PanwarJayant at SemEval-2023 Task 10: Exploring the Effectiveness of Conventional Machine Learning Techniques for Online Sexism Detection

Jayant Panwar and **Radhika Mamidi**
LTRC, International Institute of Information Technology, Hyderabad
jayant.panwar@research.iiit.ac.in and radhika.mamidi@iiit.ac.in

Abstract

The rapid growth of online communication using social media platforms has led to an increase in the presence of hate speech, especially in terms of sexist language online. The proliferation of such hate speech has a significant impact on the mental health and well-being of the users and hence the need for automated systems to detect and filter such texts. In this study, we explore the effectiveness of conventional machine learning techniques for detecting sexist text. We explore five conventional classifiers, namely, Logistic Regression, Decision Tree, XGBoost, Support Vector Machines, and Random Forest. The results show that different classifiers perform differently on each task due to their different inherent architectures which may be suited to a certain problem more. These models are trained on the shared task dataset, which includes both sexist and non-sexist texts. All in all, this study explores the potential of conventional machine learning techniques in detecting online sexist content. The results of this study highlight the strengths and weaknesses of all classifiers with respect to all subtasks. The results of this study will be useful for researchers and practitioners interested in developing systems for detecting or filtering online hate speech.

1 Introduction

The prevalence of sexist and offensive language online has severe consequences for the mental health and well-being of women. Emotional distress caused by exposure to such language can lead to anger, sadness, and fear, which can negatively impact their overall well-being. Given the severe harm caused by sexist language online, it is imperative that automated systems be developed to detect and filter such texts. The shared task (see Kirk et al., 2023 for a detailed description) deals with the online detection of sexism in English and provides the right opportunity to test the performance of automated systems in doing so effectively. The task comprises three subtasks:

- **Task A - Binary Sexism Detection:** a binary classification task where systems have to predict whether the text is sexist or not
- **Task B - Category of Sexism:** four-class classification task where systems have to predict sexist text into one of four categories: (1) threats, (2) derogation, (3) animosity, (4) prejudiced discussions.
- **Task C - Fine-grained Vector of Sexism:** an 11-class classification where systems have to predict sexist texts as one of 11 fine-grained vectors.

In this study, we employ several different conventional machine classifiers like Random Forests, Decision Trees, etc. By conventional classifiers, we mean those that have been developed before the advent of more recent techniques like deep learning and transformer-based approaches. They are simpler in design, easier to interpret and understand, and require less computational resources compared to classifiers that rely on deep learning techniques. In spite of being "conventional", these classifiers are still widely used in many practical applications and have been effective in resolving many real-world problems. The aim of this study will be to judge the effectiveness of these classifiers in detecting sexist text and also the type of sexism (as seen in subtasks B and C). Moreover, we deal with the highly imbalanced dataset by undersampling so that the classifiers do not become biased toward one class.

From the results, it is pretty clear that Random Forest Classifier is the best at detecting sexism in a binary setting. However, when we go a level or two deeper and need to identify the class of sexism as well, Random Forests are not such a good choice

Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023), pages 1531–1536
July 13-14, 2023 ©2023 Association for Computational Linguistics

and lose out to Support Vector Machines and Decision Trees. For Task-A, our best classifier achieved the rank of $73rd$ among all participants, while for Tasks-B and C, our best-performing classifiers were ranked $65th$ and $55th$ respectively, among all the participating teams.

2 Background

As stated earlier, the task had three subtasks. Only the training dataset provided by the task organizers was used in the study and no other dataset was utilized for training the conventional classifiers. The dataset consisted of 14,000 entries taken from posts/comments on two social networking platforms: Gab and Reddit. The entries also had their respective labels for each subtask. The task organizers had also provided a large amount of unlabelled data but it was not used in training the classifiers. The input for the system would be the text of the post/comment with everything preserved including the emojis, hyperlinks, and hashtags but not the usernames. The usernames are masked as [USER] in the dataset for privacy reasons. Each subtask had a different form of output.

