Team LRL_NC at SemEval-2022 Task 4: Binary and Multi-label Classification of PCL using Fine-tuned Transformer-based Models

Kushagri Tandon, Niladri Chatterjee
Indian Institute of Technology Delhi
Hauz Khas, Delhi-110016, India
{kushagri.tandon,niladri}@maths.iitd.ac.in

Abstract

Patronizing and condescending language (PCL) can find its way into many mediums of public discourse. Presence of PCL in text can produce negative effects in the society. The challenge presented by the task emerges from the subtleties of PCL and various data dependent constraints. Hence, developing techniques to detect PCL in text, before it is propagated is vital. The aim of this paper is twofold, a) to present systems that can be used to classify a text as containing PCL or not, and b) to present systems that assign the different categories of PCL present in text. The proposed systems are primarily rooted in transformer-based pretrained language models. Among the models submitted for Subtask 1, the best F1-Score of 0.5436 was achieved by a deep learning based ensemble model. This system secured the rank 29 in the official task ranking. For Subtask 2, the best macro-average F1-Score of 0.339 was achieved by an ensemble model combining transformer-based neural architecture with gradient boosting label-balanced classifiers. This system secured the rank 21 in the official task ranking. Among subsequently carried out experiments a variation in architecture of a system for Subtask 2 achieved a macro-average F1-Score of 0.3527.

1 Introduction

The aim of the current task, viz. Patronizing and Condescending Language Detection (Pérez-Almendros et al., 2022), is to identify presence of condescending and patronizing tones in text, particularly by the media when referring to vulnerable communities. Preponderance of PCL in news and different social media texts, often targeted towards marginalised and under represented communities is a major social concern these days. It can feed stereotypes, tilt the scales of superiority towards a particular community, and fuel discriminatory behaviour. Thus the task of identifying PCL particularly that directed to vulnerable communities is a crucial task.

The task is based on the English language Don't Patronize Me! dataset (Pérez-Almendros et al., 2020). It consists of two subtasks, binary classification (Subtask 1) and multi-label classification (Subtask 2). In Subtask 1, given a paragraph the aim is to predict whether the paragraph consists of any form of patronizing and condescending language (PCL). In Subtask 2, the aim is to assign each paragraph a subset of labels which express different categories of PCL. The major challenges offered in such tasks emerge from various linguistic aspects, such as use of cryptic sentences, sarcasms used, polysemous nature of the English words among others. For the present task, further challenge emerges from the imbalance of dataset as discussed in Section 2.

Experiments were conducted with various systems and the best performing ones for both the subtasks are discussed in detail. The two systems that perform best among these, for both the subtasks are ensemble techniques, which employ two or more classifier models at different stages of the system.

The paper is organized as follows. The task background is discussed in Section 2. Sections 3 and 4 discuss the details of the systems and the experimental setup, respectively. The results from the systems are given in Section 5. The paper is concluded in Section 6.

The code for the proposed systems have been made available at https://github.com/KushagriT/SemEval2022-TeamLRL_NC

2 Background

As mentioned in Section 1, this task is based on the Don't Patronize Me! dataset. The paragraphs for this dataset have been extracted form the News on Web (NoW) corpus (https://www.english-corpora.org/now/), and have been manually anno-

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 421 - 431

Label:	1	2	3	4	5	6	7
Neg/Pos	14	52	46	45	52	21	261

Table 1: Label Ratios

tated. The statistics of the subset of data used for this task are as follows. The Train subset has 8375 samples, Dev subset has 2094 samples, and the Test set has 3832 samples. The label set considered for this multi-label classification consists of seven labels, namely 'Unbalanced power relations'(1), 'Shallow solution'(2), 'Presupposition'(3), 'Authority voice'(4), 'Metaphor'(5), 'Compassion'(6), and 'The poorer, the merrier'(7). These labels are henceforth addressed by their sequence ids as mentioned above.

The number of negative examples per positive example, for the binary classification task, in the Train + Dev dataset (henceforth referred to as Training dataset), is approximately 10. For Subtask 2, the number of negative examples per positive example (approximated to nearest integer) for each label in the Training dataset is given in Table 1 (denoted as Neg/Pos). An approximation of these ratios are used as scaling factors to manage the unbalanced classes when creating gradient boosting classifiers for multi-label classification.

Work has been carried out in the field of NLP to identify different types of linguistic variations or harmful languages from text such as, sarcasm detection (Chatterjee et al., 2020), hate speech detection (Djuric et al. (2015), Gitari et al. (2015)), and fake news detection (Shu et al. (2017), Conroy et al. (2015)). Since the introduction of BERT (Devlin et al., 2019), transformer-based language models have been used for a variety of NLP tasks, such as the use of BERT for a regression task of predicting eye-tracking features for a given word of the sentence (Choudhary et al., 2021), or use of BERT for the task of document classification (**?**, Adhikari et al. (2019)).

