Don't Patronise Me! SemEval-2022 Task 4 - Patronizing and Condescending Language Detection

Jamie Couchman, Jonah Anton, Mustafa Saleem

1 Introduction

Pre-training of Deep Bidirectional Transformers [1, 2] allows for a deep sense of language context in a wide variety of NLP tasks. In this report we apply the transformer architecture for detection of Patronizing and Condescending Language (PCL) [1]. PCL towards vulnerable communities is often involuntary and unconscious, and its detection is essential in raising awareness and prompt action for under-represented individuals.

2 Background on PCL and the *Don't Patronize Me! dataset*

2.1 Why is PCL detection hard?

PCL is often subtle. Ref.[3] notes that speech constitutes PCL when it "shows a superior attitude towards others or depicts them in a compassionate way". However, formulating precise criteria for this is difficult and subjective.

The patronizing content of the *Don't Patronize Me! dataset* [3] is determined by two main annotators, each of whom assign to each paragraph a label - 0 (no PCL), 1 (borderline) or 2 (PCL). Paragraphs with final labels 0 and 1 are treated as instances of no PCL, and those with final labels 2, 3 and 4 as instances of PCL.

In many cases the distinction between the language used in any two paragraphs, in which one is established as an instance of PCL and the other as not, is nuanced. Further, there exist a large diversity within the dataset for what makes a particular paragraph patronizing and/or condescending. A high performing PCL classifier will therefore have to learn precise and diverse language semantics, a task which we anticipate to be difficult.

2.2 Exploratory Data Analysis on the *Don't Patronize Me! Dataset*

The dataset is divided into internal training and validation sets. The resulting training set has 8375 entries and the test set has 2094. There exists a clear class imbalance within the training set, with 7581 negative and only 794 positive instances. There further exists a discrepancy between the lengths of the paragraphs for the positive and negative samples. We observe that paragraphs containing PCL tend to be longer (mean of 61 tokens) than those without (mean of 55 tokens). We further find some correlation with the keyword associated with each instance, with higher PCL rates for the keywords *homeless*, *in-need* and *poor-families* (all above 16%).

3 Experiments

3.1 Implementation of RoBERTa Baseline and consideration of other models

In these initial experiments no data pre-processing is considered. All models, unless otherwise stated, are trained for 10 epochs with default hyperparameters. Re-implementation of the baseline model provided by the SemEval competition organisers, through fine-tuning pre-trained RoBERTa$_{base}$ [4], achieves an F1-score for the positive class of 0.49 upon evaluation on the validation set. We also consider fine-tuning pre-trained XLNet [5]. This marginally improves the F1-score to 0.50. Further, we consider DeBERTa [6]; fine-tuning pre-trained DeBERTa results in an F1-score of 0.51. As it has the strongest baseline performance, DeBERTa is selected as the default language model for the reminder of this analysis. All further changes, therefore, aim to build upon and improve the DeBERTa baseline.

3.2 Data pre-processing

We experiment with a number of different data pre-processing techniques. We note that all pre-processing steps are applied exclusively on the training set, allowing model evaluation on the validation set to be an accurate indicator of model generalisation performance on an unseen test set.

1. **Lemmatization** Reduction of all words to their base form. For example, *ate* and *eating* both get converted to *eat*. In this way the language becomes standardised across PCL ex-

[1] Code found in link: drive.google.com/drive/folders/1lHnrZzxZhhAys320px7phwzEyWp5SX75?usp=sharing

amples, simplifying the inputs to the model. Solely through the introduction of lemmatization, using the `spaCy` [7] 'en-core-web-sm' English pipeline, the F1-score improves to 0.53.

2. **Stop Word Removal** Removal of commonly used words which add little semantic meaning to a given text, such as *nevertheless*, *whereby*, *such* etc. This is implemented using the default `spaCy` 'en-core-web-sm' stop word list. Fine-tuning DeBERTa after the application of stop word removal on the training corpus, however, significantly reduces the F1-score of the validation set positive class to 0.48. We anticipate, therefore, that stop word removal alters the patronizing content of the text, thereby confusing the language model.

3. **Data Augmentation** We consider two different data augmentation techniques, both of which increase the size of the input text.

