NLP@UIT: Exploring Feature Engineer and Ensemble Model for Hate Speech Detection at VLSP 2019

Dang Van Thin[♣*], **Lac Si Le**[♢*], **Ngan Luu-Thuy Nguyen**[♡]
[♣]Multimedia Communications Laboratory
[♢]Department of Software Engineering
[♡]Faculty of Computer Science
University of Information Technology, Vietnam National University Ho Chi Minh City, Vietnam
{thindv,ngannlt}@uit.edu.vn, 17520669@gm.uit.edu.vn

***Abstract*—Social media use has mature exponentially in recent year. It helps Internet users range of benefits, and opportunities to empower themselves in a variety of ways, to express their opinions and viewpoints on a supportive forum. Alongside such good and distinct chances of interaction, bad things like the use of hate speech are also permitted. Digital automated hate speech identification is a major scientific issue in different ways [1]. This paper presents our team approach for the VLSP 2019 - Shared Task competition for Hate Speech Detection on Social networking sites[1]. The proposed strategy relies on feature extraction and ensemble technique to improve the classifier's performance. The experimental results show the supremacy of our model, was received an official ranking of 2 in the private test set with an F1-score of 58.883% and the public test set at 4th place with F1-score of 70.582%.**

I. Introduction

Ubiquitously, Internet access has brought profound change to our lifestyle: information and online social connections are at our fingertips; but it brings new problems with it, such as the unparalleled liberalization of hate speech. Named as "public speech that communicates hatred or encourages violence against a person or group based on race, ethnicity, sex or sexual orientation," the dissemination of online hate speech is suspected to be a primary culprit in creating a state of political violence and exacerbating ethnic violence. There has been considerable pressure on social media platforms to rapidly and delete language, as well as cyberbully and inflammatory posts.

Given these considerations, NLP hate speech work has been minimal primarily because of an insufficient description of hate speech, and an investigation of the most compelling features.

Hate speech recognition is, however, a difficult task. Due to its inherent complexity, varied types of hatred, different goals, and different means to convey the same message, the challenge remains very daunting.

- Nhóc đấy thông minh Vl - This kid is so smart [CLEAN]
- Toàn bộ sự việc chỉ kéo dài một phút trước khi điện thoại Vl tao vang lên. - The whole thing only lasted a minute before the damn phone rang. [OFFENSIVE BUT NOT HATE]
- Mầy chó Vl - You son of a bitch [HATE]

'Vl'- It is an exclamation word (in acronym state) to say something that surprises others. This phrase is often used in communication on Facebook, Zalo, and other social networking sites. 'Vl', however, is transformed into many other meanings such as: 'Vô lí' - indicates the ungainly of someone, 'Vui lắm'- show a person's happiness, and much more.

In this work, we focus on the issue of classifying social network site's comments as **CLEAN, OFFENSIVE BUT NOT HATE, HATE**. Most of the work before is about feature detection and the use of linear representation approaches. Ideally, feature extraction can be applied to any NLP task. Additionally, ensemble methods, meta-algorithms, combine multiple strategies of machine learning in one predictive model, to minimize uncertainty (bagging), bias (boosting), or maximize predictions (stacking).

Given the current hate speech approaches in Vietnamese, our paper's principal contributions are: (1) we analyze the features that improve the detection of hate

[*] Equal contribution
[1]VLSP - Shared Task 2019 http://vlsp.org.vn/vlsp2019

TABLE I
DISTRIBUTION OF THE CORPUS (%).

DATASET	CLEAN	OFFENSIVE BUT NOT HATE	HATE
TRAINING SET	91.47	5.02	3.51

speech (2) we explore, high-power, operative, functional ensemble techniques in identifying hate speech with further improvement (3) our system achieved competitive performance ranking 2nd (Private Test set) and 4th (Public Test set) in VLSP Shared Task 2019.

The rest of the paper is as follows. The next section describes the tasks; the data description, pre-processing details, are provided by Section 3. Section 4 summarily shows our results in the competition, and Section 5 concludes the work with some observations about our experiences.

