NLP Coursework

Alicia Law Jiayun
Imperial College London
MSc Computing
(Software Engineering)
`ajl115@ic.ac.uk`

Mun Fai Chan
Imperial College London
MSc Artificial Intelligence
`mc821@ic.ac.uk`

Wei Jie Chua
Imperial College London
MSc Artificial Intelligence
`wc1021@ic.ac.uk`

Introduction

Patronising behaviour is the display of kind or helpful treatment with a sense of superiority. In this paper, we analyse the Don't Patronize Me! dataset [1] and developed a Sentiment Classifier that identifies patronising content which can be found on GitHub[1]. We explore various state-of-the-art transformer architectures and data processing methods. Finally, the report addresses several interesting model findings.

1 Analysis of Training Data

1.1 Class Label Analysis

The distribution of patronising data is severely imbalanced, where only 9.49% of the data is patronising. Moreover, longer sentences were more likely to be classified with higher levels of patronising content (see Table 1). This may be because detecting condescension is highly contextual, hence longer sentences make them more easily identifiable to the dataset's annotators.

Table 1: Class Label Analysis

Multi Class	Mean Text Length	%	Binary	%
0	263.3	81.48	0	90.51
1	284.23	9.05		
2	270.17	1.38	1	9.49
3	278.82	4.38		
4	306.47	3.74		

A words frequency analysis was also carried out. The words *need*, *family* and *poor* were the most frequently mentioned in the patronising dataset. A word cloud of the patronising and non-patronising dataset are included in Figures 4 and 5 of Appendix A.

1.2 Qualitative Assessment of Dataset

Patronising detection requires the NLP algorithm to not only understand content but intention. This is concerned with semantics and is highly context dependent. This is in contrast to purely disparaging language, which are easily detected through keywords containing negative connotations.

This is evident in our corpus, where some of the most frequently used words of the patronising dataset, such as *family*, *need* and *child*, are also most frequent in the non-patronising dataset. Hence, individual keywords are not a good indication of condescension, making this task difficult.

This necessitates NLP approaches that consider word contexts (e.g. transformers), but also requires some understanding of the context in which the statement was being made. For instance, Figure 1 shows that datasets from Ghana, Nigeria and Philippines are observed to have a higher level of condescension. This may suggest that certain cultures may favour the use of sentences that appear condescending to others.

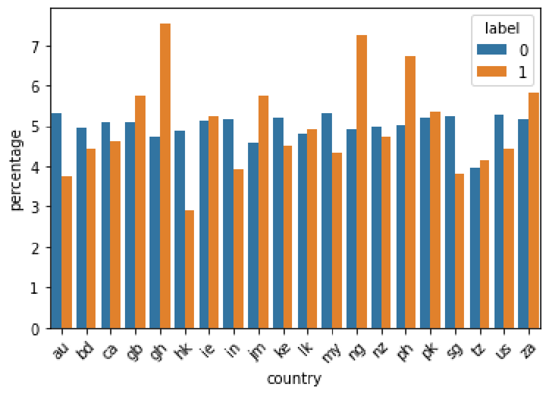


Figure 1: Condescending percentage per country

Moreover, determining if a statement is patro-

[1]https://github.com/alicialawjy/Patronizing-Language-Detection

nising is inherently subjective, making this task even more challenging. For instance, the sentence *"The charity has a number of success stories where people have been homeless and are now leading happy lives in employment"* is considered patronising by the annotators, presumably because it presupposes greater happiness for the employed. However, arguably, employment generally leads to higher living standards and thus happiness, which may be what the authors were alluding to (without taking on a position of superiority). This also points to the fact that condescension may not even be apparent to the speaker him/herself.

2 Model Implementation

The internal dataset was first split into train, evaluation and test sets in proportions of 70%, 15% and 15% respectively. F1 scores on the positive (condescending) class for the validation set are reported below.

2.1 Data Processing

We perform data processing on the training set to combat the class imbalance.

2.1.1 Data Sampling

We perform data sampling to attain a 35% proportion of condescending (Class 1) samples in the training set. Up-and-downsampling achieved the highest F1 score and was hence adopted.

