Voice@SRIB at SemEval-2020 Tasks 9 and 12: Stacked Ensembling method for Sentiment and Offensiveness detection in Social Media

Abhishek Singh
Samsung R&D Bangalore
abhi3.singh@samsung.com

Surya Pratap Singh Parmar
Samsung R&D Bangalore
s.singhparm@samsung.com

Abstract

In social-media platforms such as Twitter, Facebook, and Reddit, people prefer to use code-mixed language such as Spanish-English, Hindi-English to express their opinions. In this paper, we describe different models we used, using the external dataset to train embeddings, ensembling methods for Sentimix, and OffensEval tasks. The use of pre-trained embeddings usually helps in multiple tasks such as sentence classification, and machine translation. In this experiment, we have used our trained code-mixed embeddings and twitter pre-trained embeddings to SemEval tasks. We evaluate our models on macro F1-score, precision, accuracy, and recall on the datasets. We intend to show that hyper-parameter tuning and data pre-processing steps help a lot in improving the scores. In our experiments, we are able to achieve 0.886 F1-Macro on OffenEval Greek language subtask post-evaluation, whereas the highest is 0.852 during the Evaluation Period. We stood third in Spanglish competition with our best F1-score of 0.756. Codalab username is asking28.

1 Introduction

SemEval Task-9 Sentimix (Patwa et al., 2020) is divided into two tasks, one for Hinglish (Hindi-English) and the other for Spanglish (Spanish-English) code-mixed subtasks. In the Spanglish task, the dataset contains tweets in Spanglish (Spanish-English) code-mixed language, and it is labeled into three categories positive, negative, and neutral sentiments. The task is to classify codemixed tweets into these three sentiments. SemEval Task-12 OffensEval (Zampieri et al., 2020) is divided into different subtasks, English (Rosenthal et al., 2020), Danish (Sigurbergsson and Derczynski, 2020), Arabic (Mubarak et al., 2020), Turkish (Çöltekin, 2020), and Greek (Pitenis et al., 2020) languages. English task is divided into three subtasks- A, B, and C. Subtask-A of OffensEval is offensive language identification, subtask-B is categorization of offensive types into targeted and untargeted, and subtask-C is offensive target identification as individual, group, or other.

In the last decade, there has been proliferation in the use of social media web sites. It has led to pervasive use of hate inducing speech and offensive language to express opinions. The use of profane language has been growing in face-to-face interactions as well as online communications in recent years. The anonymity provided by these websites and lack of stringent action has led to adoption aggressive behavior by people.Youth who experienced cyberbullying, as either an offender or a victim, had more suicidal thoughts and were more likely to attempt suicide than those who had not experienced such forms of peer aggression (Hinduja and Patchin, 2010). Hence it's necessary to auto-remove offensive and profane language in an online environment.

Since the inflow of such type of content is huge, manual filtering is time-consuming and requires much manual labor; hence it becomes almost impractical to do manual filtering. Due to this reason, researchers have proposed methods to automate filtering process by training machine learning models in pre-annotated datasets hate speech and offensive language by (Davidson et al., 2017a) (Malmasi and Zampieri, 2017) , cyberbullying (Xu et al., 2012) and detection of racism by (Tulkens et al., 2016).

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1331–1341
Barcelona, Spain (Online), December 12, 2020.

In this work (team name is SRIB2020), we try to classify twitter tweets in different languages code-mixed for Sentimix tasks and Monolingual tweets in different languages in the OffensEval task into different classes. In OffensEval tasks, tweets are classified as offensive or non-offensive, whereas in the Sentimix task, tweets are classified as positive, negative, and neutral. In the Sentimix task, the "neutral" class is bit ambiguous as many positive tweets in the dataset are labeled as neutral, and negative tweets are labeled as neutral. Class "neutral" has a very thin boundary with the other two classes, "positive" and "negative".

1. ID-7229 - *WOO hoo Cricket world cup starts today. Good luck to host @englandcricket hope for a good start.* - This sentence is positive in tone, but it is labeled as neutral.

2. ID-8199- *@hardikpandya7 best wishes for WorldCup and Eid-Mubarak from MUJAFFAR Hasan National General Secretary LJP URL*- This tweet is positive in its sentiment but is labeled as neutral.

