Project Overview

The project involves detecting hate speech in Arabic tweets using a fine-tuned BERT model, combined with an evaluation of the model's robustness through adversarial attacks. The project also includes integrating this system into the Serbot platform to enable interactive features, such as responding to user input and detecting hateful content.s

System Architecture

1. **Data Pipeline**
    - **Data Collection:** Tweets in Arabic were collected, focusing on those labeled for hate speech.
    - **Preprocessing:** Text preprocessing included tokenization, stop word removal, and handling Arabic language-specific features like prefixes and suffixes.
    - **Adversarial Data Generation:** A pipeline was designed to create adversarial examples by masking words, generating substitutes using BERT, and selecting the most similar word replacements.
2. **Model Architecture**
    - **BERT Model:** We employed the 'qarib/bert-base-qarib' model, fine-tuning it on our specific dataset to enhance its ability to detect hate speech in Arabic.
    - **Fine-Tuning Process:** The model was fine-tuned using labeled data, with careful attention to hyperparameter tuning and validation to ensure high accuracy.
3. **Adversarial Attack Framework**
    - **Blackbox Greedy Approach:** This method involved three main steps: word masking, BERT-based word substitution, and similarity matching. The objective was to create challenging inputs that test the model's robustness.
    - **Evaluation:** The framework was used to generate adversarial examples, which were then employed to evaluate the model's resistance to such attacks.
4. **Integration with Serbot**
    - **User Interaction:** The system was designed to accept user input, classify it as hate speech or not, and identify specific hateful words.
    - **System Response:** When integrated with Serbot, the robot will respond based on the detection results, adding a layer of interactivity to the platform.

Implementation Details

- **Programming Languages:** Python was primarily used, leveraging libraries such as Hugging Face's `transformers` for model fine-tuning and `pop` for Serbot integration.
- **Tools & Libraries:** Jupyter Notebook was used for development, with specific tools like `BERT` for NLP tasks and `OpenCV` for Serbot's visual processing.

Challenges & Solutions

- **False Positives:** Initial versions of the model occasionally flagged non-hate speech as hate speech. We adjusted the confidence threshold and refined the preprocessing steps to mitigate this.
- **System Integration:** Integrating hate speech detection with Serbot required careful attention to ensure seamless interaction between the two systems, particularly in handling real-time input and output.

Future Work

- **Further Robustness Testing:** Continue evaluating the model against adversarial examples to strengthen its defenses.
- **System Optimization:** Work on optimizing the processing pipeline to reduce latency and improve performance in real-time applications.
- **Enhanced Features:** Explore additional features for the Serbot, such as real-time monitoring and response adjustments based on detected content.