Sentiment Analysis & Transfer Learning

Denis Kokorev
Matriculation: 7007817
Saarland University
deko00002@stud.uni-saarland.de

Mejbah Uddin Shameem
Matriculation: 2577739
Saarland University
s8mesham@stud.uni-saarland.de

1 Introduction

Due to overwhelming use of online based social media and e-commerce platforms in different contexts in recent years, it became extremely important to recognise users' behaviours and emotions for different purposes. Recognising and filtering out online bullying, spams and getting the extensive insights of a product from the customers through comments and reviews are some of the notable task in this area. Binary sentiment analysis draws the line for a sentence, text or comments being a positive one or a negative (offensive) one. A lot of techniques including multinomial logistic regression [5], multinomial naive bayes classifier [3], TfidfVectorizer and SVM based sentiment analysis [1], long short-term memory recurrent neural networks [2] and also convolution neural networks [4]. In natural language processing tasks' Convolution neural network (CNN) has been applied and has shown quite successful outcome. In our work of sentiment classification, we also use CNN based architecture and produce the evaluation metrics for HASOC Hindi and Bengali dataset. Before proceeding with the task of sentiment classification of these datasets we obtain the wor2vec embeddings of the entire datasets using the skip-gram model of word embedding to represent words as corresponding vectors showing the relationships among them in different contexts marked as task 1. In task 2, we do the actual work of sentiment classification where pre-trained Hindi and Bengali embeddings are loaded for Hindi dataset and also Bengali word embeddings are learned to be fed into our CNN based architecture to predict the polarity of text. Moreover, before passing the datasets to the actual classifier we perform data preprocessing steps that includes removing unnecessary information in the data that do not have impact on the classifier. Also, we evaluate the accuracy of trained classifier for Bengali dataset that used Hindi dataset for training to examine the success of transfer learning. Finally, we investigate the reasons of not acquiring good accuracy and try to go one step further to improve the accuracy of the classifier by doing data augmentation in task 3. The report is structured in following order: Section 2 describes the details of methodologies used in all the tasks, section 3 contains the experimental setup, section 4 summarizes the results of the model and finally section 5 concludes the report with references following the section.

2 Methodology

2.1 Task 1. Word2Vec embeddings

The first step to classify the data is to get the word embeddings. Here we use the word2vec skip-gram architecture to obtain the word embeddings of dimension 300.

The pre-processing step is quite straightforward: We remove the punctuation, the stopwords, words starting with '@' and 'http', and words which are larger than 2 or smaller than 30 characters in length. We leave the emojis in the text because they may strongly indicate users' emotions which we will be trying to predict. We leave the words starting with '#' in the text as well because these are just words which may or may not indicate users' attitude to something. The same preprocessing steps are applied to both Bengali and Hindi texts.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2.2 Task 2. The CNN classifier

In the second task we design the CNN architectures for classifying the Hindi and Bengali texts.

Before passing the sentences to the classification model, we pad them so that their sizes become equal. We choose the sentence lengths to be equal to 35 for Hindi dataset and 20 - for the Bengali dataset. The threshold has been chosen by building the histograms of sentence length distribution and setting the threshold such that more than 75% of sentences are smaller than the threshold. The sentences that are larger than the threshold has been shortened in size.

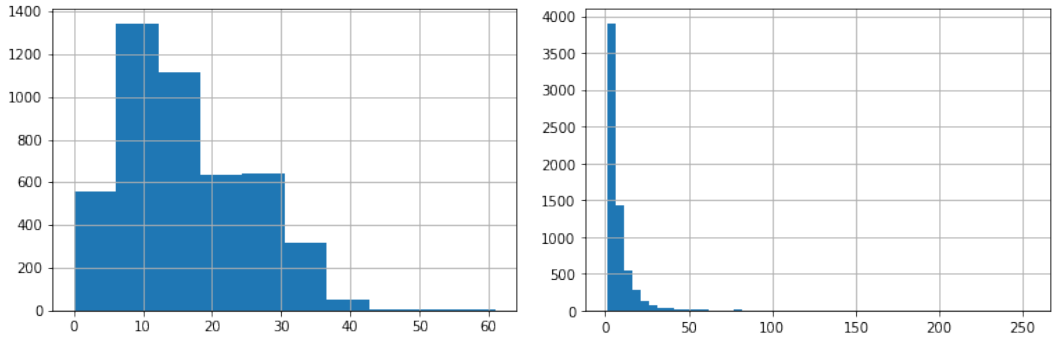

Figure 1: Histograms of the sentence lengths distributions. Left - for Hindi dataset; Right - for Bengali dataset

