A Toxic Comment Identifier App
Project Type: App Dev
DSC 478 - Winter 2023
Team Members: Jeffrey Bocek, Xuyang Ji & Anna-Lisa Vu

Overview

The goal of this project is to create a Python application that identifies whether a comment is toxic. This app can be used as a third-party library for social media sites or public sites where users are allowed to leave comments. When the application starts up, it will train multiple models against an existing data set. Once the models are trained, the user will be prompted to enter a comment. The application will classify the comments using the following categorizes below:

- Toxic
- Severe Toxic
- Obscene
- Threat
- Insult
- Identify Hate
- Non-Toxic

The categorizes will be an output from the program. Any program would be able to apply logic based on the results of this app.

Some key features of the application would include:

- Ability to train multiple models with multiple settings:
    - KNN (Euclidean Distance/Cosine Distance)
    - Rocchio (Nearest Centroid/Cosine Similarity)
    - Naïve Bayes
    - Ensemble Methods
- Feature Extraction: TD*IDF, Word2Vec vs. no weights
- Ability to re-train the models with additional data if needed.
- When the application starts up, the user can choose which model to classify the comment. At any point of the program, the user will be able to switch which model to use for classification.

Analysis Approach

DM Task 1: Pre-processing and Cleaning of Data

- Format the data into a matrix to do text categorization
- Feature Extractions:
    - Create TF*IDF matrix to train models based on TF*IDF weighted data
    - Use Words2Vec

DM Task 2: Exploratory Phase

- Perform basic exploration of the data
    - Frequency counts
    - Histograms of classes
- Create visualizations to view class breakdown
- Run clustering models against the data to see if the model would find the same number of classes we are working with (7)
- Depending on what class distribution looks like, we will look at potential resampling methods.

DM Task 3: Implement the Supervised Learning Models

- Train and Test the following Models based on our data:
    - KNN (Euclidean Distance/Cosine Distance)
    - Rocchio (Nearest Centroid/Cosine Similarity)
    - Naïve Bayes
    - TD*IDF weights vs. no weights
    - Explore an ensemble method combining all or some of these models

DM Task 4: Evaluate Models and Tune for User in the Application

- Stratified K-fold Cross Validation
- Create classification matrix/reports
- ROC Curve & AUC Analysis

DM Task 5: Develop the Application to Classify Comments

- Develop the application to train models upon startup and then ask the user which model and options they would like to use for classification. To help guide the user, we will mention strengths and weaknesses of the different models. Some options we will support are below:
    - Feature Extractions: TF*IDF, Words2Vec, vs. Simple (Non-Weighted Data)
    - Models:
        - KNN: Euclidean Distance or Cosine
        - Rocchio: Nearest Centroid or Cosine Similarity
        - Naïve Bayes

        - Ensemble methods to combine some models
- After, the application will prompt the user for a comment.
- The application will respond with the category the model has classified the comment.
- After categorizing a comment, the application will prompt the user again for a comment.
- If the user wants to change the model being used, they will be able to do so by using some special program keywords.
- If the user would like to re-train the models with additional data, they will be able to do so by using some special program keywords.

Data Schema and Size

The data set we are using to train our models is from Kaggle**:** https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data?select=test.csv.zip

- Training Data – 159,571 rows
- Test Data – 153,164 rows

Plan for Evaluation – Analysis of Results/ Discussion

To evaluate our models, we plan to run the following:

- Stratified K-fold Cross Validation
- Create classification matrix/reports
- ROC Curve & AUC Analysis

We plan to include all the models and evaluations in a separate Jupyter notebook. For the application, we will select the models which we believe provide the best results for the end user.

Plan Work Distribution

Task	**Who?**	**Target Completion Date**
Pre-Processing		
Design Juypter notebook layout/“treeview” needed for subsequent parts		
Clean/get text into a matrix		

Feature Extraction		
Create Words2Vec Matrix (non-weighted data)		
Create TF*IDF Matrix		
Data Exploration		
Basic Data Exploration		
Exploratory Visualizations		
Model Building & Evaluation **For all models evaluate by:** ● N-fold Cross Validation ● Create classification matrix/reports ● ROC Curve & AUC Analysis		
Create and tune KNN Model		
Create and tune Rocchio Classifier		
Create and tune Naïve Bayes		
Create and tune Ensemble Models		
Make sure models adhere to one code style/design for final application		
Application Development		
Design basic blueprint for application (functions/programs design)		
Implement startup logic - ● Read training data on startup ● Train models ● Prompt user on what model to use for classifying		
Implement classifier logic: ● Prompt user for a comment		

• Run the comment in the model • Output the prediction to user		
Implement Additional Features • If the user types "***Change Model***", allow the user to pick a different model. • If the user types "***Retrain Model***", allow the user to re-train the model by reading the input files again (additional input could have been added while program was running)		
Record an appendix of application of sample run		
Ensure all code is documented		
Project Deliverables		
Create Executive Summary		
Create Readme file		
Create Final Report		
Record Demo Video		