- For subtask-A, the output could be either *sexist* or *not sexist*.
- For subtask-B, the output could be one of the following four:
  - *1. threats, plans to harm and incitement*
  - *2. derogation*
  - *3. animosity*
  - *4. prejudiced discussions*
- For subtask-C, the output would be an even more fine-grained version of subtask-B's outputs:
  - *1.1 threats of harm*
  - *1.2 incitement and encouragement of harm*
  - *2.1 descriptive attacks*
  - *2.2 aggressive and emotive attacks*
  - *2.3 dehumanising attacks & overt sexual objectification*
  - *3.1 casual use of gendered slurs, profanities, and insults*
  - *3.2 immutable gender differences and gender stereotypes*
  - *3.3 backhanded gendered compliments*
  - *3.4 condescending explanations or unwelcome advice*
  - *4.1 supporting mistreatment of individual women*
  - *4.2 supporting systematic discrimination against women as a group*

2.1 Related Work

There has been a lot of work in the field of text classification with respect to online sexism or hate speech in general. We derive inspiration from some of them for our work. For example, one of the seminal works in this area was conducted by Waseem et al. (2017), wherein they used a lexicon-based approach to detect hate speech and offensive language in social media. The authors used a publicly available dataset and reported high accuracy and F1 scores. Detecting sexism in a ternary setting can also be challenging and in their work, Jha and Mamidi (2017) showed how SVMs, Seq2Seq, and Fasttext classifiers can be used in determining the class of sexism. Finally, another work that looks at the problem in a way somewhat similar to ours is a study by Davidson et al. (2017), in which they proposed a deep learning model for detecting hate speech in online social media. This work used a large dataset and showed that deep learning models outperformed traditional machine learning algorithms for detecting hate speech. We wish to explore this area a little deeper with our study. We compare the best-known conventional classifiers for the three subtasks to showcase their strengths and weaknesses.

3 System Overview

As stated earlier, the focus of the study is to explore the effectiveness of conventional machine learning classifiers, and therefore, we consider the following classifiers (some are used only in one subtask while some are used for all three subtasks):

- *Decision Tree*: a supervised learning algorithm that creates a tree-like model of decisions and their possible consequences. It uses a set of input features to classify an input data point into one of several pre-defined classes based on a sequence of decisions made on those features. The concept of decision tree based classification was originally introduced by Fisher (1936) and has been improved upon over the years.

- *Logistic Regression*: an algorithm used in machine learning for binary classification tasks. Using a logistic function, it models the probability of an input data point belonging to one of two possible classes. An in-depth application of Logistic Regression for classification was first introduced by Hosmer Jr. et al. (2013).

- *Random Forest*: an ensemble learning algorithm used in machine learning for classification tasks. It constructs multiple decision trees on different subsets of the input data and features and combines their outputs to make the final prediction. The concept of Random Forests was originally introduced by Breiman (2001).

- *SVM Classifier*: an algorithm used in machine learning for text classification tasks. The formulation of SVMs as a classification and regression method was introduced by Cortes and Vapnik (1995). It finds the hyperplane that maximally separates the input data points into different classes in a high-dimensional space, by maximizing the margin between them.

- *XGBoost Classifier*: an ensemble learning algorithm used in machine learning for classification and regression tasks. The modern-day implementation was first described by Chen and Guestrin (2016). It uses a gradient-boosting framework and optimized computing techniques to improve the accuracy and efficiency of predictions.

The general outline of the model can be noticed in Figure- 1. The first stage is also one of the most important stages of the system, i.e., Data Preprocessing, where we make sure that the data being carried forward is fit for modelling our classifiers. Then we come to the stage of Feature Engineering, which is mainly concerned with converting the text into vectors, which the classifier will be able to understand when modelling the data.

Once we have extracted the features, we split the vector dataset into two parts for training and testing. We discuss the implementation level details in the next section. From here on, the training data is used to train the classifier to understand the task and provide correct labels. Finally, the trained classifier is fed with testing data in order to evaluate its performance and see how accurately and precisely it is able to predict the correct tag of sexism.

