Don’t Patronize Me! A NLP study of Patronizing and Condescending Media

Bryan O’Donohoe[#1], Conor Cosgrove[#2], Dario Alvarez[#3], Marco Baglio[#4]

[#]*School of Computing, National College of Ireland*
Mayor Street Lower, International Financial Services Centre, Dublin, Ireland

{[#1]x20212828, [#2]x20210396, [#3]x20195940, [#4]x20199520}@student.ncirl.ie

***Abstract*—This paper describes the project undertaken which is based on the SemEval 2022 Task 4: Patronizing and Condescending Language Detection. The task establishes itself from the paper *Don’t Patronize me! An annotated Dataset with Patronizing and Condescending Language Towards Vulnerable Communities* which develops a natural language processing model to identify such language in text. This paper will discuss the approach taken to apply our own deep learning models to the given dataset and evaluate the models’ performance in terms of the implementation of the task. Based on the fact patronizing and condescending language (PCL) can be subtle in nature and often difficult to discern, it provides a challenge in machine learning due to the struggle in recognizing PCL itself.**

Keywords—patronizing, condescending, machine learning, model, analysis, validation, classification

I. Introduction

The report documents the approach taken as part of the Data Mining and Machine Learning II module project for the PGDDA programme. The project is based on SemEval 2022 Task 4: Patronizing and Condescending Language Detection. This paper investigates the presence of PCL within extracts of text published in the media. Patronizing is to treat in a way that attempts to be helpful or kind but in such a way that portrays the feeling of superiority whereas condescending is to show an attitude of patronising superiority. This PCL identification task differs from similar NLP problems such as hate speech identification in the fact that an entity can often be patronising or condescending unbeknown to themselves and genuinely not mean any harm or discontent. This provides a unique challenge in tackling this particular NLP problem as there is an element of subjectivity to the task. A reporter in the media may be authentically attempting to raise awareness of a certain societal issue but may be inadvertently directing unwanted attention, from the vulnerable community’s perspective in a condescending way. Additionally, by identifying PCL and highlighting the general issue of PCL, this can itself drive unpremeditated discrimination against recipients of PCL. As PCL in NLP research is still a relatively emerging topic, this project provided an opportunity to delve into uncharted territory and produce deep learning models that can be compared and evaluated against other solutions in the community.

The data provided as part of the SemEval task contains paragraphs extracted from news articles where there are various vulnerable communities such as homeless people, immigrant and poor families mentioned. For the purpose of this project, subtask 1 was adopted for the research question which was to perform a binary classification to predict if each given paragraph contains any form of PCL.

II. Related Work

PCL is a topic that has been analysed in various papers and similarly, other forms of offensive language references were researched as part of this project. This section of the report discusses these papers that perform similar tasks as identifying PCL or offensive language in text. The papers researched as part of related work are more focused on the harmful aspects such as hate speech, discrimination, and aggressive text.

In [1], the authors describe the detection of hate speech against immigrants and women in Spanish and English messages extracted from Twitter. Social media has in recent years become more scrutinised for the rise in online hate speech on various online platforms. Due to the exponential rise in user generated content online, the authors of [1] discuss the growing need for the timely detection of hate speech in user generated content to fight against misogyny and xenophobia. One positive aspect of the paper is it acknowledges the fact that the variety of targets (immigrants and women) provides a unique comparative setting, which can be helpful in the integration of hate speech detection tools in a wider set of applications. A limitation of [1] was when an error analysis was performed, and it was found that some instances of text were incorrectly labelled in both languages, skewed towards false positives.

Another paper written on a similar topic is [2] which also looks at hate and prejudice against immigrants on social media. A positive aspect of this paper is how the authors intend on applying the findings of their analysis. The NLP analysis was integrated with data and networks analytics and visualisation approaches and shared with local operators in geographical locations for a more integrated communal approach to create awareness of the online hate and prejudice problem. One limitation of [2] is the reliance on local domains to perform their own application and evaluation of the project, which is going to be subjective to each individual region and difficult to benchmark.

