This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

Hate Speech on Twitter
A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection

Hajime Watanabe, Mondher Bouazizi, Tomoaki Ohtsuki

Abstract—With the rapid growth of social networks and microblogging websites, communication between people from different cultural and psychological backgrounds became more direct, resulting in more and more "cyber" conflicts between these people. Consequently, hate speech is used more and more, to the point where it became a serious problem invading these open spaces. Hate speech refers to the use of aggressive, violent or offensive language, targeting a specific group of people sharing a common property, whether this property is their gender (i.e., sexism), their ethnic group or race (i.e., racism) or their believes and religion, etc. While most of the online social networks and microblogging websites forbid the use of hate speech, the size of these networks and websites makes it almost impossible to control all of their content.Therefore, arises the necessity to detect such speech automatically and filter any content that presents hateful language or language inciting to hatred. In this paper we propose an approach to detect hate expressions on Twitter. Our approach is based on unigrams and patterns that are automatically collected from the training set. These patterns and unigrams are later used, among others, as features to train a machine learning algorithm. Our experiments on a test set composed of 2 010 tweets show that our approach reaches an accuracy equal to 87.4% on detecting whether a tweet is offensive or not (binary classification), and an accuracy equal to 78.4% on detecting whether a tweet is hateful, offensive or clean (ternary classification).

Index Terms—Twitter, Hate Speech, Machine Learning, Sentiment Analysis.

1 Introduction

Online social networks (OSN) and microblogging websites are attracting internet users more than any other kind of website. Services such those offered by Twitter, Facebook and Instagram are more and more popular among people from different backgrounds, cultures and interests. Their contents are rapidly growing, constituting a very interesting example of the so-called big data. Big data have been attracting the attention of researcher, who have been interested in the automatic analysis of people's opinions and the structure/distribution of users in the networks, etc.

While these websites offer an open space for people to discuss and share thoughts and opinions, their nature and the huge number of posts, comments and messages exchanged makes it almost impossible to control their content. Furthermore, given the different backgrounds, cultures and believes, many people tend to use and aggressive and hateful language when discussing with people who do not share the same backgrounds. King et al. [1] reported that 481 hate crimes with an anti-Islamic motive occurred in the year that following 9/11, 58% of them were perpetrated within two weeks after the event. However, nowadays, with the rapid growth of OSN, more conflicts are taking place, following each big event or other.

Nevertheless, while the censorship of content remains a controversial topic with people divided into two groups, one supporting it and one opposing it [2], in OSN, such language still exists. It is even easier to spread among young people as well as older ones than other "cleaner" speeches.

For these reasons, Burnap et al. [3] claimed that collecting and analyzing temporal data allows decision makers to study the escalation of hate crimes following "trigger" events. However, "official" information regarding such events are scarce given that hate crimes are often unreported to the police. Social networks in this context present a better and more rich, yet less reliable and full of noise, source of information.

To overcome this noise and the non-reliability of data, we propose in this work an efficient way to detect both offensive posts and hate speeches in Twitter. Our approach relies on writing patterns, and unigrams along with sentimental features to perform the detection.

The remainder of this paper is structured as follows: in Section 2 we present our motivations and describe some of the related work. In Section 3 we formally define the aim of our work and describe in detail our proposed method for hate speech detection and how features are extracted. In Section 4 we detail and discuss our experimental results. Section 5 concludes this paper and proposes possible directions for future work.

2 Motivations and Related Work

2.1 Motivations

Hate speech is a particular form of offensive language where the person using it is basing his opinion either on segregative, racist or extremist background or on stereotypes. Merriam-Webster[1] defines hate speech as a *"speech expressing hatred of a particular group of people."* From a legal perspective,

Hajime Watanabe, Mondher Bouazizi and Tomoaki Ohtsuki are with the Graduate School of Science and Technology, Keio University, 3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223-8522, Japan. E-mail: h-watanabe@ohtsuki.ics.keio.ac.jp (Hajime Watanabe), bouazizi@ohtsuki.ics.keio.ac.jp (Mondher Bouazizi), ohtsuki@ics.keio.ac.jp (Tomoaki Ohtsuki)

1. http://www.merriam-webster.com/dictionary/

it defines it as a *"speech that is intended to insult, offend, or intimidate a person because of some trait (as race, religion, sexual orientation, national origin, or disability)"*. This being the case, hate speech is considered a world-wide problem that many countries and organizations have been standing up against. With the spread of internet, and the growth of online social networks, this problem becomes even more serious, since the interactions between people became indirect, and people's speech tends to be more aggressive when they feel physically safer, not to mention that internet presents for many hate groups sees it as an *"unprecedented means of communication of recruiting"* [2].

In the context of internet and social networks, not only does hate speech create tension between groups of people, its impact can also influence businesses, or start serious real-life conflicts. For such reasons, websites such as Facebook, Youtube and Twitter prohibit the use of hate speech. However, it is always difficult to control and filter all the contents. Therefore, in the research field, hate speech has been subject to some studies, trying to automatically detect it. Most of these works on hate speech detection have goals such as the construction of dictionaries of hate words and expressions [4] or the binary classification into *"hate"* and *"non-hate"* [5]. However, it is always difficult to clearly decide on a sentence whether it contains hate or not, in particular if the hate speech is hiding behind sarcasm or if no clear words showing hate, racism or stereotyping exist.

Furthermore, OSN are full of ironic and joking content that might sound racist, segregative or offensive, which in reality is not. An example is given in the following two tweets:

- *"Hey dummy. It has been a while since we last read one of your useless comments"*.
- *"If we want the opinion of a WOMAN, we'll ask you dear... For now keep quiet"*.

The first tweet sounds offensive and demeaning the person target of the tweet. However, given the mutual follow of both users, the tweet is actually a joke between two friends. The second also presents the same problem, even though the user seems to be offending women, given the context of the message (i.e., a small discussion between a group of friends), the tweet in itself was not posted to offend women, or even the person targeted by the tweet.

Such expression, and others that include reference to a particular gender, race, ethnic group or religion are widely used in a joking context, and have to be clearly distinguished from hate speeches. Therefore, the use of dictionaries, and *n*-grams in general, might not be the optimal option to perform the distinction between expressions showing hate, and those that do not.