The CNN architecture is the following. The first layer is the embedding layer. We pre-initialize it with word2vec embeddings obtained for the corresponding dataset (Hindi or Bengali). The batches of 1-hot-encoded words are passed on to this layer, where they are multiplied by the embedding matrix so that the embeddings of the words are returned as the output. Because our architecture uses the embedding layer, we can make use of transfer learning there: initially, the embeddings were obtained by training the word2vec neural network and then these embeddings are fine-tuned in the classification model, making them more relevant to the particular task of binary neural classification. The second layer is a convolutional layer with 3 kernels of sizes '3', '5', '8' and stride '1' for both of the datasets. The projection of the data onto kernels is then activated by a relu activation function to obtain the activation map which is then passed on to the maximum pooling layer to get the most relevant information out of the projections and to make the input to the feed-forward layer invariant in size. Then the max-pooled values are passed on to the feed-forward (or linear) layer followed by the softmax layer to predict the probability of the text polarity ('offensive' or 'not offensive').

The loss function is 'Binary Cross Entropy with Logits'. BCE is a common loss function to have in the classification setting. BCE is with logits because this type of loss function incorporates softmax unit in itself, so that we don't need to have a softmax layer in the network.

We have chosen Adam optimizer as it is one of the most popular and recent optimizers for neural networks.

The datasets has been split on train, validation and test sets with the ratio of 0.7/0.15/0.15 correspondingly. The validation set is used to halt the training, while the test set is used to estimate the overall performance of the model.

Regularization of the neural network is performed with the dropout and early stopping techniques. Dropout act as following: during training, it ignores some of the inter-unit connections to make the training somewhat noisy. Early stopping acts the following way: as soon as the validation error starts increasing, we halt the training and save the model with the lowest validation error.

Apart from training the language models using the two datasets separately, we also examine if some transfer learning can take place in the classification task. For that we take the model trained on the Hindi dataset, replace the embedding layer by the one obtained for the Bengali dataset and apply the classifier to the Bengali corpus.

2.3 Task 3. Dataset augmentation

After performing the experiments we figured out that the training error for the Bengali dataset was approximately 95% while the test error was about 80% which tells us that probably some overfitting took place in the training process. To attempt to decrease the test error even more, we decided to perform dataset augmentation alongside with adding some new preprocessing technique to the pipeline.

The preprocessing technique is the following: we convert emojis and emoticons (e.g., ':)' or ':(') to English words. If converting emojis was performed just as a safety measure (to make them surely not be removed in the preprocessing step), converting the emoticons may result in a slight improvement, because initially all the symbols like ':' or '(' were removed at the preprocessing stage. The relevance of keeping the emoticons and emojis in the text was outlined above: they indicate people's emotions which we are trying to predict.

For augmenting the dataset we used 2 techniques: back translation and replacement on synonyms [6]. Back translation is performed the following way: firstly, the sentence is translated on some intermediate language and then it is translated back on the original language. That way we can get a sentence which is slightly different from the original but keeps the same meaning at one time. The intermediate language was chosen to be English because as this language is international, the translation models are well trained for it, so we will have as less biased translation as possible. We used Google Translate service for translation. This service has its limitation in the number of requests per time unit, so we augmented each of the datasets only twice using this technique.

Replacement on synonyms was performed as follows: for each word in each sentence if there was a synonym for it in a synonyms list, we replaced the word with its synonym. If there were multiple synonyms for a word, we chose one by a random trial. We performed 2 experiments with data augmentation: firstly we augmented the dataset once using each of the techniques in the order that they appear in this paper, then, because the results were not satisfactory, we augmented the datasets once with back translation and 9 times with replacement on synonyms.