II. Task Description

Hate Speech Detection (VLSP 2019) offers annotated databases to participants, to solve the problem of detecting hate content on the social network sites (SNSs), and support more effective conversations on SNSs.

The training, test sets providing are composed of 20345 and 5086 instances. Each instance is composed of a text, and multi-class result: CLEAN - the content is defined not as rude or disrespectful. OFFENSIVE BUT NOT HATE - rude in a way that causes upset, insulted, or annoyed. HATE - the content is defined as rude or disrespectful. In this paper, CLEAN, OFFENSIVE BUT NOT HATE, HATE, are labeled 0,1,2, respectively.

In this paper, we focus on training a multi-class classifier to distinguish between these different categories exclusively. We participated in the competition using the team name **ABCD**.

III. Data and Methodology

A. Data Description

The summary of dataset distribution is concluded in the Table I and Table II.

TABLE II
DISTRIBUTION OF THE CORPUS (QUANTITY OF SENTENCES).

DATASET	CLEAN	OFFENSIVE BUT NOT HATE	HATE
TRAINING SET	18614	1022	709

The distribution of three classes can be easily observed from the Table I and Table II. Class 0 (CLEAN) hits the highest, over 18.0 times to Class 1 (OFFENSIVE BUT NOT HATE), particularly 26.5 times to Class 2 (HATE).

It is clear that the results were severely imbalanced, poses significant challenges. A classifier has a natural tendency to catch trends in the most popular classes and to neglect the least popular. Consequently, it can get high overall accuracy without much effort — but without generating any good insights. The overall accuracy might be high, but for the minority class, it will be a very low recall.

B. Preprocessing

A typical text classification framework with preprocessing is one of the critical components. This is a step towards cleaning up and preparing classification information. This dataset includes raw data from posts of SNSs. It is necessary to pre-process information. It includes five follow-up steps:

- **Step 1:** There are many different words relating to the same thing, for example, hashtag. Consequently, we replace these terms with particular words by using a regular expression.
- **Step 2:** These comments are made up of many user terms. It is appropriate to delete special characters and punctuation.
- **Step 3:** Replacing emoji with blank[2]. In this corpus, mostly, emoji are used, not much relevant to its context ('alert icon' inbox ngay cho mình nha , send me a message now). Also, numbers are replaced with 'num' because of these meaningless in this task.
- **Step 4:** We figure out that this dataset has many elongated words; in Vietnamese, it never goes up together with the other form and a syllable in a word. For that reason, we normalize them by true word (e.g. "đkm", "đkmm" are normalized by "đkm" (motherfuck).
- **Step 5:** The ad-hoc tokenizer then partitions each string of characters into tokens[3] made up of (lower-cased) letters separated by spaces.

C. Feature Engineer

In this section, we describe the number of features to achieve competitive results and library using for implementation.

[2]Text clean: https://github.com/trinker/textclean
[3]PyVi: https://pypi.org/project/pyvi/

1) N-gram Features: In this type of feature, there are two features are extracted from the input: word n-gram and character n-gram. In this type of feature, there are three features are extracted from the input: word n-gram, character n-gram and part-of-speech (POS) n-gram. For character n-gram, we extract all n-gram of length 2 to 5 from the original text. In the other case, we extract all word-level n-gram of length 1 to 4 from the pre-processing text. For POS n-gram, we use Pyvi Library[4] to get word part-of-speech. Then we use TF-IDF score of them as numeric features for classifiers. The Scikit-Learn library[5] is used to implement TF-IDF computation.

2) Important Features: Similar to the n-gram feature, we use the TF-IDF information of J important features of each label and form an array of words with a length of 3 x J (each label has J features). We consider this feature as a Boolean feature that indicates whether an important word is contained in the input or not. We use two types of important features: Important character and important word. In our experiment, we take J as 100 for the important word and 300 for the important character.

3) Combination Features: When looking at the data in detail, we find that between classes there are specific characteristics for each class. As follow by Robinson *et al.* [2], we select the number of accordant features with this dataset as follow:

- **Word n-gram**: We extract unigram, bigram, and trigram of text inputs and use TF-IDF to represent these features.
- **PoS n-gram**: We also TF-IDF to represent unigram, bigram, and trigram from Part-of-Speech of text.

