DeBERTa4IPC: Decoding-enhanced BERT with disentangled attention for Identifying Patronizing Content

Andy Fu af723@ic.ac.uk
Alberto Almagro Sánchez aa9723@ic.ac.uk
Ziyuan Bao zb23@ic.ac.uk

1 Introduction

Patronising and condescending language (PCL), often subtle in its delivery, can have profound negative impacts on interpersonal communication and social dynamics. To reduce such media discourse, Carla et al. introduced a labelled dataset to train NLP models that can identify and classify PCL (Perez-Almendros et al., 2022). In this coursework, we focus on identifying PCL using fine-tuned language models, our implementation and the predictions are uploaded to our GitLab Repo.[1]

The official training set contains 10,469 textual data, each with metadata such as the article's source, the search keywords or the country code for the source media outlet.

2 Data analysis

The dataset is annotated by three experts: two annotators label each paragraph as 0 (No PCL), 1 (borderline PCL), and 2 (contains PCL). Due to the subjective nature of PCL, two annotators only achieved a Kappa Inter-Annotator Agreement (IAA) of 41% across three labels. The final agreement reached 86% after being coordinated by the third annotator. The dataset annotation is the sum of labels assigned by two annotators, where 0,1 means no PCL (negative), and 2,3,4 means contains PCL (positive). Detailed information can be found on the dataset's official GitHub.

2.1 Analysis of the class labels

Firstly, the training set contains 993 positive samples, accounting for approximately 9.49% of the total training set. The severely imbalanced dataset introduced difficulty in retrieving information about the minor class. Afterwards, as shown in Table 1 in the original paper (Perez-Almendros et al., 2022), each country and keyword has a balanced sample size.

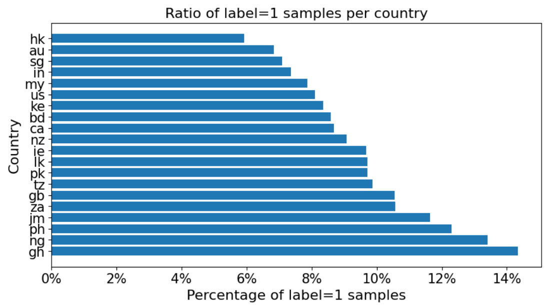

Figure 1: Bar chart displaying the ratio of positive samples of every country.

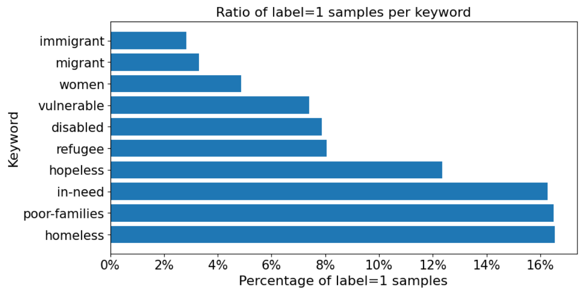

Figure 2: Bar chart displaying the ratio of positive samples of every keyword.

As shown in Fig 1, the highest positive rate of Ghana is approximately 2.4 times that of the lowest, Hong Kong, indicating that the distribution of the dataset is imbalanced across regions.

Similarly, as shown in Fig 2: the positive rates for "in-need," "poor-families," and "homeless" are the highest, around 16%, while "immigrant" and "migrant" have the lowest positive rates, around 3%. The frequency of positive data among keywords is severely imbalanced.

Finally, we also analysed the average sentence length per label and found that the positive paragraphs tend to be longer (Figure 3). This distribution could be explained as "using a literary style

[1]If the hyperlink fails to redirect, please use this link https://gitlab.doc.ic.ac.uk/zb23/nlp-coursework-1

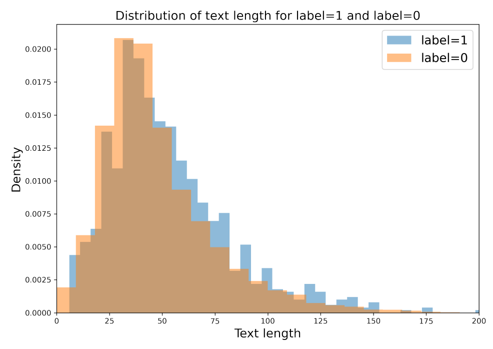

Figure 3: Histogram displaying the distribution of text length per label.

to describe people or situations is a type of PCL" (Perez-Almendros et al., 2022).