Worth noting that the test and validation sets were not augmented, because we need as unbiased sets as possible in order to evaluate the performance.

The architecture of the classification model itself stayed the same to the one in 'Task 2'.

3 Experiment

3.1 Datasets

The datasets and data distributions in the datasets are shown in Table 1. Note that, though we have a 30k samples in Bengali dataset we randomly sample the dataset to match it approximately with the size and class distributions of Hindi dataset for experiment and comparison purposes as per the project guideline. In task 3, we try to increase the number of samples by augmentation as from the table it is understandable that the data samples might not be enough to train a deep neural network without encountering overfitting issue.

Table 1: Details of datasets

Dataset	Total data	Offensive	Not Off.	Training data	Validation data	Test data
Hindi	4665	2469	2196	3265	700	700
Bengali	4700	2500	2200	3290	705	705
Hindi augmented	23694	10910	12784	22295	699	700
Bengali augmented	19853	9277	10576	18448	703	702

3.2 Hyperparameters

In both the task of word embedding and sentiment classification we experimented different values for the hyperparameters to get the best models. For example, we experimented range of values for batch

size, number of filters, filter sizes, dropout and learning rate and finally selected the values that are shown in Table 2 from which best performance is achieved.

Table 2: Hyperparameters used in different models

	Embedding model	Sentiment classifier CNN model
window size	5	-
embedding size	300	300
input size	#of vocabulary	#of vocabulary
batch size	100	64
learning rate	.05	.001
filter size task 2	-	8,5,3
filter size task 3	-	2,5,3
# of filters	-	35
dropout	-	0.5
epochs (original datasets)	100	50
epochs (augmented datasets)	10	10

3.3 Evaluation metrices

For the task of embedding we consider the average error of the datasets by looking at the cosine dissimilarity between the word vectors and their context vectors as performance metric. However, for the binary classification task we know that looking at the precision, recall and F1 score along with accuracy provides greater insights about the model's performance and so in our all different tasks of sentiment classification we use these metrices on test data to evaluate the model's performance.

4 Results

4.1 Task 1. Word2Vec embeddings

Average cosine dissimilarity between the word vectors and their context vectors after the 100th epoch:

For Hindi dataset - 0.9141

For Bengali dataset - 0.8896

We can see that the average error of Hindi dataset is larger than of Bengali one which tells us that words from the Hindi dataset are somewhat less predictable from their context than the ones from the Bengali dataset.

4.2 Task 2. Binary sentiment classifier (CNN)

Table 3 summarizes the performance of sentiment classifier in correspondence with different emdeddings and classifier trained on different datasets. We observe that for Hindi dataset with Hindi embedding the f1 score is higher (82.74) than the same setting of Bengali training with Bengali embedding from scratch (79.48).

Table 3: Performance of CNN classifiers for different datasets in task 3 in percentage

Embedding dataset	Classifier trained on	Accuracy	Precision	Recall	F1
Hindi	Hindi	80.59	80.10	85.56	82.74
Bengali	Bengali	81.38	83.69	75.68	79.48
Bengali	Hindi (transfer learning)	47.66	51.95	98.09	67.93
Bengali	Hindi (retrained on Bengali)	79.17	84.11	69.40	76.05

However, when we change the Hindi model's embedding layer to the Bengali embeddings while keeping the other layers of CNN the same that is from Hindi sentiment classifier to investigate the possibility of transfer learning, we see that the model performance drops by a large margin. Looking

at high recall in this transfer learning setting we feel that the model gets biased to one class and classify everything to that class to achieve high recall but actually performing very poorly. In fact, the accuracy drops to only 47.66% which is the expected behavior. We would expect the model to perform better if at least the word embeddings of Bengali text would be approximetely equal to the ones of Hindi text. The probability of this situation is extremely low at least due to the weight space symmetry or different contextual probabilities of the words of 2 different languages.