(Lal et al., 2019) first generates subword level representations for the sentences using a CNN architecture. The generated representations are used as inputs to a Dual Encoder Network, consisting of two different BiLSTMs - the Collective and Specific Encoder. The Collective Encoder captures the overall sentiment of the sentence, while the Specific Encoder utilizes an attention mechanism to focus on individual sentiment-bearing sub-words. (Sharma et al., 2016)have annotated the data, developed a language identifier, a normalizer, a part-of-speech tagger, and a shallow parser for sentiment analysis of code-mixed data. (Pravalika et al., 2017) used a lexicon lookup approach to perform domain-specific sentiment analysis. (Joshi et al., 2016) introduce learning sub-word level representations in LSTM (Subword-LSTM) architecture instead of character-level or word-level representations; this enables to learn the information about sentiment value of meaningful morphemes. (Choudhary et al., 2018) uses the shared parameters of siamese networks to map the sentences of code-mixed and standard languages to a common sentiment space. They introduce a primary clustering-based preprocessing method to capture variations of code-mixed transliterated words.

Supervised learning techniques for hate detection, offensive detection, and target and sentiment classification on social media datasets have been explored in recent times. (Davidson et al., 2017b) described a way of multi-class classification of offensive language and hate speech in tweets, using SVM, random forest, naive Bayes, and Logistic Regression. (Del Vigna12 et al., 2017) reported performance for a simple LSTM classifier not better than an ordinary SVM, when evaluated on a small sample of Facebook data for only two classes (Hate, No-Hate), and three different levels of strength of hatred. (Pitsilis et al., 2018) propose a detection scheme that is an ensemble of Recurrent Neural Network (RNN) classifiers. It incorporates various features associated with user-related information, such as the users' tendency towards racism or sexism.

This paper can be summarised into five key points-

1. Applied a variation of Focal Loss by applying class weight along with Gamma parameter in the loss function.

2. Applied multiple preprocessing on the raw text, since in social media platforms people tend to use incorrect grammatical forms and incorrect spellings, it helped us to increase F1 scores.

3. For Hinglish subtask we trained our own word embedding by collecting code-mixed datasets from multiple sources.

4. Ensemble model made of multiple deep-learning based models, CNN, LSTM, and Sequential self-attention on LSTM.

5. Comparing model performance on different Machine Learning and Deep Learning models.

Rest of the paper is organized as follows: **Section-2** presents the methodology in our paper, data description, pre-processing steps, model description, and parameter tuning. **Section-3** presents various

experiments performed on different models and their results. Finally in **Section-4** conclusion based on experiments performed and the future work is discussed. Code is available at github[1].

2 Methodology

2.1 Data Description

1. Sentimix Tasks-
   - Hinglish test set 3000 unlabeled tweets.
   - Spanglish test data contains 3789 unlabelled tweets.

	Positive Train/Dev	**Negative Train/Dev**	**Neutral Train/Dev**	**Total Train/Dev**
Hinglish	4634/982	4102/1128	5264/890	14000/3000
Spanglish	6005/1498	2023/506	3974/994	12002/2998

Table 1: Training and Dev data distribution Sentimix

2. Offenseval Tasks-

   Offenseval English task is divided into three subtasks A, B, and C.

	Train	**Test**
Eng. SubTask-A	8696199	3877
Eng. SubTask-B	188974	1722
Eng. Subtask-C	188974	1722
Danish	2961	329
Turkish	31756	3528
Greek	8743	1544
Arabic	7000	2000

Table 2: OffensEval data distribution

2.2 Data Preprocessing

Hinglish Data Processing

- **In Demojisation step**, different types of emojis present in the corpus is converted into corresponding text representation. Since these combined datasets contain large number of tweets and contain different types of emojis, it becomes necessary to convert emojis into corresponding text representations using the cheatsheet list [2].
- **Removing different types of patterns** such as URLs were replaced with URL token in the dataset, @USERNAME was converted to USER token and hashtags, # symbol was removed from the dataset. The dataset is cleaned for different punctuation marks, as punctuation marks are not needed to train the embeddings.
- **Acronyms and Contractions** were replaced with their corresponding English words. We replace it by creating a dictionary of acronyms and contractions mapping to their expanded form. Acronyms such as 4ever are converted to forever, abt to about, cb to comeback, etc. These acronyms are commonly used in social media platforms. Contractions such as can't, aren't, i've, etc. were again converted to their corresponding text cannot, are not, and I have for this case.