2.2 Qualitative assessment of the dataset

Given the low IAA between two annotators, the task is highly subjective. Besides, successfully identifying PCL requires background knowledge. Here are some examples from the dataset:

- **Positive:** After Vatican controversy, McDonald's helps feed homeless in Rome.
- **Negative:** More than 750 trolleys of food donated to those in need.
- **Negative:** On Wednesday 18th September the group held its first event, which was a charity drive to feed those who are homeless around Central-London.

In these examples, "feed" is considered more condescending than "donate food". However, with very similar semantics, the last example was classified as negative. To truly understand why the first example is positive, a language model needs to know what the "Vatican controversy" is and how it should relate to the homeless in Rome.

3 Basic hyper-parameter tuning

We used the DeBERTa-base model as our starting point, and we split the original training set into training and validation sets using an 80-20 proportion. We performed hyper-parameter tuning **on our validation set**, and found the optimal learning rate as 1e-5 (Table 1), weight decay as 1e-2 (Table 2. We used the default AdamW optimizer with linear learning rate decay 3.

Cased or uncased: By default, our model is cased, as we believe uppercase letters may indicate some information, like named entities. Based on our data pre-processing experiments, lower-casing all words does not improve the model performance.

Learning rate	Val_set F1-score
5e-6	0.452
1e-5	**0.530**
1e-4	0.411
5e-4	0

Table 1: Effects of learning rates on F1 score, with weight decay fixed to 1e-2, AdamW optimizer and linear learning rate decay.

Weight decay	Val_set F1-score
1e-2	**0.530**
1e-4	0.432
1e-3	0.449
1e-1	0.447

Table 2: Effects of weight decay on val F1 score, with learning rate fixed to 1e-5, AdamW optimizer and linear learning rate decay.

LR scheduler	Val_set F1-score
linear decay	**0.530**
constant	0.520
cosine	0.517

Table 3: Hyperparameter tuning of learning rate Scheduler. The initial learning rate is 1e-5 and the weight decay is 1e-2 for three experiments.

Maximum input length: We used a max sequence length of 192, as we found only 3% of the texts have longer lengths. We also found a greater max-sequence length may harm the performance.

Epochs and early stopping: We always load the best-performing model among all epochs after training. Thus, we believe there is no need to tune the number of epochs.

4 Further model improvements

The combination of data preprocessing and adding additional information performed the best. This subsection focuses mainly on the two successful techniques. As a third technique, data augmentation improved the default model performance but is not compatible with other approaches.

4.1 Data pre-processing

We attempted data pre-processing. The DontPatronizeMe dataset contains many texts from social media and online articles with varied and non-standard text formats, which may confuse the model. We analysed the dataset (Fig. 4), and implemented the follow pre-processings:

1. Removing the tag `<h>`, which comes from the HTML code used to retrieve texts.
2. Replacing tags `&amp`, `&gt`, `&lt` and `&quot` with `&`, `>`, `<` and `"` respectively; removing `More&gt;&gt;` from the end of texts.
3. Removing irrelevant Twitter usernames (`@` at the beginning or the end of the texts)
4. Converting texts to a lowercase format.
5. Since AI models do not perform well with contractions (Kacker et al., 2022), and given that the dataset separates most punctuation from words like `I 'm` or even `do n't`, we defined a mapping of more than 130 possible contractions and applied it to every text.
6. Removing duplicated quotation marks reduces the total number of `"` appearances from 17,270 to 9,281.
7. Removing duplicated white spaces.

Some techniques could harm our model's learning process, so we tried three possible pre-processing combinations. We denote as **basic pre-processing** the sequential application of steps 1, 2, 3, 5 and 7; while **medium pre-processing** refers to the application of steps 1, 2, 3, 4, 5 and 7; and **heavy pre-processing** includes all 7 steps. Basic pre-processing should always improve performance, while medium pre-processing and heavy pre-processing implement riskier strategies. Their impact on the character frequency can be seen in Fig. 4.

4.2 Data augmentation

We attempted Contextualized Word Embedding (CWE) data augmentation to introduce more diversity and balance the class sizes. We used the `nlpaug` library and the distilRoBERTa-base model to create new sentences using contextualized word embedding. This technique preserves the paragraph's contextual information but generates more diverse samples. Besides CWE, we have also attempted to create new sentences using back-translation and WordNet-synonym replacements, but they failed to improve the model.