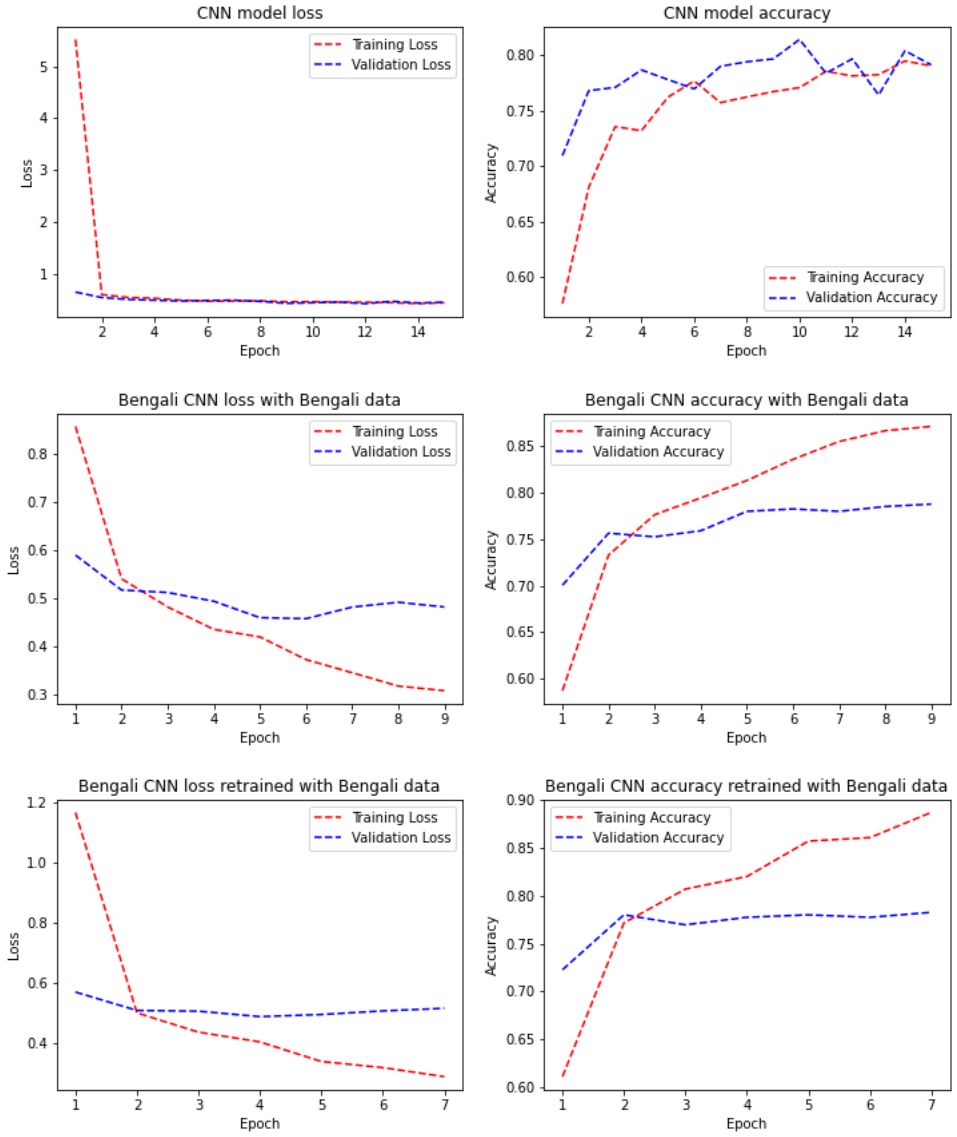

Figure 2: Training and validation loss and accuracy of CNN sentiment classifier with different dataset. Top - Hindi embedding with Hindi dataset; Middle - Bengali embedding with Bengali dataset; Bottom - Top model after retrained with Bengali dataset with Bengali embedding. All these figures are from task 2.

From this point, if the same model is further retrained on Bengali dataset, the accuracy improves from 47.66% to 79.17%, also, the balance between precision and recall is achieved to some extent while acquiring around 9% increase in f1 score causing the score set to 79.65%.

We can observe that the accuracy of the model trained on Hindi text and then retrained on Bengali text is slightly higher than of the model that was trained on the Bengali text only. This may indicate that some transfer learning took place: the classification model first learnt the parameters being trained on

the Hindi dataset and then fine-tuned the parameters being retrained on the Bengali dataset. Therefore, we conclude that because the last model actually used 2 datasets to train, it scored a higher accuracy.

