Detection of abusive messages in an on-line community

Etienne Papegnies, Vincent Labatut, Richard Dufour, Georges Linares

Detection of abusive messages in an on-line community

Etienne Papegnies[1,2] — **Vincent Labatut**[1] — **Richard Dufour**[1] — **Georges Linarès**[1]

[1] *LIA - University of Avignon (France)*
[2] *Nectar de Code, Barbentane (France)*

RÉSUMÉ. *La modération du contenu posté par les utilisateurs de communautés en ligne est majoritairement effectuée manuellement. De par la taille des données à traiter, les méthodes automatiques ont un intérêt certain pour réduire la charge de travail. Actuellement, l'industrie utilise des approches basiques à base de recherche de mots, comme par exemple le filtrage des messages contenant certains mots interdits. Nous nous intéressons dans cet article à une tâche de classification permettant de déterminer si un message est abusif ou non. Ceci est complexe, car les messages sont écrits dans un langage naturel non standardisé. Nous proposons ici une approche originale de modération automatique appliquée au français, s'appuyant à la fois sur des outils classiques et un nouveau descripteur fondé sur la modélisation du comportement utilisateur face à un message abusif. Les résultats obtenus lors de cette étude préliminaire montrent le potentiel de notre méthode, pour l'alerte automatique ou le support à la décision.*

ABSTRACT. *Moderating user content in online communities is mainly performed manually, and reducing the workload through automatic methods is of great interest. The industry mainly uses basic approaches such as bad words filtering. In this article, we consider the task of automatically determining whether a message is abusive or not. This task is complex, because messages are written in a non-standardized natural language. We propose an original automatic moderation method applied to French, which is based on both traditional tools and a newly proposed context-based feature relying on the modeling of user behavior when reacting to a message. The results obtained during this preliminary study show the potential of the proposed method, in a context of automatic processing or decision support.*

MOTS-CLÉS : *Détection d'abus, Communautés en ligne, Modération, Traitement du langage naturel*

KEYWORDS: *Abuse Detection, On-line Communities, Moderation, Natural Language Processing*

1. Introduction

The Internet has revolutionized the way we communicate, by connecting communities of users and allowing them to exchange through various means, such as text messages, videos, on-line voice calls, social networks, etc. In the context of textual communications, exchanges can be made through instant messaging (chat for example), on messages boards through Internet forums, by commenting news articles, and other ways.

As more and more people get access to the Internet, the size of these on-line communities is rapidly increasing. Maybe because of this success, on-line exchange platforms are inevitably confronted to *abusive behaviors* from users. The term *abusive* is originally defined for real-life situations, but most types of abusive behavior can be transposed to on-line exchanges, provided they do not require any physical interaction : verbal aggression, coercion or control, destruction, disruption, harassment, intimidation, isolation or emission of threats. On-line abuse is a widespread issue, impacting various types of communities, that may take different forms : unsolicited messages (*spam*), provocative messages (such as trolls), racist messages, etc.

Such abuses are generally perceived as harmful to the quality of the service provided by on-line platforms : abused users tend to leave them and look for more comfortable communication means. More importantly, some forms of abuse are also simply considered as illegal in certain countries, e.g. homophobic or racists comments in France. In addition, in the past few years, specific laws have been passed in many countries to explicitly hold the infrastructure maintainers responsible for the content hosted on their platforms, even if such content is submitted under the cover of anonymity. For these reasons, the administrators of on-line platforms started defining behaving rules, often enforced by so-called moderators. However, due to the increasing size of on-line communities, performing this task manually becomes more and more difficult and expensive.

In this paper, we try to partially automate this task. We focus on on-line abuses appearing in text messaging communications, and consider the detection of abusive messages as a supervised classification problem. We evaluate the use of different classic features extracted from textual content (average word length, number of bad words, message length...), and also propose a new feature, based on the detection of significant changes in user behavior. Our assumption is that, when confronted to an abusive situation, a user does not behave like he usually does. To model the user's regular behavior, we build a word n-gram Markov chain using messages known to have been published in a non-abusive situation. To assess and compare the discriminant power of all these features, we use textual messages obtained from the official communication platform of the massively multiplayer on-line game (MMO) SpaceOrigin. We consider two communication means : in-game messages, which could be compared to e-mails, and chat messages, *e.g.* instant messages.

