targeted attack, on the other hand, through the greedy selection algorithm, aimed to transform the most influential words based on how much their removal affected the classifier's probability. For example, words like "clever" or "unexpected" were picked out for modification because they would have strong predictive power for positive labels.

The algorithm was successful in finding those words that would tend to influence the output of the classifier, but there were cases where, despite the modifications, the classifier went on to predict the same label, which meant that something other than those specific words was influencing the final classification.

Overall, though, while the targeted attack was more precise in choosing the correct words to alter, it was still constrained by the fact that text classification models are complex and rely on more than a single feature.

Challenges and issues

1.Faced challenges applying the greedy selection component of the targeted attack, but it made me understand how the model's predictions are affected by shifting individual words.
2.Coding challenges were significant, but I managed to address them and complete the task effectively.
3.Since the project was divided into four, it was long, but I believe it was worth the trouble for learning and outcome.

Conclusion

This task highlighted the impact of adversarial attacks on machine learning models, in our case, text classification. The targeted and untargeted attacks were indicative of how slight and calculated alterations in input text significantly degrade model accuracy.

Note : Use of AI and chat gpt leveraged for understanding and assistance