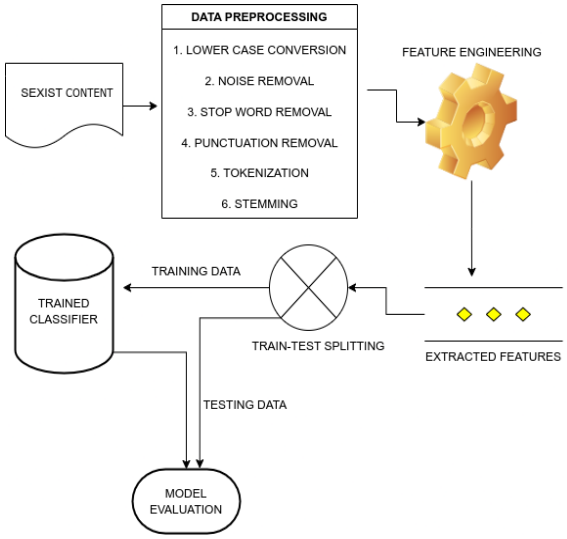

Figure 1: System Overview

4 Experimental Setup

One of the most important things to note about this shared task is the dataset itself. The dataset is highly imbalanced (see Figure- 2). The number of *non-sexist* entries is almost 3 times the number of *sexist* entries.

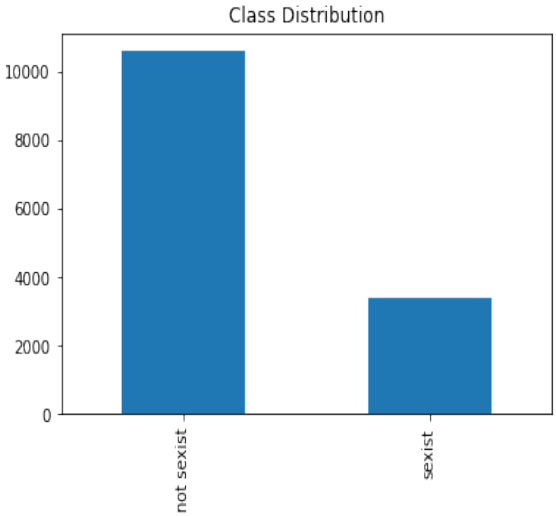

Figure 2: Imbalanced Class Distribution in the Dataset

In order to deal with this imbalance, we tried different techniques like oversampling and undersampling. Undersampling proved to be the best option as it was able to boost performance significantly which we will look at in the next section. Undersampling is a technique used in machine learning to address the problem of class imbalance in a dataset.

Class imbalance occurs when one class (in our case, the sexist class) has significantly fewer examples than the other class (the non-sexist class). This can lead to biased models that perform poorly on the minority class. Undersampling involves randomly removing examples from the majority class in order to balance the number of examples in each class. In our system, we structure the dataset such that the non-sexist class had only 1000 more instances than the sexist class. We dropped more than half of the examples in the original non-sexist dataset. This greatly improved our performance, as we will see in the later section. After undersampling, we do random splitting of the dataset to generate our train-test split.

For the train-test split, we started with a 75-25% split but noticed a jump in performance in development set evaluation when the split was changed to 85-15%. The authors had provided both the development set and the test set. Test set correct labels were not made accessible until the end of the task, so the system was developed only using the development set.

Next, we come on to the Data preprocessing details. The steps in this stage are as follows:

- Firstly, we convert the entire raw data entry into lowercase.
- Secondly, we remove noise from the raw entry. This can be in the form of hyperlinks, emojis, hashtags, etc.
- Thirdly, we remove all the stopwords as their contribution to the semantics of the sentence is negligible in sexist settings.
- Fourthly, we clean the entry by removing punctuation marks.
- Fifthly, we tokenize the entry text
- Lastly, we do stemming and lemmatization to reduce the words to their base forms before sending them to the next stage.

For the feature engineering stage, we make use of a simple TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer. It converts our words or tokens into vectors/embeddings by quantifying the relevance of a term in a sentence or a corpus by weighing how often it appears in a sentence against how often it appears in the entire corpus.