The authors of [3] also look at the prevalence of aggressive behaviour online when analysing Mexican Spanish tweets and attempts to identify the residence and occupation of the Twitter user sending out the tweets. Only the content in

relation to identifying aggressive text in tweets was acknowledged as this was seen as relevant to the task outlined in this project, therefore the section on author profiling was ignored. The authors of [3] used the results of a workshop carried out to present different approaches for the detection of abusive language in social media [4] which reduced the level of subjectiveness to the labelling of aggressive text. The performance measures used in [3] were mainly the F1 score which aligned with our approach as part of this project. It was concluded that detecting aggressiveness in tweets is a very complex task as only three systems outperformed the baselines as part of the research.

In another paper researching hate speech detection and the problem of offensive language, [5] discusses how separating hate speech from other offensive language is difficult. The authors found that racist and homophobic tweets on social media are more likely to be classified as hate speech, but sexist tweets are more likely to be classified as offensive. The difficulty with the research in [5] is differentiating hate speech and offensive language. This relates to the issues identified in this project with regard to identifying patronizing or condescending language and the subjective nature of this differentiation. The authors of [5] use basic machine learning methods such as logistic regression and decision trees and the results show 40% of hate speech is misclassified which amplifies the complexity of such a problem and the need for more advanced deep learning methodologies.

The authors of [6] go one step further when developing a model to detect hate speech and perform analysis on the targeted recipients of the online hate speech. The paper outlines that although the major social media companies such as Twitter focus on specific manifestations of the problem such as racism, they do not put any focus on the underlying issues related to online hate speech and why it happens in the first instance. [6] takes a high-level approach of first identifying the hate speech and attempt to unveil patterns to provide a better understanding of the issue but offer direction for preventative and detection measures. One limitation of [6] was the fact that their chosen methodology was not designed to capture online hate speech, but their aim was to build a dataset that allowed the authors to identify targets of online hate. They used a very manual approach whereby the authors randomly selected 100 social media posts and manually verified if these should be classed as hate speech. This type of manual approach would not be feasible in a larger scale project.

A paper that looks at a similar topic is misogyny identification where it analyses Spanish and English tweets and performs a binary classification of the data [7]. The authors used a combination of evaluation techniques which gave a comprehensive evaluation of the performance of the model. The authors used a combination of model accuracy and a macro average which is a technique used to evaluate the models as part of this research project. The authors of [7] solidify the fact that although identification of misogyny can be relatively easily identified, the target classification of monogynist behaviour is a more difficult problem.

Papers [8] and [9] describe convolution neural networks (CNN) for sentence classification, the author of [8] implemented 4 different convolution neural network variant models: CNN-rand, which is the baseline model, CNN-static, CNN-non-static and CNN-multichannel. Dropout has been implemented on the penultimate layer for regularization as the dataset is very imbalanced, dropout added 2-4% relative performance.

All models are studied in multiple benchmarks: detecting sentiment of movie reviews and of customer reviews of various items, classifying a sentence as being subjective or objective and classifying a question into six question types (if the question is about a person, location, etc.). Only CNN-rand does not perform well on its own, the rest have very good results, even simple model CNN-static, which has only one layer of convolution.

In the paper [9], the authors describe a Dynamic Convolutional Neural Network (DCNN) which uses a dynamic k-max pooling operator. The DCNN model was implemented in four different experiments: small scale binary and multi-class sentiment prediction, classification of a question into six question types and Twitter sentiment prediction. The DCNN model achieved high performance on question and sentiment classification without requiring external variables, significantly outperforming the other neural and non-neural models in the sentiment predictions, and it was one of the three best models in the six questions prediction.

Alternatively, papers [10] and [11] implemented Recurrent Neural Networks (RNNs) for text classification, Paper [10] describes a long short-term memory (LSTM) RNN model, the authors found that LSTM outperforms previous RNNs on context-free language benchmarks. LSTM variants were also the first RNN to learn a simple context-sensitive language.