For each settings of the classifier, training loss is always decreasing while the validation loss is sometimes remains almost same in different epochs and in some cases it increases as well. From the plot it is also clear that for Bengali dataset the model is clearly overfitting for the training data. Figure 2 represents the loss and accuracy plots of those different settings of model training of sentiment classifier. Note that, in the x-axis of the plots which represent the number of epochs are different in each setting. We use early stopping method to halt the model training which causes these different numbers in epochs.

4.3 Task 3. Dataset augmentation

Table 4 summarizes the performance of sentiment classifier trained on different datasets after performing the dataset augmentation techniques.

Table 4: Performance comparison of CNN classifier for different datasets in task 2 in percentage

Embedding dataset	Classifier trained on	Accuracy,	Precision	Recall	F1
Hindi	Hindi	75.47	77.30	69.53	73.21
Bengali	Bengali	77.93	74.57	79.82	77.10
Bengali	Hindi	49.53	43.95	29.97	35.64
Bengali	Hindi and retrained on Bengali	77.21	81.27	66.36	73.06

Comparing the models to the ones in 'Task 2', we observe the decline in performance with regard to the accuracy scores. That concludes that augmenting the Hindi training dataset 6.8 times and the Bengali training dataset 5.6 times doesn't lead to a better performance. Instead, it leads to a decrease in performance. There are a couple of reasons for that.

First and the most important reason is biased information representation in the augmented data. When we replace the words on their synonyms, than repeat this process again and again for the same sentence, the meaning of the sentence very often gets biased. The same applies to back translation: translating one sentence to another language and backwards many times leads to the loss of the original meaning, so the sentence that originally was offensive might become not offensive or neutral in the end.

Second reason of bad performance is poor vector representations of the words. Better results may be achieved if one pre-initialize the embedding layer of CNN by the word vectors obtained with training the word2vec model on a large corpus (for instance, on Wikipedia articles of twitter posts). As in our work we trained the embeddings on a relatively small corpus, the vector representations of the words didn't incorporate the meaning of the words objectively. The word embeddings were biased towards the training data very much.

5 Conclusions

We have been able to successfully classify the small portion of data for Hindi and Bengali languages. Also, we have examined that it is possible to apply transfer learning from one language to another in order to perform the classification. To examine the transfer learning possibility further we suggest training the classifier on a large corpus of Bengali data and retraining it on a small corpus of Hindi dataset. If one would achieve better results then by training the model on the Hindi dataset only, that would mean that transfer learning is practically applicable between the two languages.

Also, in future works we would suggest using word embeddings pre-trained on a larger corpus (twitter posts or Wikipedia pages) so that they captured the semantics better.

Considering the dataset augmentation we would suggest augmenting the dataset less times than was presented in the paper to get some less biased information representation and therefore the better results.

After all, other classifier architectures may be investigated to improve the predictions.

References

[1] V. Kumar & B. Subba (2020) *A TfidfVectorizer and SVM based sentiment analysis framework for text data corpus.* In National Conference on Communications (NCC), pp. 1-6.

[2] Karthik Gopalakrishnan & Fathi M. Salem (2020) *Sentiment Analysis Using Simplified Long Short-term Memory Recurrent Neural Networks.* arXiv:2005.03993.

[3] Arif Abdurrahman Farisi, Yuliant Sibaroni, & Said Al Faraby (2019) *Sentiment analysis on hotel reviews using Multinomial Naïve Bayes classifier.* In J. Phys.: Conf. Ser. 1192 012024.

[4] Hannah Kim & Young-Seob Jeong (2019) *Sentiment Classification Using Convolutional Neural Networks.* In Appl. Sci. 2019, 9, 2347.

[5] W.P. Ramadhan, S.T.M.T. Astri Novianty & S.T.M.T. Casi Setianingsih (2017) *Sentiment analysis using multinomial logistic regression.* In International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC), pp. 46-49.

[6] Duong, Huu-Thanh, and Tram-Anh Nguyen-Thi. (2021) *A review: preprocessing techniques and data augmentation for sentiment analysis.* Computational Social Networks 8.1 (2021): 1-16.