All the methods proposed in this paper use further pre-trained transformer-based language models to extract document embeddings. The motivation comes from some recent work that has been carried out in detecting condescending language from text. Wang and Potts (2019) introduce a new labeled dataset, namely TalkDown, of condescending acts in context, and establish baselines for this dataset using BERT.

3 System Overview

Experiments were conducted with two types of models for each of the two subtasks. Each system uses transformer-based language models for generating text embeddings. These language models were further pre-trained on Masked Language Modeling (MLM) task on the given data. The paragraphs from the training dataset were prepared for MLM with the model specific tokenizer by masking tokens in the input with probability 0.15, and using truncation and padding to maximum length of 256.

The present experiments use further pre-trained RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019) language models[1]. These further pre-trained RoBERTa (roberta-base) and XLNet (xlnet-base-cased) models are henceforth denoted as MyRoberta and MyXlnet, respectively. This further pre-training is done using the given data on the existing pre-trained RoBERTa and XLNet models in order to fine-tune the models for the two specific subtasks, so they become adapted to the given corpus irrespective of the task at hand, whether it is binary classification or multi-label classification.

The systems discussed in Sections 3.1 and 3.2, use a custom attention head architecture[2]. The input to this attention head is of dimension $batch\ size \times sequence\ length \times embedding\ dimension$, and the output is of dimension $batch\ size \times embedding\ dimension$. The attention mechanism is used so that the system learns the emphasis of different tokens towards the classification. This layer is henceforth referred to as Attn_head.

Embeddings are generated using these models to extract document representations, which will be used as features in each system. For training the proposed models, discussed in sections 3.1 and 3.2, and for the training of the MyRoberta and MyXlnet models, smart batching is used, with a maximum token length limit of 256. In smart batching the dataset is sorted by length of the sequences before creating batches, and padding is done to each sequence in each batch to the length of the longest sequence in that batch. In the following subsections the model algorithms are discussed in detail. Smart batching has been used to optimize the training speed for the transformer-based models, since now most of the resulting batches will have shorter

[1]Training details are discussed in the appendix
[2]Architecture details discussed in the appendix

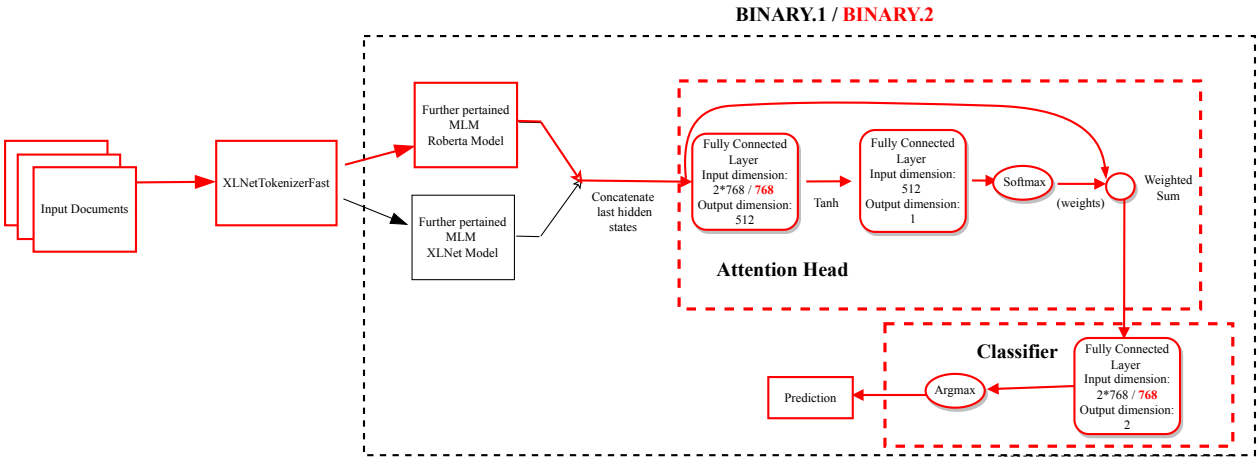

Figure 1: BINARY.1 and BINARY.2 (flow indicated in red)

sequence lengths which would prevent memory overload.

The following notations have been used in the further sections. The two subtasks of Binary and Multi-label classification have been referred to as, Subtask 1 and Subtask 2, respectively. The two systems in for each of these tasks have been denoted as BINARY.1, and BINARY.2 for Subtask 1, and MULTI.1 and MULTI.2 for Subtask 2. MULTI.1 and MULTI.2, each have two sub-models which are denoted as MULTI.1.A and MULTI.1.B for MULTI.1, and MULTI.2.A and MULTI.2.B for MULTI.2.