The authors of the paper [11] focused on the sentiment classification of four datasets from IMDB and Yelp. This paper tried to identify the overall sentiment polarity from 1 to 5 stars on the Yelp review sites and the sentiment label (from ten different classes) in the IMDB datasets. The paper studied the LSTM KNN model, and when evaluating the results compared to CNN and other Support Vector Machine (SVM) models, the LSTM model shows a stronger performance, similar to CNN. The report also found that the traditional RNN model was extremely weak in modelling document composition, but when neural gates were included, it boosted the performance.

Another research paper [12] combined convolution and recurrent layers for a RNN and CNN hybrid model which efficiently identify document classification tasks. The paper then evaluated the model in eight different large datasets from Amazon reviews, Yahoo answers, Yelp, DBPedia, Sogou news and AG's news. The sizes of the dataset range from 200,000 to 4 million documents, and include sentimental analysis, news categorization, ontology classification and question type classification.

The results are then compared with only convolution models. The hybrid model showed comparable performances for all the eight datasets with significantly less convolutional layers, results also showed that accuracy does not always increase with the number of convolutional layers and decrease if more are added to the model. Other results of the hybrid model were that it performed better when the number of classes was large, training size was small, and the number of convolutional layers were two or three.

The last research paper [13] presented a classification problem using shallow machine learning technique fast text classifier 'fastTest', and then compared the results with convolution neural network techniques. The paper evaluated

the model in eight datasets which are the same as [12]. The results indicate that the accuracy of 'fastText' using 10 hidden units and for 5 epochs was slightly better than CNN and character-based CNN, and a little bit worse than very deep CNN.

The authors of the paper noted that although deep learning networks have in theory much higher representational power than shallow ones, 'fastText' obtains a similar performance to deep learning techniques, while being much faster in sentiment classification. 'fastTest' model takes less than a minute to train the evaluated datasets, while deep learning techniques take hours per epoch.

III. Data Mining & Methodology

A. *Data Selection & Exploration*

In today's world, there are huge volumes of data being generated on a daily basis and careful consideration must be taken to manage this data and to extract the business value from the same. Raw data can often be useless, a greater emphasis must be placed on interpretation and insights derived from data to reap the benefits of analysis such as what has been completed as part of this project. Knowledge discovery in databases (KDD) is an approach that can assist with discovering the usefulness of a given dataset. Figure 1 outlines the steps taken using the KDD approach[1].

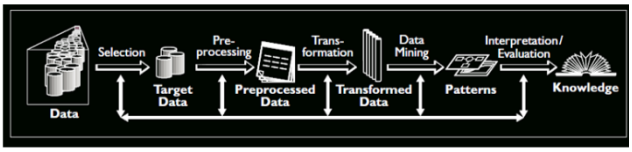

Fig. 1. Steps in KDD Methodology

The data for this project was sourced from the SemEval 2022 Task 4: *Patronizing and Condescending Language Detection* group and contains a collection of paragraphs extracted from media publications, mentioning vulnerable communities in 20 different countries. The data contains around 4,000 paragraphs which have been manually annotated and labelled with PCL annotation. For the purpose of building the machine learning models, the data was split into train and test, 80% and 20% respectively. Exploratory analysis was performed to gain an understanding of the format of the data and how we would proceed with the task. Figure 2 outlines the number of documents versus the number of characters in each paragraph.

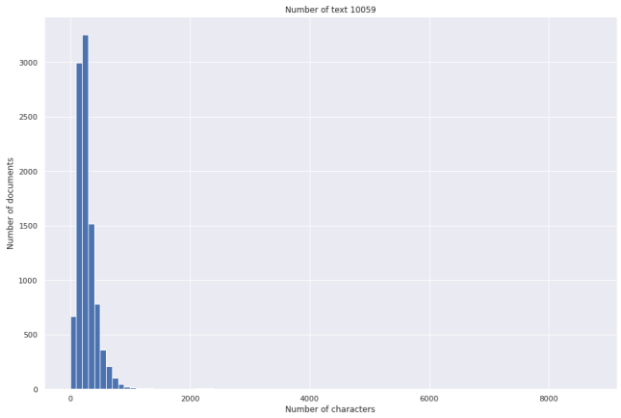

Fig. 2. Histogram of characters per paragraph

We can see from figure 3 and 4, the most common words that are in the data. The transformation section below outlines the steps taken to process the raw data before proceeding with the model building.