It is arguable that sentiment analysis techniques can be used to perform hate speech detection. However, this is a different task, which requires more sophisticated techniques: In sentiment analysis, the main task is the detection of sentiment polarity of the tweet, which goes back to the idea of the detection of any existing positive/negative word or expression. This makes it easy to rely on the direct meaning of words: words have usually the same sentiment polarity regardless of the context or the actual meaning with very few exceptions (e.g. the word "bad" cannot be interpreted, under any circumstance, in a positive way). However, in the case of hate speech, some words might be negative, might even have the meaning of hate, but the context makes them not hate speech-related. A typical example can be seen in the following two examples:

- *"I hate seeing them losing every time! It's just unfair!"*:
Even though the word "hate" has been employed here, the given sentence does not fall under the category of hate speech, simply because the context is not a context of offending a person, let alone to be offending him for his gender, race, etc.
- *"I hate these neggers, they keep making life much painful"*:
this is obviously a hate speech towards a specific ethnic group.

This makes the task of hate speech detection quite different and more challenging than sentiment analysis: not only is it context-dependent, but also, we should not rely on simple words or even n-grams to detect it.

On a related context, writing patterns have proven to be effective in text classification tasks such as sarcasm detection [6] [7], multi-class sentiment analysis [8] or sentiment quantification [9]. The types of patterns, and the way they are built and extracted depend on the application. Therefore, during this work, we try to extract patterns of hate speech and offensive texts using a pragmatic approach, and use these, along with other features to detect hate speech in short text messages on Twitter.

Therefore, in this work, we propose different sets of features including writing patterns and hate speech unigrams. We use these features together to perform the classification of texts collected from Twitter (i.e., tweets) into three classes we refer to as *"Clean"*, *"Offensive"* and *"Hateful"*. Further description of the different classes will be given in the next section.

The main contribution of this paper are as follows:

1) We propose a pattern-based approach to detect hate speech on Twitter: patterns are extracted in pragmatic way from the training set and we define a set of parameters to optimize the collection of patterns.
2) In addition to patterns, we propose an approach that collects, also in a pragmatic way, words and expressions showing hate and offense, and use them with patterns, along with other sentiment-based features to detect hate speech.
3) The proposed sets of unigrams and patterns can be used as already-built dictionaries for future works related to hate speech detection.
4) We classify tweets into three different classes (instead of only two) where we make distinction between tweets showing hate, and those being just offensive.

2.2 Related Work

The analysis of subjective language on OSN has been deeply studied and applied on different fields varying from sentiment analysis [10] [11] [12] to sarcasm detection [6] [7] or detection of rumors [13] etc. However, relatively fewer works (compared to the aforementioned topics) have been addressed to the hate speech detection. Some of these works targeted sentences in the world wide web such as the work

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

of Warner et al. [5] and Djuric et al. [14]. The first work reached an accuracy of classification equal to 94% with an $F1$ score equal to 63.75% in the task of binary classification, and the second reached an accuracy equal to 80%.

Gitari et al. [15] extracted sentences from some major *"hate sites"* in United States. They annotated each of the sentences into one of three classes: *"strongly hateful (SH)"*, *"weakly hateful (WH)"*, and *"non-hateful (NH)"*. They used semantic features and grammatical patterns features, run the classification on a test set and obtained an F1-score equal to 65.12%.

Nobata et al. [16] used lexicon features, n-gram features, linguistic features, syntactic features, pretrained features, *"word2vec"* features and *"comment2vec"* features to perform the classification task into two classes, and obtained an accuracy equal to 90%.

Nevertheless, some other works targeted the detection of hateful sentences in Twitter. Kwok et al. [17] targeted the detection of hateful tweets against black people. They used unigram features which gave an accuracy equal to 76% for the task of binary classification. Obviously, the focus on the hate speech toward a specific gender, ethnic group, race or other makes the collected unigrams related to that specific group. Therefore, the built dictionary of unigrams cannot be reused to detect hate speech towards other groups with the same efficiency. Burnap et al. [3] used typed dependencies (i.e., the relation between words) along with bag of words (BoW) features to distinguish hate speech utterances from clean speech ones.

3 Proposed Approach

Given a set of Tweets, the aim of this work is to classify each of them into one of three classes which are:

- **Clean:** this class consists of tweets which are neutral, non-offensive and present no hate speech.
- **Offensive:** this class contains tweets that are offensive, but do not present any hate or a segregative/racist speeches
- **Hateful:** this class includes tweets which are offensive, and present hate, racist and segregative words and expressions.

We use machine learning to perform the classification: we extract a set of features from each tweet, we refer to a training set and perform the classification.

3.1 Data

For the sake of this work, we have collected and combined 3 different data sets:

- A first data set publicly available on Crowdflower[2]: this data set contains more than 14 000 tweets that have been manually classified into one of the following classes: "Hateful", "Offensive" and "Clean". All the tweets on this data set have been manually annotated by three people.
- A second data set publicly available also on Crowdflower[3]: which has been used previously in [19] and which has also been manually annotated into one of the three classes: "Hateful", "Offensive" and "Neither", the last referring to the "Clean" class mentioned previously.
- A third data set, which has been published in github[4] and used in the work [18]: Tweets on this data set are classified into one of the following three classes: "Sexism", "Racism" and "Neither". The first two ("Sexism", "Racism") referring to specific forms of hate speech, they have been included as a part of the class "Hateful", whereas the tweets of the class "Neither" have been discarded because there is no indication whether they are clean or offensive (several tweets were manually checked, and they have been identified as belonging to both classes).

As stated above, the three data sets were combined to make a bigger data set, that we split as we will describe later in this section.

To perform the task of classification, the data set is split into three subsets as follows:

- **A training set:** this set contains 21 000 tweets, distributed evenly among the three classes (i.e., *"Clean"*, *"Offensive"* and *"Hateful"*): each class has 7 000 tweets. This set will be referred to as the *"training set"* in the rest of this work.
- **A test set:** this set contains 2 010 tweets: each class has 670 tweets.This set will be referred to as the *"test set"* and will be used to optimize our proposed approach.
- **A validation set:** this set contains 2 010 tweets: each class has 670 tweets. This set will be referred to as the *"valication set"* and will be used to evaluate our proposed approach.