Downsampling Minority Class (F1 = 0.51)
Decrease majority class proportion to 10%
Total train data size = 1300
Upsampling Majority Class (F1 = 0.50)
Increase minority class proportion by 100%
Total train data size = 10328
Combined Up-and-Downsampling (F1 = 0.52)
Increase minority class proportion by 100%
Decrease majority class proportionby 40%
Total train data size = 4055

2.1.2 Data Augmentation

To improve the quality of our upsampled data, data augmentation was explored. 3 methods were considered (see below), each augmenting the full set of minority class training data.

Paraphrasing (F1 = 0.61)
Paraphrasing was explored as it changes the positional encoding and hence tokenisation input to the transformer. This was implemented using the pegasus_paraphrase huggingface model.

Selective Synonym Replacement (F1 = 0.58)
This was implemented using nltk wordnet. Following an initial implementation which replaces all text with its synonyms, a poor quality of text replacement was noted. Hence, selective replacement was used instead. Stop words, numeric text, and uppercase words were not replaced. From the remaining bag of words, only alternate words were replaced to prevent drastic changes.

Paraphrasing & Synonym Replacement (F1 = 0.60)
Synonym replacement was applied onto the paraphrased text. While this method was thought to yield the best performance, it performed slightly poorer than the paraphrase-only dataset. This may be due to the limitations of synonym replacement, which replaces text without context.

Overall, data augmentation brought a significant improvement to model performance, yielding almost 10% increase in F1 scores. Hence, the final train set uses up-and-down sampling, with upsampling using paraphrased minority datasets.

2.2 Hyperparameter Tuning

2.2.1 Parameters

The following parameters were explored:

Pre-trained Models:
Bert, Roberta, Distilbert, Longformer, Twitter[2]
Model size: base, large
Tokenizer: cased, uncased
max_length: 32, 64, 128, 256, 512
(See Section 3)
Learning rate: 4e-4 to 4e-6
Epoch: 1 to 5
Optimiser: AdamW and AdaFactor
Scheduler:
constant schedule, constant schedule with warmup, linear schedule with warmup, cosine schedule with warmup, cosine with hard restarts schedule with warmup, polynomial decay schedule with warmup
Cross Entropy Loss Weights [1.0, x] where x range from: 1.0 to 2.5

[2]cardiffnlp/twitter-roberta-base-sentiment.

2.2.2 Findings and Final Model

We found that the choice of model and the epoch resulted in the most significant changes to the results. Results for hyperparameter tuning are reported in the Appendix B. The best hyperparameters were used in the model and they are:
- **Pre-trained Models**: RoBERTa-base
- **Learning rate**: 2e-5
- **Epoch**: 4
- **Optimiser**: AdamW
- **Scheduler**: Linear schedule with warmup
- **Loss weights**: None

F1 on the official testset: 0.5227[3]

3 Analysis

To what extent is the model better at predicting examples with a higher level of patronising content? As expected, there is a significant impact on the level of patronising content with the model's performance. The model performs strongest when the content is Level 4, identifying 88% of the cases. This performance diminishes to 67.61% for Level 3 and finally 25% at Level 2. Where patronising content is low and even human annotators have found it difficult to discern condescension, hence the model is expected to perform similarly.

How does the length of the input sequence impact the model performance? We varied the maximum input sequence length from 64, 128, 256 and 512[4]. We hypothesised that a longer input length can lead to better performance as there is more information. However, this is not true. Having a max length of 256 results in the worst F1 of 0.45 while a max length of 512 results in only a marginally better F1 (0.51) than with 64 and 128 (0.50). There may not be much differences because condescension may be more concerned with the general tone of the statement and hence a short text will suffice. Furthermore, it may be that attention weights are too spread out for longer sequences and hence performance does not improve much.

To what extent does the categorical data provided influence the model predictions? Keywords of high frequency in the condescending dataset such as *homeless*, *in-need* and *poor-families* were noted to over-predict condescending labels (see Figure 2). This may indicate the influence of keyword frequency during model training. The same pattern is noted for countries as shown in Appendix C Figure 10, where countries with higher amounts of condescending dataset yielded more over-predictions, and this error rate decreased for countries with smaller condescending datasets. In light of this, we can perform further upsampling on these keywords/countries to balance the dataset and remove any biases to improve performance.