[1] https://github.com/asking28/sentimix2020
[2] https://www.webfx.com/tools/emoji-cheat-sheet/

Spanglish Data Processing- The NLTK Snowball Stemmer [3] package was used because it offers to stem in both English and Spanish. The flexibility to use the stemmer in both languages played a key role in the Spanglish Sentiment Analysis system. The list of stop words was constructed from the stop words corpus provided in NLTK. While pre-processing tokenized tweets, any word included in the NLTK English stop word corpus is excluded. Close attention is paid to elongated words (i.e. – "hellooooo" , "orrrrrale"), and after considering possible features of elongated words, spelling normalization is applied to these tokens. It would also be beneficial to apply spelling normalization to slang or purposely misspelled words that are common in tweets or other informally written texts. We removed character repetition by removing characters that occurred more than two times continuously. Emoticons are replaced with their corresponding text in the tweets.

English Data Preprocessing- Pre-processing steps such as emoticon replacement, contraction replacement, acronym replacements are done in a similar manner as in previous datasets. In social media platforms, people tend to use short forms such as **forget** maybe written as **frgt**. So to deal with this problem we have applied multiple spell correction steps. We have used PySpellchecker [4], it uses a Levenshtein Distance algorithm to find permutations within an edit distance of 2 from the original word. Then it compares all permutations (insertions, deletions, replacements, and transpositions) to known words in a word frequency list. Those words that are found more often in the frequency list are more likely the correct results. Then we delete characters having more than two continuous occurrences, as it is very rare that a character occurs more than twice continuously.

Turkish Data preprocessing- We have followed pre-processing steps as mentioned above with an extra step of Turkish word lemmatization using lemmatization model by (Sak et al., 2008) which is trained by nearly one million Turkish sentences.

Arabic, Danish, and Greek Data processing- Arabic data is first transliterated to Roman script using Classic Language Toolkit (CLTK)[5] and then all the steps used for other languages are applied to datasets. Danish data was used as it is. We applied Greek Stemmer from [6] for Greek language competition, and then followed pre-processing steps as described above.

2.3 Model Description

We have used Ensemble model for all of the tasks mentioned above by combining CNN, self-attention, and LSTM based model.

Algorithm 1: Stacking Ensemble Algorithm

Input: training data D= $\{x_i, y_i\}_{i=1}^{m}$
Output: Ensemble classifier H
Step 1: Learn base level classifiers
for *t*=1 to *T* **do**:
    learn h_t based on dataset *D*
end for
Step 2: Construct new dataset of predictions
for *i*=1 to *m* **do**:
    D_h= $\{x_i', y_i\}$, where $x_i' = \{h_1(x_i), h_2(x_i), ..., h_T(x_i)\}$
end for
Step 3: Learn a meta classifier H
learn H based on D_h
return **H**

In the above algorithm, *T* base level classifiers are trained on training dataset *D*. These base classifiers are named as h_t, where *t* ranges from 1 to T. In step two new dataset D_h is created for meta classifier

[3] https://www.nltk.org/_modules/nltk/stem/snowball.html
[4] https://github.com/barrust/pyspellchecker
[5] http://docs.cltk.org/en/latest/
[6] https://deixto.com/greek-stemmer/