To get fair result, we use the same number of tweets for each set. Given that the number of tweets in *"Hateful"* class was 8 340 and it is the least among the three classes, we set the number of training tweets for each class to 7 000 tweets, that of the test tweets to 670 tweets and that of the validation tweets to 670.

3.2 Data Pre-Processing

In this section, we briefly describe how the tweets were preprocessed. Fig 1 shows the different steps done during this phase.

In a first step, we clean up the tweets. This includes the removal of URLs (which starting either with *"http://"* or *"https://"*) and tags (i.e., *"@user"*) and irrelevant expressions (words written in languages that is not supported by ANSI coding). This is because these do not add any information on whether the tweet might express hate or not. In particular, for the case of tags, if the relationship between the author of the tweet and the person tagged is known, this information might be valuable. However, since no background is given regarding the author and the tagged person, we believe that the use of tags is not useful for our work.

The second step consists of the tokenization, Part-of-Speech (PoS) Tagging, and the lemmatization (using both

2. https://www.crowdflower.com/data-for-everyone/
3. https://data.world/crowdflower/hate-speech-identification
4. https://github.com/ZeerakW/hatespeech

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

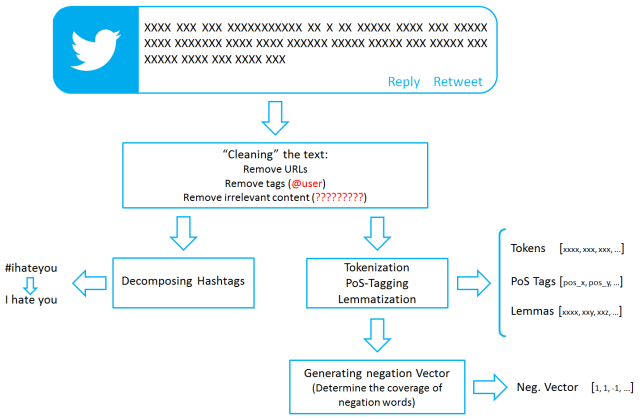

Fig. 1: Pre-processing phases of the tweets

tokens and PoS tags) of the different words. For this sake, we used OpenNLP[5] to perform the Natural Language Processing (NLP) tasks of tokenization and lemmatization. However, to perform the Part-of-Speech (PoS) tagging, we rely on Gate Twitter PoS Tagger [20]. This is because OpenNLP presents poor performances on PoS tagging of informal and noisy texts such as tweets.

Afterwards, we generate what we qualify as negation vector: we detect the position of negation words (e.g., *"not"*, *"never"*, etc.) and detect the coverage of these words. The approach we used is quite simple and inspired from the work of Das et al. [21]: basically, a negation word covers all the words that follows it until the next punctuation mark or the occurrence of a contrast word (e.g., *"but"*, *"however"*, etc). Words covered by a negation word are given a negation score equal to -1 while the rest of the words will be given a score equal to 1. This will be used later on the count of positive and negative words: a positive word (negative word) having a negation score equal to -1 will be considered as a negative word (positive word), and it is attributed the opposite of its original score (This will be explained in the next subsection).

On a separate step, we extract all the hashtags, and use a small tool we developed to decompose it into the words that compose it (e.g., the hashtag *"#ihateyou"* will give the expression *"I hate you"*) and are kept aside to be used when needed.

3.3 Features Extraction

In this subsection, we describe how features are extracted from the tweets, and which we will use later to perform the classification. However, we first explain the choice of our sets of features.

Hate is basically a sentiment among others, a negative sentiment to be precise. Therefore, we believe that relying on sentiment polarity of the tweet is an important indicator of whether or not it can be a potential hateful tweets.

In addition, punctuation marks and use of all-capitalized words can significantly change the meaning of the tweet, or make explicit some intention hidden in a text. Therefore, such features need to be extracted along with sentiment features to detect hate.

However, hate manifests mainly on the words and expressions a person uses. Therefore, the content of the words itself is even more important than the aforementioned features. For this, we extract from the training set, in a pragmatic way, a set of words (to which we refer as unigrams) and expressions (to which we refer as patterns), that are most likely to be related to hate and use them as extra features for hate detection.

As explained early on this work (Section 2.1), unlike sentiment analysis, it is not very useful to rely only on the sentiment polarity of the words to detect hate speech: not only do the words' meanings change according to the context, but also hate speech has different manifestations. Patterns, in such cases, are useful to detect longer hateful expression. Therefore, we extract patterns referring to words, as well as part-of-speech tags, to make sure that we do not get exclusive patterns that apply to only very specific situations, but general ones that reflect hate regardless of the content. In other words, we make sure that an expression extracted that shows hate, is a general one that applies to different contexts of hate. This will be elaborated later on this work, when we set some parameters to make sure that a certain expression occurs enough times in a given class (i.e., it is not specific to a single case or scenario) and does not occur in the other classes (i.e., it is not a general expression that has nothing to do with that class).

To conclude, mainly 4 sets of features are extracted which we qualify as "sentiment-based features", "semantic features", "unigram features", and "pattern features". By combining these sets, we believe it is possible to detect hate speech: "sentiment features" allow us to extract the polarity of the tweet, a very essential component of hate speech (given that hateful speeches are mostly negative ones). "Semantic features" allow us to find any emphasized expression. "Unigram features" allow us to detect any explicit form of hate speech, whereas patterns allow the identification of any longer or implicit forms of hate speech. In the rest of this subsection, we describe how these features are extracted.

3.3.1 Sentiment-based Features

Although the task of detection of hate speech differs drastically from that of sentiment analysis and polarity detection, it still makes sense to use sentiment-based features as the most basic features that allow the detection of hate speech. This is because hate speech is most likely to be present in a *"negative"* tweet, rather than a *"positive"* one.

Consequently, we first extract features that would help to determine whether a tweet is positive, negative or neutral. As mentioned above, the detection of the polarity in itself is not the purpose of this work, but an extra step to facilitate the main task which is the detection of hate speech.

Therefore, from each tweet t we extract the following features:

- the total score of positive words (PW),
- the total score of negative words (NW),
- the ratio of emotional (positive and negative) words $\rho(t)$ defined as: $\rho(t) = \frac{PW-NW}{PW+NW}$; $\rho(t)$ is set to 0 if the tweet has no emotional words,

5. https://opennlp.apach.org

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

- the number of positive slang words,
- the number of negative slang words,
- the number of positive emoticons,
- the number of negative emoticons,
- the number of positive hashtags,
- the number of negative hashtags.

