VAIS Hate Speech Detection System: A Deep Learning based Approach for System Combination

1st Thai Binh Nguyen
Vietnam Artificial Intelligence System
Hanoi University of Science and Technology
Hanoi, Vietnam
binhnguyen@vais.vn

2nd Quang Minh Nguyen
Vietnam Artificial Intelligence System
Hanoi, Vietnam
minhnq@vais.vn

3rd Thu Hien Nguyen
Thai Nguyen University of Education
Thai Nguyen, Vietnam
nguyenthuhien@dhsptn.edu.vn

4rd Ngoc Phuong Pham
Vietnam Artificial Intelligence System
Thai Nguyen University
Thai Nguyen, Vietnam
phuongpn@tnu.edu.vn

5rd The Loc Nguyen
Vietnam Artificial Intelligence System
Hanoi University of Mining and Geology
Hanoi, Vietnam
locnguyen@vais.vn

6rd Quoc Truong Do
Vietnam Artificial Intelligence System
Hanoi, Vietnam
truongdo@vais.vn

***Abstract*—Nowadays, Social network sites (SNSs) such as Facebook, Twitter are common places where people show their opinions, sentiments and share information with others. However, some people use SNSs to post abuse and harassment threats in order to prevent other SNSs users from expressing themselves as well as seeking different opinions. To deal with this problem, SNSs have to use a lot of resources including people to clean the aforementioned content. In this paper, we propose a supervised learning model based on the ensemble method to solve the problem of detecting hate content on SNSs in order to make conversations on SNSs more effective. Our proposed model got the first place for public dashboard with 0.730 F1 macro-score and the third place with 0.584 F1 macro-score for private dashboard at the sixth international workshop on Vietnamese Language and Speech Processing 2019.**

***Index Terms*—hate speech detection, ensemble, social network comment, natural language processing, text analysis**

I. Introduction

Currently, social networks are so popular. Some of the biggest ones include Facebook, Twitter, Youtube,... with extremely number of users. Thus, controlling content of those platforms is essential. For years, social media companies such as Twitter, Facebook, and YouTube have been investing hundreds of millions euros on this task [1], [2]. However, their effort is not enough since such efforts are primarily based on manual moderation to identify and delete offensive materials. The process is labour intensive, time consuming, and not sustainable or scalable in reality [1], [3], [4].

In the sixth international workshop on Vietnamese Language and Speech Processing (VLSP 2019), the Hate Speech Detection (HSD) task is proposed as one of the shared-tasks to handle the problem related to controlling content in SNSs. HSD is required to build a multi-class classification model that is capable of classifying an item to one of 3 classes (*hate*, *offensive*, *clean*). Hate speech (*hate*): an item is identified as hate speech if it (1) targets individuals or groups on the basis of their characteristics; (2) demonstrates a clear intention to incite harm, or to promote hatred; (3) may or may not use offensive or profane words. Offensive but not hate speech (*offensive*): an item (posts/comments) may contain offensive words but it does not target individuals or groups on the basis of their characteristics. Neither offensive nor hate speech (*clean*): normal item, it does not contain offensive language or hate speech.

The term ‘hate speech’ was formally defined as ‘any communication that disparages a person or a group on the basis of some characteristics (to be referred to as types of hate or hate classes) such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics’ [5]. Many researches have been conducted in recent years to develop automatic methods for hate speech detection in the social media domain. These typically employ semantic content analysis techniques built on Natural Language Processing (NLP) and Machine Learning (ML) methods. The task typically involves classifying textual content into non-hate or hateful. This HSD task is much more difficult when it requires classify text in three classes, with *hate* and *offensive* class quite hard to classify even with humans.

In this paper, we propose a method to handle this HSD problem. Our system combines multiple text representations and models architecture in order to make diverse predictions. The system is heavily based on the ensemble method. The next section will present detail of our system including data preparation (how we clean text and build text representation), architecture of the model using in the system, and how we combine them together. The third section is our experiment and result report in HSD shared-task VLSP 2019. The final

