Coursework for Natural Language Processing (70016)

Emotional Damage Team: Yuze An, Boyu Han, Xinyu Bai

1 Introduction

The Patronizing and Condescending Language (PCL) has been long presented as an interesting and technical challenge in NLP. Although PCLs are usually used unconsciously out of goodwill, it still feeds stereotypes, routinizes discrimination and drives to greater exclusion. Therefore, detecting PCL have become especially important. In this paper, we proposed a text-classification pipeline based on Microsoft's DeBERTa transformer (He et al., 2021b) and introduced various techniques to increase the overall performance. Our code is available on Github (Bai et al., 2022).

2 Exploratory Data Analysis

2.1 Data Description

The **Don't Patronize Me!** dataset (Pérez-Almendros et al., 2020) contains 10,636 paragraphs about vulnerable communities extracted from News on Web corpus (Davies, 2013). Each sample was annotated by label from 0 (not containing PCL) to 4 (being highly PCL). In task 1, we merged these five labels into two categories; setting labels 0 & 1 as new label 0 (no PCL) and setting labels 2 to 4 as new label 1 (contains PCL).

2.2 Data Analysis

First, we studied the distribution of the newly defined labels. As shown in Table 1, samples in training set for each class is highly unbalanced, with a proportion of nearly 9:1. In addition to the analysis of class labels, we also calculated the length distribution of all sequences and the result is drawn in Figure 1. Nearly 80% of the samples are between 200 and 500 tokens, while some of them are longer than 600 tokens. This result inspires us to properly choose the size of the tokenizer to retain as much information as possible.

Types of Label	0	1	2	3	4
Number of Label	6825	756	126	369	299
Proportion of Label	81.49%	9.03%	1.50%	4.41%	3.57%
Types of New Label	0		1		
Proportion of New Label	90.52%		9.48%		

Table 1: Statistics per label in training set

Tokenization is to encode a string of text into transformer-readable token ID integers. Limited by the positional embeddings mechanism in transformers architecture, common tokenizer input sizes are 256, 512 and 1024. In consideration of memory consumption and conservation of input information, it is crucial to select proper size for it. Table 2 presents the number of samples per class given four different length intervals in the training set. From the table, we can see the proportion of two classes in each length interval are evenly distributed, and the majority of samples are shorter than 512 tokens.

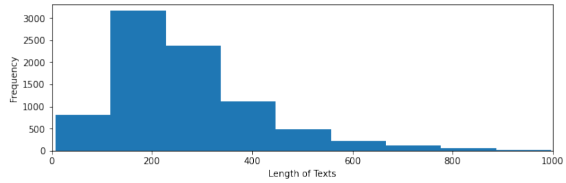


Figure 1: Histogram for the length of each sentence

Length Interval	Total Samples	Label 0	Label 1
[0,256)	4814 (57.48%)	4410 (91.61%)	404 (8.39%)
[256,512)	2986 (35.65%)	2667 (89.32%)	319 (10.68%)
[512,1024)	565 (6.75%)	496 (87.79%)	69 (12.21%)
[1024,5501]	10 (0.12%)	8 (80.00%)	2 (20.00%)

Table 2: Statistics for samples in each length interval

2.3 Qualitative Evaluation

To begin with, the imbalance situation in dataset is hard to resolve given that text data is not very easy to manipulate. Also, compared to other types of data such as images, text data is sequential and more sensitive to contexts. The complexity in contexts adds another burden to this task. Furthermore, some of the texts are too short to form complete semantics (disagreement cases), which is also a key factor that makes our task especially difficult. For example, keyword **disabled** appears in both of the following samples, but the context and semantics here are quite different.

- Label 1: *Ireland labels Eoghan **disabled** and has **disabled** him.*
- Label 0: *The Ghana Aids Commission (GAC) is currently engaging persons with disability for the preparation of a strategic plan which would include them in the national response to the HIV/AIDS , to step up its prevention among the **disabled**.*

Pérez-Almendros et al. (2020) mentioned that when divergence of labelling occured, they introduced the third annotator as a referee to provide a final label. This indicates that the task is subjective even for human annotators.

3 Modelling Decisions

3.1 Data Preprocessing

Considering the imbalanced situation as well as the small volume (8375) of the training dataset, several solutions were proposed and tested on the baseline RoBERTa model to reflect the quality of these methods, as shown in Table 3. Both the undersampling and upsampling methods are designed to generate an equal number of samples for both classes, while the weight for weighted loss method is set to be the inverse of the proportion of each class with respect to the entire training set. We are surprised to find that the upsampling method outperforms more advanced data augmentation methods and obtained the highest accuracy on the baseline model. The weighted loss is not as effective as the augmentation method simply because most of the times, the input does not contain samples from positive class thus weakens the effect of the weights. Based on initial experiments, we decided to adopt upsampling method for data balancing and augmentation.