3.1 Subtask 1: Binary Classification

In this section the two proposed systems for the task of binary classification are discussed.

3.1.1 BINARY.1: MyRoberta + MyXlnet + Attn_head

The first component of BINARY.1 consists of MyRoberta and MyXlnet. Concatenation of the last hidden states from the two models is passed as an input to Attn_head. The output from this layer is sent to a fully connected layer which serves as the classifier unit. The architecture for BINARY.1 is given in Figure 1. The entire flow indicates the algorithm for BINARY.1.

This system is trained for the task of binary classification with Cross Entropy loss using AdamW optimizer which is Adam (Kingma and Ba, 2014) optimizer with weight decay (Loshchilov and Hutter, 2017). A linear learning rate scheduler with 50 warm-up steps is used. The hyperparameters for this system are given in Table 2.

Hyperparameter	Value
Train Epochs	5
Batch Size	16
Initial Learning Rate	2e-5

Table 2: Hyperparameters for BINARY.1

By using ensemble technique the quality of representation of these documents is improved, and using attention this model learns the emphasis of each token in the text sequence in contributing towards each label in the predicted label set. This can be observed from the ablation experiments discussed in Section 5.

3.1.2 BINARY.2: MyRoberta + Attn_head

The basic architecture used in BINARY.2 is similar to that of BINARY.1 described in Section 3.1.1. This system, however, trains the following model.

The first component of this system is MyRoberta. The last hidden states from this model is passed as input to Attn_head. The output from this layer is sent to a fully connected layer which serves as the classifier unit. The architecture for BINARY.2 is given in Figure 1, with its flow indicated in red. In case of BINARY.2, the input dimension for the first fully connected layer in the Attention Head unit and the classifier unit is 768 instead of 2*768, as for BINARY.1.

In this system, while training the weights in the pooler layer and the last 5 layers of MyRoberta are re-initialized. The weights for the fully connected layers in MyRoberta are re-initialized with mean 0 and standard deviation same as that of the

initializer range of MyRoberta[3]. The weights and biases of the layer normalization in MyRoberta are re-initialized to constant values of 1 and 0, respectively.

This system is trained for the task of binary classification with Cross Entropy loss using AdamW optimizer. A linear learning rate scheduler with 50 warm-up steps is used. For BINARY.1 a smaller batch size is used to prevent memory overload.

Hyperparameter	Value
Train Epochs	5
Batch Size	32
Initial Learning Rate (Group 1)	1e-5
Initial Learning Rate (Group 2)	2e-5
Initial Learning Rate (Group 3)	4e-5
Initial Learning Rate (Group 4)	5e-5

Table 3: Hyperparameters for BINARY.2

The initial layers of the MyRoberta model encode the more general information that is present in the text. Additionally, since MyRoberta has already been trained on the task-specific data, the embedding layer and the first four layers of MyRoberta are given a lower initial learning rate of 1e-5. This parameter group is denoted as Group 1. As the layers move closer the output or the classifier layer, the model encodes task-specific information. Hence for the next four layers (Group 2), the learning rate is chosen as 2e-5, and for the last four layers (Group 3) the learning rate is chosen to be 4e-5. The classifier and the pooler layers are assigned a higher learning rate of 5e-5. This parameter group is denoted as Group 4. Each of these layers have weight decay of 0.01, except for bias and layer normalization weights. Table 3 shows the values of the different hyperparameters for this system.

3.2 Subtask 2: Multi-Label Classification

In this section the two proposed systems for the Subtask 2, namely MULTI.1 and MULTI.2 are discussed. MULTI.1 consists of two sub-models, MULTI.1.A and MULTI.1.B. MULTI.1.A is MyRoberta + FCL_1 where FCL_1 denotes a fully connected layer, and MULTI.1.B is a collection of label-balanced XGBoost Classifiers (XGB). MULTI.2 also consists of two sub-models. In MULTI.2.A, a similar architecture is used as in MULTI.1.A, but with different dimensions and with an additional layer named FCL_2 which a fully connected classifier unit. MULTI.2.B has two parts: i) Fuzzy C-Means clustering to extract features, ii) a multi-label classifier model based on fuzzy membership. These two parts will be denoted as FCM and Fuzzy_CLF, respectively. The flow of output between the two models are explained in Section 3.2.2.

3.2.1 MULTI.1: MyRoberta + FCL_1 + XGB

This system has two parts, namely a model to extract features from the text (MULTI.1.A), and a model to assign a set of labels to this text using a classifier (MULTI.1.B). The architecture for this model is given in Figure 2.