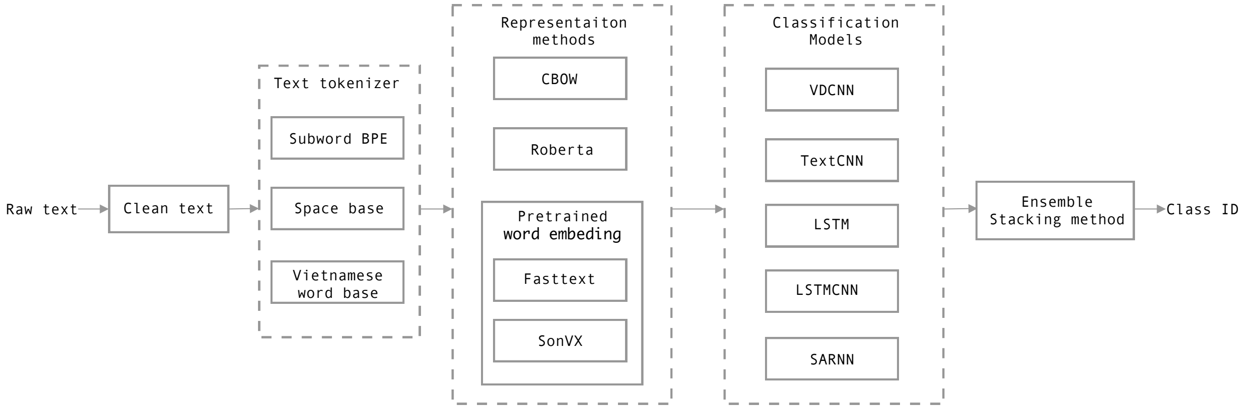

Figure 1. Hate Speech Detection System Overview

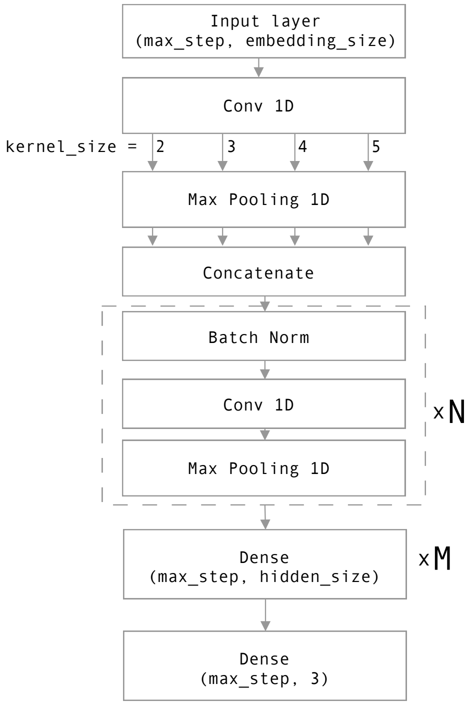

Figure 2. TextCNN model architecture

section is our conclusion with advantages and disadvantages of the system following by our perspective.

II. System description

In this section, we present the system architecture. It includes how we pre-process text, what types of text representation we use and models used in our system. In the end, we combine model results by using an ensemble technique.

A. System overview

The fundamental idea of this system is how to make a system that has the diversity of viewing an input. That because of the variety of the meaning in Vietnamese language especially with the acronym, teen code type. To make this diversity, after cleaning raw text input, we use multiple types of word tokenizers. Each one of these tokenizers, we combine with some types of representation methods, including word to vector methods such as continuous bag of words [6], pre-trained embedding as fasttext (trained on Wiki Vietnamese language) [7] and sonvx (trained on Vietnamese newspaper) [8]. Each sentence has a set of words corresponding to a set of word vectors, and that set of word vectors is a representation of a sentence. We also make a sentence embedding by using RoBERTa architecture [9]. CBOW and RoBERTa models trained on text from some resources including VLSP 2016 Sentiment Analysis, VLSP 2018 Sentiment Analysis, VLSP 2019 HSD and text crawled from Facebook. After having sentence representation, we use some classification models to classify input sentences. Those models will be described in detail in the section II-C. With the multiply output results, we will use an ensemble method to combine them and output the final result. Ensemble method we use here is Stacking method will be introduced in the section II-D.

