QutNocturnal@HASOC'19: CNN for Hate Speech and Offensive Content Identification in Hindi Language

Md Abul Bashar[0000−0003−1004−4085] and Richi Nayak[0000−0002−9954−0159]

School of Electrical Engineering and Computer Science
Queensland University of Technology, Brisbane, Australia
{m1.bashar, r.nayak}@qut.edu.au

Abstract. We describe our top-team solution to Task 1 for Hindi in the HASOC contest organised by FIRE 2019. The task is to identify hate speech and offensive language in Hindi. More specifically, it is a binary classification problem where a system is required to classify tweets into two classes: (a) *Hate and Offensive (HOF)* and (b) *Not Hate or Offensive (NOT)*. In contrast to the popular idea of pretraining word vectors (a.k.a. word embedding) with a large corpus from a general domain such as Wikipedia, we used a relatively small collection of relevant tweets (i.e. random and sarcasm tweets in Hindi and Hinglish) for pretraining. We trained a Convolutional Neural Network (CNN) on top of the pretrained word vectors. This approach allowed us to be ranked first for this task out of all teams. Our approach could easily be adapted to other applications where the goal is to predict class of a text when the provided context is limited.

Keywords: Hate Speech · Offensive Content · Hindi · CNN · Deep Learning.

1 Introduction

The "Hate Speech and Offensive Content Identification in Indo-European Languages" track[1] (HASOC) is one of the tracks in FIRE 2019 conference[2] [16]. Task 1 in this track is identification of hate speech and Offensive (HOF) language in English, German and Hindi in social media posts. In this paper, we describe our approach to the solution of Task 1 in Hindi. The goal is to label a tweet written in Hindi as HOF if it contains any form of non-acceptable language such as hate speech, aggression or profanity; otherwise it is labelled as NOT. There has been significant research on hate speech and offensive content identification in several languages, especially in English [3, 2, 6, 25, 24]. However, there is a lack of work in most other languages. People are now realising the urgency of such research in other languages. Recently, SemEval 2019 Task 5 [4] was carried out

---

[1] https://hasoc2019.github.io/call_for_participation.html
[2] http://fire.irsi.res.in/fire/2019/home

on detecting hate speech against immigrants and women in Spanish and English messages extracted from Twitter, GermEval Share Task [22] was carried out on the Identification of Offensive Language in German language tweets, and TRAC-1 [11] conducted a shared task on aggression identification in Hindi and English. Therefore, HASOC Task 1 for Hindi intends to find out the quality of hate speech and offensive content identification technology in Hindi.

The training dataset is comprised of 4665 labelled tweets in Hindi. The training dataset is created from Twitter and participants are allowed to use external datasets for this task. In the competition setup, the testing dataset is comprised of 1319 unlabelled tweets that were also created from Twitter. The testing dataset and leaderboard were kept unknown to participants until the results were announced. Competitors had to split the training set to get validation set and use the validation set through the competition to compare models. The testing set was only used at the end of the competition for the final leaderboard.

Th proposed approach relies on very little feature-engineering and preprocessing as compared to many existing approaches. Section 2 discusses our top-ranked model building approach. It consists of two steps: (a) pretraining word vectors using a relevant collection of unlabelled tweets and (b) training a Convolutional Neural Network (CNN) model using the labelled training set on top of the pretrained word vectors. Section 3 describes other sophisticated alternative models that we tried. Though these models did not perform as good as compared to our winning model in this track, their performance provides further insight into how to use machine learning models for identifying hate speech and offensive language in Hindi. Section 4 provides experimental results comparing and analysing our various models both on testing set and validation set. The source code of our model can be found online at [1].

2   The Winning Model: QutNocturnal

2.1   Data Collection

Labelled Contest Dataset The goal of Task 1 for Hindi is to predict the class (HOF or NOT) of a given tweet written in Hindi. Out of 4665 labelled tweets in the training set, 2469 (52.92%) are HOF and 2196 (47.07%) are NOT. We randomly kept 20% of training data for validation set. We used ten cross validation in the remaining training set for hyper parameter setting.