   Synonym Replacement
   Words are randomly replaced with synonyms using WordNet[8] to over-sample the positive class. We find that this reduces the F1-score from 0.52 to 0.49. This is due to the loss of semantic meaning in the text through replacement with synonyms, masking subtle PCL.

   Backtranslation
   Text paragraphs are translated into a target language and then re-translated back to the source language. This leads to subtle paraphrasing of the original text, thereby automatically crafting similar but ultimately linguistically different text samples. Following [9], we perform backtranslation through the Google Translate API on the entire training corpus, using Spanish and Chinese as intermediary languages, augmenting the dataset threefold. To give one example, the sentence *'These are the two countries with key macroeconomic challenges including unemployment."* is backtranslated, via Chinese, to *'Both countries face major macroeconomic challenges, including unemployment."*. Applying DeBERTa with backtranslation and lemmatization, gives us an F1-score of 0.57.

4. **Addressing the class imbalance** To address the class imbalance, we consider both over-sampling of the positive class, down-sampling of the negative class and weighting the loss of both classes (with two different sets of weights). The resulting F1-scores are shown in Table 1.

5. **Tokenization** & **Casing** We use the default DeBERTa tokenizer, which implements end-to-end tokenization. By default, the DeBERTa tokenizer lowercases the input when tokenizing. Since this is how the model is pretrained, we anticipate stronger performance with uncased inputs. We also train a model with cased inputs and find this reduces F1-score to 0.51.

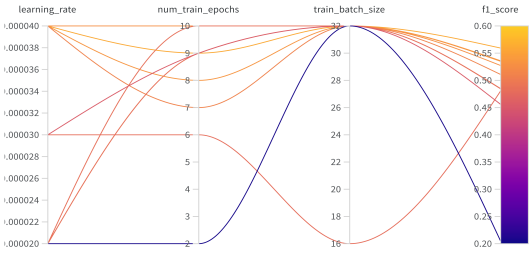

Figure 1: Hyperparameter optimisation using wandb sweeps with Bayesian Optimisation on DeBERTa-base.

Data Augmentation	F1-score
DeBERTa baseline	0.51
+ lemma	0.53
+ stop-word	0.48
+ synonym	0.49
+ lemma + backtrans	0.57
+ lemma + over-sample	0.51
+ lemma + down-sample	0.50
+ lemma + weights$_{v1}$	0.52
+ lemma + weights$_{v2}$	0.53

Table 1: The effect of different data pre-processing and in-processing techniques on the F1-score of the positive class (contains PCL) when evaluated on the internal validation set. All changes are added on top of the DeBERTa baseline, training for 10 epochs with default hyperparameters. weights$_{v1}$ gives weights [0.25, 1] to the negative and positive classes, respectively, and weights$_{v2}$ gives weights [0.55, 5.27].

3.3 Model hyperparameter selection

Initial hyperparameters for DeBERTa are set to those in the original DeBERTa paper [6]. We consider tuning a subsection of these hyperparameters on the validation set, varying learning rate, batch size and number of training epochs. This is implemented using 10 Weights & Biases (*wandb*) sweeps [10], where each successive sweep location is chosen via Bayesian Optimization. The F1-scores for 10 hyperparameter configurations are shown in Fig.1. We find that optimal performance is reached with a batch size of 32, 9 training epochs and a learning rate of 4×10^{-5}.

3.4 Results

We fine-tune a DeBERTa$_{base}$ language model on the entire backtranslated training set, applying lemmatization and weighting the two classes within the loss with weights$_{v2}$. The model is trained with

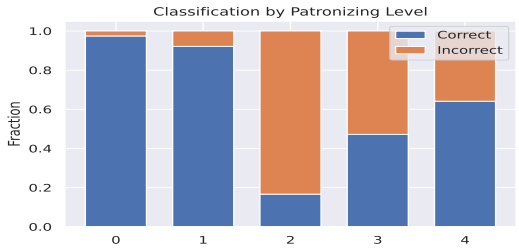

Figure 2: Classification by PCL patronizing level on the validation set.

