Approche multimodale par plongements de texte et de graphes pour la détection de messages abusifs

Noé Cecillon, Richard Dufour, Vincent Labatut

Approche multimodale par plongements de texte et de graphes pour la détection de messages abusifs

Noé Cécillon* — **Richard Dufour¤** — **Vincent Labatut***

* *Laboratoire Informatique d'Avignon - LIA EA 4128, Avignon Université, France*
¤ *Laboratoire des Sciences Numériques de Nantes (LS2N), Equipe TALN, Nantes Université, France*

RÉSUMÉ. *Les comportements abusifs sont de plus en plus courants sur les plateformes d'échange en ligne, ce qui oblige leurs propriétaires à trouver de nouvelles solutions de modération. Des méthodes automatiques utilisant le contenu textuel ou la structure de la conversation ont alors été proposées. Par ailleurs, de nouvelles méthodes génériques de représentation de texte et de graphes sont apparues, fondées sur la notion de plongement (ou embedding), tirant notamment profit de l'augmentation drastique des données et de la puissance de calcul disponibles. Dans ce travail, nous évaluons cinq méthodes de plongement lexical et quatre méthodes de plongement de graphes sur une tâche de détection de messages abusifs. Ces deux types d'approches ne s'appuyant pas sur les mêmes informations, nous étudions également différentes combinaisons de ces plongements. Nous obtenons des résultats comparables voire supérieurs pour le texte, à des approches classiques avec sélection manuelle de caractéristiques. La combinaison des plongements de texte et de graphes apporte enfin une nette amélioration des performances.*

ABSTRACT. *Abusive behaviors are common on online social networks, forcing hosts of such platforms to find new moderation tools. Various methods based on the textual content or the structure of the conversation have thus emerged. Furthermore, new generic embedding methods have been proposed to represent text and graphs. Those approaches benefit the current exponential growth in available data and computing power. In this work, we evaluate five text embedding methods and four graph embedding methods on an abusive message detection task. These two types of embedding are not based on the same information, therefore we also study various combinations of these methods. Our results are comparable, and even better for text, to standard approaches based on feature engineering. The combination of text and graph embeddings finally brings a clear improvement in performance.*

MOTS-CLÉS : *Plongement de texte, Plongement de graphes, Détection d'abus.*

KEYWORDS: *Text embedding, Graph embedding, Abuse detection, Conversational graphs*

1re soumission à *TAL 62-2*, le 13/09/2021

1. Introduction

Au fil du temps, les réseaux sociaux et forums en ligne sont devenus des éléments incontournables de notre quotidien. Cependant, la popularité de ces plateformes les confronte à un nombre croissant d'utilisateurs au comportement abusif. Cela a éveillé l'attention des gouvernements qui demandent aux propriétaires de ces plateformes d'accroître leurs efforts pour lutter contre ce phénomène, à travers différentes lois et règlements. Selon la taille des communautés, la modération peut s'avérer très coûteuse car elle est généralement effectuée par un opérateur humain. De plus, pour que la modération soit efficace, il faut qu'elle soit rapide et disponible à tout instant. Ainsi, l'automatisation de cette tâche de modération suscite l'intérêt de nombreuses plateformes d'échange. Ce procédé reste toutefois complexe car la manière de modérer une plateforme en ligne est dépendante de sa taille, sa communauté, son pays, ou encore son contexte d'utilisation. Dans ce cadre, ce qui est considéré comme abusif dans un pays peut ne pas l'être dans un autre. Les premiers travaux dans ce domaine s'appuient sur le traitement automatique des langues (TAL). Ils utilisent des approches statistiques s'appuyant sur le contenu textuel des messages pour détecter différents types d'abus tels que des propos racistes (Waseem et Hovy, 2016), haineux (Djuric *et al.*, 2015 ; Salminen *et al.*, 2018 ; Warner et Hirschberg, 2012 ; Badjatiya *et al.*, 2017), offensants (Chen *et al.*, 2012 ; Nobata *et al.*, 2016 ; Xiang *et al.*, 2012 ; Okky Ibrohim et Budi, 2018) ou encore du cyber-harcèlement (Dinakar *et al.*, 2011). La principale limitation de ces approches réside dans leur sensibilité à certaines techniques utilisées par les utilisateurs malveillants pour camoufler leurs commentaires abusifs. Par exemple, les mots `c*n` et `c0n` sont facilement compréhensibles pour un humain mais peuvent ne pas être détectés par un système automatique. Les possibilités pour créer ce genre de termes étant infinies, il est très difficile de trouver des données d'entraînement, ou simplement définir une liste de règles, reflétant complètement ce phénomène. De plus, le vocabulaire employé peut être très riche, incluant des termes spécifiques à certains domaines ou communautés, ainsi qu'un registre de langue pouvant varier de très familier à soutenu. Ainsi, bien que ces outils issus du TAL, et en particulier les modèles statistiques, soient efficaces pour assister des modérateurs humains, leurs performances ne suffisent pas à automatiser complètement la modération de communautés en ligne.