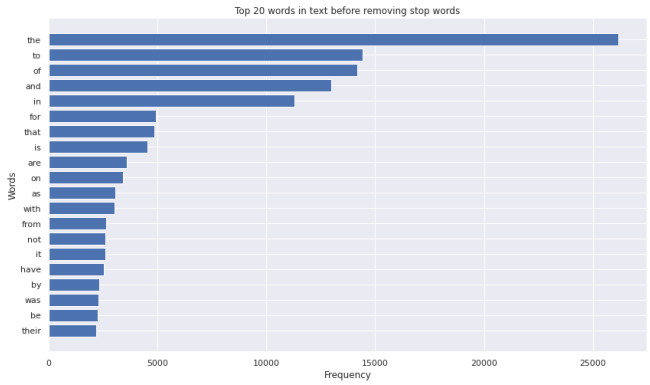

Fig. 3. Histogram of top words in the data

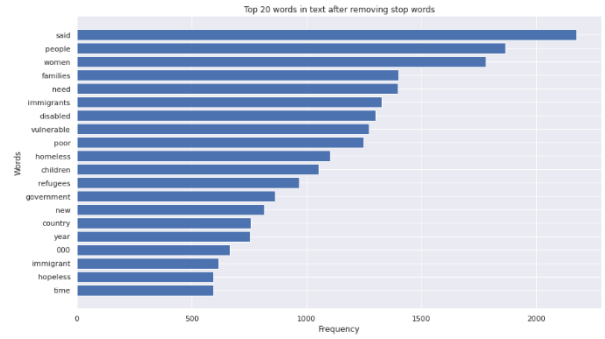

Fig. 4. Histogram of top words in the data (excluding stopwords)

The same approach was taken as of the research paper upon this project is based [14], where the authors selected 10 key words that were identified as potential groups vulnerable to PCL. These words included disabled, homeless, immigrant, in need, migrant, poor families, refugee, vulnerable and women. To reduce the impact of subjectiveness in identifying PCL, three annotators annotated the dataset – two identifying what they thought was PCL and the third acting as a reviewer to confirm if they thought the text did in fact include PCL.

1 https://www.ceine.cl/the-kdd-process-for-extracting-useful-knowledge-from-volumes-of-data/

B. Pre-Processing

The first steps of the pre-processing were removing the upper cases, URL and HTML text and emoji. Also, abbreviations were removed and replaced by their corresponding word, such as "not" instead of " n't".

A dataset was included in order to identify the words which are the most common words in the English language, we will keep them in our model as they are important in order to identify if text contains PCL.

A pre-trained layer has been added to the model. Many text processing applications nowadays rely on pre-trained word representations estimated from large text corporations such as Wikipedia and Common Crawl, these pre-trained representations provide distributional information about words which typically improve the generalization of models learned on limited amounts of data [15]. An important aspect of their training is to efficiently capture as much statistical information as possible from rich sources of data [15]. The pre-trained layer in the model has been created using a fastText pre-trained word vector (FastText Common Crawl bin). The FastText Common Crawl contains 2-million-word vectors trained on Common Crawl, and around 600 billion tokens.

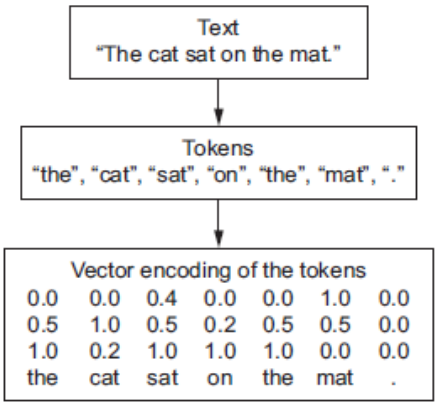

Fig. 5. Histogram of top words in the data (excluding stopwords)

C. Transformation

1) Tokenization

The different units into which text can be broken down (words, characters, or n-grams) are called tokens, and breaking text into suck tokens is called tokenization. Text-vectorization processes consist of applying tokenization and then associating numeric vectors to the created tokens, these vectors will be led into the deep neural network. There are many ways to associate a vector with a token, in our model token embedding, also called word embedding has been implemented. Word embeddings are low-dimensional and dense vectors which are learnt from data, in contrast to one-hot encoding of tokens, another way to associate a vector with a token.