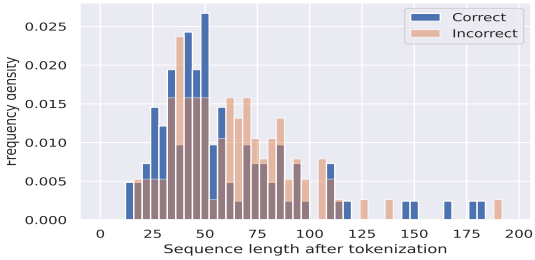

Figure 3: Distribution of sequence lengths for correctly (blue) and incorrectly (orange) classified instances in the validation set. The sequence lengths are calculated after tokenization with the DeBERTa default tokenizer.

the optimal hyperparameter configuration. All other hyperparameters are default as in [6]. We obtain an F1-score of 0.58. Upon submission to CodaLab, the model achieves an F1-score of 0.52 on the unseen test set.

4 Analysis

To what extent is the model better at predicting examples with a higher level of patronising content?

The model is better at correctly classifying language with no PCL content (F1-score of 0.96 for the negative class). As alluded to in section 2.1, many of the paragraphs are subtle in their PCL. This is reflected in Fig.2, as for level 2, where both annotators consider each instance as borderline PCL cases (1 + 1), we see extremely poor performance, with only 3/18 PCL instances picked up on. We do observe an increase in model performance for more obvious PCL instances (levels 3 (42/89 correct classifications) and 4 (59/92)), which mirrors the difficulty levels the experts themselves found in classifying the text paragraphs for PCL.

How does the length of the input sequence impact the model performance?

Fig.3 shows the distribution of sequence lengths for both correctly and incorrectly classified text instances in the validation set. We observe that the model is clearly better at classifying shorter sequence lengths. We anticipate that this is correlated with that negative samples are classified more accurately, as, as noted in section 2.2, the negative samples in the training data tend to have a shorter sequence length than the positive ones.

We further explore this dependency of model performance on sequence length by varying the maximum sequence length allowed as input into the DeBERTa model, considering values of 128 and 512 (defaults to 512). However we find that this variation has no significant impact on the F1-score, presumably because the significant majority of instances have sequence length under 128.

To what extent does the categorical data provided influence the model predictions?

We observe that there exists some correlation between the fraction of instances predicted as positive and the country codes. For example, 12.5% of instances originating from Ireland are predicted to contain PCL, compared to 3% for instances originating from Bangladesh. However, the proportion of paragraphs containing PCL varies between countries within the validation set, and we find minimal correlation between the fraction of correctly classified (both as positive and negative) instances and country.

There exists some correlation between keyword and correct classification, with, for example, 216/226 instances (96%) with keyword *in-need* being classified correctly but only 159/190 (84%) for the keyword *poor-families*.

5 Conclusion

We fine-tune the pre-trained DeBERTa$_{base}$ model on the *Don't Patronize Me! Dataset*, applying lemmatization, backtranslation (through Spanish and Chinese intermediary languages) and class weighting within the training loss. After performing hyperparameter optimization, we achieve an F1-score of 0.58 on the held-out validation set and 0.52 on the official test set.

We find that the model is able to better predict examples with a higher level of patronizing content and struggles with longer sequence lengths, suggesting it fails to capture long range dependencies in PCL. We anticipate that further work on patronizing and condescending language detection may consider using POS tagging of adjectives, verbs and adverbs, considered to capture the majority of the sentiment of a text sample, as additional input embeddings into the language model to help guide the model towards the text sentiment during training, building on the work of Ref.[11].

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Carla Perez-Almendros, Luis Espinosa-Anke, and Steven Schockaert. 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, 2020.

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

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

[6] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.

[7] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[8] Ingo Feinerer and Kurt Hornik. *wordnet: WordNet Interface*, 2020. R package version 0.1-15.

[9] Jean-Philippe Corbeil and Hadi Abdi Ghadivel. BET: A backtranslation approach for easy data augmentation in transformer-based paraphrase identification context. *CoRR*, abs/2009.12452, 2020.

[10] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[11] Kai Shuang, Mengyu Gu, Rui Li, Jonathan Loo, and Sen Su. Interactive pos-aware network for aspect-level sentiment classification. *Neurocomputing*, 420:181–196, 2021.