Then, we combine the above features with numerical features from observing the training dataset. These features are described as follow:

- **Number of hashtags**: The hashtags are usually represented with a "#" at the beginning of the word. Therefore, we can easily count the number of words representing the hashtag value using the regular expression from the original text.
- **Number of words**: We count the number of words from original text and number of unique words in the pre-processing text as two features.
- **Number of websites**: We count the number of words which indicate the link websites in the text input.

[4]https://pypi.org/project/pyvi/
[5]https://scikit-learn.org/

TABLE III
PUBLIC AND PRIVATE RANKING LIST. THE SYMBOL DENOTES THAT THESE TEAMS ALSO GOT INTO THE TOP3 OF THE PRIVATE-TEST.

#	Public	Score	Private	Score
1	Try hard	0.73019	SunBear	0.61971
2	HH_UIT	0.71432	**ABCD**	0.58883
3	titanic	0.70747	Try hard	0.58455
4	**ABCD**	0.70582	Cr4zy	0.57357
5	TIN HUYNH	0.70576	BA	0.56281

- **Number of icons**: This is an important feature to distinguish the "CLEAN" class with others. We can easily count the number of icons appeared in the text input by using emoji library[6].

D. Stacking Ensemble Method

In this task, we employ the two-layers stacking ensemble composed of five based classifiers in the first layer for each type of feature and a single meta-model to predict the final output. We split the training set into K folds, K-1 will be used as the training set while the k^{th} will be used as the test set. For each the type of feature, we train them with five independent classifiers such as Logistic Regression, two types of ExtraTrees Classifier (*criterion = entropy and criterion = gini*) and two type of Random Forest (*criterion = entropy and criterion = gini*). All classifiers are set *"balanced"* class weight and implemented in Python Scikit-learn Library [7]. For the ExtraTrees classifier, we choose 150 estimators. After K folds training, we feed their predictions of base classifiers into meta-classifier Logistic Regression with C = 0.1 and *"balanced"* class weight.

IV. RESULT

The top systems of competition is concluded in the Table III. Given the strong disparity between instances of different classes, in this case VLSP used the macro F1-score as the official metric for the assessment.

It is clear to see the significant change; some teams ranked on the public but not on the private data set.

V. DISCUSSION

A. Imbalance Data

As mention earlier, the main problem with our results is class inequality. It was shown that imbalance can have a significant influence on the value and meaning of precision and some other well-known performance metrics [3].

[6]https://github.com/carpedm20/emoji
[7]https://scikit-learn.org/

TABLE IV
CONFUSED LABELS SENTENCES.

ID	Text	LABEL
train_fdgiakgzyx	nhìn cưng mà bẩn vl	**1**
train_bohppjueey	Bẩn vl	**0**
train_ahcvvtlvne	Hoang Lien Le hài vkl	**0**
train_vpkqyssdth	tèo sút chất vkl	**1**
train_rwmkmxegcv	ViAnh LàGió vkl :)))	**1**
train_vxfycvizog	"Thy Nguyen.Vkl"	**0**
train_xvalwkcrvj	Cái đéo j nhìn dị vl	**0**
train_swvjhbxxwi	Thế mà mày có nhớ tao đéo đâu :((	**1**

Classification is primarily based on the training data as a supervised learning process. The learning standard plays an essential role in the classifier's accuracy. In this case, the imbalanced existence of the data sets is a significant downfall. The classifier is inadequately trained, and therefore makes incorrect predictions due to the limited occurrence of the small classes.

For multi-class classifier, these imbalances lead to poor quality of entries and inevitably to complete ignorance of these entries [4]. Most classifiers find their data indirectly to be equal; thus, generic classifiers are prejudicial towards the majority. [5]

The different approaches were proposed for dealing with the class imbalance by Jason Wei and Jou [6] - Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. Based on an idea of using Synonym Replacement (one of Jsaon Wei and Jou proposed methods), we renew this technique with certain steps follow-up:

1) Collect top 100 most tokens (by frequency) in the corpus (preprocessed)
2) Manually, check the list of top tokens for re-evaluation, remove items that not relevant ('cô giáo' - teacher in label 1,etc.).
3) Word-dictionary for list words, including its value and PoS[8] value using as a key.
4) Scan our corpus, and substitute each of these terms by a randomly selected synonym in a word- dictionary (based on PoS key).