There are two ways to obtain word embeddings, loading pre-trained word embeddings and learning word embeddings jointly with the main task you care about. We have loaded word embeddings into the model which were precomputed using a different machine learning task, the pre-trained word vectors dataset (FastText Common Crawl bin), which was previously explained. This pre-trained word embedding is used as there is not enough data available to learn powerful features on their own, but it is expected that many features are fairly generic.

2) Regularization

Overfitting occurs when the model learns the detail and noise in the training dataset to the extent that it negatively impacts the performance of the model on the new dataset, as the model learns specific to the training data and does not generalize to data outside the training dataset. When overfitting occurs, the training accuracy loss separates from validation accuracy loss, it happens in every machine learning task, and we have tried to reduce the overfitting following some overfitting techniques:

- Keeping the model simple with a small number of layers.
- Adding dropout layers (Dropout and SpatialDropout1D), one of the most effective regularization techniques for neural networks, consists of randomly dropping out (converting to zero) a number of output features of the layer during the training. Dropout rate is the fraction of the features that are converted to zero, and we have tried different values between 0.2 to 0.3 in the model.

3) Imbalanced dataset

Imbalanced datasets are datasets where one or more classes have very low proportions in the training data as compared to other classes, this results in models that have poor predictive performance. Our model studies a binary classification problem where the dataset is very imbalanced. In order to deal with imbalance datasets, we have implemented some techniques:

- Added weight regularization to our model, which makes the distribution of weight values more regular. In the model, the weight value is 0.5582 for class: 0, and the weight value is 4.7966 for class: 1.
- Reduced the number of the majority class(class:0) in the training dataset, in our model we have removed 10% of the majority class in the training dataset.

4) Embedding Layers

The embedding layer is known as a dictionary that maps integer indices to dense vectors, it takes as input a 2D tensor of integers, of shape (sample, sequence length), where each entry is a sequence of integers, and returns a 3D floating tensor which shape is (ample, sequence length, embedding_dimsionality), this 3D vector can then be processed by RNN or 1D convolution layer. The embedding layer is set to no trainable, as we only want the weights from the dense and other layers to be trained, not the weights in the Embedding layer.

D. Data Mining

The selection of the optimal model learning is strictly related to the nature of the research question. In this case, we have a problem of text classification, which requires the application of algorithms for sequence processing. The two

fundamental deep-learning algorithms are recurrent neural networks (RNN) and 1D convnet (CNN).

The RNN is a network with the capacity to retain memory of previous states: each step processes the current state and stores the information that has been accumulated. The library Keras offers the SimpleRNN recurrent layer, which is the basic layer applicable to RNN networks. The SimpleRNN does not perform well with long sequences due to the vanishing gradient problem. The LSTM and GRU are the layers that offer a solution to this problem. LSTS has more representation power whereas the GRU requires less computational power. The RNNs are order dependent, it performs effectively where the order is meaningful. In particular, the layers process the sequence in the chronological order. The bi-directional layers analyse the sequence chronologically and anti-chronologically, merging the final representation with the possibility to detect patterns overlooked by the other layers.

The CNN provides a fast alternative to RNNs for simple tasks. The convolution treats the time as a spatial dimension, similarly to a dimension of an image to be processed. The convolution layers can learn local patterns in a sequence and recognize them in a different location; de facto, the CNNs are translation invariant.

In this study, we have tested combinations of the aforementioned layers, with the only exception of a simple and cheap model, based on dense layers. The next subsections show a description of each model involved in the analysis. All models were sequential and contained the following.