This paper is organized as follows. In Section 2, we quickly review the main works related to abuse detection and automatic moderation. In Section 3, we describe the fea-

tures we use to automatically detect abusive messages, including both classic features and the one we propose, based on the modeling of the user's behavior. In Section 4, we present our dataset, describe the experimental protocol we followed, and discuss the results we obtained. Finally, we summarize the main points of our work in Section 5, and show how it could be extended.

2. Related Work

Several solutions have been proposed to automatically detect abusive messages in on-line communities. While most of the works are evaluated on English datasets, a vast majority of the methods used are language- and community-independent, and can therefore be applied to any on-line text messaging community, which makes them relevant to us. In this review, we first focus on the preprocessing step, because the considered medium exhibits particular linguistic characteristics (ungrammaticality, community-specific linguistic traits, misspelling...), and requires a specific preparation in order not to impact negatively the performance of the methods applied subsequently (Subsection 2.1). Then, we distinguish two types of approaches, depending on the type of information they use to detect abusive messages : content- vs. context-based methods. In the former case, the classification is performed based only on the textual content of the considered message, whereas in the latter, it takes advantage of additional data : structure of the conversation, user metadata, etc. (Subsection 2.2). The best performance is globally reached with approaches taking advantage of the information described in this review (preprocessing step, content messages, and contextual information).

2.1. *Preprocessing step*

Preprocessing is usually a simple but important step when dealing with messages posted on-line. The Internet medium introduces specific difficulties : disregard of syntax and grammar rules, out-of-vocabulary words, heavy use of abbreviations, typos, presence of URLs... The *Denoising* and *Deobfuscation* tasks both consist in mapping an unknown word back into a dictionary of known words. In the first case, a word is out of the vocabulary for unintentional reasons such as typos, e.g. "I uess so" for "I guess so". In the second case, it is due to a more deliberate attempt to conceal the word, e.g. "F8ck3r" for "fucker". Globally, mapping the word back into the dictionary increases the performance of probabilistic learning methods, since these methods need the cleanest possible text to achieve their maximum performance.

In (Sood *et al.*, 2012), the Levenshtein distance (also called Edit Distance) is used in an attempt to match unknown words against words in a crowd-sourced list of manually annotated messages containing profanity. The Levenshtein distance (Levenshtein, 1966) measures the number of insertions, deletions and replacements needed to convert a string into another (*e.g.* The Levenshtein distance between "@ss" and "ass" is 1). Computing the Levenshtein distance between two words of length n

and m is computationally expensive : the runtime is $\mathcal{O}(n \cdot m)$, and for this specific task each word has to be matched against each word in the dictionary, which is huge. We base some of our features on the Levenshtein Distance.

Other works, such as (Brill et Moore, 2000), proposed to improve on simple Levenshtein distance based denoising (for the purpose of spell checking) by considering the context of the string edition in a word (where the edited character is) and in a sentence (does the edit maps the word into a high n-gram probability ?). However, this approach is based on the assumption that out-of-vocabulary words are mainly due to unintentional spelling mistakes and is therefore not applicable to deobfuscation.

In (Lee et Ng, 2005), the authors use a Hidden Markov Model customized with dictionary and context awareness for the purpose of deobfuscating spam messages. Those types of messages often include deliberately misspelled words in an attempt to bypass filters. Their model showed impressive results with the ability to correct both unintentional misspellings as well as deliberate obfuscation using weird characters or digits and could even map segmented words back into complete words. (ie : "ree movee" -> remove). This process is nonetheless also computationally intensive and it requires curated data such as examples of word obfuscation in the given language, therefore we will not use it in this paper, but will come back to it in future works.

Preprocessing is an important step in an automated abuse detection framework, but it should be applied with caution. The goal of preprocessing is to increase the amount of relevant information in a message, but it can have the opposite effect. For example, the tendency of a user to misspell words can be viewed as an important feature to describe this user, but blind preprocessing would hide that. In order to prevent this, in this paper we use the preprocessing only when it is consistent with the considered feature, and we use the raw text otherwise.

2.2. *Text messaging classification*

In this section we review existing classification approaches that consider the content of the messages to detect abusive messages, and then the context of the exchanged messages.

Content-based approaches (Spertus, 1997) was one of the first work to automate detection of hostility in messages. While hostility does not imply abuse, abuse often contains hostility. The paper defined a number of rules to identify certain characteristics of the messages, such as Imperative Statement, Profanity, Condescension, Insult, Politeness and Praise. A Decision Tree classifier was then used to categorize messages into Hostile and Non-hostile classes. The setup showed good results but was limited when dealing with sarcasm, noise or innuendo. This approach is interesting but highly tuned for the grammar rules, semantics and idioms of a specific language.