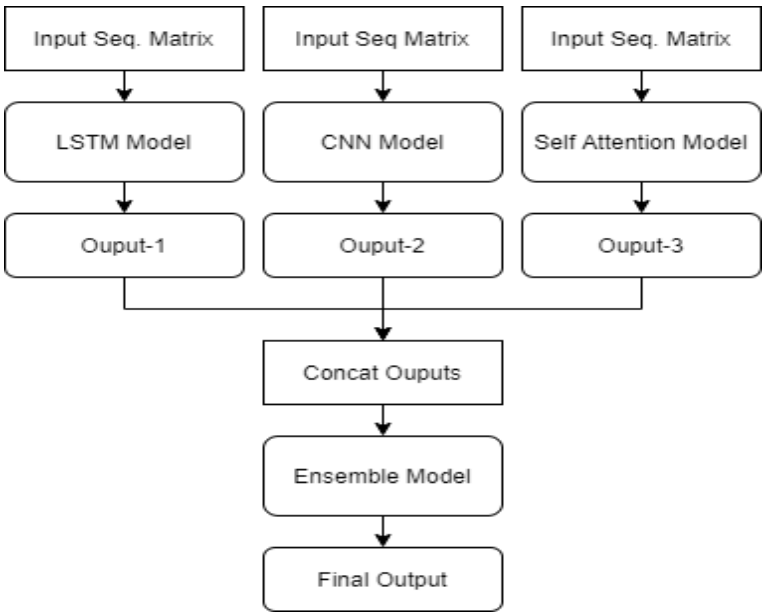


Figure 1: Ensemble Model

where input is taken as base classifiers' output and model output as y_i. Once the dataset is created meta classifier "H" is trained on dataset D_h. In the end, algorithm returns trained meta-classifier "H".

Stacking ensemble model is used for training RNN, CNN, Sequential self-attention with LSTM based architecture together in our model. In stacking, the algorithm takes output of sub-models as inputs and attempts to learn how to best combine inputs to get better output results. The idea of stacking is to learn several weak learners and combine them by training a meta-model to output predictions based on multiple predictions returned by these weak models (Zhou, 2012). In the above algorithm we have three different Deep Learning models with labeled tweets separately. Then output of these models are used as independent variable for stacked model training and labels are same as previous steps.

2.4 Parameters and Hyper-Parameters tuning of models

2.4.1 LSTM

After preprocessing, dataset is split into two parts i.e. training set= 90%, and validation set=10%. For Recurrent Neural Network based model, we have used single LSTM layer with 256 cell units and then MaxPooling Layer to get maximum of all the tokens and then four dense layers with Dropout (dropout rate = 0.3) and BatchNormalization. Final layer is softmax or sigmoid layer depending upon the task. Softmax is used in Sentimix tasks and OffensEval English Subtask-C, and for rest of the subtasks sigloid is used. Model is trained on Focal loss function (Lin et al., 2017), Adam optimizer (Kingma and Ba, 2014), and metrics as accuracy and F1-score. Since maximum number of characters in a tweet are limited to 140 characters including space, we have taken the maximum number of words in a sentence to be 25 considering average lengths of words to be 5 and 3 to 4 spaces.

2.4.2 CNN

A stack of convolutional neural networks (CNN) is used for capturing the hierarchical hidden relations among embedding features. We trained data using CNN model with three convolution layer having filter sizes of (3,4, and 5) respectively, three max pooling layers with filter size of 2 and stride of 2, dense layers of size 4096 and 2048 with Dropout rate of 0.2. Dense layer is connected to softmax or sigmoid layer depending upon the task. Model is trained on Focal loss function (Lin et al., 2017), Adam optimizer (Kingma and Ba, 2014), and metrics as accuracy and F1-score. Tweets are padded in same way as in LSTM model.