Unlabelled External Dataset It is a difficult task to separate abusive tweets from tweets that are sarcastic, joking, or contained abusive keywords in a non-abusive context [3]. Lexical detection methods tend to have low accuracy [6, 23] because they classify a tweet as abusive if it contains any abusive keywords. Also tweets are significantly noisy and do not follow a standard language format. For example, words in tweets are often misspelled, altered, written in Roman letters, include local dialects or foreign languages. To transfer the knowledge of these contexts to the CNN based deep learning model, we pretrain word vectors

using 0.5 million relevant tweets. More specifically, we collected 4,94,311 random tweets in Hindi (i.e. topic of discussion can be anything) using TrISMA[3] and 5251 sarcasm tweets in Hinglish [14] (i.e. sarcasm in Hindi language but written in Roman letters) from [19] for pretraining.

Preprocessing We de-identified person occurrence (e.g. @someone) with *xxatp*, url occurence with *xxurl*, source of modified retweet with *xxrtm* and source of unmodified retweet with *xxrtu*. We fixed the repeating characters (e.g. goooood) in word and removed common invalid characters (e.g. $< br/ >$, $< unk >$, $@ - @$, etc). We used html unescape to replace hexadecimal escape sequences with the character that it represents. We used multi-language spaCy module[4] to lemmatize words and a lightweight stemmer for Hindi language [18] for stemming the words.

2.2 Word Embedding

Embedding models quantify semantic similarities between words based on their distributional property that a word is characterised by the company it keeps. These models quantify semantic properties of words by mapping co-occurring words close to each other in an Euclidean space. Given a sizeable corpus, these models can effectively learn a high-quality word embedding from the co-occurrence of words in the corpus. Word embedding maps each word from the vocabulary to a vector of real numbers. Mikolov et al. [15] proposed two popular models for word embedding based on the feed-forward neural network: Skip-gram and Continuous Bag-of-Words as shown in Figure 1.

In embedding models, a sliding window of a fixed size moves along the text of a corpus. For a given position of the sliding window, let the word in the middle is current word w_i and the words on its left and right within the sliding window are context words C. The continuous bag-of-words model predicts the current word w_i from the surrounding context words C, i.e. $p(w_i|C)$. In contrast, the skip-gram model uses the current word w_i to predict the surrounding context words C, i.e. $p(C|w_i)$. In Figure 1, for example in this corpus, if the current position of a running sliding window contains the phrase *tum sirf chutiya kat ti ho*. In continuous bag-of-words, the context words {tum, sirf, kat, ti, ho} can be used to predict the current word {chutiya}, whereas, in skip-gram, the current word {chutiya} can be used to predict the context words {tum, sirf, kat, ti, ho}.

The objective of model training is to find a word embedding that maximises $p(w_i|C)$ or $p(C|w_i)$ over a corpus. In each step of training, each word is either (a) pulled closer to the words that co-occur with it or (b) pushed away from all the words that do not co-occur with it. A *softmax* or *approximate softmax* function can be used to achieve this objective [15]. At the end of the training, the embedding brings closer not only the words that are explicitly co-occurring

---

[3] https://research.qut.edu.au/dmrc/projects/trisma-tracking-infrastructure-for-social-media-analysis/

[4] https://spacy.io/models/xx

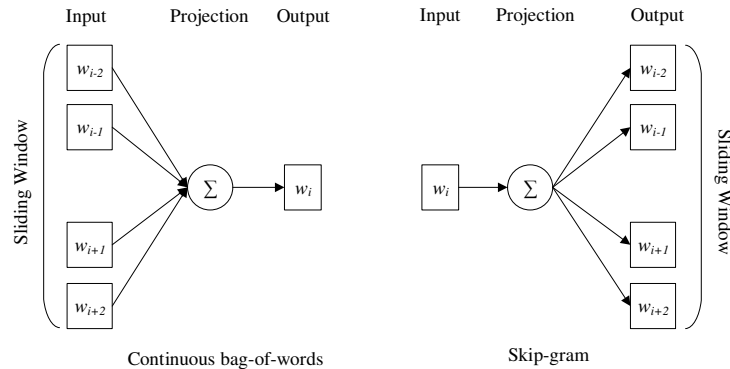Input   Projection   Output   Input   Projection   Output

Fig. 1: Continuous Bag-of-Words and Skip-gram Word Embedding Models [3]

in a training dataset, but also the words that implicitly co-occur. For example, if w_1 explicitly co-occurs with w_2 and w_2 explicitly co-occurs with w_3, then the model can bring closer not only w_1 to w_2, but also w_1 to w_3.