In (Chen *et al.*, 2012), the authors note that the mere presence of an offensive word in a message is not a strong enough indication that the message itself is offensive, ie

"You are stupid." is way more offensive than "This is stupid.". This is an important observation and the authors showed that the lack of context can be mitigated by looking at word n-grams instead of unigrams.

Another work by (Dinakar *et al.*, 2011) used features computed from tf-idf weights, a list of words reflecting negative sentiment and widely used sentences containing verbal abuse to detect cyberbullying in comments associated with Youtube videos. Their model showed good results for instances of verbal abuse and profanity, but was limited with regard to sarcasm and euphemism.

Finally, the work by (Chavan et Shylaja, 2015) reviewed machine learning approaches for the classification of aggressive messages in On-line Social Networks, described a full pipeline for achieving classification of raw comments and introduced two new features : Pronoun Occurrence and Skip-Gram features. They allow for the detection of targeted phrases such as "He sucks" or "You can go die", and for the identification of long distance relationship between words, respectively.

Content-based text classification performs relatively well as a starting point. The computational cost to implement these approaches is usually reasonable. Nonetheless, these approaches have severe limitations. For instance, abuse can cross message boundaries, and therefore a message can be abusive only because of the presence of an earlier message. In other cases, messages can be abusive because they reference a shared history between two users. Therefore, studying the context of a message, its recipient, and its author might also be important.

Context-based approaches To go beyond the limitations of content-based approaches, authors proposed to take into account the context of messages, usually in addition to the textual content itself.

Some works explore the use of the content neighboring the targeted message. In (Yin *et al.*, 2009), authors used a supervised classifier working on n-gram features, sentence-level regular expression patterns and the features derived from the neighboring phrases of a given message to detect abuse on the Web. Their approach focused on detecting derivations in the context of a discussion around a given topic and their context features significantly improved the performance of their system. For this reason, we want to adopt a similar approach for our own method, but by focusing on derivations of users themselves from their usual patterns.

Other works have focused on modeling the users' behaviors by introducing higher-level features than the textual context. A comprehensive study of antisocial behavior in on-line discussion communities has been proposed by (Cheng *et al.*, 2015). Their exploratory work reinforced the weight of classic features used to classify messages such as misspellings and length of words, and provided insight into the devolution of users over time in a community, regarding both the quality of their contributions and their reactions towards other members of the community. This analysis is a good step towards modeling abusive behavior. One of the essential results of the analysis is that instances of antisocial messages tend to generate a bigger response from the commu-

nity compared to benign messages. The number of respondents to a given message is a feature we use in this work.

A selection of contextual features aiming at detecting abuse in on-line game communities has also been investigated in (Balci et Salah, 2015). These features form a model of the users of the game by including information such as their gender, number of friends, investment in the platform, avatars, and general rankings. The goal was to help human moderators dealing with abuse reports, and the approach yielded sufficiently good results to achieve this goal. The work was however limited in applicability because of the specifics of that given community, and of the raw amount of data needed to perform similar experiments in other games.

When quantifying controversy in social media (Garimella *et al.*, 2016), the structure of the community network is exploited to identify topics that are likely to trigger strong disagreements between users. The approach relies on a network whose nodes are Twitter users and links represent communications between them. It is interesting, however hardly applicable to our case, since we cannot infer the exact network structure from our dataset unless we restrict the network to private conversations between two users. As per the game's terms of use, the content of these conversations is only available to moderators when an abuse case is reported.

3. Features for abusive message detection

In this work, we propose to use a selection of features to characterize an abusive message, including one newly proposed by us. In this section, we describe according to their type : low-level morphological features (Subsection 3.1), higher level language features (Subsection 3.2), and user-related features (Subsection 3.3).

3.1. *Morphological Features*

Message length. This feature corresponds to the length of a message, expressed in number of characters, before any pre-processing such as normalization. The intuition is that abusive messages are usually either kept short (*e.g.* "Go die."), or extremely long, which is symptomatic of a massive copy/paste.

We also consider the length expressed in terms of words. In conjunction with the character count, it can match certain overly emphasized messages (e.g. : "Shuuuuuuuuuuuuuuuuuuuuuuuut uuuuuuuuuuuuuuuuuuuup !").

Character classes. We split characters into 5 classes : *Letters*, *Digits*, *Punctuation*, *Spaces* and *Others*. We keep track of the number of characters in those classes and the ratio of those classes in the message. This is done before any preprocessing on the raw message.