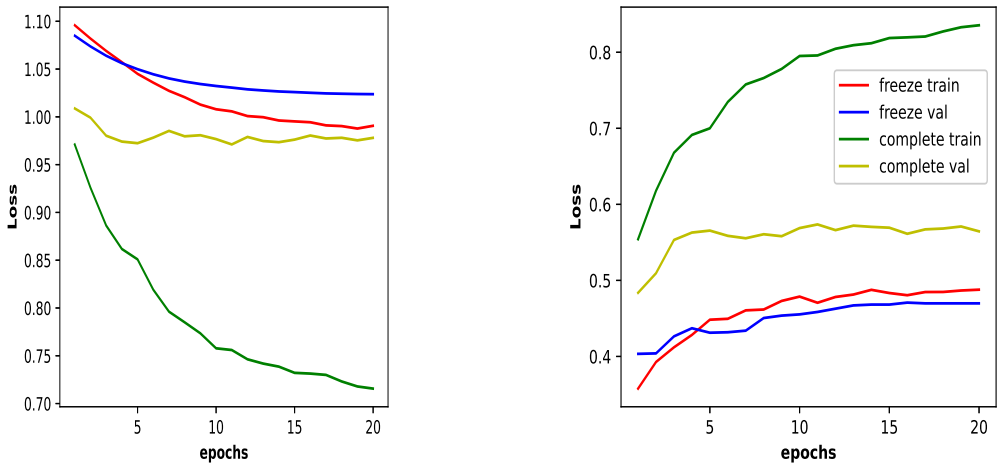
2.4.3 Sequential self-attention model

We have used attention as explained in (Bahdanau et al., 2014), after Gated Recurrent Unit layer (Chung et al., 2014) by returning cell outputs from each steps. This model has 256 GRU cells and each cells return output state which are then fed to self-attention layer. Then there are same number of Dense layers and dropout rate as used in LSTM model. Rest parameters are same as in LSTM layer.

3 Experiments and Results

We performed our experiments on three Deep-Learning Models CNN, LSTM, and ensemble of CNN, LSTM, and self-attention. In Spanglish, we achieved F1-Macro of 0.770, precision of 0.749, and recall of 0.803 with Ensemble model on pre-processed data, whereas it was 0.709, 0.755 and 0.672 respectively for raw text without pre-processing of the data. In Hinglish challenge, Ensemble model out performed other models on pre-processed dataset. Ensemble model on pre-processed data achieved F1-score of 0.682, precision of 0.695 and recall of 0.679 whereas same model when trained on raw data achieved 0.665, 0.681 and 0.665 respectively. From these results we can infer that cleaning steps involved helped to improve the results . We have also experimented our models with and without pretrained Embeddings in Hinglish task but it did not help in improving the scores. We trained Hinglish embeddings by collecting code mixed Hinglish data from various sources such as blogs and scraping twitter data using Fasttext library (Bojanowski et al., 2016). Post evaluation on Spanglish task was not performed since gold labels were not released after the competition. Error analysis of Hinglish and Spanglish tasks is presented in **Appendix A and B** respectively.

Bert multilingual and Bert-uncased mode (Devlin et al., 2018) are trained and fine-tuned by adding three delta layers (dense) layers on top of pre-trained models. We trained Bert model using both ways one by freezing Bert pre-trained parameters and other by keeping parameters as trainable during the complete training process. From our experiments on Hinglish and Spanglish datasets using these models and techniques [7] we found that Bert-uncased model performed better than the Bert-multilingual model. Keeping the pre-trained parameters of Bert as trainable during the complete process performed better than freezing the Bert parameters during fine-tuning. We attribute this behavior of Bert to the difference in data distributions of Bert pre-training and Sentimix tasks.



(a) Loss (b) Accuracy

Figure 2: Bert Model Performance on Hinglish Dataset

In Offenseval tasks, we performed post-evaluation experiments by tuning only few hyper-parameters in the model like changing class-weights and changing loss function. In Greek language subtask our model[8]

[7] https://github.com/asking28/sentimix2020/blob/master/multilingual_bert.py
[8] https://github.com/asking28/offenseval2020/blob/master/offens_2020_greek.ipynb