4.3 Adding more information to input text

So far, we have only considered using as input for the language models the text column itself. However, the dataset provides information as well of the community the text may be patronizing to (from now on, the `keyword` or `k`) and of the country the text comes from (the `country` or `c`). Thus, we tried training the DeBERTa-base model using several combinations of all this information. To do this, we chose one character that was not present in the data, which was `|`, and created the input text as one of the following:

- **ck setup:** `c + | + k + | + text.`
- **c setup:** `c + | + text.`
- **k setup:** `k + | + text.`

5 Baselines

The two baseline models we have chosen are the T5 and Bag of Words, the implementation details are as follows.

- Sequence-to-sequence models can sometimes outperform other models in classification tasks. We compared T5-base (Raffel et al., 2019) to other baselines in this task. The T5 model's F1-score on the official dev-set is 0.525.
- We used the Bag of Words model for feature extraction and then trained a Logistic Regression model as a binary classifier, which classifies these features. The BoW model's F1-score on the official dev-set is 0.376.

These baselines were trained on the whole training dataset and tested on the official validation test. For the T5-base model, we added `patronize:` as the task-specific prefix.

Since the T5 model also contains a decoder, fine-tuning it with limited data is hard. The Bag of Words model is not powerful enough for this task. Our DeBERTa-base model with data pre-processing and extra keywords outperforms both baselines.

5.1 Analysis of features and misclassifications

Misclassified sample: "Many refugees do not want to be resettled anywhere, let alone in the US ." True label: 1. Predicted label: 0. Category: Presupposition.

After training the BoW model with the preprocessed data, we obtained a total of 26,842 features,

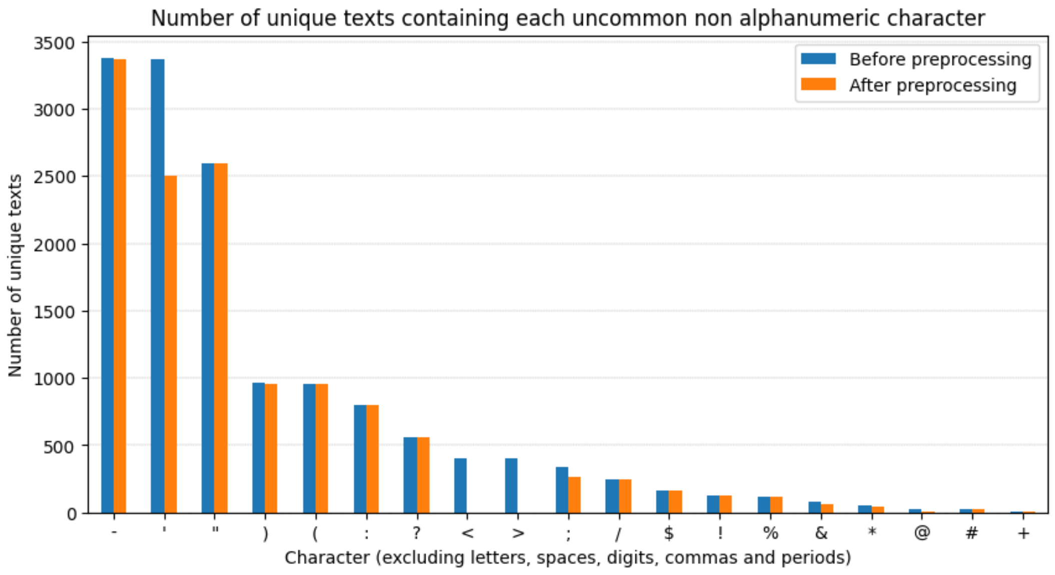

Figure 4: Histogram displaying the number of unique texts containing each non-alphanumeric (and excluding the space, comma and period characters) in the official training data, containing 10,469 total texts.