We use the continuous bag-of-words model in this contest as this model is faster and has a slightly better accuracy for the words that appear frequently based on our experimental results. We implemented this model using the module Word2Vec in Gensim Python library. We set the word vector dimension to 200, minimum word count to 2, number of iteration in pretraining to 10, sliding window size to 5 and maximum vocabulary count to 0. We run this model on the unlabelled external dataset described in Section 2.1 to get the pretrain word vectors. Our pretrained word vectors and corresponding python code to use them in classifier are available online at [1].

2.3   Model Architecture

The proposed architecture of our top-ranked model CNN to identify hate speech and offensive language in Hindi is given in Figure 2. This is an empirically customised and regulated version of the architecture that we have used in our prior work of misogynistic tweets identification on Tweeter [3]. In this architecture, we use word embedding to represent each word w in an n-dimensional word vector $\mathbf{w} \in \mathbb{R}^n$. We represent a tweet t with m words as a matrix $\mathbf{t} \in \mathbb{R}^{m \times n}$. We apply convolution operation to the tweet matrix with one stride. Each convolution operation applies a filter $\mathbf{f}_i \in \mathbb{R}^{h \times n}$ of size h. Empirically, based on the accuracy improvement in ten-fold cross validation, 256 filters are used for $h \in \{3, 4\}$ and 512 filters for $h \in \{5\}$. The convolution is a function $\mathbf{c}(\mathbf{f}_i, \mathbf{t}) = r(\mathbf{f}_i \cdot \mathbf{t}_{k:k+h-1})$, where $\mathbf{t}_{k:k+h-1}$ is the kth vertical slice of the tweet matrix from position k to $k + h - 1$, $\mathbf{f}_i$ is the given filter and r is a Rectified Linear Unit (ReLU) function [17]. The function $\mathbf{c}(\mathbf{f}_i, \mathbf{t})$ produces a feature c_k similar to nGrams for each slice k, resulting in $m - h + 1$ features. The max-pooling operation [20] is applied over these features and the maximum value is taken, i.e. $\hat{c}_i = max(\mathbf{c}(\mathbf{f}_i, \mathbf{t}))$. Max-pooling captures the most important feature for each filter. As there are a total

of 1024 filters (256+256+512) in the proposed model, the 1024 most important features are learned from the convolution layer.

Then, we pass these features to a fully connected hidden layer with 256 perceptrons that use the ReLU activation function. This fully connected hidden layer learns the complex non-linear interactions between the features from the convolution layer and generates 256 higher level new features. Finally, we pass these 256 higher level features to the output layer with single perceptron that uses the sigmoid activation function. The perceptron in output layer generates the probability of the tweet being HOF or NOT.

In this architecture (Figure 2), a proportion of units are randomly dropped-out from each layer except the output. This is done to prevent co-adaptation of units in a layer and to reduce overfitting. We set 50% units droppedout from the input layer, the filters of size 3 and the fully connected hidden layer based on best empirical results. Only 20% units are droppedout from the filters of size 4 and 5. Python code for this model is available online at [1].
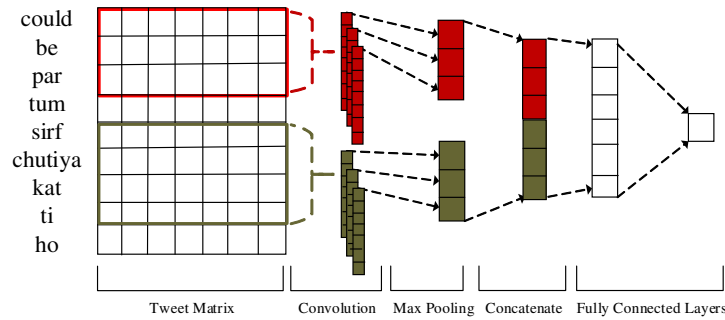

Fig. 2: Architecture of our top-ranked CNN Model for the Hate Speech and Offensive Content Identification track in Hindi Language [3]

3 Alternative Models

We have implemented eight other models in addition to the winning CNN model to see the performance of hate speech and offensive language detection in Hindi.

