NLP Coursework

Joanna Ye
jdy15@ic.ac.uk

Lew Hwi Hsien, Astrid
hl1022@ic.ac.uk

Chia-Hsin Lin
cl222@ic.ac.uk

1 Introduction

This report aims to develop a binary classification model to predict whether a text contains patronising or condescending language (PCL) for task 4-1 in the SemEval 2022 competition. Specifically, with the provided Don't Patronize Me! dataset (Perez Almendros et al., 2020), we aim to develop a model which outperforms the baseline model given by the task organisers in terms of F1 score.

Several tasks were undertaken to achieve the aim and this report is structured as follows. Section 2 conducts an exploratory data analysis on the dataset. Section 3 performs model selection and hyperparameter tuning. Section 4 considers three approaches to further improve the model performance. Sections 5 and 6 analyse the performance of the final model and compare it with two naive classifiers. Finally, Section 7 summarises the key insights and provides directions for future work.

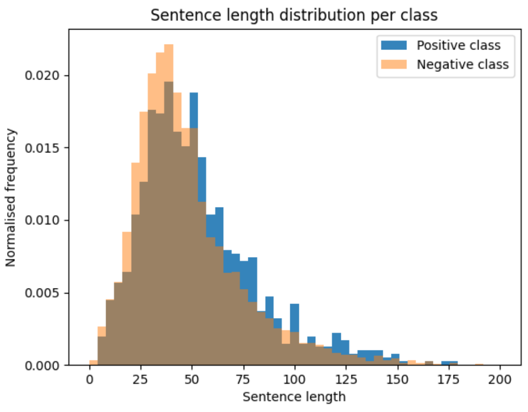

Figure 1: Distribution of sentence lengths per class.

2 Exploratory data analysis

2.1 Class labels

The provided dataset for detecting PCL is unbalanced, with a total of 993 positive examples and 9476 negative examples. Figure 1 shows the distribution of sentence lengths for the two classes and shows that the positive class contains a slightly higher proportion of longer sentences. Furthermore, figure 2 shows the proportion of examples from each keyword for both classes. This shows that while the examples in the negative class are fairly uniformly distributed across the 10 keywords, this is not the case for the positive class; more examples come from the keywords 'homeless', 'in-need' and 'poor-families'.

2.2 Qualitative assessment

The task of training a classification model to detect PCL is expected to be very difficult. Firstly, there is limited data, and in particular positive examples of PCL, to train the model with. Secondly, the use of PCL is generally subtler than other types of language that are targeted in NLP, such as hate speech. This means that the model may need to understand more complex relationships between words and the subtle implication of long phrases of text to identify PCL. Finally, whether or not a sentence contains PCL is subjective; even the two expert annotators disagreed on the classification of 1457 (14%) of the training examples (Perez Almendros et al., 2020). It is therefore reasonable to expect that non-experts and machine learning models may find it even more difficult to identify PCL. One illustration of the difficulty of the task is given by the two sentences: "Fast food employee who fed disabled man becomes internet sensation" and "More than 750 trolleys of food donated to those in need". Both concern food being given to a disadvantaged group. However, the implication of the word 'fed' in the first example means that this has been classified as PCL by the expert annotators, while the second one has not. This is subtle and may be difficult for a model to understand.

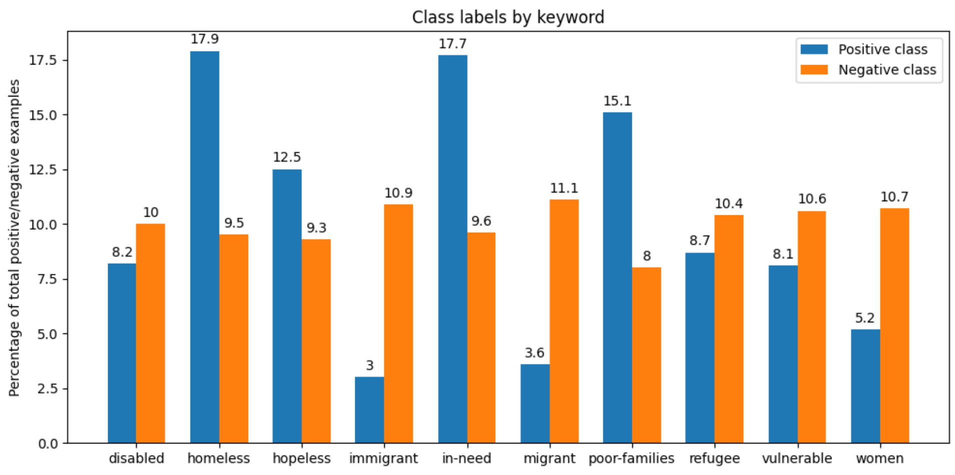

Figure 2: Distribution of keywords per class.