Pre-processing	Prefix setup	Augmentation	Dev F1-score	Test (official dev) F1-score
-	-	-	0.553	0.594
Basic	-	-	0.569	0.616
Medium	-	-	0.562	0.622
Heavy	-	-	0.549	0.613
-	ck	-	0.571	0.620
-	c	-	0.517	0.595
-	k	-	0.583	0.596
-	-	CWE	0.580	0.601
Basic	k	-	**0.591**	**0.628**
Basic	-	CWE	0.508	0.526
-	k	CWE	0.551	0.575
Basic	k	CWE	0.563	0.585

Table 4: F1-scores for our different technique combinations. Similar techniques are grouped, and the best-performing one among each group is underlined. The overall best combination is highlighted with a bold font.

which is the total number of unique words in the training set. The vector form for the above example is [14814, 19946, 7404, 16707, 26023, 24403, 2841, 20316, 1862, 13997, 1544, 11886, 24150, 25495], each item being the index of the word in the feature list.

’Presupposition’ means this sample assumes “Refugees don’t want to be resettled anywhere, let alone in the US” without reliable information sources. The reason for this misclassification is BoW models treat text as a collection of individual words without considering the context or the order in which they appear. Words in the sentence such as “refugees” “don’t” and “want”, when combined, could imply a condescending sentiment depending on the context. However, individually, these words might not be strong indicators of PCL to the model.

6 Results and hyper-parameter tuning

We first performed basic hyper-parameter tuning, as mentioned in section 3; minor differences in basic hyper-parameters may significantly affect performance. For example, when the learning rate is greater than 5e-4, the model gets 0 F1-score.

We then used the found hyper-parameters to explore other techniques further. Further improving the performance is challenging, most techniques

original label	0	1	2	3	4
error rate	0.03	0.18	0.72	0.39	0.22

Table 5: Error rate per original label.

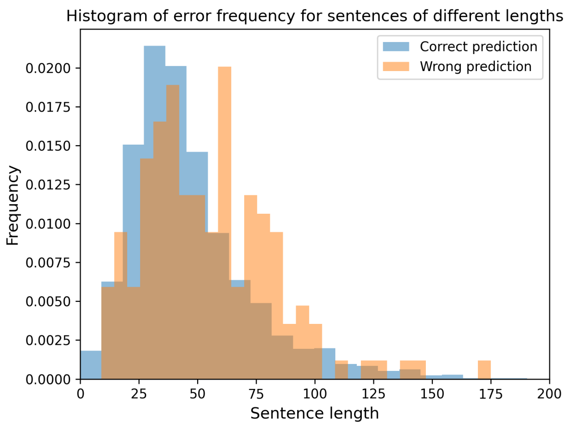

Figure 5: Histogram of error frequency and correct frequency for sentences of different lengths in the test set.

did not work as expected thus yielding poor performance, e.g., Replacing words (WordNet synonym) from positive text samples may create ill-formed sentences, which completely confused the model (0.242 F1-score).

Some techniques are not compatible. Both data augmentation using contextualized word embeddings are proven beneficial, whereas its combination with pre-processing decreased the F1 Score.

Our results for further experiments are shown in Table 4. We included performance on the official validation set, although we did not use it to select which techniques were the best.

7 Analysis of the model

7.1 Model performance on different levels of PCL content

From Table 5, we can see that the model has the highest error rate when the original label is 2. This is because data labelled as 2 are considered borderline cases by both annotators. Although it is also uncertain whether the text contains PCL when the label is 1, both labels 0 and 1 are considered as no PCL. And since the dataset is imbalanced, the model tends to classify borderline cases as 0.

7.2 The impact of the sequence length of the input on the performance of the model

According to Figure 5, by comparing the distribution of the data lengths that were misclassified

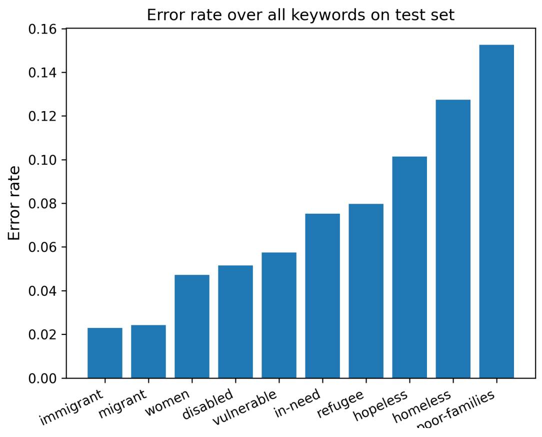

Figure 6: Bar graph of error rate over all keywords in the test set.

with the data that were correctly classified, we can find that the sequence lengths of the misclassified data are longer. Besides dealing with longer inputs being more complicated, our model tends to misclassify positive samples. As the data analysis section mentioned, positive samples tend to have longer sentence lengths. Therefore, the model has a higher error rate with long sentences.