We selected these features based on several observations we made on the abusive messages. First, some of them have an unusual number of charac-

ters in the "Other" class, e.g. : "8==================D". Second, some use digits to obfuscate their meaning, in violation of the game rules. For instance, "01000111011011110010000001100100011010010110010100101110" and "476F206469652E" are obfuscated versions of the text "Go die." : the first one is coded in binary, and the second in hexadecimal.

Abusive users also commonly "yell" insults using capital letters, which is why we keep track of both the number of caps and the corresponding ratio of caps in the message.

Compression ratio. This feature is defined as the ratio of the length of the compressed message to that of the original message, both expressed in characters. It is based on the observation that certain users tend to repeat *exactly* and many times the same text in their abusive messages. We use the Lempel–Ziv–Welch (LZW) compression algorithm (Batista et Meira, 2004), and the features therefore directly relates to the number of copy/pastes made in the same message.

Unique characters. By counting the number of distinct characters in the message, we can detect the use of binary or hexadecimal obfuscation, as well as the overuse of punctuation. For instance, for the message "01000111011011110010000001100100011010010110010100101110", this feature has only a value of 2. This value is also computed before any preprocessing.

Collapsed characters. This feature is processed after normalization, and more particularly after the removal of consecutive characters. When three or more identical consecutive characters are found in the message, they are collapsed down to two characters. For instance, "looooooool" would be collapsed to "lool". The feature is the difference between the length of the original message and that of the collapsed one. This preprocessing step has been widely used in the classification of Tweets, for instance in (Roy *et al.*, 2013).

3.2. *Language Features*

Word length. This feature is a component of the Automated Readability Index (ARI) (Senter et Smith, 1967). It measures how proficient someone is at creating text documents. While abusive messages are sometimes surprisingly well written, this remains rare.

Unique words. We consider the number of unique words in the message. The intuition is that messages with more words are likely to be more constructive in terms of their content.

tf-idf **Scores.** Those features are the sums of the tf-idf scores of each individual word in the message. We use two distinct scores : the so-called *non-abuse score* is processed relatively to the non-abuse class, whereas the *abuse score* is processed over the abuse class.

If the considered word is unknown, in the sense it does not appear in the training set, we process an approximation of these scores. For this purpose, we first search for known words located within a given Levenshtein distance from the word of interest, and average their own scores.

Computing the full Levenshtein Distance between two words is computationally expensive. For this reason, solutions proposed in this paper never compute the full Levenshtein Distance between two words. Instead, a specialized tree-based index data-structure with a search function that yields all words in the tree within a given maximum edit distance is used. We use 2 as the maximum edit distance. This is considerably faster because branches of the tree are pruned as soon as we reach a state where the maximum edit distance is exceeded. It is still the most computationally intensive operation in our experiments.

Sentiment Scores. These features are numeric values derived from the number of words in the message that have a sentiment weight. It is based on the sentiment corpus presented in (Chen et Skiena, 2014), which was automatically generated for the French language and we augmented it manually by selecting words from a large list of insults.

Bad Words. This feature corresponds to the number of words in the message that appear in a manually crafted list of bad words. The list of words was created from a list of insults in French and then augmented with common Internet shorthand and symbolism. (*e.g* : 'connard', 'fdp', 'stfu' but also '..l..', '8==D' etc)

When we cannot match the considered word to any of the bad words in our list, we try to perform a fuzzy match using the Levenshtein distance. This is supposed to allow us picking up some obfuscated bad words.

We also perform the same tests using the collapsed version of the message (as described for the *Collapsed characters* feature).

3.3. *User Behavior Features*

Number of respondents. Given a fixed size window after a target message, this feature tracks how many distinct users replied to this message. This feature is relevant because it has been shown that abusive comments tend to trigger a higher number of responses from the community (Cheng *et al.*, 2015).

Probability of n-gram emission. We investigate if the abusiveness of a user's message can be detected by considering the effect it has on the *other* users participating to the same conversation. To do this, we compare the writing behavior of these other users before and after the apparition of the targeted message. We model this behavior through a user-specific Markov chain, which we use to compute how likely some text is to have been generated by the considered user.