B. Data pre-processing

Content in the dataset that provided in this HSD task is very diverse. Words having the same meaning were written in various types (teen code, non tone, emojis,..) depending on the style of users. Dataset was crawled from various sources with multiple text encodes. In order to make it easy for training, all types of encoding need to be unified. This cleaning module will be used in two processes: cleaning data before training and cleaning input in inferring phase. Following is the data processing steps that we use:

- **Step 1**: Format encoding. Vietnamese has many accents, intonations with different Unicode typing programs which may have different outputs with the same typing type. To make it unified, we build a library named *visen*[1]. For example, the input "thíêt kế" will be normalized to "thiết kế" as the output.
- **Step 2**: In social networks, people show their feelings a lot by emojis. Emoticon is often a special Unicode character, but sometimes, it is combined by multiple

[1] https://github.com/nguyenvulebinh/visen

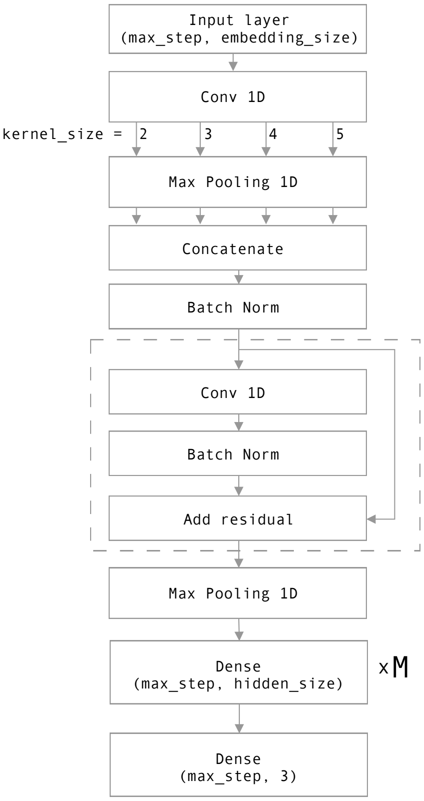

Figure 3. VDCNN model architecture

normal characters like ‘: (=]’. We make a dictionary mapping this emoji (combined by some characters) to a single Unicode character like other emojis to make it unified.

- **Step 3**: Remove unseen characters. For human, unseen character is invisible but for a computer, it makes the model harder to process and inserts space between words, punctuation and emoji. This step aims at reducing the number of words in the dictionary which is important task, especially with low dataset resources like this HSD task.
- **Step 4**: With model requiring Vietnamese word segmentation as the input, we use [10]–[12] to tokenize the input text.
- **Step 5**: Make all string lower. We experimented and found that lower-case or upper-case are not a significant impact on the result, but with lower characters, the number of words in the dictionary is reduced.

RoBERTa proposed in [9] an optimized method for pre-training self-supervised NLP systems. In our system, we use RoBERTa not only to make sentence representation but also to augment data. With mask mechanism, we replace a word in the input sentence with another word that RoBERTa model proposes. To reduce the impact of replacement word, the chosen words are all common words that appear in almost three classes of the dataset. For example, with input *‘nhổn làm gắt vl’*, we can augment to other outputs: *‘vl làm gắt qá’*, *‘còn làm vl vậy’*, *‘vl làm đỉnh vl’* or *‘thanh chút gắt vl’*.

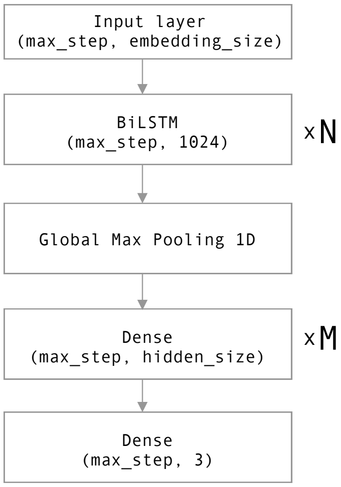

Figure 4. LSTM model architecture

C. Models architecture

Social comment dataset has high variety, the core idea is using multiple model architectures to handle data in many viewpoints. In our system, we use five different model architectures combining many types of CNN, and RNN. Each model will use some types of word embedding or handle directly sentence embedding to achieve the best general result. Source code of five models is extended from the GitHub repository[2]

The first model is TextCNN (figure 2) proposed in [13]. It only contains CNN blocks following by some Dense layers. The output of multiple CNN blocks with different kernel sizes is connected to each other.

The second model is VDCNN (figure 3) inspired by the research in [14]. Like the TextCNN model, it contains multiple CNN blocks. The addition in this model is its residual connection.

The third model is a simple LSTM bidirectional model (figure 4). It contains multiple LSTM bidirectional blocks stacked to each other.

The fourth model is LSTMCNN (figure 5). Before going through CNN blocks, series of word embedding will be transformed by LSTM bidirectional block.