Preprocessing Methods	**F1-Score** (%)
Undersampling the majority	48.12
Upsampling the minority	**52.57**
Back Translation	51.26
Synonym Replacement	51.44
Weighted Loss	50.89

Table 3: F1-Score on RoBERTa baseline model

3.2 Model Selection and Tuning

To find a suitable architecture for this task, we explored a number of pretrained models from HuggingFace (Wolf et al., 2020) and fine-tuned them on the upsampled official training set (Table 4). We followed the general guidelines (Sun et al., 2020) as well as the model configuration profiles to determine the hyper-parameters. Since sequences that are longer than 512 tokens only take up about 6.87% of the entire training set, we used 512 as the maximum input token size. In the fine-tuning process, the precision is increasing as the model learns the task, while the recall keeps decreasing, resulting in an improvement in F1-score. At certain point of the training, the precision surpasses recall and keeps increasing, which indicates that the model begins to over-fit on the training set as depicted in Figure 2.

Model	Precision	Recall	**F1-Score** (%)
DeBERTa (He et al., 2021b)	0.675	0.492	0.569
DeBERTaLarge (He et al., 2021b)	0.681	0.492	0.571
DeBERTaV2XLarge (He et al., 2021b)	0.549	0.673	**0.605**
DeBERTaV3Large (He et al., 2021a)	0.609	0.587	0.598
XLNet (Yang et al., 2020)	0.535	0.636	0.581
Longformer (Beltagy et al., 2020)	0.540	0.673	0.600
Hyper-parameters			**Value**
Learning Rate			$[2 \cdot 10^{-5}, 4 \cdot 10^{-5}]$
Decay Epsilon			$1 \cdot 10^{-4}$
Fine-tune Epoch			$[3, 4]$
Input Token Length			512
Batch Size			$[2, 4]$

Table 4: Fine-tuned model performance

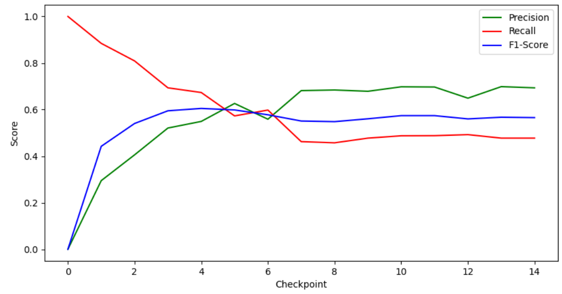


Figure 2: Metrics over training checkpoints

Based on the observation, we think it is crucial to perform early stopping in the fine-tuning stage. We used F1-score as the sign and stop the training when it started to decrease. When determining the perfect spot for early stopping, we find that a learning rate larger than $4 \cdot 10^{-5}$ is not desirable since it makes the model over-fit too quickly.

From our experiment, larger models tend to perform better than smaller models in the fine-tuning stage. Among all the best-performing models, DeBERTaV2XLarge returns the top F1-score (60.5%) on the upsampled dataset with early stopping.

To train the model on GPU, we decided to reduce model precision from FP32 to FP16 instead of using smaller batch size and input token length. Based our experiment, compromising either batch size or input size severely affects model performance since 35.7% of the sequences are located within the range $[256, 512]$ and a small batch size would greatly undermine the quality of gradients.

Our final optimization focused on the imbalanced precision and recall score. It is mainly due to the inconsistency in data distribution between training set and test set. We introduced an extra step based on the predicted probability of the output. Using Bayesian Optimization methods, we can redefine the split criteria for two classes and test it on the official dev set to fit the data distribution. The optimal split confidence returned by Bayesian Optimizer is 90.056% with a F1-score of $\mathbf{61.423}\%$ on official dev set as illustrated in Figure 4.

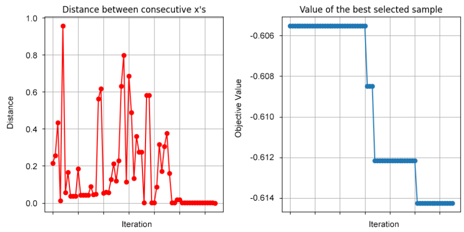


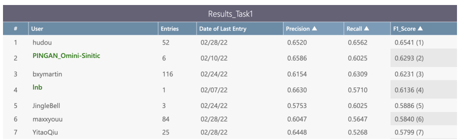
Figure 3: Bayesian optimization on split criteria

Our further analyses are based on the model that is trained on the official train set and evluated on official dev set.

3.3 Performance Analysis

3.3.1 Performance on Test Set

Model is trained with entire labelled dataset in final submission. Our final submission obtained 0.6154 in precision, 0.6309 in recall and **0.6231** in F1-score (**3rd Place** among all submissions in Post-Evaluation Section) on CodaLab.