The total score of positive words, and that of negative words are extracted using SentiStrength[6], a tool that attributes sentiment scores to sentences as well as the words of which it is composed. The scores range from -5 to -1 for negative words, and from 1 to 5 for positive words. Given a tweet t, we count the sum of the scores of individual words that have a positive polarity and attribute the obtained sum to the first features; and we do the same for the negative words and attribute the absolute value of the obtained sum to the second features (i.e., both features take positive values).

To detect the polarity of emoticons and slang words, we rely on two manually-built dictionaries containing the emoticons/slang words along with their polarity. As for Hashtags, we developed our own tool that splits a hashtag into the words that composes it and used SentiStrength scores to decide on its polarity.

Sentiment-related features are good indicators whether or not a text is negative. As mentioned above, a negative text is most likely to present hate speech. However, not all negative texts do. Therefore, more features need to be extracted for the sake of detection of hate speech.

3.3.2 Semantic features

Semantic features are ones that describe how an internet user uses punctuation, capitalized words, and interjections, etc. Although hate speech on social networks and microblogging websites do not have a specific and a common use of punctuation or employment of capitalization, in some cases, some of these reflect some sort of segregation or others, such as the following example:

> *"Why don't you simply go back to YOUR COUNTRY and leave us in peace?"*

The tweet is obviously offensive and shows some hate, however, there is no explicit use of hate words, or any sentimental word (except the word *"peace"* which is obviously a positive word.).

Therefore, we believe that punctuation features, including the capitalization, the existence of question and exclamation marks, etc. help detecting hateful speech, and they cannot be simply discarded. In our work, we make use of the following features:

- the number of exclamation marks,
- the number of question marks,
- the number of full stop marks,
- the number of all-capitalized words,
- the number of quotes,
- the number of interjections,
- the number of laughing expressions,
- the number of words in the tweet.

6. http://sentistrength.wlv.ac.uk/

3.3.3 Unigram features

Unigram features are simply unigrams collected from the training set in a pragmatic way, and are used each as an independent feature which can take one of two values: *"true"* and *"false"*.

All unigrams that have a part-of-speech (PoS) tag of a noun, verb, adjective or adverb are extracted from the training set and stored in three different lists (one list for each class) along with their number of occurrences in the corresponding class. We keep only words that occur at least min^u_{occ} (a threshold that represents minimal number of occurrences of unigrams to be taken into account).

Given a word w that appeared in one of the three lists (for convenience we call it C_1), we measure two ratios we refer to as ρ_{12} and ρ_{13} defined as follows:

$$\rho_{12}(w) = \frac{N_1(w)}{N_2(w)} \tag{1}$$

$$\rho_{13}(w) = \frac{N_1(w)}{N_3(w)} \tag{2}$$

where $N_i(w)$ is the number of occurrences of the word in a class i. If the denominator of the ratio is 0, the value is set to 2.

This is done for all the words of the three classes that satisfy the condition mentioned above regarding the number of occurrences. We keep only words that satisfy a second condition defined as follows:

$$\rho_{ij}(w) \geq Th_u \tag{3}$$

where Th_u is a threshold we set for the ratios, that needs to be tuned to maximize the accuracy.

As mentioned above, each of the resulting words will be used as a unique feature: for a word w, in each tweet, we check whether it is employed or not. If the tweet contains the word, the value of the corresponding feature is set to *"true"*, otherwise, it is set to *"false"*.ta

Given the optimal values of the two parameters min^u_{occ} and Th_u (we will describe the optimization process of the different parameters later in this section), the most occurring top words extracted from the tweets of the class, *"hateful"* are given in Fig. 2 and *"offensive"* are given in Fig. 3.

While most of the used words from both classes are just general words that people use when insulting or demeaning someone, some of them have a racist content or content that refers to a specific gender, ethnic group or others (e.g., *"muslims'*, *"islamic"*, *"faggot"*, *"spic"* etc.). We believe that using a bigger training set, we can use the approach we proposed above for *"unigram-features"* to build a dictionary of hate-related words that can be used for future works.

In total, we extracted 1 373 words. Consequently, 1 373 unigram features are defined.

3.3.4 Pattern Features

Pattern features are extracted the same way we extract unigrams: however, before we describe how pattern features are attributed their values and are extracted from the training set, we first introduce a pattern in our context.

In a first step, we divide the words of a tweet into two groups based on whether or not they can be sentimental into

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

Fig. 2: Hateful Class Top Words

Fig. 3: Offensive Class Top Words

two categories: a category *"SW"* (i.e., sentimental word) and a category *"NSW"* (i.e., non-sentimental word). Words that can be sentimental are simply nouns, verbs, adjective and adverbs. Therefore, any word in the tweet that has a PoS that refers to a noun, verb, adjective or adverb is qualified as belonging to *"SW"*. A word that has another PoS tag is qualified as belonging to *"NSW"*.

A pattern is extracted from a tweet as follows: for each word, if it belongs to *"SW"*, it is replaced by its simplified PoS tag as described in TABLE 1 along with its polarity. For example the word *"coward"* will be replaced by the expression *"Negative_ADJECTIVE"*. Otherwise, if the word

TABLE 1: List of PoS tags and their Corresponding Simplified Tags

PoS Tags	**Simplified Tags**
NN, NNS, NNP, NNPS	NOUN
VB, VBD, VBG, VBN, VBP, VBZ	VERB
RB, RBR, RBS	ADVERB
JJ, JJR, JJS	ADJECTIVE
CC	COORDCONJUNCTION
CD	CARDINAL
DT	DETERMINER
EX	EXISTTHERE
FW	FOREIGNWORD
IN	PREPOSITION
LS	LISTMARKER
MD	MODAL
PDT	PREDETERMINER
POS	POSSESSIVEEND
PRP, PRP$	PRONOUN
RP	PARTICLE
SYM	SYMBOL
TO	TO
UH	INTERJECTION
WDT, WP, WP$, WRB	WHDETERMINER
Punctuation marks	.

belongs to *"NSW"* it is simply replaced by its simplified PoS tag as described in TABLE 1.