- Initial embedding layer
- Root mean squared propagation (RMSProp)
- BinaryCrossentropy loss function
- Validation split of 80/20
- Early stopping monitoring the validation loss
- Final dense layer with sigmoid activation

All models were also tuned using a "RandomizedSearchCV". The number of combinations checked varied between models as there was a wide range of fitting times for each model. This led to the number of combinations ranging from as little as 3 to up to 60.

1) Dense Model

The dense model is the simple model and contains the following layers:

- GlobalAveragePooling1D layer
- Dense layer with "relu" activation
- Dense layer with "relu" activation

2) CNN Model

The CNN model is based on convolution layers only and contains the following:

- Five stacks of 1D Convolution layer with relu activation. Each stack includes a dropout layer of 0.2 and MaxPooling1D or GlobalMaxPooling1D as the last layer
- Dense layer with "relu" activation

3) Bi-directional GRU Model

The Bi-Directional GRU model contains the following layers.

- Bi-directional GRU layer with dropout of 0.32 and recurrent dropout of 0.32.
- Two stacks of dense layers with activation "relu" and dropout layers of 0.25

4) RCNN Model

The RCNN model contains the following layers:

- Bi-Directional LSTM layer with dropout and recurrent dropout of 0.25. This layer returns the entire sequence instead of the last step. The activation used is "relu".
- 1D convolutional layer with "relu" activation, l1 and l2 regularizer values of 0.001. A GlobalMaxPooling1D layer is included after this layer.
- Dense layer with activation "relu"

5) CNN-LSTM Model

The convolutional lstm model contains the following layers:

- 1D convolutional layer with "relu" activation and an l2 regularizer of 0.01
- Dropout layer of 0.4
- 1D maximum pooling layer
- LSTM layer with l1 and l2 regularizer values of 0.1 and 0.01 respectively

6) RNN-LSTM Model

The rnn-lstm model contains the following layers:

- LSTM layer with l2 regularizer value of 0.001
- Dropout layer of 0.3

7) Bi-directional LSTM Model

The bi-directional lstm model contains the following layers:

- Bi-directional LSTM layer
- Dropout layer of 0.5

IV. Evaluation

When fitting all models, plots were created to evaluate overfitting. The accuracy and loss for the train and validation sets were plotted for each epoch of the model fitting. The aim being to select the point where the validation and training accuracy are at their local maximum.

Using the test set, each model calculated predicted values. These values were then aggregated in a confusion matrix and precision, recall and weighted-f1 scores were calculated. Precision is given by the following equation.

$$precision = \frac{TP}{TP + FP}$$

where,
TP = True positives
FP = False positives

Recall is given by the following equation.

$$recall = \frac{TP}{TP + FN}$$

where,
TP = True positives
FN = False negatives

F1 is given by the following equation.

$$f1 = \frac{2\ x\ precision\ x\ recall}{precision + recall}$$

Weighted-f1 is given by the following equation;

$$weighted - f1 = \frac{n_a f1_a + n_b f1_b}{n}$$

where,
n_j = Number of samples of class j
$f1_j$ = f1 score of class j

V. Results

Table 1 below illustrates the weighted precision, recall and f1 scores for the validation data using the 7 tuned candidate models.

TABLE 1
Results for the problem of detecting PCL

	P	R	F1
Dense	0.88	0.91	0.877
CNN	0.89	0.91	0.871
BI-GRU	0.88	0.91	0.883
RCNN	0.89	0.91	0.896
CNN-LSTM	0.82	0.91	0.861
RNN-LSTM	0.89	0.91	0.881
BI-LSTM	0.89	0.90	0.894

The best performing model is the RCNN with the f1-score of 0.896. Fig. 6 below illustrates the training and validation loss of the RCNN model. This shows that there is a similar trend between the training and validation loss until epoch 5, with a stabilisation until epoch 15 and overfitting beginning to occur from 16 onwards.