Results_Task1

#	User	Entries	Date of Last Entry	Precision ▲	Recall ▲	F1_Score ▲
1	hudou	52	02/28/22	0.6520	0.6562	0.6541 (1)
2	PINGAN_Omini-Sinitic	6	02/10/22	0.6586	0.6025	0.6293 (2)
3	bxymartin	116	02/24/22	0.6154	0.6309	0.6231 (3)
4	lnb	1	02/07/22	0.6630	0.5710	0.6136 (4)
5	JingleBell	3	02/24/22	0.5753	0.6025	0.5886 (5)
6	maxxyouu	84	02/28/22	0.6047	0.5647	0.5840 (6)
7	YitaoQiu	25	02/28/22	0.6448	0.5268	0.5799 (7)

Figure 4: Model final performance on CodaLab

3.3.2 Patronising Level Analysis

Labelled with different level of patronising content, it is likely that samples at a higher PCL level contains more distinguishable features, which helps the model to detect. Performance statistics of PCL at different levels shown in Table 5 verifies our initial assumption. In a deeper analysis, we found that higher level contents are classified to more PCL categories which means they have more PCL features, resulting in better performance in these samples.

PCL Level	PCL Category Count							Accuracy
	1	2	3	4	5	Total	Avg Count	
Level 2	61	50	11	4	0	126	1.67	0.167
Level 3	113	158	71	23	4	369	2.04	0.607
Level 4	69	127	77	23	3	299	2.21	0.836

Table 5: Model performance and categorical statistics

3.3.3 Sequence Length Analysis

We divide input length into 4 categories and summarize model performance into Table 6. The model achieves its best performance when input length are within range of $[128, 256)$ while performs the worst when input length are longer than 512. This is mainly because when input length are longer than 512, our model will truncate the first 512 characters and thus may not gain sufficient information to predict such a long sequence. When input length are less than 128, it may not contain enough contexts for the model to make prediction. When input length are in $[128, 256)$, the model is able to take all characters into consideration and the length of samples guarantees sufficient context for the model to make the right predictions.

Length Interval	$[1, 128)$	$[128, 256)$	$[256 - 512)$	$[512, \infty)$
Precision	0.519	0.560	0.569	0.458
Recall	0.667	0.709	0.631	0.733
F1-score	0.583	0.755	0.599	0.564

Table 6: Model performance w.r.t. different length

3.3.4 Categorical Data Analysis

As seen in Table 7, our model performs well in non-English speaking regions like *Hong Kong*, *Malaysia* and *Kenya* while performs poorly in English speaking regions like *Australia*,*Great Britain* and *Singapore*. It is reasonable to infer that for native speakers they are more familiar with English and are able to express thoughts in plenty of ways which makes it hard to predict the real meaning. However, for non-native speakers they may only have certain forms of expressions so the model is able to identify the patterns easily. In terms of keywords, some keywords are strong indicators of PCL while others are more versatile and can be used in other non-condescending contexts, causing the model to predict false positives. For example, the word 'refugee' is often used in a political news to illustrate a objective phenomenon without being condescending to any specific group.

Country	au	bd	ca	gb	gh	hk	ie	in	gm	ke
F1	0.308	0.444	0.588	0.421	0.692	0.750	0.667	0.471	0.533	0.749
Total	98	103	118	127	84	93	112	104	104	113
Country	lk	my	ng	nz	ph	pk	sg	tz	us	za
F1	0.700	0.778	0.709	0.559	0.588	0.581	0.333	0.632	0.560	0.640
Total	85	116	108	101	105	109	95	94	114	110
Keyword	dis	hom	hop	imm	need	mig	poor	ref	vul	wom
F1	0.438	0.657	0.588	0.667	0.765	0.444	0.559	0.357	0.653	0.551
Total	194	212	217	218	226	206	190	188	209	233

Table 7: F1-score w.r.t. country and keyword

4 Conclusion

We proposed a DeBERTa based detection pipeline combining with multiple techniques which ranked 3rd on CodaLab. We also discovered that features including patronising levels, sequence lengths, countries and keywords have significant impact on model performance. To further improve our model, we can try training in full-precision mode, add label smoothing and use ensemble learning. It is also possible to use transfer learning and utilize the data in task 2 to boost performance.

References

Xinyu Bai, Yuze An, and Boyu Han. 2022. Github repository for dontpatronizeme task 1. `https://github.com/BXYMartin/BERT-Emotional_Damage`.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer.

Mark Davies. 2013. Corpus of news on the web (now): 3+ billion words from 20 countries, updated every day. *Retrieved January*, 25:2019.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

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

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.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

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. Huggingface's transformers: State-of-the-art natural language processing.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. Xlnet: Generalized autoregressive pretraining for language understanding.