For implementing the conventional classifiers, we use the in-built classifiers of sklearn[1] library. For classifiers like Support Vector Machine, we performed hyperparameter tuning to ensure that we get the best performance out of the model. We use a SVM classifier with *linear* kernel, 1.0 as the penalty parameter (C), degree set to 3, and gamma set to auto.

For evaluation, like the authors of the shared task, we stick to the macro average F-1 scores of the sklearn library. It is best to take the macro average instead of just the average because our dataset is highly imbalanced and the macro average technique treats each class of the data adequately. It does so by first calculating the F-1 score for each class separately and then taking the arithmetic mean of those scores. As a result of this, each class is given equal importance in the final score, regardless of its size or prevalence in the dataset.

5 Results

Regarding the performance of the system in the shared task, it stood at rank 73 for task-A, rank 65 for task-B, and rank 55 for task-C. The best F1 scores obtained for task-A were as expected, as these statistical classifiers do not employ any deep learning or language techniques like large pre-trained language models or deep neural networks. Still, as we see in table 1, they were able to output decent F1-scores for the test dataset. Random Forest performed the best with a score of 75+. Others came close, but Random Forest marginally outperformed them.

Model	*Development Set*	*Test Set*
Decision Tree	60.15	57.87
Logistic Regression	71.78	73.73
Random Forest	**73.91**	**75.83**
SVM Classifier	71.87	72.79
XGBoost Classifier	71.27	72.72

Table 1: Macro Avg. F-1 Scores of Classifiers on Subtask-A: Binary Classification

Shifting our focus to task-B (see table 2), we only used 3 classifiers here because Logistic Regression was not suited for multi-class classification problems and XGBoost had very poor performance (<10 F-1 Score). As expected, the SVM classifier outperformed Decision Trees and Random Forests by quite some margin. This is because

[1] https://scikit-learn.org/stable/

of their ability to handle unbalanced data quite seamlessly by adjusting the penalty parameter (C) and by assigning higher weights to the minority classes during training to make the SVM more sensitive to minority classes. In contrast, decision trees and random forests may struggle with imbalanced data, as they tend to favour the majority class of the dataset.

Model	*Development Set*	*Test Set*
Decision Tree	32.66	37.58
Random Forest	15.91	15.94
SVM Classifier	**40.26**	**38.19**

Table 2: Macro Avg. F-1 Scores of Classifiers on Subtask-B: 4-class classification

Finally, we come to the performance of classifiers in task-C (see table 3). As expected, all of them performed quite underwhelmingly. The best F-1 score barely managed to cross the 20 mark.

Model	*Development Set*	*Test Set*
Decision Tree	**21.84**	**21.85**
Random Forest	12.14	12.22
SVM Classifier	19.94	20.42

Table 3: Macro Avg. F-1 Scores of Classifiers on Subtask-C: 11-class classification

It is worth noting that a different classifier has performed the best for every subtask. Subtask A was a simple binary classification problem where most of the classifiers eased past the F-1 score of 70. Random forest classifier performed the best as it is not prone to overfitting. There were only 3 major classifiers in the running for subtasks B, and C. SVM was the best for subtask B, while Decision Trees were the best for subtask C. In general, the performance of classifiers can vary for different subtasks due to various factors, such as dataset characteristics, feature representation, hyperparameter tuning, and randomness. The performance of classifiers can be influenced by the hyperparameters chosen for each classifier. If the hyperparameters of the classifiers are not optimized or tuned specifically for each subtask, it can impact their performance. Finally, the architecture of the classifier itself is one of the causes of different performances across the subtasks. For example, SVM can outperform Decision Tree in subtask B because it treats the 4-class classification problem as four binary classification problems and is well suited for it but when the data becomes highly non-linear and complex like in subtask C, Decision Tree will outperform SVM.

Furthermore, we can see how our little tweaks and optimizations gave a lot of boost to the performance of our model (see table 4). Increasing our training split from 75% to 85% increases the F-1 score by merely 1 point but utilizing the Undersampling technique had a great effect on the final F-1 score, improving the score by more than 5 points. Oversampling was not useful as it only ended up reducing the performance even from the system without any sampling technique.