3 Model selection and hyperparameter tuning

3.1 Chosen model

To cope with the high difficulty level of the task, we decided to use the pre-trained Decoding-enhanced BERT with disentangled attention (DeBERTa) developed by Microsoft (He et al., 2020). This model was improved from the BERT and RobERTa models using a disentangled attention mechanism where each word is represented using two vectors to encode its content and positions, and the attention weights are calculated using disentangled matrices based on these two properties. In addition, there is also an enhanced mask decoder that replaces the original softmax layer in the output to predict the masked tokens during model pre-training.

3.2 Model training

Fine tuning of the transformer was conducted using the Trainer API from Hugging Face. The provided dataset was first converted into a binary classification problem by assigning all examples with labels 0-1 a negative classification (non-PCL), and all examples with labels 2-4 a positive classification (PCL). We then created an internal validation set by splitting the training dataset into train-train and train-val sets using a 0.8:0.2 ratio. Due to the low percentage of positive samples present in the full dataset, we checked the data labels after the split to ensure a relatively equal percentage of positive samples (8.7-9.6%) were present in both the train-train and train-val sets. The training data texts were then tokenized using the deberta-base tokenizer with padding. Using the Trainer API, the training and validation data and their respective labels were given as arguments. We set the evaluation strategy to be at every epoch (for 10 epochs) and customised a “compute metrics” function to calculate the accuracy and F1 score. We used the F1 score of the positive class as the performance metric for the training. The optimizer used was AdamW.

3.3 Hyperparameter tuning

To arrive at the optimal hyperparameters, we varied various hyperparameters for the model and evaluated the performance on the internal validation set. We used a trial and error approach where we varied the hyperparameters systematically, and used engineering assessment to decide on the next parameter to tune, as the high complexity of the model meant that a more exhaustive hyperparameter search was not computationally feasible.

The hyperparameters varied were: 1) Learning Rate, 2) Learning Rate Schedule, 3) Weight Decay, 4) Number of Training Epochs. The results are shown in tables 1 and 2 and explained below.

LR	LR Sched	Best F1	Epoch No.
1e-1	Linear	0.0	-
1e-3	Linear	0.0	-
1e-5	Linear Decay	0.57	5
1e-5	Constant	0.55	4

Table 1: Hyperparameter tuning results (learning rate and learning rate schedule).

Learning rate. Only a learning rate of 1e-5 gave satisfactory F1 scores, with the F1 score for 1e-1

WD	Best F1	Epoch No.
0.5	0.57	4
0.05	0.57	4
0.005	0.55	8
0.0	0.57	5

Table 2: Hyperparameter tuning results (weight decay).

and 1e-3 being 0.0. We suspect that the higher learning rates led to unstable parameter updates hence there was no convergence.

Learning rate schedule. We compared a constant learning rate with decreasing the learning rate linearly and adaptively with AdamW. The results showed that the latter provided better performance.

Weight decay. Given our learning rate, it was found that the best weight decay rate (both the highest and the average) was 0.05. This was likely due to a reasonable level of regularisation present in the optimisation process to prevent the model from overfitting.

Number of training epochs. We used 10 epochs as the default as we noticed that the F1 scores for the model had stabilised after this amount of training. However, through the Trainer API, we were able to evaluate the F1 scores at each epoch and the resultant best model is saved. So if early stopping was required, we were able to obtain the best model for the run.

We trained our model using cased inputs as the base DeBERTa model is cased, unlike BERT which has options of cased or uncased base models.

If possible, we would also have explored varying the batch size, but due to GPU memory limitations, we could only use batch sizes of 8 or below. However, the F1 score for a batch size of 8 was satisfactory. In addition, without compute resource limitations we would have wanted to adopt a k-fold cross validation methodology to get more representative validation results.

4 Further model improvements

A number of techniques were investigated to further improve the model performance. These are shown in table 3 with the F1 score achieved on the internal validation set for each technique. It can be seen that only contextual data augmentation improves the model performance beyond the best F1 score of 0.57 from section 3. Each technique is discussed in more detail below.

Model improvement	F1 score
Data pre-processing	0.50
Downsampling negative class	0.48
Upsampling positive class	0.56
Data augmentation: synonyms	0.55
Data augmentation: contextual	0.62

Table 3: F1 score on internal validation set for further model improvements.

