Task 10: Towards Explainable Detection of Online Sexism

Perna, Grazia
g.perna4@studenti.uniba.it
Matr. n° 785321

Zaza, Maria Elena
m.zaza16@studenti.uniba.it
Matr.n°787626

1 Introduction

In our study, we tackled two tasks, namely Task A and Task B, as part of the "SemEval 2023 - Task 10 - Explainable Detection of Online Sexism (EDOS)" challenge on CodaLab. Task A involved identifying whether a message or sentence exhibited sexism, while Task B aimed to categorize the specific type of sexism. To accomplish these tasks, we leveraged the dataset provided for the challenge and tailored it to our specific needs by obtaining six different datasets. Through rigorous data preprocessing and AI techniques, we developed models that were well-suited for executing the aforementioned tasks. Our primary objective was not only to accurately detect instances of sexism but also to gain insights into the underlying nuances and manifestations of sexism within the language used. This allowed us to delve deeper into the different forms of sexism and their prevalence in various contexts.

2 Istructions

To install the required libraries to run the system, uncomment the first code section in "MLPipeline.ipynb". This section contains the necessary commands to install the libraries using the Python package manager, pip. Simply remove the comment tags "#" from the code and run the cell. Once the installation is complete, you can proceed with the rest of the steps in the notebook.
For executing the GUI app, it's necessary to run 'MLPipeline.ipynb'.

3 Task

Natural Language Processing plays a crucial role in addressing sexism by analyzing and understanding natural language. It enables the identification of gender discrimination and sexist language in texts and conversations.
By detecting and raising awareness of sexist content, NLP helps foster a more inclusive and equitable communication environment. It also facilitates the examination of sexism's impact on society, including its presence in media, online discussions, and various communication contexts.
By leveraging NLP techniques, we can gain insights into the prevalence and patterns of sexism, enabling us to tackle these issues more effectively. NLP empowers us to monitor, analyze, and mitigate the harmful effects of gender bias in language. This technology serves as a valuable tool in promoting gender equality and challenging societal norms that perpetuate discrimination.
In order to improve the interpretability, it could be helpful to detect when something is sexist and also explaining why.
The task contains two hierarchical subtasks:

- TASK A - Binary Sexism Detection: a two-class (or binary) classification where systems have to predict whether a post is sexist or not sexist.
- TASK B - Category of Sexism: for posts which are sexist, a four-class classification where systems have to predict one of four categories: (1) threats, (2) derogation, (3) animosity, (4) prejudiced discussions.

4 Model

In the experimental settings of our study, we aimed to tackle two tasks related to the detection of online sexism: Task A, which involved binary sexism detection, and Task B, focused on categorizing the types of sexism. To accomplish these tasks, we leveraged the dataset provided for the "SemEval 2023 - Task 10 - Explainable Detection of Online Sexism (EDOS)" challenge on CodaLab.
For Task A, we implemented a custom text cleaning function to remove spaces, convert text to lowercase and eliminate punctuation. We used TreeBank-

Tokenizer, a powerful NLP library, for tokenization using the word regular expressions and, only in the task A, the WordNetLemmatizer for lemmatization. The cleaned text was then fed into a CountVectorizer, which transformed the text into numerical features suitable for classification.
The LinearSVC classifier was employed with class weights set to address the imbalance between the "not sexist" and "sexist" classes.
Both individual words and word pairs (n-grams) were considered by the vectorizers.
To evaluate the performance of our models we used the validation set and then we employed GridSearchCV that performs hyperparameter tuning in order to determine the optimal values for a given model. It uses the Cross-Validation method, that we fixed to 10. After achieving satisfactory results we evaluated the models using the test set to assess their performance.

5 Related work

The top three submissions for each task used multiple models or an ensemble approach. Many of the top systems employed additional techniques such as pre-training and multi-task learning.
The teams "stce" and "PASSTeam" achieved top results in multiple tasks by using multi-task learning and pre-training on unlabelled data. "stce" utilized RoBERTa-large and ELECTRA, while "PASSTeam" employed fine-tuned RoBERTa and HateBERT.
In Task A, PingAnLifeInsurance used a multi-task DNN structure with further pre-training of DeBERTa-v3 and TwHIN-BERT on the starter kit's unlabelled data and an additional Kaggle dataset. FiRC-NLP used an ensemble of various fine-tuned DeBERTa models on the labelled task data.
In Task B, JUAGE relied on prompt-based learning and achieved first place using an instruction-tuned PaLM model with a parameter-efficient prompt tuned on the task data and majority voting over six iterations.
The popular methods used by participants included transformers-based architectures such as RoBERTa, DeBERTa, BERT, BERTweet and DistilBERT. Prompted language models like GPT-2, GPT-3, PaLM and OPT were also used. Some participants employed traditional machine learning methods or non-transformer deep neural networks, often in combination with other techniques.
Regarding additional training and data, the majority of participants applied fine-tuning on the target task, while some also performed further pre-training, fine-tuning on auxiliary tasks or prompt-based learning. Around 40% of participants used the unlabelled data in the starter kit and some used external resources, both labelled and unlabelled.
Additional features were not commonly used by participants, with around 66% not utilizing any. If additional features were employed, they primarily included additional embeddings, emojis or lexicons.