Optimization	*Macro Avg F-1 score*
Undersampling	**73.91**
Oversampling	67.50
No-sampling	68.69
75-25% data split	73.10
85-15% data split	**73.91**

Table 4: Macro Avg. F-1 Scores of Random Forest Classifier on Development Set of Subtask-A: Binary Classification with different optimizations

Lastly, our best classifiers were able to outperform the baseline models B0, B1, and B2 from the task paper. These 3 models were based on conventional classifiers like Uniform and XGBoost. For comparison (see table 5), we only report B2 model because it had the best score among all baseline conventional classifier techniques for each subtask. On the other hand, our classifiers failed to reach the performance of the B6 baseline model, which was the best among B3, B4, B5, and B6 models, all of which were transformer-based approaches, utilizing variations of DistilBERT and DeBERTa.

6 Conclusion

In conclusion, through this study, we were able to explore the effectiveness of some of the conventional machine learning classifiers in the task of online sexism detection. Although they performed commendably in the binary classification task, it is clear that they are not cut out for multi-class classification tasks. Random Forests seem to be the best choice for simple binary classification of sexist texts out of conventional machine learning classifiers.

In terms of future work, deep learning neural networks and pre-trained language models can be used to improve upon the score achieved by our

Subtask	B2 Baseline model	Our best model	B6 Baseline model	Top ranked model
A: 2 class	49.33	75.83	82.35	87.46
B: 4 class	22.97	38.19	59.26	73.26
C: 11 class	8.81	21.85	31.71	56.06

Table 5: Comparison of the best F-1 scores attained for each subtask by our best model, top ranked model and baseline models from task paper: B2 and B6. B2 was the best baseline for conventional classifying approaches while B6 was the best baseline for transformer-based approaches.

system. Moreover, instead of just removing emojis and hashtags from our post data, we should try to inculcate them in the feature engineering stage and extract features from them. For example, Eisner et al. (2016) showed how emojis can be converted to vector embeddings for deep neural networks to make use of.

Limitations

The very nature of our approach has limitations. We delved only into conventional classifiers which are very good in text classification tasks but not as good as transformer or deep learning based approaches. Table 5 showcases this exactly. We were able to beat the conventional classifiers' baseline models but not transformer-based ones, let alone beating the top ranked system. Transformer-based approaches easily outperform us in all subtasks. For subtask A, our model comes close but it is in subtasks B and C, where the true performance of transformer-based approaches comes to light.

Another form of limitation, which we have already discussed briefly, is how we handle data preprocessing or cleaning. For this study, we completely remove any forms of hashtags and emojis from the raw content of the post during the data pre-processing stage. In order to better our feature extraction, we must make use of techniques like hashtag segmentation and emoji2vec so we can have representations for them as well. This will directly facilitate the classifiers to learn the difference between sexist and non-sexist posts better and perform better in all subtasks.

References

Leo Breiman. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA. Association for Computing Machinery.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):512–515.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. In *Proceedings of the Fourth International Workshop on Natural Language Processing for Social Media*, pages 48–54, Austin, TX, USA. Association for Computational Linguistics.

Robert A. Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(Part II):179–188.

David W. Hosmer Jr., Stanley Lemeshow, and Rodney X. Sturdivant. 2013. *Applied Logistic Regression*, 3rd edition edition. John Wiley & Sons, Hoboken, NJ.

Akshita Jha and Radhika Mamidi. 2017. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data. In *Proceedings of the Second Workshop on NLP and Computational Social Science*, pages 7–16, Vancouver, Canada. Association for Computational Linguistics.

Hannah Rose Kirk, Wenjie Yin, Bertie Vidgen, and Paul Röttger. 2023. SemEval-2023 Task 10: Explainable Detection of Online Sexism. In *Proceedings of the 17th International Workshop on Semantic Evaluation*, Toronto, Canada. Association for Computational Linguistics.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada. Association for Computational Linguistics.