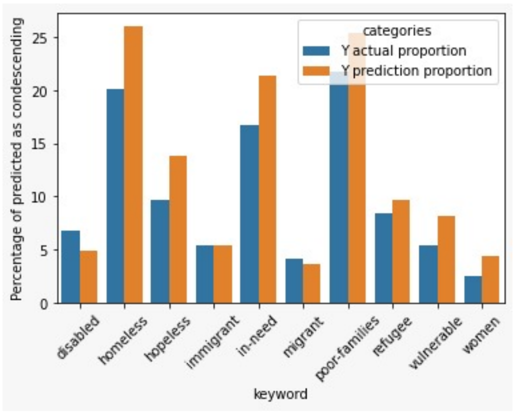


Figure 2: Predictions for keywords

4 Conclusion

In conclusion, our project investigated different means to improve model performance. This includes data sampling, data augmentation, and hyperparameter tuning on multiple variables. We found that the Roberta-Base model is still the best performing model and attained the most significant improvements through combined upsampling (paraphrasing) and downsampling. Our best model attained an F1 score of 0.62 on our internal test set and 0.5227 on the competition test set.

We notice in Figure 9 that texts with a lower degree of patronising content have a lower recall. Hence, an extension can be to perform more targeted up-sampling on these categories. Other extensions to the project can involve different truncation methods (e.g. truncating in the middle) and experimenting with other models. For instance, we experimented with the Longformer model to deal with the long sequence length of this corpus as well as a pre-trained sentiment analysis model, but both were not as effective as our best model. Further tuning of these models may be required.

[3]See Figure 11.

[4]Full results are presented in Appendix Table 2.

References

[1] Carla Pérez-Almendros, Luis Espinosa-Anke, and Steven Schockaert. *Don't Patronize Me! An Annotated Dataset with Patronizing and Condescending Language towards Vulnerable Communities*. 2020. arXiv: 2011.08320.