In the low-weight classes, we created roughly 9000 sentences for an increase without changing their actual labels. Our estimate has nevertheless declined . Then, we took a more in-depth look at our data set for a better approach. We found, after all, that data quality was a concern.

B. Data Labelling

As mentioned above, data quality is problematic, significantly, issues in labels.

[8]ViPosTagger from PyVi

TABLE V
DATA LABELLING ISSUES, 'PROPOSE' COLUMN DENOTES THESE LABELS SHOULD BE CHANGED

Text	LABELED	PROPOSE
Nguyễn Thọ Ars thể cái mặt chó m k học lcs à bêu trường à thằng ngu	**1**	2
Ở trên sân ad ngu vl	**1**	2
Các bạn ơi khi các bạn thấy một cô gái quyết định cắt tóc thì ko có nghĩa là cô ý đang buồn hay thất tình đâu mà là cô gái ấy bị nứng lồz, bị một thế lực thù địch ma xui quỷ khiến nào đó làm cô nhổng lên đi cắt tóc và giờ cô đang cảm vãi cả lồn khi nghĩ đến những lời đùa cợt của mấy con đĩ bạn thân ở lớp vì mái tóc cô giờ đây nhìn đéo khác gì cái đb cuốn giẻ. .	**0**	1

We also noted that some of the sentences have not always the correct label by using human judgement. Since labelling is a challenge because psychological experiences are largely ambiguous and vague [7]. Additionally, emotions are in turn associated with people, objects and things, current, past and foresaw [8]. One of the toughest problems in the area is how these relations can be translated into a tag. A succinct overview of our results is given in Table IV and V.

VI. CONCLUSION AND FUTURE WORK

We introduced our approach and experiments for Hate Speech Identification VLSP shared-task 2019. The tests, configurations and model variants have also been discussed. Ensemble technique uses this variety of methodology to give predictions (clean, offensive but note hate or hate). Each model has been trained on the given corpus separately. After the contest was done, our ensemble system received an F1-score rating of 0.5883% on the final private set.

Improvements could be a re-evaluate the importance of features in this process and explore additional ensemble models and DNN to get a picture of this case.

ACKNOWLEDGMENT

Our thanks, especially, go to the VLSP organizers for organizing the Hate Speech Detection challenge and for the exhilarating and inspiring task, the data, and addressing all our questions promptly.

REFERENCES

[1] I. Ameer, M. H. F. Siddiqui, G. Sidorov, and A. Gelbukh, "CIC at SemEval-2019 task 5: Simple yet very efficient approach to hate speech detection, aggressive behavior detection, and target classification in twitter", in *Proceedings of the 13th International Workshop on Semantic Evaluation*, Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 382–386.

[2] D. Robinson, Z. Zhang, and J. Tepper, "Hate speech detection on twitter: Feature engineering vs feature selection", in *European Semantic Web Conference*, Springer, 2018, pp. 46–49.

[3] A. Luque, A. Carrasco, A. Martın, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix", *Pattern Recognition*, vol. 91, pp. 216–231, 2019.

[4] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, Jul. 2012.

[5] A. Somasundaram and U. S. Reddy, "Data imbalance: Effects and solutions for classification of large and highly imbalanced data", Jan. 2016.

[6] J. W. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks", *arXiv preprint arXiv:1901.11196*, 2019.

[7] R. Cowie, C. Cox, J.-C. Martin, A. Batliner, D. Heylen, and K. Karpouzis, "Issues in data labelling", in *Emotion-Oriented Systems*, Springer, 2011, pp. 213–241.

[8] S. Aman and S. Szpakowicz, "Identifying expressions of emotion in text", in *International Conference on Text, Speech and Dialogue*, Springer, 2007, pp. 196–205.