- Long Short-Term Memory Network (LSTM) [9]. We implement LSTM with 100 units, 50% dropout, binary cross-entropy loss function, Adam optimiser and sigmoid activation.
- Feedforward Deep Neural Network (DNN) [7]. We implement DNN with five hidden layers, each layer containing eight units, 50% dropout applied to the input layer and the first two hidden layers, softmax activation and 0.04 learning rate. We manually tuned hyper parameters of all neural network based models (CNN, LSTM, DNN) based on cross-validation.

– Non NN models including Support Vector Machines (SVM) [8], Random Forest (RF) [13], XGBoost (XGB) [5], Multinomial Naive Bayes (MNB) [12], k-Nearest Neighbours (kNN) [21] and Ridge Classifier (RC) [10]. We automatically tune hyper parameters of all these models using ten-fold cross-validation and GridSearch from scikit-learn. Among all the models, only CNN and LSTM use transfer learning.

4 Experimental Results

A total of nine machine learning models, including the winning customised CNN model, were trained to identify hate speech and offensive language in Hindi. We used transfer learning of word vectors for both CNN and LSTM. The word vectors were pre-trained on a collection of relevant tweets and tuned with the training dataset during the model training.

4.1 Results

The experimental results comparing models in custom validation set are given in Table 1. The detailed results of the winning CNN model in test dataset are

Table 1: Model Comparison Results in Custom Validation Set

	CNN	DNN	kNN	LSTM	MNB	RF	RC	SVM	XGB
Macro Average of Classes									
precision	**0.83**	0.72	0.61	0.79	0.76	0.74	0.73	0.68	0.74
recall	**0.82**	0.72	0.56	0.78	0.75	0.74	0.72	0.61	0.75
f1-score	**0.81**	0.72	0.51	0.78	0.75	0.74	0.72	0.58	0.74
support	933	933	933	933	933	933	933	933	933
Weighted Average of Classes									
precision	**0.84**	0.72	0.61	0.79	0.76	0.74	0.73	0.68	0.75
recall	**0.82**	0.72	0.58	0.78	0.76	0.74	0.73	0.63	0.74
f1-score	**0.81**	0.72	0.52	0.78	0.75	0.74	0.73	0.58	0.75
support	933	933	933	933	933	933	933	933	933
Accuracy									
	0.82	0.72	0.58	0.78	0.76	0.74	0.73	0.63	0.74

given in Table 2.[5]

---

[5] In the absence of any other information except the email message about the top-team performance, we are not able to provide the comparative results with other submitted team results. We will update this table with the rest of the team performance, once we receive information from the track organisers.

Table 2: Detailed Results of Winning Model CNN in Test Dataset

Confusion Matrix		
HOF	NOT	
446	159	HOF
80	633	NOT

Class Wise Performance			
	Precision	Recall	F_1-score Support
HOF	0.85	0.74	0.79 605
NOT	0.8	0.89	0.84 713
Accuracy			0.82 1318
Macro avg	0.82	0.81	0.81 1318
Weighted avg	0.82	0.82	0.82 1318

4.2 Analysis of the results

Experimental results in both validation and test set show that CNN outperforms all other models. CNN is able to outperform LSTM and other baseline models because of the specific nature of tweets. For example, tweets can be super condensed and indirect texts (e.g. satire), may not follow the standard sequence of the language and be full of noise.

Traditional models (e.g. SVM, XGBoost, RF, kNN, etc.) are based on bag-of-words assumption. The bag-of-words (or bag-of-phrases) representation cannot capture sequences and patterns that are very important to identify hate speech and offensive contents in tweets. For example, if a tweet ends saying *if you know what I mean*, there is a high chance that it is an offensive tweet, even though individual words are innocent.

A LSTM model is popularly used in natural language processing research because of its effectiveness of handling sequences in text datasets. Empirical results in Table 1 show that it performed as a second best model. However, the sequence in a tweet can be highly impacted by the noise [3, 23], consequently LSTM finds it difficult to identify the class. On the other hand, CNN can identify many small and large patterns in a tweet, if some of them are impacted by noise it can still use other patterns to identify the class.