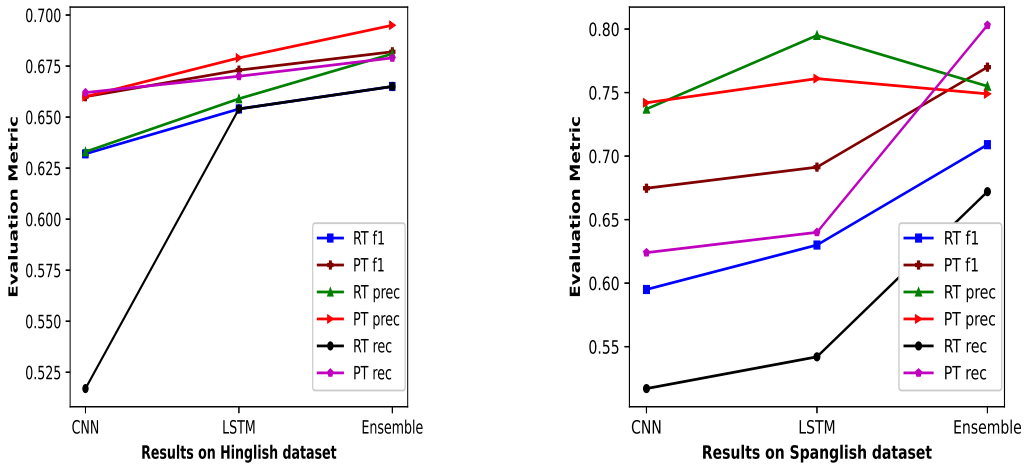


Figure 3: Above plots show F1, Precision and recall when trained on LSTM, CNN and Ensemble models. Here f1 represents F1-macro, prec represents Precision, rec represents Recall, RT represents Raw Text, and PT represents Pre-processed Text.

	Raw (F1-Macro)	**processed (F1-Macro)**	**Raw (Precision)**	**Processed (Precision)**	**Text (Recall)**	**Processed (Recall)**
CNN	0.595	0.6747	0.737	0.742	0.517	0.624
LSTM	0.630	0.6913	0.795	0.761	0.542	0.640
Ensemble	0.709	0.770	0.755	0.749	0.672	0.803

Table 3: Spanglish Testset Evaluation

achieved F1-score of 0.886, which is more than the highest F1-score of 0.852 achieved during evaluation period. In English Subtask-B, our model[9] is able to achieve F1-score of 0.685 in post-evaluation, which was 0.580 in Evaluation period. In Subtask-A and C F1-score in post-evaluation period is 0.9084 and 0.5106 respectively, with very small difference from evaluation period. In Danish task, our model[10] was able to achieve F1-score of 0.6585 in post evaluation period and 0.613 during evaluation period. For Turkish and Arabic tasks there is small difference in the results in evaluation and post-evaluation experiments. Error analysis of English subtasks B and C are presented in **Appendix C, D** respectively.

[9] https://github.com/asking28/offenseval2020/blob/master/offens_task2_bilstm_attention.ipynb

[10] https://github.com/asking28/offenseval2020/blob/master/offens_2020_danish.ipynb

	Raw Text (F1-Macro)	Processed (F1-Macro)	Raw Text (Precision)	Processed (Precision)	Raw Text (Recall)	Processed (Recall)
CNN	0.632	0.660	0.633	0.660	0.517	0.662
LSTM	0.654	0.673	0.659	0.679	0.654	0.670
Ensemble	0.665	0.682	0.681	0.695	0.665	0.679

Table 4: Hinglish Testset Evaluation

	English Subtask-A	English Subtask-B	English Subtask-C	Turkish	Arabic	Danish	Greek
Precision	0.8891	0.7028	0.5268	0.721	0.8014	0.643	**0.882**
Recall	0.9437	0.6813	0.5013	0.707	0.8092	0.683	**0.889**
F1-Score	0.9084	0.6855	0.5106	0.713	0.8052	0.6585	**0.886**

Table 5: Offenseval Testset Post Evaluation

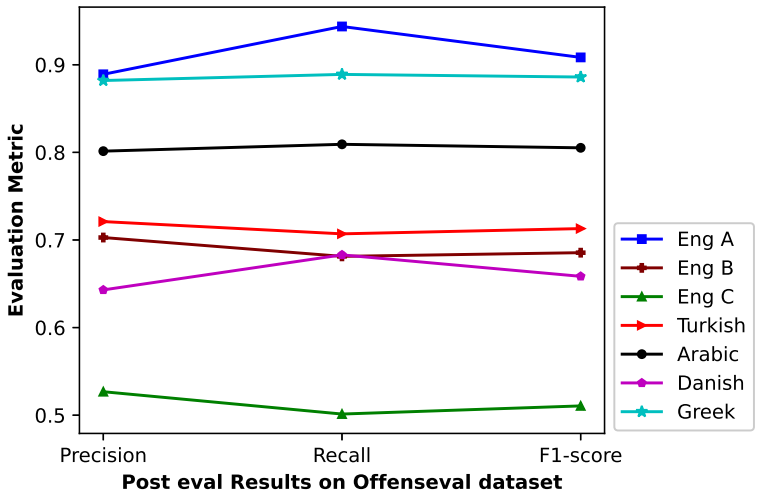


Figure 4: This plot represents Precision, Recall and F1-score for all the OffensEval tasks. Eng A represents English Subtask-A and so on.