6 Experimental setting

6.1 Data

The initial dataset provided for our project consisted of a comprehensive set of features. In order to address the specific requirements of Task A and Task B, we performed data preprocessing and feature extraction to create six distinct datasets:

- train_category
- train_sexist
- dev_category
- dev_sexist
- test_category
- test_sexist

The "sexist" datasets refer to the task A, instead the other ones to the other task. In the task A, the feature considered are: ID, text, label_sexist.
On the other hand, in the other task we have: ID, text, label_category.
As regards the prefix of the name (train/dev/test), they refer to the split in which they are used. To accomplish this data transformation, we leveraged the pandas library.

6.2 Implementation

Our model implementation relied on various tools and libraries to develop a solution for online sexism detection. We used Visual Studio as our integrated development environment and employed the Python programming language.
For text processing tasks, we used libraries such as TreeBankTokenizer and NLTK. TreeBankTokenizer provided functionalities like tokenization, while NLTK offered additional support for text processing operations.

In terms of machine learning, we relied on popular libraries such as Scikit-learn and NumPy. Scikit-learn provided a comprehensive set of tools for building and evaluating machine learning models. To streamline our machine learning workflow, we used Scikit-learn's Pipeline class, which allowed us to chain together various steps such as text vectorization and model training. Additionally, we employed the pandas library for data manipulation and analysis, and NumPy for efficient array operations.
These tools and libraries formed the backbone of our project, enabling us to preprocess the data, build and train machine learning models, and evaluate their performance in a systematic and efficient manner. In order to tune the hyperparameters we used the GridSearchCV of the sklearn library.
For Task A and B, we employed "balanced" as weight setting. This was necessary due to the significant class imbalance. The class distribution for Task A was as follows:

- TRAIN SET:
  - not sexist: 10602
  - sexist: 3398
- DEV SET:
  - not sexist: 1514
  - sexist: 486
- TEST SET:
  - not sexist: 3030
  - sexist: 970

As regards task B, the class distribution was:

- TRAIN SET:
  - derogation: 1590
  - animosity: 1165
  - prejudiced discussions: 333
  - threats, plans to harm and incitement: 310
- DEV SET:
  - derogation: 227
  - animosity: 167
  - prejudiced discussions: 48
  - threats, plans to harm and incitement: 44
- TEST SET:
  - derogation: 454
  - animosity: 333
  - prejudiced discussions: 94
  - threats, plans to harm and incitement: 89

7 GUI

We developed a graphical user interface (GUI) for an application designed to detect sexism in text messages. It provides a simple and intuitive way for users to interact with the application.
When the program runs, a window is displayed with a logo at the top.
Below the logo there is an instruction label that prompts the user to enter a message. A text box is provided where the user can type the message they want to analyze for sexism.
To process the user's input there is a "Send" button. When the user clicks the button a prediction function is triggered. This function takes the text entered by the user and performs the necessary analysis using pre-trained models.
After the analysis is completed, the prediction results are displayed in a label on the interface. This label shows whether the message contains sexism or not, based on the prediction made by the models.
The graphical interface provides a user-friendly way for individuals to input text, receive predictions and gain insights into the presence of sexism in their messages.

7.1 Implementation

For the graphical user interface (GUI), the code uses the Tkinter library, which is a standard Python library for creating GUI applications. In this case, the code creates a window using the Tk() function and sets the window title. The window contains a canvas for organizing the interface elements.
To display the application logo, the code uses the PIL (Python Imaging Library) library to open and resize the logo image. The image is then converted into a format compatible with Tkinter using the ImageTk.PhotoImage() function. Finally, a label widget is created to display the logo image.
The interface includes an instruction label that prompts the user to input a message. This is achieved using the tk.Label() function. Additionally, a text box widget is created using the Text() function, allowing the user to enter the message to be analyzed.
When the user clicks the "Send" button, a function

is called to retrieve the text entered by the user from the text box widget. This function then applies the loaded models to the input text to make predictions regarding the presence of sexism and potentially categorize the type of sexism.
The prediction results are displayed in a label widget, which is updated with the appropriate text based on the model's prediction.

8 Evaluation

As said before our data are divided into three splits: train, validation and test. As a matter of fact, the models were trained using the train set.
Then the validation set was used to assess their performance and make any necessary adjustments or tuning. Finally, the test set was employed to evaluate the models' generalization ability and obtain their final performance metrics. The best SOTA results are related to the task A, while in the task B the results are not so promising due to the complexity of the problem. In this competition, the metric considered is "f1 macro", but we computed aldo the following ones:

- Accuracy
- Precision
- Recall

We obtained the following results when using the validation data:

Table 1: Metrics - Validation

N	Accuracy	Precision	Recall	F1
A	0.8340	0.7899	0.7255	0.7481
B	0.5453	0.5258	0.4491	0.4675

Instead, concerning the usage of the test data we obtained:

Table 2: Metrics - Test

N	Accuracy	Precision	Recall	F1
A	0.8405	0.0.7947	0.7444	0.0.7637
B	0.5124	0.4932	0.4438	0.4597

9 References

@inproceedingskirkSemEval2023, title = SemEval-2023 Task 10: Explainable Detection of Online Sexism, url = http://arxiv.org/abs/2303.04222, doi = 10.48550/arXiv.2303.04222, author = Kirk, Hannah Rose and Yin, Wenjie and Vidgen, Bertie and Röttger, Paul, booktitle = Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023), publisher = Association for Computational Linguistics, year = 2023