The resulting vectors extracted from different tweets have different lengths, therefore, we define a pattern as a vector of consecutive words having a fixed length L where L is a parameter to optimize. If a tweets has more than L words, we extract all possible patterns. If it has less words than L, it is simply discarded.

We extract different patterns as described from the training set and save them in three different lists along with their number of occurrences. We filter out the ones that appear less than min^p_{occ}. Afterwards, given a pattern p that appeared in one of the three lists (we call it C_1), we measure two ratios we refer to as ρ_{12} and ρ_{13} defined as follows:

$$\rho_{12}(p) = \frac{N_1(p)}{N_2(p)} \tag{4}$$

$$\rho_{13}(p) = \frac{N_1(p)}{N_3(p)} \tag{5}$$

where $N_i(p)$ is the number of occurrences of the pattern p in a class i. If the denominator of the ratio is 0, the value is set to 2.

Only patterns that satisfy the condition

$$\rho_{ij}(p) \geq Th_p \tag{6}$$

are kept, where Th_p is a threshold we define and tune.

Using the optimal values of the two parameters min^p_{occ} and Th_p, 1875 patterns features are extracted in total.

Given a pattern p, the corresponding feature is attributed a numeric value measuring the resemblance of the tweet to that pattern. Therefore, given a tweet t and a pattern p, we define the following resemblance function [6]:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

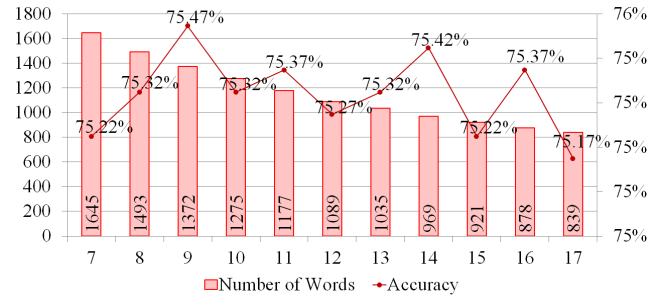

Fig. 4: Classification accuracy (right axis) and number of words collected (left axis) for different values of the parameter min^u_{occ}

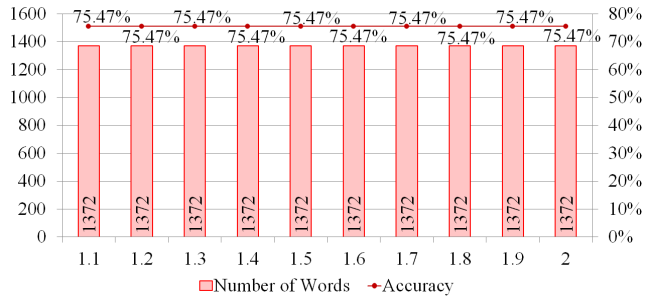

Fig. 5: Classification accuracy (right axis) and number of words collected (left axis) for different values of the parameter Th_u

$$res(p,t) = \begin{cases} 1, & \text{if the pattern appears in the tweets as it is,} \\ \alpha \cdot n/N, & \text{if the tweet contains } n \text{ out of the } N \text{ tags of the pattern in the correct order,} \\ 0, & \text{if the tweet doesn't contain any of the tags of the pattern.} \end{cases}$$

where α is a parameter to optimize.

3.4 Parameters Optimization

The proposed sets of features present different parameters that need to be optimized to obtain the maximum accuracy of classification. The parameters to be optimized are the following:

- the minimal occurrence of words min^u_{occ}
- the word ratios threshold Th_u
- the minimal occurrence of patterns min^p_{occ}
- the pattern ratios threshold Th_p
- the pattern length L
- the coefficient α

To tune these parameters, each time we fix all the parameters except one, and look for its optimal value. Therefore, to determine the best value of the parameter min^u_{occ}, we set the values of the the different parameters as follows:

- $Th_u = Th_p = 1.4$,
- $min^p_{occ} = 3$,
- $L = 7$
- $\alpha = 0.1$

The choice of these values was based on an earlier set of experiment in which we tried to limit the intervals of the values of the parameters: we ran our experiments on each family of features independently using the values of similar parameters that we introduced in a previous work [6]. Then we adjusted the features to get the current values.

We try different values of the parameter min^u_{occ}. The results are given in Fig. 4. The optimal value was obtained for $min^u_{occ} = 9$.

We then keep the values of the different parameters as they are, set min^u_{occ} to 9, and adjust the parameter Th_u. Different values from 1.1 to 2 have been checked, and the optimal value was obtained for $Th_u = 1.4$ as shown in Fig. 5. In total, 1 373 words are collected.

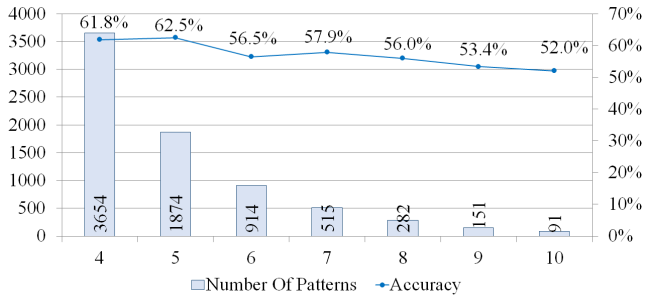

Fig. 6: Classification accuracy (right axis) and number of patterns collected (left axis) for different values of the parameter L

To determine the best length of patterns (i.e., L), we set the values of the parameters related to unigram features to their optimal values and try different values of the parameter L as shown in Fig. 6. we kept the other parameters as we set them initially. The optimal value was obtained for $L = 5$, and the total number of patterns obtained is 1 875.

We proceed the same way to obtain the optimal values of min^p_{occ} and Th_p. The optimal values of the parameters are 7 and $1.3 \sim 1.9$ (in the rest of this work the value 1.4 is considered) respectively as show in Figs. 7 and 8.