4.1 Data pre-processing

We investigated whether performing traditional pre-processing on the training data (lower-casing, punctuation and stopwords removal) improved model performance. It was found that the F1 score on the internal validation set decreased; we believe this is because DeBERTa has been trained on data that was not pre-processed in this way. Removing punctuation, stopwords and capital letters can alter or make less clear the semantics of the text, therefore making it more difficult for the classification model to determine if there is patronising or condescending intent behind it.

4.2 Data sampling

Two different data sampling approaches were tried: random downsampling of the negative class to match the number of positive examples in the training data, and upsampling of the positive class to match the number of negative examples. Table 3 shows that neither approach led to an improvement in performance; it is hypothesised that this is because no new training examples are introduced to aid the learning of the model. Downsampling the negative class leads to worse results than upsampling the positive class. This is likely because the downsampling removes some of the training data and therefore reduces the knowledge that the model can gain.

4.3 Data augmentation

Two different data augmentation techniques from the nlpaug.augmenter.word library were investigated. The first replaces randomly selected words in the text with synonyms sourced from WordNet, while the second leverages contextual word embeddings from DistilBERT to replace words with similar counterparts. An example is given below of each augmentation technique applied to a sentence from the training data:

- Original sentence: “This nostalgia of homelessness is the fate of my generation and tribe. I am not the only one walking on this pathway of anguish, a whole caravan of wounded souls is walking with me.” (Ashfaq Ahmed)

- Sentence augmented with synonyms: “This nostalgia of homelessness is the luck of my coevals and tribe. Atomic number 53 am not the only 1 walking on this pathway of anguish, a whole caravan of hurt souls is walking with me.” (Ashfaq Ahmed)

- Sentence augmented with contextual word embeddings: “This nostalgia beyond homelessness is our fate facing my generation and tribe. I will surely the only one passing through this pathway of anguish, a whole caravan comprising tormented souls is walking with them.” (Ashfaq Ahmed)

Both data augmentation techniques were applied to the positive class only to increase the number of positive examples in the training set by a factor of 4. Table 3 shows that only the contextual augmenter led to an improvement in performance on the internal validation set. We believe this is because augmenting in this way attempts to preserve the overall semantics of the text by taking into account the context of each word. In contrast, augmenting with synonyms can significantly degrade the quality of the training data; in the example above, the word ‘I’ has been replaced with ‘Atomic number 53’ which has no meaning in the context of the sentence, and therefore is likely to confuse the classification model rather than provide it with more useful data to learn from.

4.4 Final model

Using the optimal hyperparameters and contextual data augmentation, a final model was trained on the full training dataset and tested on the official development set. This achieved a final F1 score of 0.61, which exceeds the RoBERTa-base score of 0.48.

5 Analysis of final model

To understand the performance of our final model, we analysed the predictions of the model based on three input features: level of patronising content, length of input sequence, and input keyword categories.

5.1 Level of patronising content

Table 4 shows the percentage of misclassified examples for each PCL level, where the levels are defined by the expert annotations of the data. The model shows lower error rates on texts with extreme (rather than higher) PCL levels 0, 1, and 4, and higher error rates on moderate levels 2 and 3. We suspect that the better performance on PCL levels 0 and 1 over level 4 is due to the unbalanced label distribution in the data, as more samples with low PCL levels are provided during training.

PCL level	**0**	**1**	**2**	**3**	**4**
Misclassified %	1	13	89	55	27

Table 4: Misclassified percentage of the trained DeBERTa model on texts with varying PCL levels.

5.2 Length of input sequence

Figure 3 shows the distribution of the input length given by the misclassified samples. For false negatives (texts with PCL), the distribution is similar to the input data distribution in figure 1, suggesting equal performance over different sentence lengths. For false positives (text without PCL), the error rate drops at text lengths between 50 and 75. We suspect that the rich number of negative samples in the data and the attention mechanism in DeBERTa enable the model to perform well on sentences with moderate length.

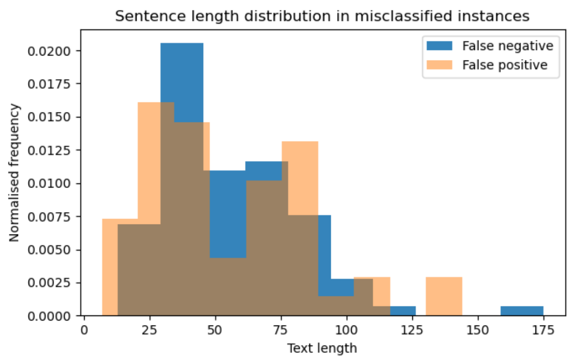

Figure 3: Distribution of sentence length per misclassification type of the trained DeBERTa model.