To extract features MyRoberta is fine-tuned for the task of multi-label text classification. The first component of MULTI.1.A is MyRoberta. The CLS embedding from MyRoberta is passed to a fully connected classifier layer with output dimension 7, corresponding to the seven labels. In this model, while training the weights in the pooler layer and the last 5 layers for the MyRoberta are re-initialized as described in Section 3.1.2.

MULTI.1.A is trained using Binary Cross Entropy loss between the true multi-labels with the output, applied with sigmoid activation. It is trained using AdamW optimizer with. A linear learning rate scheduler is used, with 100 warm-up steps and number of total steps corresponding to 25 epochs. This is done to avoid a lower learning rate at 5 epochs. The hyperparameters for this system are given in Table 4.

Hyperparameter	Value
Train Epochs	5
Batch Size	16
Weight Decay	0.01
Initial Learning Rate	5e-5

Table 4: Hyperparameters for MULTI.1.A

The output from the above model is used as input to the final multi-label classifier (MULTI.1.B). Two sets of experiments were conducted, one with the CLS embeddings from MyRoberta, and another with the output of the fully connected classifier layer from MULTI.1.A, as the desired input to MULTI.1.B. These results are given in Section 5. MULTI.1.B consists of individual binary XGBoost (Chen and Guestrin, 2016) classifiers for each of the seven labels. These classifiers account for data imbalance for each given label by

[3] `MyRoberta.config.initializer_range`

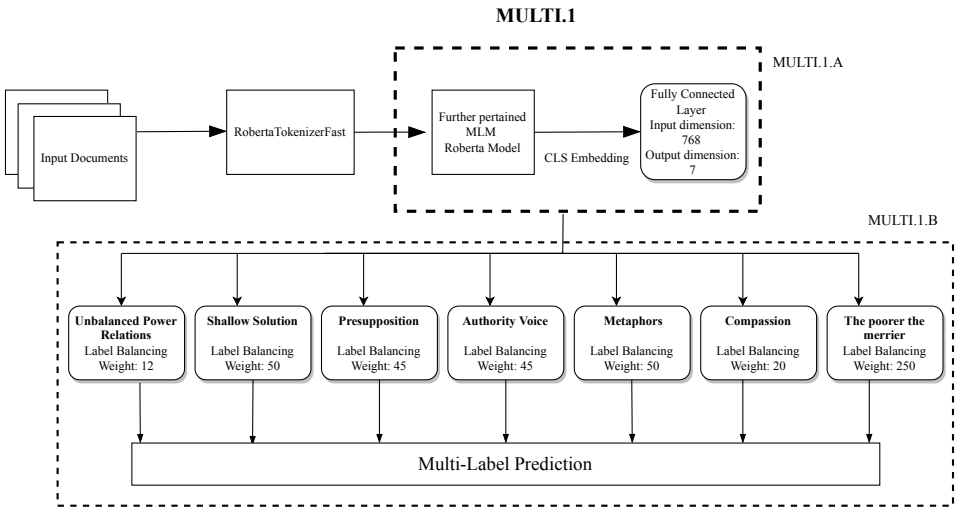

Figure 2: MULTI.1

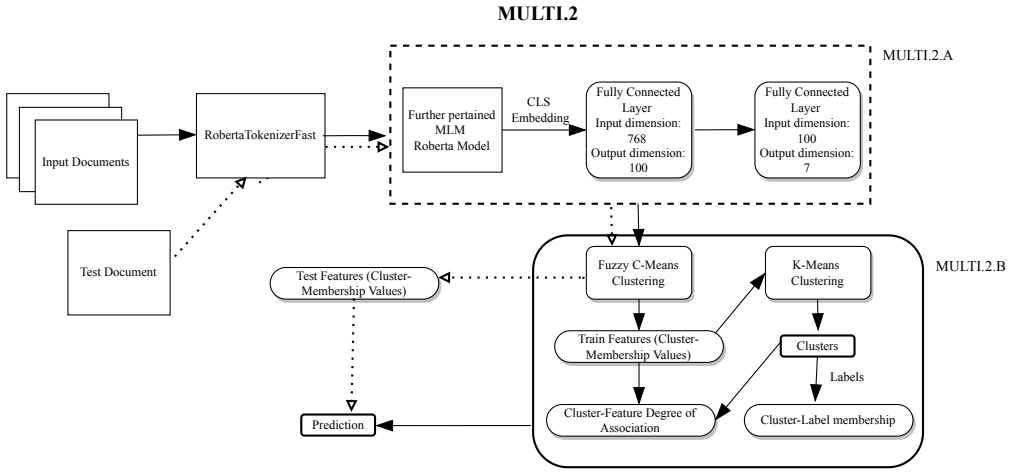

Figure 3: MULTI.2

using the parameter `scale_pos_weight` to indicate the number of negative examples per positive example in the training dataset[4]. The value of `scale_pos_weight` is chosen as the approximation of this value in different partitions of the dataset (Train, Dev and Train+Dev).