The final model is the system named SARNN (figure 6). It adds an attention block between LTSM blocks.

D. Ensemble method

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. Have the main three types of ensemble methods including Bagging, Boosting and Stacking. In this system, we use the Stacking method. In this method, the output of each model is not only class id but also the probability of each class in the set of three classes. This probability

[2] https://github.com/petrpan26/Aivivn_1

Embedding	VDCNN	TextCNN	LSTM	LSTMCNN	SARNN
comment	0.6812	0.6743	0.6612	0.7012	0.7056
comment_bpe	0.6643	0.6665	0.6514	0.6918	0.6901
comment_tokenize	0.7098	0.7143	0.6832	0.7123	0.7167
fasttext	0.6954	0.7123	0.6812	0.7012	0.7012
roberta	0.6734	0.6636	0.6345	0.6704	0.6566
sonvx_wiki	0.6534	0.6624	0.6456	0.6745	0.6316
sonvx_baomoi_w2	0.6612	0.6712	0.6549	0.6822	0.6513
sonvx_baomoi_w5	0.6656	0.6645	0.6601	0.6756	0.6647

Table I
F1_MACRO SCORE OF DIFFERENT MODEL

will become a feature for the ensemble model. The stacking ensemble model here is a simple full-connection model with input is all of probability that output from sub-model. The output is the probability of each class.

III. Experiment

The dataset in this HSD task is really imbalance. *Clean* class dominates with 91.5%, *offensive* class takes 5% and the rest belongs to *hate* class with 3.5%. To make model being able to learn with this imbalance data, we inject class weight to the loss function with the corresponding ratio (*clean*, *offensive*, *hate*) is $(0.09, 0.95, 0.96)$. Formular 1 is the loss function apply for all models in our system. w_i is the class weight, y_i is the ground truth and $\hat{y}_i$ is the output of the model. If the class weight is not set, we find that model cannot adjust parameters. The model tends to output all *clean* classes.

$$J = -\frac{1}{N}(\sum_{i=1}^{N} w_i * y_i * log(\hat{y}_i)) \tag{1}$$

We experiment 8 types of embedding in total:

- **comment**: CBOW embedding training in all dataset comment, each word is splited by space. Embedding size is 200.
- **comment_bpe**: CBOW embedding training in all dataset comment, each word is splited by subword bpe[3]. Embedding size is 200.
- **comment_tokenize**: CBOW embedding training in all dataset comment, each word is splited by space. Before split by space, word is concatenated by using [10]–[12]. Embedding size is 200.
- **roberta**: sentence embedding training in all dataset comment, training by using RoBERTa architecture. Embedding size is 256.
- **fasttext, sonvx*** is all pre-trained word embedding in general domain. Before mapping word to vector, word is concatenated by using [10]–[12]. Embedding size of fasttext is 300. (sonvx_wiki, sonvx_baomoi_w2, sonvx_baomoi_w5) have embedding size corresponding is (400, 300, 400).

In our experiment, the dataset is split into two-part: *train* set and *dev* set with the corresponding ratio $(0.9, 0.1)$. Two subsets have the same imbalance ratio like the root set. For each combination of model and word embedding, we train model in *train* set until it achieve the best result of loss score in the *dev* set. The table I shows the best result of each combination on the f1_macro score.

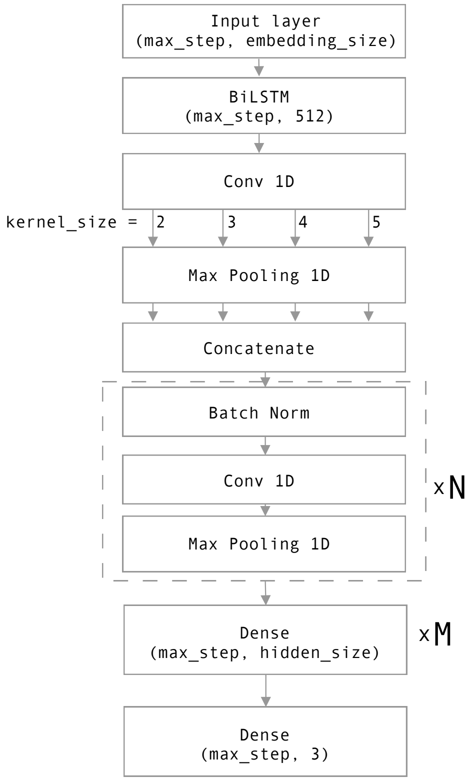

Figure 5. LSTMCNN model architecture

For each model having the best fit on the *dev* set, we export the probability distribution of classes for each sample in the *dev* set. In this case, we only use the result of model that has f1_macro score that larger than 0.67. The probability distribution of classes is then used as feature to input into a dense model with only one hidden layer (size 128). The training process of the ensemble model is done on samples of the *dev* set. The best fit result is 0.7356. The final result submitted in public leaderboard is 0.73019 and in private leaderboard is 0.58455. It is quite different in bad way. That maybe is the result of the model too overfit on *train* set tuning