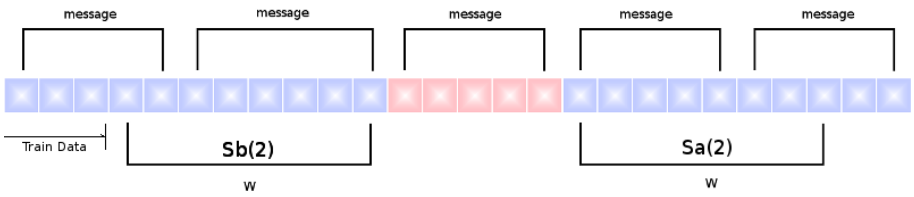

Figure 1 : A sequence of messages broken down into n-grams. Each square represents an n-gram : red for the targeted message from user1, blue for the surrounding messages written by user2.

We first sort the messages in chronological order. For each participant other than the user who wrote the targeted message, we build a word n-gram Markov chain using all but the last W n-grams in the messages posted before the target message.

The Markov chain is a convenient way to store the transition probabilities for all couples of n-grams in a participant's history. We compute two values : the average emission probabilities of the n-grams in the W-length window before and after the target message, as represented in Figure 1.

Let $P_{i,i+1}$ be the emission probability of a transition between the i^{th} and $i+1^{th}$ n-grams in the window of length W. Then we define the average emission probability S over the set of W n-grams as :

$$S = \frac{\sum_{i=0}^{W-1} P_{i,i+1}}{W} \quad [1]$$

We note $S_B(u)$ and $S_A(u)$, respectively, the average probabilities processed before and after the targeted message, for the same user u. The final score $S(u)$ for user u corresponds to their difference :

$$S(u) = S_A(u) - S_B(u) \quad [2]$$

This score is processed for every respondent to a message in a window of fixed length after the message. We then compute our feature by averaging this score over all the responding users.

4. Experiments

In this section we will describe the data used in our experiments (Subsection 4.1), the experimental protocol we applied to the data (Subsection 4.2), and finally discuss the results obtained with our proposed system (Subsection 4.3).

4.1. *Dataset*

We have access to a database of user in-game interactions for the SpaceOrigin MMO. This user-generated content was manually verified, in the sense the game users had the ability to flag parts of the content as inappropriate (*i.e. abusive*). There are many types of reportable contents, but, in this paper, we focus on two of them : ingame-messages and chat-messages, collectively referred to as messages.

Ingame-messages are on-line messages with a clearly defined reach. They are the equivalent of e-mails and can be sent to specific users or groups of users. They can be edited *a posteriori* by moderators when an abuse case is reported. The reach of *chat-messages* is loosely defined because it is limited to users currently active in a chatroom. However, there is no way to determine which user has actually seen a specific chat-message based on the available data. Users are fed recent scroll history for a chatroom upon joining, but it is not possible to reliably determine who has joined when from the chat logs. Chat-messages cannot be edited by moderators afterwards.

The database contains $474,599$ in-game messages and $3,554,744$ chat-messages. We extract 779 *abusive* messages (0.02%), which constitute what we call the *Positive Class* (Class 1) of messages. These messages were first flagged by the game users using a built-in reporting tool, and then confirmed as being abuse cases by the game community moderators. Of these 779 abusive messages, 14% are ingame-messages, and the rest are chat-messages. We then extract *non-abuse* messages from the database, in order to constitute the so-called *Negative Class* (Class 0). They are chosen at random from a pool of messages written by users which have *never* been flagged by a confirmed abuse report. For each message, we also gather context data : a window of messages occurring before and after each message.

We run the experiments with different versions of the corpus : in-game messages only (*iM*), chat-messages only (*cM*) and messages of both types combined (*iM+cM*). Since the non-abuse class of message is chosen automatically at random from the database, we also experiment with different number of messages in that class. We run experiments with *balanced* classes (noted *B*), where the number of "abuse" messages equals that of "non-abuse" ones ; and with *unbalanced* classes (noted *U*) where the number of "non-abuse" messages is twice that of "abuse" ones. Those numbers are listed in Table 1.

Experiment	**Abusive Messages**	**Non-Abusive Messages**
iM+cM U	779	1558
iM+cM B	779	779
iM U	111	222
iM B	111	111
cM U	668	1336
cM B	668	668

Tableau 1 : Corpus sizes depending on the considered experimental setup.