3.2.2 MULTI.2: MyRoberta + FCL_1 + FCL_2 + FCM + Fuzzy_CLF

This system uses a fuzzy membership-based ensemble classifier (Tandon and Chatterjee, 2022), consisting of two sub-models, namely MULTI.2.A and MULTI.2.B. The sub-model MULTI.2.A uses fine-tuned MyRoberta for feature extraction, which is fed to the sub-model MULTI.2.B for classification. The architecture for this system is given in Figure 3.

[4]The other parameters for these XGBoost Classifiers are given in the Appendix

The first component of MULTI.2.A is MyRoberta. The CLS embedding from MyRoberta is passed to a fully connected layer with output dimension 100, followed by a fully connected classifier layer with output dimension 7. While training the weights in the pooler layer and the last five layers for the MyRoberta are re-initialized as described in Section 3.1.2. This model is trained using Binary Cross Entropy loss applied with sigmoid activation, using AdamW optimizer. A linear learning rate scheduler is used with 100 warm-up steps and number of total steps corresponding to 25 epochs. This model uses 10 train epochs. All the other hyperparameter values are same as given in Table 4.

Output from the first fully connected layer in MULTI.2.A is used as an input to MULTI.2.B. MULTI.2.B uses this input for Fuzzy C-Means (Bezdek et al., 1984) clustering algorithm, in which

the clusters are considered as fuzzy sets over the set of all samples. Each cluster is represented by a fuzzy membership function. The parameters for this model are c (number of clusters) and m (the weighting exponent). This algorithm outputs Fuzzy C-partition $\mathbf{X} = [X_{i,j}]_{l \times c}$ which is used as the set of features for the main classification model. Here, l is the number of training documents. The use of this as the set of features aims at measuring the underlying uncertainty using membership-based measures.

Using these features the set of input documents $\mathcal{D}$ are clustered in k hard clusters. Each cluster thus formed is considered as a fuzzy set on the prescribed label set. In case of k clusters and p labels, the cluster is represented by a $p-$dimensional vector of fuzzy membership values which are utilized to assign the label set for an unseen example. Next, a measure of association of clusters to each of the extracted features (generated from Fuzzy C-Means clustering algorithm) is derived, which aids in retrieving a value to represent a new instance as a $k-$dimensional vector $(R_1, \cdots, R_k)$, where each R_i is the projection of the instance in the i^{th} cluster. Top s ($s \leq k$) clusters with highest R values are chosen. For a threshold value α the predicted label set for this instance is computed as the set of all labels whose membership to the union of these clusters is greater than or equal to α. The mathematical details of the algorithm are given in the Appendix.

4 Experimental Setup

The experiments were carried on Google Colaboratory in Python 3.7.12 with Nvidia Tesla P100 GPU. PyTorch (Paszke et al., 2019) and Huggingface Transformers (Wolf et al., 2020) are the key frameworks used to carry out the experiments.

The text from the training data was used to further pre-train RoBERTa and XLNet models, without any preprocessing. The text was tokenized using the fast implementations of RobertaTokenizer and XLNetTokenizer from the transformers library i.e., RobertaTokenizerFast and XLNetTokenizerFast, respectively.

The input text was tokenized using XLNetTokenizerFast for BINARY.1 and using RobertaTokenizerFast for BINARY.2.

For MULTI.1 and MULTI.2, the text in the training data was preprocessed using the following preprocessing steps.

- The punctuations `",;` – were removed from the text.
- Extra spaces between contractions were removed, and digits were removed from the text.
- The contractions were fixed using contractions library.[5]
- HTML symbols were removed.
- Any additional punctuations in particular, `!"#$%&()*+,-./:;<=>?@[\]^_'|~` were removed.

For MULTI.1 and MULTI.2, the preprocessed text is tokenized using RobertaTokenizerFast from the transformers library. The XGBoost classifier has been implemented using the XGBoost[6] library and the MiniBatchKmeans has been implemented using scikit-learn[7] Python library (Pedregosa et al., 2011). Fuzzy C-means is implemented using the 'fuzzy-c-means'(Dias, 2021) Python framework. For MULTI.2, the parameters are given in Table 5. These parameters are chosen by performing hyperparameter tuning using Train and Dev partitions of the dataset.

Parameter	Value
m	1.6
c	22
k	38
α	0.55
s	4

Table 5: MULTI.2.B Parameters

5 Results

Several experiments were conducted using the systems described in Section 3 and some variations of these systems. The systems were trained on the Training dataset (Train + Dev) and the metrics obtained on Test set are discussed in this section.

