Advance NLP: Hate Speech detection using Transformers (Deep Learning)

Group name: VerbalVigilantes

Team member names: Zohra Bouchamaoui

Email: Zohra.bouchamaoui@outlook.com

Country : United Kingdom

Company: Data Glacier

Batch code: LISUM21

Specialisation: NLP

PROBLEM DESCRIPTION

Hate speech is defined as any form of verbal, written, or behavioural communication that uses derogatory or discriminatory language to insult or attack an individual or a group based on attributes such as religion, ethnicity, nationality, race, colour, ancestry, gender, or other identity factors. In this project, our objective is to design a machine learning model, utilising Python, that can accurately detect instances of hate speech.

Hate speech detection typically falls under the umbrella of sentiment classification. To train a model capable of discerning hate speech in a given text, we will utilise a dataset commonly used for sentiment classification. Specifically, for this task, we will train our hate speech detection model using Twitter data, with the aim of identifying tweets that contain hate speech.

DATA CLEANSING AND TRANSFORMATION

1. **Data Loading:** The train and test datasets are loaded using the pd.read_csv() function from the pandas library. The file paths for the datasets are provided as arguments to the function.
2. **Data Exploration:** We conducted some initial exploratory analysis to understand the data. The following steps are performed:
   - We displayed the first few rows of the train and test datasets using the head() function to get a brief look at the data.
   - We calculated the basic statistics of the train dataset using the describe() function to get information like count, mean, standard deviation, minimum, and maximum values for numerical columns.
3. **Data Preprocessing:**

- Text Preprocessing: A function named preprocess_text() is defined to pre-process the text data in the "tweet" column. The function applies the following steps:
  - Convert the text to lowercase using the lower() function.
  - Remove URLs using regular expressions (re.sub() function) to replace URLs starting with "http", "www", or "https" with an empty string.
  - Remove special characters and numbers using regular expressions to keep only alphabets and spaces.
  - Remove stopwords using the NLTK library's stopwords corpus. Stopwords are common words like "the", "is", "and" that do not contribute much to the overall meaning of the text.
  - Return the pre-processed text.

4. **Applying Data Preprocessing:**
   - The preprocess_text() function is applied to the "tweet" column of both the train and test datasets using the apply() function from pandas. The pre-processed text is then stored in a new column named "clean_text" in both datasets.
5. **Data Visualization:**
   - The length distribution of tweets in the train dataset is visualized using a histogram using the hist() function from matplotlib.
   - The word frequency distribution of the tweets is visualized using a bar chart using the bar() function from matplotlib.
   - The sentiment distribution of the tweets is also visualized using a bar chart.
   - The hashtag analysis is performed by extracting hashtags from tweets using regular expressions and counting their occurrences. The top 10 most used hashtags are visualized using a bar chart.
6. **Feature Engineering:**

- TF-IDF Vectorization: The TfidfVectorizer function from scikit-learn is used to convert the pre-processed text data into a numerical representation. The fit_transform() function is then applied to the "clean_text" column of the train dataset in order to obtain the TF-IDF feature matrix. The transform() function is then applied to the "clean_text" column of the test dataset to obtain the corresponding TF-IDF feature matrix.
- Bag-of-Words (BoW) Representation: The CountVectorizer class from scikit-learn is used to convert the pre-processed text data into a bag-of-words representation. The fit_transform() function is applied to the "clean_text" column of the train dataset to obtain the BoW feature matrix. The transform() function is then applied to the "clean_text" column of the test dataset to obtain the corresponding BoW feature matrix.