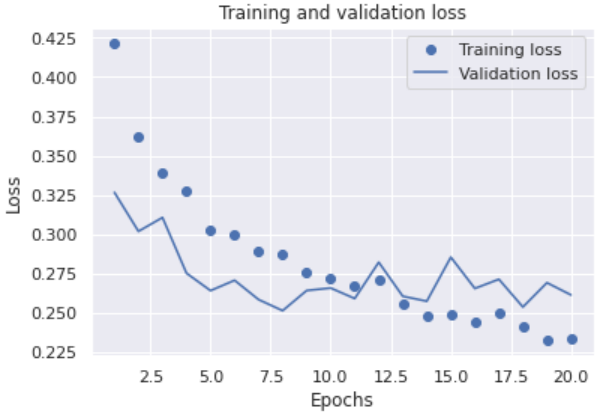

Fig. 6. Training and validation loss for RCNN model

Fig. 7 below illustrates the training and validation accuracy of the RCNN model. Until epoch 8, there is unremarked improvement in both training and validation accuracy. The training accuracy then begins to gradually increase with some noise evident on the validation accuracy.

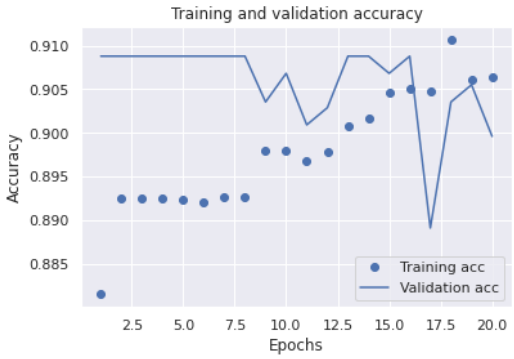

Fig. 7. Training and validation accuracy for RCNN model

Fig. 8 below illustrates the confusion matrix for the RCNN model.

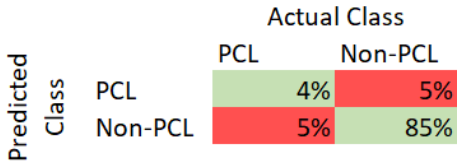

Fig. 8. Confusion Matrix for RCNN model

Table 2 below illustrates the weighted-f1 scores for each of the vulnerable groups in the test dataset using the RCNN model.

TABLE 2
Weighted-f1 of each vulnerable group

Vulnerable Group	Weighted-f1
Disabled	88%
Homeless	84%
Hopeless	84%
Immigrant	0%
In-need	81%
Migrant	0%
Poor-families	84%
Refugee	95%
Vulnerable	91%
Women	95%

Table 3 below illustrates the weighted-f1 scores for each of the continents in the test dataset using the RCNN model.

TABLE 3
Weighted-f1 of each continent

Continent	Weighted-f1
Africa	87%
Americas	90%
Asia	90%
Europe	89%
Oceania	94%

VI. Discussion

The 7 models proposed in this paper provide similar values in terms of precision and recall. The f1-score allows us to classify the models by considering the effect of both the metrics. The worst models are CNN-LSTM and CNN, with the lowest f1-scores as well as bad training/validation loss. Several combinations of convolutional layers and parameters have been tested but the convolutional layer does not perform well for this problem.

The Bi-LSTM and RCNN are the best models with f1-score almost equivalent. The importance of sequence order appears to be relevant and the bidirectionality provides additional information extracted by these two models. The RNN-LSTM and Bi-GRU are very close to the best performing models. One reason they are not the best is the usage of the GRU layer performing worse than LSTM and the usage of uni-directionality .

Finally, the deep model provides an interesting compromise between speed and effectiveness. Despite the model being based only on cheap layers, the results obtained are very good when the training/validation loss is evaluated, and the results compared to the other models.

Fig. 8 illustrates the confusion matrix of the predicted values on the test data for the best performing model. Due to the nature of this problem, there is a challenge in predicting the positive PCL values. This is reflected in that most of the models considered here have an F1 score of around 40% for the PCL response class.

Looking at the values in Table 2, we can see that the optimal model performs poorly on the “Immigrant” and “Migrant” categories on the test data, unable to predict any of the PCL instances for those groups.