5.1 Subtask 1

In BINARY.1, to understand the impact of the layers MyRoberta, MyXlnet, and Attn_head, ablation experiments were carried out. For each of these experiments, each one of these layers are systematically removed and the results are reported. Since

[5]https://github.com/kootenpv/contractions
[6]https://xgboost.readthedocs.io/en/stable/
[7]https://scikit-learn.org/stable/index.html

Model	Precision	Recall	F1-Score
BINARY.1: MyRoberta + MyXlnet + Attn_head	0.607	**0.492**	**0.544**
BINARY.1: MyRoberta + Attn_head	0.606	0.486	0.539
BINARY.1: MyXlnet + Attn_head	0.621	0.470	0.535
BINARY.1: MyRoberta + CLS	0.606	0.470	0.529
BINARY.1: MyXlnet + CLS	0.568	0.489	0.525
BINARY.1: MyRoberta + MyXlnet + CLS	0.588	0.476	0.526
BINARY.2: MyRoberta + Attn_head	**0.631**	0.470	0.539
BINARY.2: MyRoberta + CLS	0.598	0.483	0.534
roberta-baseline	0.394	0.653	0.491

Table 6: Experiments: Subtask 1

Model	1	2	3	4	5	6	7	Average
MULTI.1: MyRoberta + FCL_1 + XGB	0.521	0.427	0.252	0.304	0.288	0.433	0.148	0.339
MULTI.1: MyRoberta + XGB	0.534	0.395	**0.253**	0.276	**0.395**	**0.454**	0.162	**0.353**
MULTI.2: MyRoberta + FCL_1 + FCL_2	0.545	0.410	0.231	**0.308**	0.279	0.432	0.214	0.345
MULTI.2: MyRoberta + FCL_1 + FCM + Fuzzy_CLF	0.480	0.390	0.176	0.268	0.247	0.350	0	0.273
MULTI.2: MyRoberta + FCL_1 + FCM + Base_CLF	0.534	0.421	0.232	0.276	0.235	0.420	0	0.303
MULTI.2: MyRoberta + FCM + Fuzzy_CLF	0.504	**0.457**	0.163	0.283	0.256	0.390	0	0.293
MULTI.2: MyRoberta + FCM + Base_CLF	**0.548**	0.405	0.219	0.256	0.286	0.362	0	0.297
MULTI.2: MyRoberta + FCL_1 + Base_CLF	0.541	0.368	0.227	0.288	0.273	0.413	0.148	0.322
MULTI.2: MyRoberta + Base_CLF	0.539	0.400	0.200	0.286	0.286	0.416	**0.267**	0.342
roberta-baseline	0.354	0	0.167	0	0	0.209	0	0.104

Table 7: Experiments: Subtask 2

BINARY.1 considers an ensemble of MyRoberta and MyXlnet, exactly one of these is eliminated in each set of experiments. Additionally, the attention mechanism on the last hidden state is substituted with using CLS embeddings (denoted CLS) from the previous layer. The results from these experiments are given in Table 6. This table also includes the result for BINARY.2 and the official roberta-baseline. Here, the proposed model in BINARY.2 is denoted by MyRoberta + Attn_head. For each of these models, the final layer is a fully connected layer which serves as a classifier unit.

The best precision is observed for the model MyXlnet + Attn_head among the BINARY.1 ablations, and for BINARY.2, overall. However proposed model i.e., MyRoberta + MyXlnet + Attn_head with the ensemble embedding layers and an attention head performs the best according to the overall F1-Score.

5.2 Subtask 2

In MULTI.1, two variations of the proposed model can be achieved, depending on the input to MULTI.1.B. The CLS embedding from MyRoberta in MULTI.1.A can be considered as an input to MULTI.1.B, or the output of the fully connected layer in MULTI.1.A can be considered as input to MULTI.1.B.

MULTI.2.A consists of three components, namely MyRoberta followed by two fully connected layers denoted as FCL_1 and FCL_2, respectively. MULTI.2.B has two parts, namely the

model consisting of Fuzzy C-Means clustering to extract features followed by a multi-label classifier model based on fuzzy membership. These two parts will be denoted as FCM and Fuzzy_CLF, respectively, as mentioned in Section 3.2. Ablation experiments were performed by removing components of these model to understand their impact in the overall system. For these experiments the main classifier layer is substituted with a One-vs-the-rest classifier which fits one classifier per class. The base estimator for this classifier is taken as a support vector classifier (This classifier was designed using the scikit-learn library) . This classifier is denoted as Base_CLF.

The results from these experiments are given in Table 7. This table consists of F1-Scores for each of the seven labels and the macro-average F1-Score.