We finally set the values of the four parameters to their optimal and tried different values of α. The obtained results

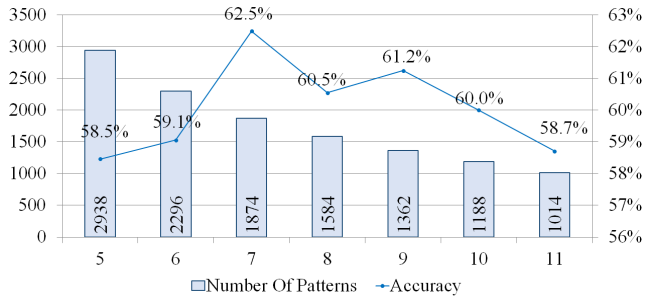

Fig. 7: Classification accuracy (right axis) and number of patterns collected (left axis) for different values of the parameter min^p_{occ}

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

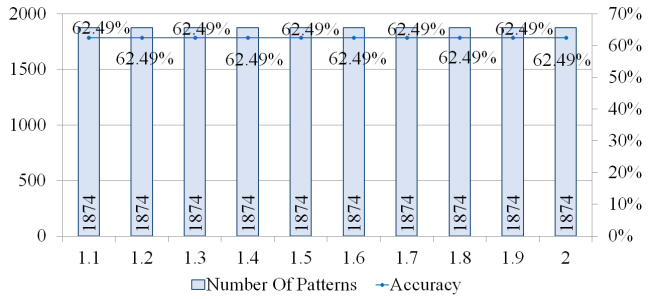

Fig. 8: Classification accuracy (right axis) and number of patterns collected (left axis) for different values of the parameter Th_p

TABLE 2: Accuracy, Precision, Recall and F1-Score of Classification Using Different Classifiers

	TP Rate	FP Rate	Prec.	Recall	F1-Score
Rand. Forest	0.592	0.204	0.605	0.592	0.589
SVM	0.57	0.276	0.643	0.57	0.599
J48graft	0.784	0.108	0.793	0.784	0.784

did not differ much (keeping in mind α should have a low value). The optimal value of this parameter is equal to 0.01.

Therefore, for the rest of this work, we considered the first case and keep the values of the parameters as follows:

$$\begin{cases} min^u_{occ} &= 9, \\ Th_u &= 1.4, \\ min^p_{occ} &= 7, \\ Th_p &= 1.4, \\ L &= 5, \\ \alpha &= 0.01. \end{cases}$$

4 Experimental Results

After the extraction of features and optimization of parameters, we proceed to our final experiments. The classification is done using the toolkit weka [22]. Weka presents variety of classifiers organized into groups based on the type of the algorithm (e.g., decision tree-based, rule-based, etc.).

To evaluate the performance of classification, we use 4 different key performances indicators (KPIs) which are the percentage of true positives, the precision, the recall and the $F1\text{-}score$ defined as:

$$F1\text{-}score = 2 \times \frac{Precision \cdot Recall}{Precision + Recall} \tag{7}$$

For the sake of our work, to perform the classification, we use the machine learning algorithm *"J48graft"* [23]. The algorithm *"J48graft"* presents a main parameter to tune which is the confidence threshold for pruning (*C*). The optimal value of this parameter, obtained during this work is $C = 0.04$. This is because this classifier presents better performances than other classifiers (even powerful ones such as Support Vector Machine (SVM) and Random Forest, etc.). The fact that *"J48graft"* outperforms SVM might be due to the existence of hundreds of binary features (that take the values *"true"* or *"false"*), since SVM is better dealing with numeric features. However this does not explain why *"J48graft"* outperforms Random Forest.

Table 2 shows the performances of classification using *"J48graft"* compared to that using some other classifiers.

Initially, we perform the classification on the test set, which has been used to optimize the features' parameters we defined. This is to optimize also the parameters of the classifier used (i.e., *"J48graft"*). Once the parameters are optimized we re-run the classification again on the validation set. This is to make sure that the features as well as the classifier parameters were not overfitting to the current test set, and that they perform well for a completely different set.

TABLE 3: Binary Classification Performances on the Test Set

Class	TP Rate	FP Rate	Prec.	Recall	F1
Sentiment-based Features					
Offensive	0.963	0.767	0.715	0.963	0.821
Clean	0.233	0.037	0.757	0.233	0.356
Overall	0.719	0.524	0.729	0.719	0.666
Semantic Features					
Offensive	0.984	0.976	0.669	0.984	0.796
Clean	0.024	0.016	0.432	0.024	0.045
Overall	0.664	0.656	0.590	0.664	0.554
Unigram Features					
Offensive	0.778	0.091	0.945	0.778	0.853
Clean	0.909	0.222	0.671	0.909	0.772
Overall	0.821	0.135	0.8534	0.821	0.826
Pattern Features					
Offensive	0.690	0.284	0.830	0.690	0.754
Clean	0.716	0.310	0.536	0.716	0.613
Overall	0.699	0.292	0.732	0.699	0.707
All features combined					
Offensive	0.876	0.122	0.935	0.876	0.904
Clean	0.878	0.124	0.780	0.878	0.826
Overall	**0.877**	**0.123**	**0.883**	**0.877**	**0.878**

TABLE 4: Binary Classification Confusion Matrix

Class	Classified as	
	Offensive	Clean
Offensive	**1174**	166
Clean	82	**588**

4.1 Binary Classification

In a first step, we combined the tweets of the two classes *"hateful"* and *"offensive"* under one class we refer to as *"offensive"* (since hateful tweets are indeed offensive and aggressive). This is to make the classification a binary classification task. In the training set, in total we have 14 000 tweets for class *"offensive"* and 7 000 tweets for the class *"clean"*. As for the test set, the number of tweets of the class *"offensive"* is 2,680 while that of the class *"clean"* is 1 340. Using these sets, run the classification. The obtained results are given in TABLE 3, while the confusion matrix is given in TABLE 4.

We then perform the binary classification on the validation set (which, to remind, has not been involved in any of the optimization process steps). The results of the classification are given in Table 5 and the confusion matrix is given in Table 6.