English Subtask-A (F1-Macro)	English Subtask-B (F1-Macro)	English Subtask-C (F1-Macro)	Turkish (F1)	Arabic (F1)	Danish (F1)
0.905	0.580	0.514	0.699	0.800	0.613

Table 6: Offenseval Testset Evaluation

4 Conclusion and Future work

In this paper, we present description of the system that we have used in all OffensEval and Sentimix tasks. With our best model we were able to achieve third position in Spanglish task in evaluation period. In post evaluation experiments our model is able to achieve F1-score more than the highest score in Greek task evaluation period. From our experiments, we have found that pre-processing steps played a huge role in increasing F1-scores. Since we have used deep learning models, our model could not perform very well in the tasks where dataset was small like in English Subtask-C. In this work paper we present

different data pre-processing steps that played important role. In English Subtasks we experimented with pre-trained Embeddings [11] trained in twitter corpus, and found that pre-trained embeddings helped to increase F1-score in English sub-tasks but did not help in Hinglish task. The results obtained through our experiments in test data are lower than the results obtained in development set data. We inferred from our experiments that F1-score partly depends upon data distribution of different classes in training and development data which is used to tune hyper-parameters. In most of the tasks, data is not distributed equally among different classes. Exploratory data analysis reveals that there is huge difference in class distribution in the datasets.

Our system presents a solid baseline for Sentiment analysis of code-mixed languages and Offensiveness detection in multiple languages. In our future work, we plan to add handcrafted features along with current features and train it on different machine learning models. We also plan to explore techniques of data augmentation as Deep learning models need large amount of data to train. Corpus used to train code-mixed language models and languages other than English is very small as compared to corpus used to train English language models. Lot of research needs to be done in this direction.

5 Acknowledgement

We deeply thank Ashutosh Singh, Gaurav Kumar, Himanshu Mangla, Suraj Tripathi, and Dr.Tribikram Pradhan for reviewing our work. These experiments have been performed on Google Colab. We thank Colab for providing GPUs and RAM free of cost.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Çağrı Çöltekin. 2020. A Corpus of Turkish Offensive Language on Social Media. In *Proceedings of the 12th International Conference on Language Resources and Evaluation*. ELRA.

Nurendra Choudhary, Rajat Singh, Ishita Bindlish, and Manish Shrivastava. 2018. Sentiment analysis of code-mixed languages leveraging resource rich languages. *arXiv preprint arXiv:1804.00806*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017a. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017b. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*.

Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, pages 86–95.

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

Sameer Hinduja and Justin Patchin. 2010. Bullying, cyberbullying, and suicide. *Archives of suicide research : official journal of the International Academy for Suicide Research*, 14:206–21, 07.

Aditya Joshi, Ameya Prabhu, Manish Shrivastava, and Vasudeva Varma. 2016. Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2482–2491.

[11] `https://github.com/FredericGodin/TwitterEmbeddings`

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yash Kumar Lal, Vaibhav Kumar, Mrinal Dhar, Manish Shrivastava, and Philipp Koehn. 2019. De-mixing sentiment from code-mixed text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 371–377.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting hate speech in social media. *CoRR*, abs/1712.06427.

Hamdy Mubarak, Ammar Rashed, Kareem Darwish, Younes Samih, and Ahmed Abdelali. 2020. Arabic offensive language on twitter: Analysis and experiments. *arXiv preprint arXiv:2004.02192*.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, December. Association for Computational Linguistics.

Zeses Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. Offensive Language Identification in Greek. In *Proceedings of the 12th Language Resources and Evaluation Conference*. ELRA.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

A Pravalika, Vishvesh Oza, NP Meghana, and S Sowmya Kamath. 2017. Domain-specific sentiment analysis approaches for code-mixed social network data. In *2017 8th international conference on computing, communication and networking technologies (ICCCNT)*, pages 1–6. IEEE.

Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. A Large-Scale Semi-Supervised Dataset for Offensive Language Identification. In *arxiv*.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *International Conference on Natural Language Processing*, pages 417–427. Springer.

