TOXICITY CLASSIFICATION OF COMMENTS

Team 19

			
Anja Matic	Mija Pilkaite	Emma Moncia	Person #4
• TF-ID • Text preprocessing	• SBert • Data Analysis	• BagOfWords	• Keyword 1 • Keyword 2 • Keyword 3

Highlights (4–6):

- Data analysis to get to know and understand a large dataset with lots of features and parameters, analyzing the most common types of toxicities.
- Develop a machine learning model to detect and quantify the toxicity of Wikipedia comments. Different sentence transformation techniques are used with different prediction models to find the best suitable one.
- Explored various feature representations to capture the essence of the text data: Bag of Words, TF-IDF vector and SBert.
- Trained and evaluated Logistic Regression, SGD Regressor, and Decision Tree Regressor on each feature representation: this allowed for a comprehensive comparison of the models' performance on different text embeddings.
- Employed Mean Squared Error (MSE) and accuracy score as evaluation metrics for the models: fine-tuning hyperparameters using MSE for SGD and Decision Tree regressors to ensure the best performance.

Brief Report:

Social media becoming a larger and larger part of our lives everyday inevitably reminds us of the downsides that come with it. Toxicity in social media, forums and comments can lead to unhealthy communication and hinder knowledge exchange. Thus, we aimed to develop a machine learning model that could classify toxic comments and quantify the level of toxicity of each comment. We acquired our dataset from Kaggle that was collected from Wikipedia Talk Corpus, consisting of annotated comments from Wikipedia talk pages.

To work with **Natural Language Processing**, we made sure to explore different types of feature representations including **Bag of Words, Term Frequency-Inverse Document Frequency (TF-IDF)** and **Sentence-Bert embeddings (SBert).** We use three different models for both classification

and regression. We use **Logistic Regression** to classify the comments into toxic and non-toxic ones and fine-tune the hyperparameters of **Stochastic Gradient Descent** and **Decision Tree regressors** to predict the toxicity value (also known as target value) to be able to quantify it. For our model evaluation, we use **Mean Squared Error (MSE)** metric and **accuracy score** from Sklearn library to assess the quality of our predictions and to improve them.

Our various techniques of feature representation and using different models to predict and quantify toxicity, lets us thoroughly research the best combination of methods to potentially identify harmful Wikipedia comments and hopefully making the internet forum a more pleasant and more of a safe place. This research and extensive dataset is a starting point for future work while using more advanced techniques and bettering the detection of toxicity in not only Wikipedia, but also other social media comments.

Description of Appendices (Optional):

Besides the report, we are also attaching our Jupyter notebook, where one can expect thorough research and code for all parts of the project. We used some online sources to find out more about natural language processing and pick the strategy for our project.
Among these we will cite our sources of reference for:
1. **Tokenization**: Splitting the text into individual words to break down the text into separate words and to create fundamental units of analysis for subsequent processing at the word level
2. **Removing Punctuation and Special Characters** as well as only considering lower case characters. For this we used the built-in functions such as: re.sub('[^A-Za-z0-9]+', ' ', text_string) and .lower(): https://lzone.de/examples/Python%20re.sub
3. **Stemming:** Reducing words to their base form by removing affixes to simplify the vocabulary thus variations of words, such as tense or plural forms, were reduced to their core form, enabling consolidation and simplification of the vocabulary.
https://www.nltk.org/api/nltk.stem.snowball.html
4. **Undersampling for Balancing the Dataset:** Randomly removing samples from the majority class to balance the dataset and thus get a more evenly represented dataset across the different classes, helping to mitigate biases and enhance the performance of the classification models.
https://imbalanced-learn.org/stable/under_sampling.html
5. **Bag-of-Words (BoW) Method**: Representing text as word frequencies, ignoring grammar and word order. In this project, the BoW method was employed to construct a feature vector for each document by counting the frequency of each word and thus capturing the frequencies of words.
https://www.mygreatlearning.com/blog/bag-of-words/
https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer
6. **TF-IDF:** Assigning weights to words based on their importance in a document and rarity in the corpus. t takes into account the frequency of a word in a document (term frequency) and the rarity of the word in the entire corpus (inverse document frequency). By using TF-IDF, more weight is assigned to words that are both frequent in a document and rare in the overall corpus
https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer
7. **SBERT Embedding**: Generating fixed-dimensional vectors that capture the semantic meaning of sentences. In this project, the sentence_transformers library, specifically the SentenceTransformer model, was utilized to generate high-quality sentence embeddings.
https://www.sbert.net/, "Sentence Transformers and Embeddings | Pinecone," n.d. https://www.pinecone.io/learn/sentence-embeddings/.