The overall accuracy obtained when all the features are used is equal to 87.4% with a precision equal to 93.2% for the class *"offensive"*. The performances per family of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2806394, IEEE Access

TABLE 5: Binary Classification Performances on the Validation Set

Class	TP Rate	FP Rate	Prec.	Recall	F1
Sentiment-based Features					
Offensive	0.963	0.767	0.715	0.963	0.821
Clean	0.233	0.037	0.757	0.233	0.356
Overall	0.719	0.524	0.729	0.719	0.666
Semantic Features					
Offensive	0.989	0.976	0.669	0.984	0.796
Clean	0.024	0.016	0.432	0.024	0.045
Overall	0.664	0.656	0.590	0.664	0.554
Unigram Features					
Offensive	0.778	0.091	0.945	0.778	0.853
Clean	0.909	0.222	0.671	0.909	0.772
Overall	0.821	0.135	0.854	0.821	0.826
Pattern Features					
Offensive	0.710	0.321	0.816	0.710	0.759
Clean	0.679	0.290	0.539	0.679	0.601
Overall	0.700	0.311	0.723	0.700	0.706
All features combined					
Offensive	0.875	0.128	0.932	0.875	0.902
Clean	0.872	0.125	0.777	0.872	0.821
Overall	**0.874**	**0.127**	**0.88**	**0.874**	**0.875**

TABLE 6: Binary Classification Confusion Matrix of the Validation Set

Class	Classified as	
	Offensive	Clean
Offensive	**1172**	168
Clean	86	**584**

features show that the unigram features as well as the pattern features present the highest accuracy with values respectively equal to 82.1% and 70%. This is because the way these features are extracted (pragmatic approach) made them highly related to the different classes. In other words, while punctuation-based and sentiment-based marks have not been selected to reflect any specific aspect, and have been extracted from the different tweets as they are, patterns and top words are polarized features and the existence of any of them in a tweet has a high influence on the decision whether it is offensive or not.

Semantic features on the other hand does not have a good classification accuracy. This is because, when they are used alone, these features cannot tell whether or not a text is hateful, offensive or clean. In other words, these features need to be combined with the other sets of features to make sense. The same goes for sentiment-based features: even though offensive language is more likely to appear in negative tweets, this information alone (whether the tweet is positive or negative) is not enough to judge on the content of the tweet and the language used.

4.2 Ternary Classification

Again, we run the classification on the validation set, to confirm the efficiency of the features and the parameters used. The results of the classification on the validation set are given in TABLE 9 and the confusion matrix is given in TABLE 10.

While the binary classification discussed in the previous subsection is important since it allows to automatically detect offensive, aggressive and hateful speeches with a precision equal to 93.2%, it is a more challenging task to go deeper in the classification, and separate tweets containing hate speech from those that are just offensive. Hate speech as discussed in the motivation usually targets groups of people based on their backgrounds, while an offensive text might just target the one person to whom the message is sent. Even for humans with no background about the speaker, it is usually very hard to judge whether a tweet is hateful or just offensive.

TABLE 7: Ternary Classification Performances on the Test Set

Class	TP Rate	FP Rate	Prec.	Recall	F1
Sentiment-based Features					
Hateful	0.390	0.216	0.474	0.390	0.428
Offensive	0.363	0.161	0.529	0.363	0.655
Clean	0.696	0.399	0.466	0.696	0.671
Overall	0.483	0.259	0.490	0.483	0.648
Semantic Features					
Hateful	0.219	0.227	0.326	0.219	0.262
Offensive	0.655	0.509	0.392	0.655	0.490
Clean	0.284	0.185	0.434	0.284	0.343
Overall	0.386	0.307	0.384	0.386	0.365
Unigram Features					
Hateful	0.639	0.058	0.846	0.639	0.728
Offensive	0.701	0.045	0.887	0.701	0.783
Clean	0.931	0.261	0.641	0.931	0.759
Overall	0.757	0.121	0.791	0.757	0.757
Pattern Features					
Hateful	0.281	0.117	0.545	0.281	0.370
Offensive	0.734	0.047	0.886	0.734	0.803
Clean	0.858	0.399	0.518	0.858	0.646
Overall	0.624	0.188	0.650	0.624	0.726
All features combined					
Hateful	0.685	0.1	0.774	0.685	0.727
Offensive	0.793	0.036	0.917	0.793	0.850
Clean	0.913	0.169	0.730	0.913	0.812
Overall	**0.797**	**0.101**	**0.807**	**0.797**	**0.796**

TABLE 8: Ternary Classification Confusion Matrix of the Test Set

Class	Classified as		
	Hateful	Offensive	Clean
Hateful	**459**	41	170
Offensive	83	**531**	56
Clean	51	7	**612**

Using the same sets of features we ran the classification. The classification results obtained are given in TABLE 9. Obviously, the accuracy dropped remarkably compared to the binary classification for the simple reason that hateful and offensive speeches are hard to distinguish from each other and In TABLE 10, and we observed that many of the tweets of the class *"hateful"* were misclassified as belonging to the class *"clean"*. This also explains the low recall of the class *"hateful"*, and the low precision of the class *"clean"*. This is because we can't distinguish some *"hateful"* and *"clean"* tweets. We can also see it from *"clean"* tweets misclassified as *"hateful"* is more than misclassified as *"offensive"* in TABLE 10.

The overall accuracy obtained reaches 78.4%. In addition, the same sets of features that performed well during the binary classification are ones that performed well during the ternary classification, for the same reasons mentioned above.

In particular, hate-related unigrams are very close to those offensive. As shown in Fig. 2, words highly related to hate are almost the same as those usually used to offend

TABLE 9: Ternary Classification Performances on the Validation Set

Class	TP Rate	FP Rate	Prec.	Recall	F1
Sentiment-based Features					
Hateful	0.337	0.205	0.451	0.337	0.386
Offensive	0.394	0.182	0.520	0.394	0.448
Clean	0.664	0.415	0.445	0.664	0.533
Overall	0.465	0.267	0.472	0.465	0.456
Semantic Features					
Hateful	0.233	0.232	0.334	0.233	0.274
Offensive	0.634	0.467	0.404	0.634	0.494
Clean	0.300	0.217	0.409	0.300	0.346
Overall	0.389	0.305	0.382	0.389	0.371
Unigram Features					
Hateful	0.636	0.073	0.813	0.636	0.714
Offensive	0.652	0.050	0.867	0.652	0.744
Clean	0.924	0.271	0.630	0.924	0.749
Overall	0.737	0.131	0.770	0.737	0.736
Pattern Features					
Hateful	0.328	0.114	0.590	0.328	0.422
Offensive	0.721	0.053	0.872	0.721	0.789
Clean	0.845	0.386	0.523	0.845	0.646
Overall	0.631	0.184	0.661	0.631	0.619
All features combined					
Hateful	0.699	0.104	0.770	0.699	0.732
Offensive	0.763	0.048	0.889	0.763	0.821
Clean	0.891	0.172	0.722	0.891	0.798
Overall	**0.784**	**0.108**	**0.793**	**0.784**	**0.784**

TABLE 10: Ternary Classification Confusion Matrix of the Validation Set

Class	Classified as		
	Hateful	Offensive	Clean
Hateful	**468**	48	154
Offensive	83	**511**	76
Clean	57	16	**597**

people, demean them or insult them (i.e., offensive speech). That being the case, even features qualified as *"Unigram"* present lower accuracy when we split the class *"offensive"* from the pervious subsection (binary classification) into two classes which are *"hateful"* and *"offensive"*.