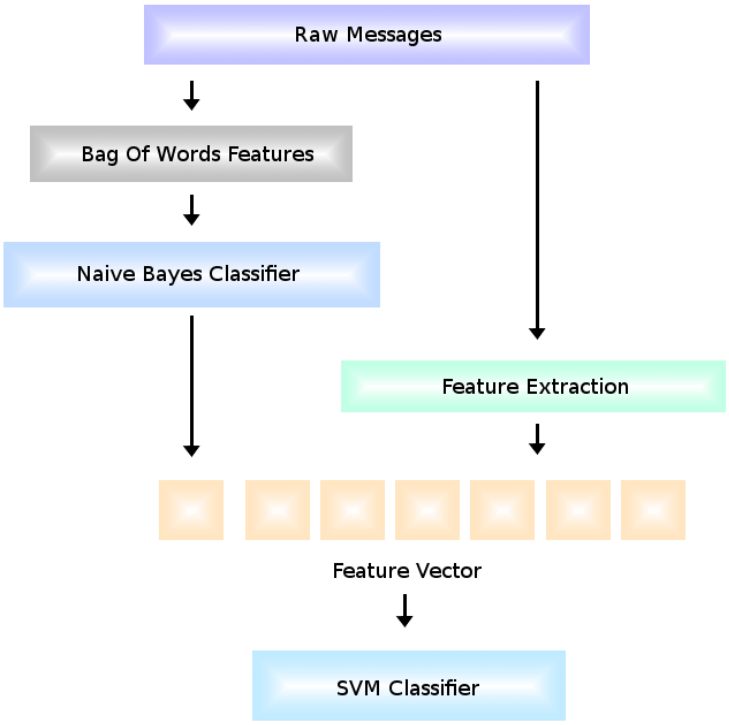

Figure 2 : The full experimental setup

4.2. *Experimental Setup*

Our experiment is designed as a multi-stage classifier pipeline, as described in Figure 2 (each box corresponds to a stage). The first stage (*Raw Messages*) consists in building the corpus. Messages from both the Abuse and Non-abuse classes are extracted from the database as explained in Subsection 4.1. The corpus is then split into a *Train* set containing 70% of the messages, and a *Test* set containing the remaining 30%.

In the second stage (*Bag of Words Features*), messages are normalized, tokenized and converted into Bags of Words (BoW). In the third stage (*Naive Bayes Classifier*), the BoW representations of the *Train* messages are used to build a Naive Bayes classifier. This classifier is then used to generate predictions for the class of the *Test* messages. The output of this classifier is fetched as a feature to the SVM classifier, described below.

In the fourth stage (*Feature Extraction*), we extract the features described in Section 3 from the messages. As explained also in Section 3, some of these features are derived from the messages before any normalization or preprocessing, whereas some

features require a specific preprocessing. We then use another classifier, a Support Vector Machine (SVM). We could directly feed the BoW to the SVM. However, given the size of the vocabulary in our experiments, this would lead to a dimensionality issue, with a number of features greatly exceeding the number of instances in the corpus. Therefore, we prefer to consider the decision from the Naive Bayes classifier (third stage) as an additional feature given to the SVM. All the features, including the Naive Bayes decision, are gathered into an array.

The fifth stage (*SVM Classifier*) is the final classification : the feature arrays from the Train set are fetched to an SVM classifier, and the resulting model is then used to generate class predictions for the Test set.

4.3. *Results*

We evaluate the performance of our proposed abusive message detection system in terms of the traditional Recall, Precision and F-Measure, defined below :

$$Recall = \frac{TP}{TP + FN} \quad [3]$$

$$Precision = \frac{TP}{TP + FP} \quad [4]$$

$$Fmeasure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad [5]$$

where TP, FP and FN are the numbers of True Positives (number of abusive messages correctly classified as abusive), False Positives (number of non-abusive messages incorrectly classified as abusive), and False Negatives (number of abusive messages incorrectly classified as non-abusive), respectively.

Given the relatively low number of abusive samples of the targeted corpus, the whole dataset was split into 10 parts and every result given in this section is the average value over a 10-fold cross validation. We use the notations defined in Table 1, i.e. : *U* for unbalanced classes, *B* for balanced ones, *iM* for ingame-messages only, *cM* for chat-messages only and *iM+cM* for all messages at once.

Table 2 and 3 present the results obtained when using only our content-based features, with the Naive Bayes classifier and the SVM, respectively. Table 4 displays the results obtained with the SVM when considering both content and context-based features. We could not process all the possible cases because of the high computational cost of this approach.

The behavior observed in Tables 2 and 3 is quite similar. The results are clearly better for balanced classes than for unbalanced ones, be it for *cM*, *iM* or *cM+iM*, with both Naive Bayes and the SVM. It is particularly strong for the precision obtained

Data	Recall	Precision	F-Measure
iM+cMU	0.656	0.650	0.653
iM+cMB	0.674	0.761	0.715
iMU	0.585	0.433	0.498
iMB	0.718	0.739	0.724
cMU	0.590	0.737	0.656
cMB	0.645	0.785	0.708

Tableau 2 : Classification results with a Naive Bayes classifier using BoW features (Baseline).

on *iM*. So our first observation is that the performances are clearly better when we process a balanced dataset, even if this amounts to using less data. In the rest of our discussion, we consequently focus on the results obtained on balanced classes.