Appendix A. Analysis of Training Data

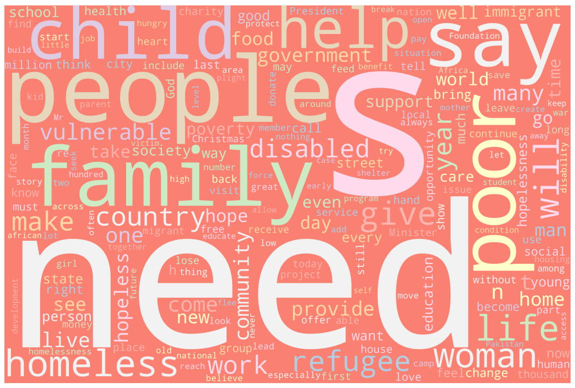


Figure 3: Part 1.1 Wordcloud: Wordcloud of condescending samples

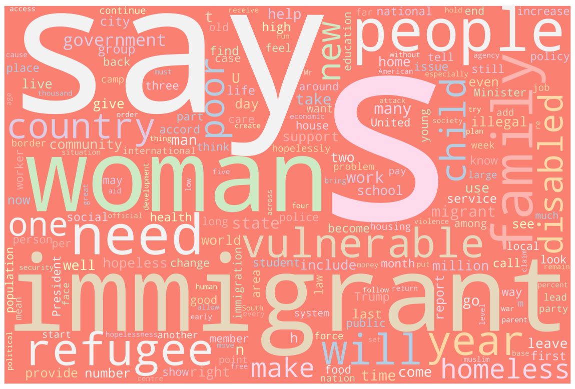


Figure 4: Part 1.1 Wordcloud: Wordcloud of overall samples

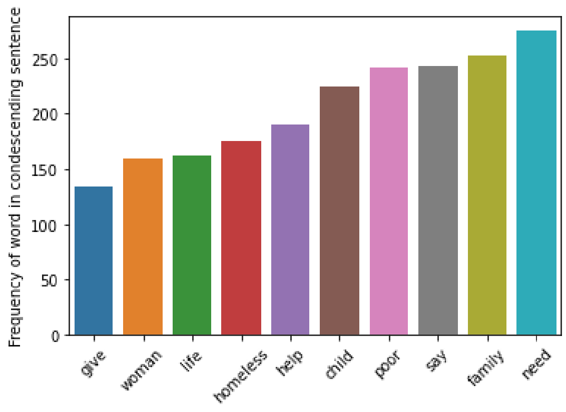


Figure 5: Part 1.2: Most frequent words of condescending samples

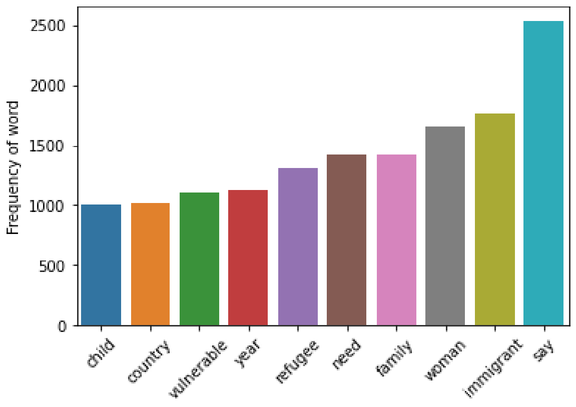


Figure 6: Part 1.2: Most frequent words of overall samples

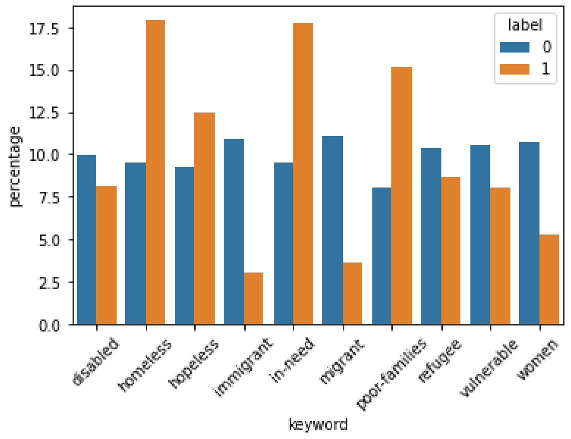


Figure 7: Part 1.2 Graph: Condescending percentage per keyword

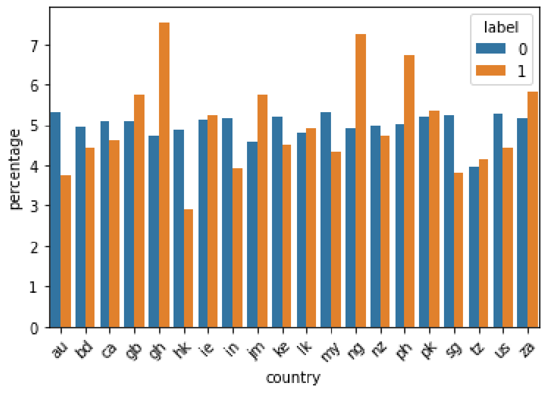


Figure 8: Part 1.2 Graph: Condescending percentage per country

Appendix B. Hyperparameter Tuning results

Max Sequence Length	Positive F1 Score
64	0.5
128	0.5
256	0.45
512	0.51

Table 2: Effect of maximum sequence length on positive F1 score. In our best model, we used a max length of 128 as a max length of 512 leads to a marginal improvement in performance but signficantly longer computational time.

Scheduler	Positive F1
Constant Schedule	0.44
Constant Schedule w/ Warmup	0.47
Linear Schedule w/ Warmup	**0.51**
Cosine Schedule w/ Warmup	0.50
Polynomial Decay	0.37

Table 3: Effect of scheduler on Positive F1 score. Hyperparameter tuning was performed on a Roberta-Base model for 1 epoch.

Weight of minority class	Positive F1 Score
1	0.44
1.5	0.52
2	0.47
2.5	0.46

Table 4: Effect of weighting on positive F1 score. Hyperparameter tuning was performed on a Roberta-Base model for 1 epoch.

Models	Positive F1 Score
bert-cased	**0.52**
bert-uncased	0.50
distilbert-cased	0.49
distilbert-uncased	0.48
longformer	0.48
roberta-base	0.48
roberta-large	0.39
sentiment-roberta-large-english	0.43
twitter roberta base	0.51

Table 5: Effect of models on positive F1 score. Hyperparameter tuning was performed on a Roberta-Base model for 1 epoch.