Even though, performing such a comparison on patterns is quite challenging (since patterns do not show a direct relation to a specific class), we believe that the same kind of problem occurs and the patterns extracted from both classes are very close and related to one another.

5 Conclusion

In this work, we proposed a new method to detect hate speech in Twitter. Our proposed approach automatically detects hate speech patterns and most common unigrams and use these along with sentimental and semantic features to classify tweets into hateful, offensive and clean. Our proposed approach reaches an accuracy equal to 87.4% for the binary classification of tweets into offensive and non-offensive, and an accuracy equal to 78.4% for the ternary classification of tweets into, hateful, offensive and clean.

In a future work, we will try to build a richer dictionary of hate speech patterns that can be used, along with a unigram dictionary, to detect hateful and offensive online texts. We will make a quantitive study of the presence of hate speech among the different genders, age groups and regions, etc.

Acknowledgment

The research results have been achieved by "Cognitive Security: A New Approach to Securing Future Large Scale and Distributed Mobile Applications," the Commissioned Research of National Institute of Information and Communications Technology (NICT) , JAPAN.

References

[1] R.D. King and G.M. Sutton, "High Times for Hate Crime: Explaining the Temporal Clustering of Hate Motivated Offending", in *Criminology* pp. 871–894, 2013.

[2] Peter J. Breckheimer, "A Haven for Hate: The Foreign and Domestic Implications of Protecting Internet Hate Speech Under the First Amendment," in *South California Law Review,* vol. 75, no. 6, Sep. 2002.

[3] P. Burnap, and M. L. Williams, "Cyber Hate Speech on Twitter: An Application of Machine Classification and Statisitcal Modeling for Policy and Decistion Making," in *Policy and Internet* pp. 223–242, June 2015.

[4] A. H. Razavi, D. Inkpen, S. Uritsky, S. Matwin, "Offensive Language Detection Using Multi-level Classification," Advances in Artificial Intelligence, vol. 6085, pp. 16–27, Springer, Ottawa, Canada, June 2010.

[5] W. Warner and J. Hirschberg "Detecting hate speech on the World Wide Web," in *Proc. Second Workshop Language Social Media,* pp. 19–26, June 2012.

[6] M. Bouazizi and T. Ohtsuki, "A pattern-based approach for sarcasm detection on Twitter," *IEEE Access,* Vol. 4, pp. 5477–5488, 2016.

[7] D. Davidov, O. Tsur, and A. Rappoport, "Semi-supervised recognition of sarcastic sentences in Twitter and Amazon," In *Proc.14th Conf. on Computational Natural Language Learning*, pp. 107–116, July 2010.

[8] M. Bouazizi and T. Ohtsuki, "Sentiment Analysis: from Binary to Multi-Class Classification - A Pattern-Based Approach for Multi-Class Sentiment Analysis in Twitter," in *Proc. IEEE ICC*, pp. 1–6, May 2016.

[9] M. Bouazizi and T. Ohtsuki, "Sentiment analysis in Twitter: from classification to quantification of sentiments within tweets," *IEEE Globecom*, Dec. 2016, to be published.

[10] J. M. Soler, F. Cuartero, and M. Roblizo, "Twitter as a tool for predicting elections results," in *Proc. IEEE/ACM ASONAM,* pp. 1194–1200, Aug. 2012.

[11] S. Homoceanu, M. Loster, C. Lofi, and W-T. Balke, "Will I like it? Providing product overviews based on opinion excerpts," in *Proc. IEEE CEC,* pp. 26–33, Sept. 2011.

[12] U. R. Hodeghatta, "Sentiment analysis of Hollywood movies on Twitter," in *Proc. IEEE/ACM ASONAM,* pp. 1401–1404, Aug. 2013.

[13] Z. Zhao, P. Resnick and Q. Mei, "Enquiring Minds: Early Detection of Rumors in Social Media from Enquiry Posts," in *Proc. Int. Conf. World Wide Web,* pp. 1395–1405, May 2015.

[14] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate Speech Detection with Comment Embeddings," in *Proc. WWW'15 Companion,* pp. 29–30, May 2015.

[15] Njagi Dennis Gitari, Z. Zuping, Hanyurwimfura Damien, and Jun Long, "A Lexicon-based Approach for Hate Speech Detection," in , pp., Apr. 2015.

[16] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang, "Abusive Language Detection in Online User Content," in *Proc. WWW'16,* pp. 145–153, Apr. 2016.

[17] I. Kwok, and Y. Wang, "Locate the Hate - Detecting Tweets against Blacks," in *Proc. AAAI'13,* pp. 1621–1622, July 2013.

[18] Z. Waseem and D. Hovy, "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter," in *Proc. NAACL'16 Student Research Workshop,* pp. 88–93, June. 2016.

[19] T. Davidson, D. Warmsley, M. Macy, and I. Weber "Automated Hate Speech Detection and the Problem of Offensive Language," in *Proc. ICWSM'17,* May. 2017.

[20] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva, "Twitter part-of-speech tagging for all: Overcoming sparse and noisy data," in *Proc. Int. Conf. RANLP,* pp. 198–206, Sept. 2013

[21] S. Das and M. Chen, "Yahoo! for Amazon: Extracting market sentiment from stock message boards," in *Proc. 8th Asia Pacific Finance Assoc. Annu. Conf.*, vol. 35, pp. 43, July 2001.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten "The WEKA data mining software: An update',' *SIGKDD Explor. Newsk.,* vol. 11, no. 1, pp. 10–18, June 2009.
[23] G. I. Webb, "Decision Tree Grafting from the All-tests-but-one Partition," in *Proc. IJCAI'99,* pp. 702–707, CA, USA, Aug. 1999.