It is observed that the F1-Score for the label 'Unbalanced power relations' is the highest among all other labels, at 0.548. This value has been observed for the the MULTI.2 ablation MyRoberta + FCM + Base_CLF. The highest F1-Score for the label 'Shallow Solution' is 0.457 using the model MyRoberta + FCM + Fuzzy_CLF. The highest F1-Scores for the labels 'Presupposition', 'Metaphor' and 'Compassion' are observed using MULTI.1 with the CLS embedding from MyRoberta considered as an input to MULTI.1.B. The highest F1-Score for the labels 'Authority Voice' is observed for MyRoberta + FCL_1 + FCL_2. The highest F1-Score for the label 'the poorer, the merrier' is observed for the MULTI.2 ablation MyRoberta + Base_CLF. Overall best performance is observed for MULTI.1 with the CLS embedding from MyRoberta considered as an input to MULTI.1.B. It is noted that all the presented experiments perform better than the roberta-baseline.

6 Conclusion

In this paper, systems for the task of binary and multi-label classification for detection of PCL in text, have been proposed. For the binary classification task, the first proposed system uses an ensemble transformer-based language model architecture. The other system detailed in the present work, for binary classification is a fine-tuned transformer-based language model with some custom layers. The first system proposed for the task of multi-label classification combines outputs from a transformer-based model fine-tuned for multi-label classification, and uses the embeddings from this model as features for individual label-balanced XGBoost models. Another system for multi-label classification has also been discussed, which uses a transformer-based model fine-tuned for multi-label classification, and uses the outputs from this model as inputs to a fuzzy-membership based ensemble classifier.

7 Acknowledgements

The experiments were conducted on Google Colaboratory.

A Appendix: Hyperparameters for Further pre-training

The RoBERTa and XLNet models were further pre-trained on the text from the Train+Dev set, with the parameter setting given in Table 8. This pre-training was carried out using RobertaForMaskedLM (RoBERTa Model with a language modeling head on top) and XLNetLMHeadModel (XLNet Model with a language modeling head on top) for RoBERTa and XLNet models, respectively. The value of batch size was chosen based on computation specifications, and the values of learning rate and weight decay were chosen based on empirical results presented in Sun et al.. To tune the hyperparameter representing the number training steps, the Train subset is used for training and the Dev set is used for validation.

Experiment Setting	Value
Train steps	30000
Batch Size	64
Initial Learning Rate	2e-5
Weight Decay	0.1
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Warm-up steps	10

Table 8: Experimental Setting for further pre-training

B Appendix: Attention Architecture

This is defined with a fully connected layer with input dimension as the size of the document embeddings of dimension $batch\ size \times sequence\ length \times embedding\ dimension$ from the previous layer, and output dimension taken as 512. This is followed by a Tanh activation and another fully connected layer with input dimension as 512, and output dimension as 1. A softmax activation is applied to this layer to get the weights

corresponding to each token in the sequence. Using these weights a weighted sum of the document embeddings of the size which were an input to this layer, is calculated producing an output of size $batch\ size \times embedding\ dimension$.

C Appendix: Hyperparameters for XGBoost Classifiers

The seven XGBoost classifiers trained in System B use the hyperparameter values as given in Table 9. For the hyperparameters not mentioned in the table, their default values given in the Python implementation of XGBoost Classifier are used. In the table, the parameter n_estimators denotes the number of gradient boosted trees, eta indicates the learning rate, min_split_loss indicates the minimum loss reduction required to make a further partition on a leaf node of the tree, max_delta_step indicates the maximum delta step allowed to each leaf output, subsample indicates the subsample ratio of the training instances, reg_alpha denotes the L1 regularization term on weights, and tree_method denotes the tree construction algorithm used in XGBoost.

Parameter	Value
n_estimators	100
eta	0.5
min_split_loss	0.1
max_delta_step	2
subsample	0.6
reg_alpha	0.5
tree_method	gpu_hist
objective	binary:logistic

Table 9: XGBoost Classifier Parameters

D Appendix: MULTI.2.B Algorithm

The algorithm for MULTI.2.B is given in two parts, the model generation algorithm and the prediction algorithm, as given below.