It is noted that the model performs well when the paragraph relates to women. Looking at some of the paragraphs correctly identified, it seems that mentions of the family are common amongst them.

As there are 20 countries in the dataset, continent level groups were created for analysis. Table 3 seems to indicate that the model performs the worst on the paragraphs from African countries. Diving into this further it appears that Ghana is the worst performing country in the test set which is dragging down the African average.

VII. Conclusion & future Work

In this work, we have focused on creating a model able to identify PCL, and we operated with the Don’t Patronize Me! dataset, whose aim is to identify and classify PCL towards vulnerable people. In order to get a better performance, we have used a pre-trained layer (FastText Common Crawl), we had also to deal with an imbalanced dataset, and for evaluating the level of success of our model we have used precision-recall and f1. Our findings show that the RCNN model shows the best results, but also a simple dense model indicates a very good performance using the layer GlobalAveragePooling1D.

As there was a limit on the time cost available to fit the optimum model, future work could include a stack of bi-directional recurrent layers if there was not a time consideration for the study. This should better model the sequence of the words in the sentence.

There were two pieces of information provided in the dataset that were not included in the exercise, vulnerable group and country the paragraph was taken from. Future work could look to include these features to avoid underfitting. It is also important to be mindful that in the event of these models being implemented in the context of screening articles on social media, that each vulnerable group is equally flagged, and that bias is not introduced.

In future, there should be a conscious attempt to make the raw dataset larger with reduced imbalance. Effectiveness of the models could be improved by training the embedding layer on the task itself. Due to the size of the dataset, the embedding layer here was pre trained as opposed to fitted on the task itself. For a minimisation of bias, the use of a greater number of individuals in assessing PCL in the paragraphs would be beneficial.

VIII. References

[1] Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In Proceedings of the 13th International Workshop on Semantic Evaluation

[2] Cristina Bosco, Patti Viviana, Marcello Bogetti, Michelangelo Conoscenti, Giancarlo Ruffo, Rossano Schifanella, and Marco Stranisci. 2017. Tools and Resources for Detecting Hate and Prejudice Against Immigrants in Social Media. In Proceedings of First Symposium on Social Interactions in Complex Intelligent Systems (SICIS), AISB Convention 2017, AI and Society.

[3] Miguel Angel ´ Alvarez Carmona, Estefan ´ ´ıa Guzman- ´ Falcon, Manuel Montes-y-G ´ omez, Hugo Jair Es- ´ calante, Luis Villasenor Pineda, Ver ˜ onica Reyes- ´ Meza, and Antonio Rico Sulayes. 2018. Overview of MEX-A3T at IberEval 2018: Authorship and Aggressiveness Analysis in Mexican Spanish Tweets. In Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018).

[4] M.S. Forbes, Centre for Culture and Media Studies, Workshop on Abusive Language Online, 2017, Vancouver Canada. Available at: https://aclanthology.org/W17-3000.pdf

[5] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. CoRR.

[6] Silva, L. A.; Mondal, M.; Correa, D.; Benevenuto, F.; and Weber, I. 2016. Analyzing the targets of hate in online social media. In ICWSM, 687–690

[7] Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018b. Overview of the Task on Automatic Misogyny Identification at IberEval 2018. In Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018). CEUR-WS.org.

[8] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

[9] N. Kalchbrenner, E. Grefenstette, P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In Proceedings of ACL 2014.

[10] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In EMNLP, pages 1746– 1751.

[11] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1422–1432, Lisbon, Portugal, September. Association for Computational Linguistics.

[12] Graves, A. Mohamed, G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In Proceedings of ICASSP 2013

[13] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, volume 2, pages 427–431

[14] Carla Perez-Almendros, Luis Espinosa-Anke, Steven Schockaert, Dont Patrionize me ! An annotated Dataset with Patrionizing and Condescending Laguge Towards Vulnerable Communities, Cardiff University, United Kingdom, 2020.

[15] Francois Chllet, Deep Learning with Python, Manning Shelter Island, 2018.