Hate Speech Detection using Transformers (Deep Learning)

Team Member Details

Group Name: NLP Pioneers

Name: Duc Duong

Email: duongduc@grinnell.edu

Country: United States

College: Grinnell College

Specialization: Data Science & NLP

Problem Description

Hate speech refers to written, verbal, or behavioral communication that attacks or discriminates against a person or group based on religion, ethnicity, nationality, race, gender, or identity factors. The goal of this project is to detect hate speech in text data (Twitter tweets) using deep learning transformer models.

Business Understanding

Social media platforms face growing challenges in moderating harmful content. Automated detection of hate speech reduces reliance on manual moderation, ensuring safer online communities and regulatory compliance. A transformer-based detection system can provide scalable, accurate, and real-time solutions to support business needs in content moderation, brand safety, and policy compliance.

Project Lifecycle & Deadline

1. **Business Understanding & Data Collection** (Week 1–2)
2. **Exploratory Data Analysis & Preprocessing** (Week 3–4)
3. **Model Development (Transformers)** (Week 5–6)
4. **Model Evaluation & Optimization** (Week 7–8)
5. **Deployment & Documentation** (Week 9–10)

Deadline: 10 weeks from project start

GitHub Repository

https://github.com/duongduc388222/twitter_hate_speech

EDA Highlights

1) Data Overview: column types, missing values; label distribution plot.
2) Text Cleaning: URLs, @mentions, hashtags normalized, digits mapped to <num>, emoji removed.
3) NA Handling: (a) simple fill with empty string, (b) length-aware placeholder imputation.
4) Outliers: character length distribution; treated with (a) IQR capping, (b) Winsorization.
5) Token Patterns: top unigrams/bigrams by frequency to surface salient terms.

Final Recommendation

Start with char-level TF-IDF (3–5 grams) + Linear SVM as the production baseline due to strong performance on noisy short texts. Improve macro-F1 via class weights and hyperparameter search. As a next step, fine-tune DistilBERT with careful preprocessing and early stopping, and compare against SVM. Include fairness checks (group-wise precision/recall) before deployment.