D.1 Algorithm for Model Generation

1 MiniBatchKMeans (Sculley, 2010) clustering algorithm is applied to $\mathbf{X} = [X_{i,j}]_{l\times c}$ to obtain k clusters namely $(D_1, \cdots, D_k)$, resulting in $\Omega = [\omega_{i,j}]_{k\times l}$ where

$$\omega_{i,j} = \begin{cases} 1 & \text{, if the } j^{\text{th}}\text{document} \\ & \text{is in cluster } D_i \\ 0 & \text{, otherwise} \end{cases}$$

2 A degree of association $\zeta(i,j)$ of each cluster D_i with each feature f_j is calculated as average of the feature values of f_j over all the documents in cluster D_i i.e., if the indices of the samples in cluster D_i are $\{e_1^{(i)}, \cdots, e_{|D_i|}^{(i)}\}$, then,

$$\zeta(i,j) = \frac{\sum_{r=1}^{|D_i|} X_{e_r^{(i)},j}}{|D_i|}$$

3 Cluster D_i is modeled as a fuzzy set of labels, with membership of label c_r to D_i as $\mu_{D_i}(c_r)$, for $r = 1, 2, \cdots, p$, which is modeled as the proportion of occurrences of label c_r in the label set of each document in the cluster D_i. Thus,

$$\mu_{D_i}(c_r) = \frac{\sum_{s=1}^{l} \omega_{i,s} G_{s,r}}{\sum_{s=1}^{l} \omega_{i,s}}$$

where $\mathbf{G} = [G_{i,j}]_{l\times p}$ is defined as

$$\begin{bmatrix} \mathbf{Y}_1 & \mathbf{Y}_2 & \cdots & \mathbf{Y}_l \end{bmatrix}^T$$

$\mathbf{Y}_i = (Y_{i,1}, \cdots, Y_{i,p})$ is the label vector for the i^{th} sample where $Y_j = 1$ if the document has label j and $Y_j = 0$ otherwise.

D.2 Algorithm for Prediction

Given an input text the following steps are performed to compute its label set.

1 This text is passed as input to MULTI.2.A and the output from the first fully connected layer is sent to the trained Fuzzy C-Means clustering algorithm. The Fuzzy C-partition from this is denoted by $\mathbf{v} = (v_1, \cdots, v_c)$.

2 Using $\zeta(i,j)$, a vector $\mathbf{R} = (R_1, \cdots, R_k)$ is calculated as,

$$R_i = \frac{\sum_{j=1}^{c} v_j \zeta(i,j)}{\sum_{j=1}^{c} \zeta(i,j)}$$

Here R_i represents the weighted average of $(v_1, \cdots, v_c)$ with weights as the degree of association vector $(\zeta(i,1), \cdots, \zeta(i,c))$ corresponding to the i^{th} cluster.

3 Top s, $s \leq k$ clusters with highest R values, say $\{D_{\gamma_1}, D_{\gamma_2}, \cdots, D_{\gamma_s}\}$ are chosen and their fuzzy union (or fuzzy t-conorm) is considered. In this case the fuzzy t-conorm algebraic sum[8] is used.

[8] $u(a,b) = a + b - ab$

4 For a given threshold α, predicted set of labels is computed as $\{c_i | \mu_{D_{\gamma_1} \cup D_{\gamma_2} \cup \cdots, D_{\gamma_s}}(c_i) \geq \alpha\}$.

References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

James C. Bezdek, Robert Ehrlich, and William Full. 1984. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203.

Niladri Chatterjee, Tanya Aggarwal, and Rishabh Maheshwari. 2020. Sarcasm detection using deep learning-based techniques. In *Deep Learning-Based Approaches for Sentiment Analysis*, pages 237–258. Springer.

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*, pages 785–794.

Shivani Choudhary, Kushagri Tandon, Raksha Agarwal, and Niladri Chatterjee. 2021. MTL782_IITD at CMCL 2021 shared task: Prediction of eye-tracking features using BERT embeddings and linguistic features. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 114–119, Online. Association for Computational Linguistics.

Nadia K Conroy, Victoria L Rubin, and Yimin Chen. 2015. Automatic deception detection: Methods for finding fake news. *Proceedings of the association for information science and technology*, 52(1):1–4.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Madson Dias. 2021. omadson/fuzzy-c-means. Original-date: 2019-05-13T16:29:01Z.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, page 29–30, New York, NY, USA. Association for Computing Machinery.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Carla Pérez-Almendros, Luis Espinosa-Anke, and Steven Schockaert. 2020. Don't Patronize Me! An Annotated Dataset with Patronizing and Condescending Language towards Vulnerable Communities. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5891–5902.

Carla Pérez-Almendros, Luis Espinosa-Anke, and Steven Schockaert. 2022. SemEval-2022 Task 4: Patronizing and Condescending Language Detection. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

D. Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web - WWW '10*, page 1177, Raleigh, North Carolina, USA. ACM Press.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to Fine-Tune BERT for Text Classification?

Kushagri Tandon and Niladri Chatterjee. 2022. Multilabel text classification with an ensemble feature space. *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–12.

Zijian Wang and Christopher Potts. 2019. TalkDown: A corpus for condescension detection in context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3711–3719, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.