7.3 Model performance on different keywords

The impact of keywords on the model's performance is similar to what was mentioned earlier. According to Figure 6, 'homeless' and 'poor-families' have the highest error rate because compared to other keywords, "homeless" and "poor-families" have more positive samples. Since the model performs worse on positive samples, the samples corresponding to "homeless" and "poor-families" have a higher error rate.

8 Conclusion

We achieved the best F1-score of **0.631** on the official dev set using a combination of the DeBERTa-base model with basic data preprocessing and keyword setup. We also found that the model performs worst when dealing with borderline cases (label 2), that the accuracy decreases with longer sentences, and that the accuracy is lower when facing the text of keywords 'homeless' and 'poor-families'. For future works, we believe that using original articles to further pre-train the model and then fine-tuning the model on the PCL dataset might yield better performance.

References

Michal Haltuf. 2019. Best loss function for f1-score metric. Available through this link. Lastly checked on 2 March 2024.

Prateek Kacker, Andi Cupallari, Aswin Gridhar Subramanian, and Nimit Jain. 2022. Abb-bert: A bert model for disambiguating abbreviations and contractions.

Carla Perez-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)*, pages 298–307, Seattle, United States. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

A Appendices

A.1 Additional Experiments

Here, we detail all attempted techniques (with suboptimal results) for interested readers. These techniques are not included in our final model and thus should not contribute to marking. Tabel 7 contains results of methods we did not mention in the main text.

A.1.1 Data sampling

Data sampling is another technique that deals with unbalanced datasets. By sampling the minor class with higher probability, the model is able to learn from classes with insufficient data. We attempted upsampling the minor class by a factor of 2. This obtained an F1-score of 0.575 on our dev-set, which is better than the base model, but worse than the CWE data augmentation. Since both methods have similar purposes, we chose CWE as our main data augmentation mechanism.

A.1.2 Data augmentation

Beyond simply upsampling the minor class, we want to introduce more diversity by data augmentation. We have attempted three data augmentations: WordNet synonym replacement, back-translation and contextual word embedding augmentation. We only performed data augmentation on the positive class and combined the augmented data with the original dataset.

WordNet: randomly replacing 10% words with synonyms from the WordNet, skip the replacement if a selected word has a length smaller or equal to 3. However, the technique created many low-quality samples, actually poisoning the dataset.

Back-translation: French is considered a resource-rich language that is close to English, we used the `googletrans` library to translate sentences into French, then translate back. However, this back-translation may also distort the sentence's true meaning, the resulting F1-score on our validation set was only 0.518.

A.1.3 F1 loss function

We noticed the binary cross-entropy does not correlate well to the observed F1-score, which is not optimal for training since the model does not try to optimise its weights in the best direction. Following (Haltuf, 2019), we tried to use a differentiable version of the F1-score as our loss function. However, the best result we obtained was 0.548 for the validation set, worse than the 0.553 by our DeBERTa-base model with the default loss function. Consequently, we discarded this idea.

A.1.4 Weighted loss function

Weighted loss is commonly used to deal with unbalanced datasets, which attempt to balance the weight update for each class by multiplying the loss for each label with a constant. We did not use a weighted loss as it did not improve the F1-score on the validation set.

Max input tokens	64	96	128	192	256	384	512
Texts needing higher number of tokens	8361	2960	966	27	8	4	2
	79.9%	28.3%	9.23%	3.01%	0.26%	0.04%	0.02%
Maximum batch size	96	64	48	32	16	12	8
Time per epoch (min)	2	3	4	6	12	18	24

Table 6: Analysis of different possible values for the maximum number of input tokens. Each value contains the number of texts that need a higher number of tokens to be fully encoded (both the absolute number and the percentage to the total number of texts), the maximum batch time that this value would allow without exceeding the GPU capacity[2] and the corresponding time per epoch under these conditions.

Other improvement methods	Val_set F1-score	test_set F1-score
Upsampling	0.575	0.582
Downsampling	0.541	0.594
Weighted loss	0.535	**0.596**
WordNet replacement	0.242	0.269
Back translate	0.518	0.595
CWE Augmentation	**0.580**	0.582

Table 7: Evaluation of methods not mentioned in the main text. These methods are discarded ideas. The best result is highlighted with a bold font.