Data	Recall	Precision	F-Measure
iM+cMU	0.696	0.662	0.679
iM+cMB	0.731	0.765	0.748
iMU	0.625	0.458	0.529
iMB	0.697	0.718	0.707
cMU	0.658	0.747	0.700
cMB	0.705	0.793	0.747

Tableau 3 : Classification results with a SVM classifier using content features only.

We now focus on the effect of the message type (only on balanced classes). In the case of Naive Bayes, the Recall is better for ingame-messages, whereas the precision is better for chat-message. For the SVM, we observe that the Recall is about the same, whereas the Precision is better for chat-messages. So, the type of message does appear to affect the performance. When combining both types of messages (*iM+cM*), the resulting performance is similar to that of the chat-messages only so it seems like there is not downside to grouping the types. Finally, the disappointment comes from the performances obtained when incorporating our context-based features (Table 4). Indeed, no improvement in terms of performance has been observed. It would seem that the features linked to the context, in the way that we made it, do not incorporate information in addition to the content ones.

Data	Recall	Precision	F-Measure
iM+cMU	0.701	0.663	0.681

Tableau 4 : Classification results using both content and context features.

Let us now compare both classifiers. When considering the Recall, the SVM is always at least as good, and generally better, as Naive Bayes. The difference is es-

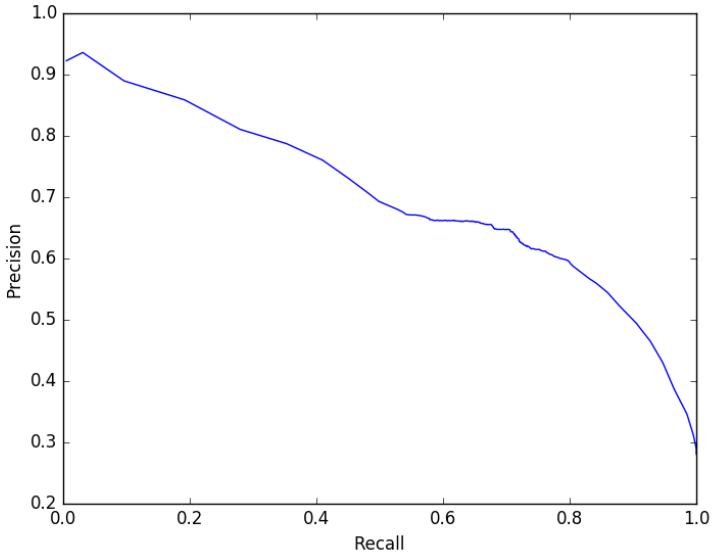

Figure 3 : Precision and Recall by varying a threshold on the posterior probability of the SVM classifier on the abusive message class.

pecially strong for *cM* (0.705 vs. 0.645). For the Precision too, the SVM classifier dominates, but with a smaller margin (0.765 vs. 0.761 on the *iM*+*cM* data).

Previous experiments showed that, even if acceptable results could be obtained with our abusive message detection system (best F-measure of more than 70%), performance is still not good enough to be directly used as a fully automatic system that replaces human moderation. Nonetheless, we think that this system could be useful to help moderators to focus on messages considered has potentially abusive, instead of having to analyze all messages. In this context, Figure 3 presents the curve obtained (in terms of precision and recall) using our proposed combined content-context approach, by varying a threshold on the posterior probability of the SVM classifier on the abusive message class. For example, a threshold of 0.8 means that if the classifier associates a posterior probability greater than 0.8 to a message for the abusive class, then the abusive class is chosen (otherwise, the message is considered as a non-abusive one). This posterior probability is obtained using SKlearn's Platt Scalling implementation (Platt *et al.*, 1999).

As expected, we can see that the posterior probability is an interesting information to take into consideration depending on the targeted application. Indeed, if the targeted application requires to be very precise in the decision to take (*i.e.* be sure that the message is abusive), such as a fully automatic system, then a higher threshold should

be used, with a loss in terms of retrieved abusive messages (*i.e.* lower recall). On the contrary, if the targeted application wants to recover as many abusive messages as possible, such as a moderator helper, then a lower threshold should be used, which will have the effect of having a higher recall (more abusive messages are retrieved) associated to a lower precision (more non-abusive messages will be wrongly considered by the automatic system as abusive).