Appendix C. Analysis

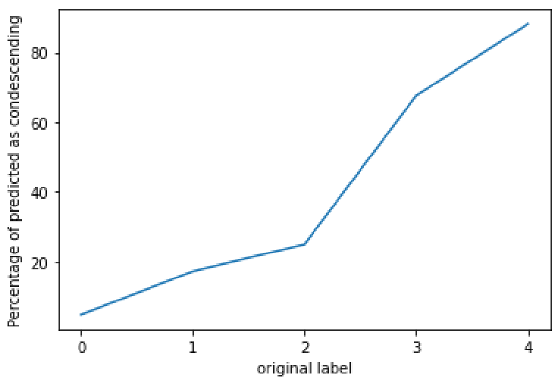


Figure 9: Part 3.1 Graph

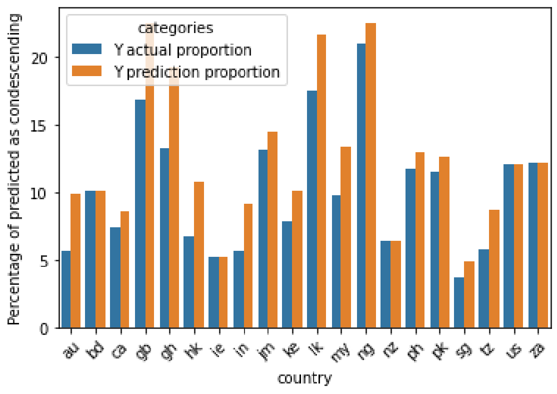
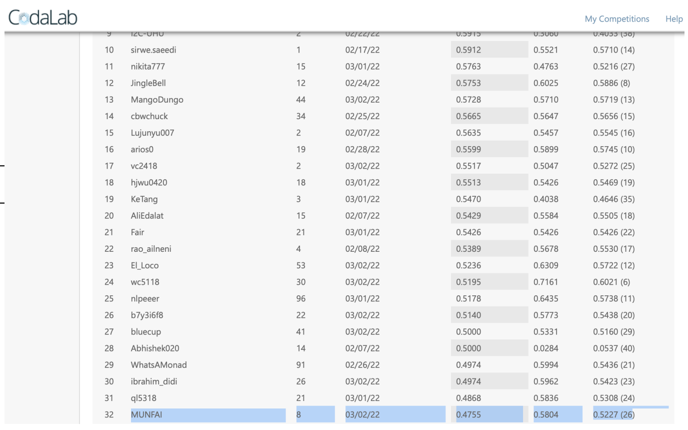


Figure 10: Predictions for each country

Appendix D. CodaLab submission



9	I2C-UHU	2	02/22/22	0.5915	0.3060	0.4033 (38)
10	sirwe.saeedi	1	02/17/22	0.5912	0.5521	0.5710 (14)
11	nikita777	15	03/01/22	0.5763	0.4763	0.5216 (27)
12	JingleBell	12	02/24/22	0.5753	0.6025	0.5886 (8)
13	MangoDungo	44	03/02/22	0.5728	0.5710	0.5719 (13)
14	cbwchuck	34	02/25/22	0.5665	0.5647	0.5656 (15)
15	Lujunyu007	2	02/07/22	0.5635	0.5457	0.5545 (16)
16	arios0	19	02/28/22	0.5599	0.5899	0.5745 (10)
17	vc2418	2	03/02/22	0.5517	0.5047	0.5272 (25)
18	hjwu0420	18	03/01/22	0.5513	0.5426	0.5469 (19)
19	KeTang	3	03/01/22	0.5470	0.4038	0.4646 (35)
20	AliEdalat	15	02/07/22	0.5429	0.5584	0.5505 (18)
21	Fair	21	03/01/22	0.5426	0.5426	0.5426 (22)
22	rao_ailneni	4	02/08/22	0.5389	0.5678	0.5530 (17)
23	El_Loco	53	03/02/22	0.5236	0.6309	0.5722 (12)
24	wc5118	30	03/02/22	0.5195	0.7161	0.6021 (6)
25	nlpeeer	96	03/01/22	0.5178	0.6435	0.5738 (11)
26	b7y3i6f8	22	03/02/22	0.5140	0.5773	0.5438 (20)
27	bluecup	41	03/02/22	0.5000	0.5331	0.5160 (29)
28	Abhishek020	14	02/07/22	0.5000	0.0284	0.0537 (40)
29	WhatsAMonad	91	02/26/22	0.4974	0.5994	0.5436 (21)
30	ibrahim_didi	26	03/02/22	0.4974	0.5962	0.5423 (23)
31	ql5318	21	03/01/22	0.4868	0.5836	0.5308 (24)
32	MUNFAI	8	03/02/22	0.4755	0.5804	0.5227 (26)

Figure 11: CodaLab submission (F1 = 0.5227)

Appendix E. GitHub Repo Link

https://github.com/alicialawjy/Patronizing-Language-Detection