[3] https://github.com/rsennrich/subword-nmt

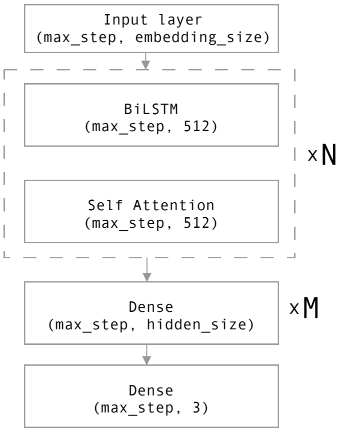

Figure 6. SARNN model architecture

on public *test* set.

Statistics of the final result on the *dev* set shows that almost cases have wrong prediction from *offensive* and *hate* class to *clean* class belong to samples containing the word ‘vl’. (62% in the *offensive* class and 48% in the *hate* class). It means that model overfit the word ‘vl’ to the *clean* class. This makes sense because ‘vl’ appears too much in the *clean* class dataset.

In case the model predicts wrong from the *clean* class to the *offensive* class and the *hate* class, the model tends to decide case having sensitive words to be wrong class. The class *offensive* and the *hate* are quite difficult to distinguish even with human.

IV. Conclusion

In this study, we experiment the combination of multiple embedding types and multiple model architecture to solve a part of the problem Hate Speech Detection with a signification good classification results. Our system heavily based on the ensemble technique so the weakness of the system is slow processing speed. But in fact, it is not big trouble with this HSD problem when human usually involve handling directly in the before.

HSD is a hard problem even with human. In order to improve classification quality, in the future, we need to collect more data especially social networks content. This will make building text representation more correct and help model easier to classify.

References

[1] B. Gambäck and U. K. Sikdar, “Using convolutional neural networks to classify hate-speech,” in *Proceedings of the first workshop on abusive language online*, 2017, pp. 85–90.

[2] Theguardian, “Zuckerberg on refugee crisis: ‘hate speech has no place on Facebook’,” *Last accessed: July 2017, https://www.theguardian.com*, 2016.

[3] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, “Detecting offensive language in social media to protect adolescent online safety,” in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. IEEE, 2012, pp. 71–80.

[4] Z. Waseem and D. Hovy, “Hateful symbols or hateful people? predictive features for hate speech detection on twitter,” in *Proceedings of the NAACL student research workshop*, 2016, pp. 88–93.

[5] J. T. Nockleby, “Hate speech in context: The case of verbal threats,” *Buff. L. Rev.*, vol. 42, p. 653, 1994.

[6] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[7] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, “Fasttext.zip: Compressing text classification models,” *arXiv preprint arXiv:1612.03651*, 2016.

[8] X.-S. Vu, “Pre-trained word2vec models for vietnamese,” https://github.com/sonvx/word2vecVN, 2016, commit xxxxxxx.

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.

[10] T. Vu, D. Q. Nguyen, D. Q. Nguyen, M. Dras, and M. Johnson, “VnCoreNLP: A Vietnamese natural language processing toolkit,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 56–60.

[11] D. Q. Nguyen, D. Q. Nguyen, T. Vu, M. Dras, and M. Johnson, “A fast and accurate vietnamese word segmenter,” in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*, 2018. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2018/summaries/55.html

[12] D. Q. Nguyen, T. Vu, D. Q. Nguyen, M. Dras, and M. Johnson, “From word segmentation to POS tagging for Vietnamese,” in *Proceedings of the Australasian Language Technology Association Workshop 2017*, Brisbane, Australia, Dec. 2017, pp. 108–113.

[13] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751.

[14] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, “Very deep convolutional networks for text classification,” *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 2017. [Online]. Available: http://dx.doi.org/10.18653/v1/e17-1104