5. Conclusion

In this paper we developed a system to classify abusive messages from an on-line community. It is developed on top of a first-stage Naive Bayes classifier and relies on multiple types of features : morphological features, language features and context features, that have proven useful in previous research. We added several features that we derived directly from observations of our corpus, and developed a context feature that aims to capture abnormal reactions from users caused by an abusive message.

This preliminary work shows that abusive messages, in their particular form, have characteristics that can be caught by an automatic system, our proposed system achieving a recall of 70% with a precision of 66% on our dataset, with a false positive rate 10%. While the performance of the system is not yet good enough to be deployed as an automated moderation tool, this can already help moderators to focus on messages being identified as abusive, with then a manual verification. However, because some features used in the system are specific to the community in which it is meant to operate, care must be taken when adapting the system to work on a different dataset.

We now plan to pursue our work in several ways. First, we will experiment with computationally more demanding preprocessing methods, and evaluate their contribution to the classifier performance. One such approach is the HMM-based preprocessor. (Lee et Ng, 2005) Second, we want to derive variants of our Markov chain-based feature, and assess which one is the most appropriate in our situation. More generally, we want to propose other context-based features, especially ones based on the network of user interactions.

6. Bibliographie

Balci K., Salah A. A., « Automatic analysis and identification of verbal aggression and abusive behaviors for online social games », *Computers in Human Behavior*, vol. 53, p. 517-526, 2015.

Batista L. V., Meira M. M., « Texture classification using the Lempel-Ziv-Welch algorithm », *Brazilian Symposium on Artificial Intelligence*, p. 444-453, 2004.

Brill E., Moore R. C., « An improved error model for noisy channel spelling correction », *38th Annual Meeting on Association for Computational Linguistics*, p. 286-293, 2000.

Chavan V. S., Shylaja S. S., « Machine learning approach for detection of cyber-aggressive comments by peers on social media network », *International Conference on Advances in Computing, Communications and Informatics*, p. 2354-2358, 2015.

Chen Y., Skiena S., « Building Sentiment Lexicons for All Major Languages », *52nd Annual Meeting of the Association for Computational Linguistics*, p. 383-389, 2014.

Chen Y., Zhou Y., Zhu S., Xu H., « Detecting offensive language in social media to protect adolescent online safety », *International Conference on Privacy, Security, Risk and Trust and International Confernece on Social Computing*, p. 71-80, 2012.

Cheng J., Danescu-Niculescu-Mizil C., Leskovec J., « Antisocial behavior in online discussion communities », *preprint arXiv :1504.00680*, 2015.

Dinakar K., Reichart R., Lieberman H., « Modeling the detection of Textual Cyberbullying. », *The Social Mobile Web*, vol. 11, p. 02, 2011.

Garimella K., De Francisci Morales G., Gionis A., Mathioudakis M., « Quantifying controversy in social media », *9th ACM International Conference on Web Search and Data Mining*, p. 33-42, 2016.

Lee H., Ng A. Y., « Spam Deobfuscation using a Hidden Markov Model », *2nd Conference on Email and Anti-Spam*, 2005.

Levenshtein V. I., « Binary codes capable of correcting deletions, insertions and reversals », *Soviet Physics Doklady*, vol. 10, p. 707, 1966.

Platt J. *et al.*, « Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods », *Advances in large margin classifiers*, vol. 10, n° 3, p. 61-74, 1999.

Roy S., Dhar S., Bhattacharjee S., Das A., « A lexicon based algorithm for noisy text normalization as pre processing for sentiment analysis », *International Journal of Research in Engineering and Technology*, vol. 2, p. 67-70, 2013.

Senter R. J., Smith E. A., Automated readability index, Technical Report n° AMRL-TR-6620, Wright-Patterson Air Force Base, 1967.

Sood S. O., Antin J., Churchill E. F., « Using Crowdsourcing to Improve Profanity Detection. », *AAAI Spring Symposium : Wisdom of the Crowd*, 2012.

Spertus E., « Smokey : Automatic recognition of hostile messages », *14th National Conference on Artificial Intelligence and 9th Conference on Innovative Applications of Artificial Intelligence*, p. 1058-1065, 1997.

Yin D., Xue Z., Hong L., Davison B. D., Kontostathis A., Edwards L., « Detection of harassment on Web 2.0 », *WWW Workshop : Content Analysis in the WEB 2.0*, p. 1-7, 2009.