5 Conclusion

We introduce an effective method for the task of hate speech and offensive content identification in Hindi. We propose a custom CNN architecture built on word vectors pre-trained on a relevant corpus from the task-specific domain. The proposed model was the top-ranked model in this task under the track. We conducted a series of experiments conducted using state-of-the-art models. Experimental results show that the contexts of hate speech and offensive content can be captured through transfer learning of word embeddings (a.k.a. word vectors) and those contexts can significantly improve the performance of hate speech and offensive content identification. We also observed that when transfer learning through word vectors is utilised, CNN performs better than LSTM because of the noisy nature of tweets. CNN can identify many small and large patterns in a tweet, if some of them gets altered by noise it can still use other

patterns to identify the class of the tweet. On the other hand, LSTM uses the sequence of a tweet to identify its class, but noise in the tweet can alter the sequence and make it hard for LSTM to identify the class.

References

1. Python code and pretrained word vectors of qutnocturnal-hasoc2019. https://github.com/mdabashar/QutNocturnal-Hasoc2019, accessed: 04-10-2019
2. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: Proceedings of the 26th International Conference on World Wide Web Companion. pp. 759–760. International World Wide Web Conferences Steering Committee (2017)
3. Bashar, M.A., Nayak, R., Suzor, N., Weir, B.: Misogynistic tweet detection: Modelling cnn with small datasets. In: Australasian Conference on Data Mining. pp. 3–16. Springer (2018)
4. Basile, V., Bosco, C., Fersini, E., Nozza, D., Patti, V., Pardo, F.M.R., Rosso, P., Sanguinetti, M.: 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. pp. 54–63 (2019)
5. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794. ACM (2016)
6. Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language. arXiv preprint arXiv:1703.04009 (2017)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)
8. Hearst, M.A., Dumais, S.T., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. IEEE Intelligent Systems and their applications **13**(4), 18–28 (1998)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
10. Hoerl, A.E., Kennard, R.W.: Ridge regression: applications to nonorthogonal problems. Technometrics **12**(1), 69–82 (1970)
11. Kumar, R., Ojha, A.K., Malmasi, S., Zampieri, M.: Benchmarking aggression identification in social media. In: Proceedings of TRAC (2018)
12. Lewis, D.D.: Naive (bayes) at forty: The independence assumption in information retrieval. In: European conference on machine learning. pp. 4–15. Springer (1998)
13. Liaw, A., Wiener, M., et al.: Classification and regression by randomforest. R news **2**(3), 18–22 (2002)
14. Mathur, P., Shah, R., Sawhney, R., Mahata, D.: Detecting offensive tweets in hindi-english code-switched language. In: Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media. pp. 18–26 (2018)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
16. Modha, S., Mandl, T., Majumder, P., Patel, D.: Overview of the HASOC track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In: Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation (2019)

17. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp. 807–814 (2010)
18. Ramanathan, A., Rao, D.: A lightweight stemmer for Hindi. In: Workshop on Computational Linguistics for South-Asian Languages, EACL (2003)
19. Swami, S., Khandelwal, A., Singh, V., Akhtar, S.S., Shrivastava, M.: A corpus of english-hindi code-mixed tweets for sarcasm detection. arXiv preprint arXiv:1805.11869 (2018)
20. Tolias, G., Sicre, R., Jégou, H.: Particular object retrieval with integral max-pooling of cnn activations. arXiv preprint arXiv:1511.05879 (2015)
21. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research **10**(Feb), 207–244 (2009)
22. Wiegand, M., Siegel, M., Ruppenhofer, J.: Overview of the germeval 2018 shared task on the identification of offensive language (2018)
23. Xiang, G., Fan, B., Wang, L., Hong, J., Rose, C.: Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In: Proceedings of the 21st ACM international conference on Information and knowledge management. pp. 1980–1984. ACM (2012)
24. Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R.: Predicting the type and target of offensive posts in social media. arXiv preprint arXiv:1902.09666 (2019)
25. Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R.: Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In: Proceedings of the 13th International Workshop on Semantic Evaluation. pp. 75–86 (2019)