5.3 Keyword categories

Figure 4 shows the misclassified percentages for different keyword categories. For texts without PCL (false positives), our model excels on classifying samples from the keywords “disabled”,

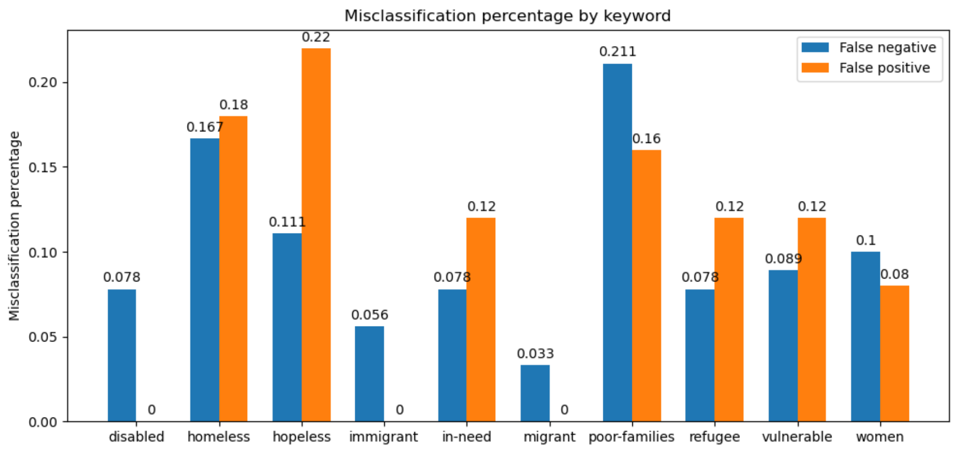

Figure 4: Distribution of keywords per misclassification type of the trained DeBERTa model.

"immigrant", and "migrant" but worse on "homeless" and "poor-families". For false negatives, the misclassified percentage distribution over the keywords is similar to that of figure 2, except for the keyword "in-need".

6 Model comparison

We compared our trained DeBERTa model with two naive classifers: a unigram bag of words with logistic regression (BoW) and a multi-layer perceptron (MLP) with two hidden layers and 50-dimensional word embeddings. For both models, the data is pre-processed by lower-casing and stop-words and punctuation removal. The prediction performances on the official development set are summarised in table 5.

Model	F1 score	Accuracy
DeBERTa (cased)	0.61	0.93
BoW (uncased)	0.27	0.90
MLP (uncased)	0.21	0.89

Table 5: Performance of the trained DeBERTa model and two naive classifiers on the official development set.

Since BoW obtains better performance than the two-layer MLP, we decided to analyse BoW.

6.1 Analysis of bag of words model

BoW is a text representation constructed by simply counting the occurrence of different words in a sentence. The BoW representation has several properties; 1. the resulting vectors are high-dimensional and sparse, leading to high memory costs, and 2. BoW treats sentences as sets of unordered words, which may be insufficient to encode semantic meanings given by the word order.

6.2 Misclassified example

To understand the properties of BoW better, we provide a misclassified example made by the BoW model in the official development set:

- "Bond went out of his way to help the less fortunate, often going on the road with Kim to take food to the homeless."

The above example is a false negative prediction of the BoW model. Two reasons are suspected for the wrong prediction: 1. BoW only focuses on the frequency of each word occurring in a text: there is only one word "homeless" in the text that could relate to PCL. The other words like "help" or "fortunate" existing on their own do not imply PCL. 2. BoW ignores the word order; it cannot comprehend the meaning of whole sentences and pay attention to relationships between words like transformer models do. For comparison, our final model correctly labels this example as PCL.

7 Conclusion

In this report, we developed a DeBERTa model which outperforms the baseline RoBERTa in classifying PCL in the Don't Patronize Me! dataset.

Although the model yields good performance, there is space for future improvements, such as classifying texts with moderate PCL levels. One solution is to use model ensembling, where multiple models are trained to improve the prediction accuracy. Other approaches including modifying the model architecture or the loss function are areas that are worth exploring in the future.

References

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention.

Carla Perez 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, Barcelona, Spain (Online). International Committee on Computational Linguistics.