Arnav Sharma, Sakshi Gupta, Raveesh Motlani, Piyush Bansal, Manish Srivastava, Radhika Mamidi, and Dipti M Sharma. 2016. Shallow parsing pipeline for hindi-english code-mixed social media text. *arXiv preprint arXiv:1604.03136*.

Gudbjartur Ingi Sigurbergsson and Leon Derczynski. 2020. Offensive Language and Hate Speech Detection for Danish. In *Proceedings of the 12th Language Resources and Evaluation Conference*. ELRA.

Stéphan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, and Walter Daelemans. 2016. A dictionary-based approach to racism detection in dutch social media. *CoRR*, abs/1608.08738.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 656–666, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.

Zhi-Hua Zhou. 2012. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition.

Appendices

A Hinglish Error Analysis

This section presents the cases where the model did not give correct predictions and their possible reasoning. During the analysis of the Hinglish dataset on cross-validation data of 1869 tweets 40% tweets

were incorrectly predicted. Our study found that in most cases, either predicted class or ground truth class were labeled as neutral. In the hinglish dataset, our model failed to predict 747 tweets out of 1869 tweets in the validation dataset, and out of 747 tweets, 618 tweets (82%) were labeled as "neutral" in either ground truth labels or predicted labels. From this, we can say that "neutral" class is a bit ambiguous in the Sentimix dataset. Consider a tweet- "rubika di umar mein aap se kaafi chota hun par am big fan of yours kabhi naseeb ne chaha to ". It is labelled as "neutral" whereas we feel that it should be labeled as "positive". In most of the cases, our proposed model gets confused when it is/can be labeled as "neutral".

B Spanglish Error analysis

We found the same distribution in the Spanglish dataset. We performed validation in 2998 tweets, out of which 1025 were erroneous predictions. Out of 1025 tweets, 764 (74%) of the tweets were labeled as "neutral" in either predicted or ground-truth class. Hence from our analysis, we found that labeling a tweet as "neutral" is somewhat ambiguous even for human annotators.

C OffensEval English Subtask-B Error Analysis

We analyzed the categorization of offensive tweets on 37795 tweets as validation data. Out of this validation set, we got 7339 incorrect predictions, and out of these inaccurate predictions, 6912 tweets labeled as "targeted" were incorrectly predicted as "Un-Targeted". Most of the targeted tweets have pronouns like "you, your, they, them, ur, u, she, her, these, him, his, he", contain names of personalities who is targeted, and words like "people, bitch, boys, girls, variations of nigga ". Our model is biased towards such kind of words in a sentence. It can predict a sentence as "targeted", which contains such type of terms. We feel that the dataset includes some incorrect labels, for example - "might fuck around and sleep without my feet covered". It is not explicitly directed towards a person or a group. Our model sometimes fails to identify tweets directly targeting with names; for example, "This is some high level shit. Someone needs to dumb it down for Trump voters". In most of the cases where our model fails to determine a tweet as targeted, the target is from "others" category where the objective is some event, situation, organization or an issue for example- "And here's another fucking breakdown", "I'm sick of it all, April to August has been utter bullshit".

D OffensEval English Subtask-C Error Analysis

We analyzed target identification of OffensEval English subtask-C on 213 targeted tweets validation set. Out of 213 validation data points, 64 tweets' target is incorrectly predicted by our model. In this dataset we found that targets of some tweets are incorrectly labeled for example - "he should be ashamed of himself but he's not because he's #zionel" is targeted towards an individual but labeled as other in the dataset and "#arunjaitleystepdown he is most shameless #fm in history of india and audacity and shamelessness with which is lies in public is disgrace to post." is labeled as "Group" targeted but it is targeted towards an individual. Our model is able to classify these tweets as targeted towards an "Individual" correctly. Our model may be biased towards some pronouns, for example- "Dollar for a phone. you all are fucking dumb." is classified as "Individual" targeted, but its correct label is "Group" targeted possibly due to the presence of "you" in the sentence. Also in tweet "anyway this game sucks", model predicts as "Individual" targeted possibly because it is not able to decode what "this" refers to in the context, here "this" refers to an event "game".