Pour contourner les limites des approches s'appuyant sur le contenu textuel, des travaux ont proposé d'intégrer des informations sur la structure de la conversation et sur le comportement des utilisateurs (Balci et Salah, 2015 ; Chatzakou *et al.*, 2017 ; Dadvar *et al.*, 2013 ; Mishra *et al.*, 2018a ; Yin *et al.*, 2009). Papegnies *et al.* (2019) proposent d'utiliser des caractéristiques fondées sur des graphes conversationnels pour détecter des messages abusifs dans le cadre de conversations virtuelles d'un jeu vidéo en ligne. Ces graphes permettent de modéliser les interactions entre joueurs (c.-à-d. *qui parle à qui*) tout en ignorant le contenu textuel des messages échangés. Les auteurs ont sélectionné manuellement une grande quantité de mesures topologiques à calculer pour représenter la structure des graphes conversationnels. Ces mesures sont ensuite utilisées pour entraîner un classificateur à détecter des messages abusifs. Cette approche n'utilisant pas le texte s'abstrait alors des problèmes relatifs

aux méthodes de TAL évoqués précédemment. En revanche, les auteurs, ne connaissant pas à l'avance quelles mesures topologiques seront pertinentes pour caractériser des messages abusifs, ont fait le choix de considérer une grande quantité de mesures tout en sachant qu'une majorité d'entre elles n'auront pas de pouvoir discriminant dans le processus de classification. Cette sélection manuelle non-exhaustive de mesures topologiques implique donc un temps de calcul élevé dû à la grande quantité de valeurs à calculer et ne garantit pas que les mesures sélectionnées soient les plus pertinentes pour cette tâche. On peut même supposer que, dans certains cas, aucune mesure définie dans la littérature ne permette de capturer les informations utiles pour la tâche considérée. Ce sont des limitations importantes de cette méthode, et même, plus généralement, des méthodes s'appuyant sur l'ingénierie des caractéristiques, qu'elles soient par exemple issues de contenus textuels ou de représentations par graphes.

Les méthodes utilisant le texte et les graphes fonctionnent raisonnablement bien lorsqu'elles sont prises séparément. Cécillon *et al.* (2019) montrent que leur combinaison atteint de meilleurs résultats puisqu'il s'agit de représentations fondées sur des aspects complètement différents. Quelques travaux proposent d'utiliser le texte ainsi que des informations s'appuyant sur des graphes pour détecter du contenu abusif. Mishra *et al.* (2018a) proposent de construire un graphe représentant les utilisateurs et leurs interactions, et utilisent une méthode de plongement de graphes pour obtenir un profil de chaque utilisateur qui est ensuite combiné à des caractéristiques extraites par une méthode de TAL agissant uniquement sur le texte. Cécillon *et al.* (2019) combinent une méthode de TAL utilisant des caractéristiques pré-définies et des mesures topologiques représentant un graphe conversationnel pour détecter des messages abusifs.

Contrairement à l'extraction de caractéristiques définies *a priori*, les méthodes de plongement permettent de transformer automatiquement des objets en représentations vectorielles de taille fixée. Ces représentations conservent des propriétés des objets sans que celles-ci soient définies et peuvent être utilisées comme entrées d'un classificateur. Ainsi, les méthodes de plongement de graphes permettent de représenter un graphe, ou des éléments d'un graphe, sous la forme de vecteurs en préservant, au moins, une partie des propriétés topologiques du graphe. Ces représentations sont apprises automatiquement, elles ne nécessitent donc aucune sélection manuelle de caractéristiques à calculer. Elles sont ainsi beaucoup plus efficaces en temps de calcul que les méthodes évoquées ci-dessus. En revanche, on ne sait pas explicitement quelles informations sont capturées par les représentations apprises. Les méthodes de plongement lexical permettent, quant à elles, de représenter des mots par des vecteurs qui conservent leur sens. Ces approches peuvent permettre de surpasser les limitations des méthodes de TAL traditionnelles en proposant des représentations plus robustes. Elles ont été appliquées dans le cadre de la détection de contenu abusif en ligne pour détecter des propos racistes ou sexistes (Mishra *et al.*, 2018b) ainsi que des propos haineux ou offensants (Founta *et al.*, 2019 ; Padilla Montani et Schüller, 2018).

Dans ce travail, nous reprenons l'idée de combiner le contenu textuel et les interactions entre utilisateurs pour traiter une tâche de détection automatique de mes-

sages abusifs en ligne. Cependant, nous proposons d'utiliser des plongements pour apprendre automatiquement les représentations et éviter d'avoir à sélectionner manuellement un ensemble pré-établi de caractéristiques à calculer. Dans ce contexte, notre première contribution consiste à étudier l'efficacité de cinq approches de plongement lexical. Nous les évaluons et comparons sur une tâche de détection d'abus. Notre deuxième contribution est de les combiner avec les approches par plongement de graphes offrant les meilleures performances pour cette tâche selon Cécillon *et al.* (2021). Notre troisième contribution est de proposer trois variants de Graph2vec intégrant des informations relatives à notre tâche.

La suite de cet article est organisée comme suit. Tout d'abord, dans la section 2, nous faisons une revue de la littérature concernant les plongements lexicaux et les plongements de graphes. Puis, dans la section 3, nous détaillons le processus d'extraction des graphes, les méthodes de référence ainsi que les modèles de plongement que nous utilisons dans nos expérimentations. Nous présentons nos données, notre protocole expérimental, les méthodes de fusion ainsi que les résultats obtenus dans la section 4. Enfin, nous résumons notre travail et présentons quelques pistes de développement dans la section 5.

2. Représentations par plongement

Les plongements sont des approches qui consistent à transformer un objet en un vecteur numérique de taille fixée. Ces représentations par plongement peuvent ensuite être utilisées dans d'autres processus, comme par exemple en entrée d'un classificateur. Parmi ces approches, les plongements lexicaux permettent de représenter des mots, phrases, ou documents, sous la forme de vecteurs. Les plongements de graphes permettent, quant à eux, de transformer des nœuds, des liens ou même des graphes entiers en vecteurs.

2.1. *Plongement lexical*

On peut décomposer les approches de plongement lexical en deux catégories : celles qui génèrent une représentation vectorielle fixée par mot, qu'importe le contexte dans lequel il est utilisé, et celles dépendantes du contexte, qui génèrent des représentations différentes pour un même mot en fonction du contexte dans lequel il est utilisé.

2.1.1. *Représentation par un vecteur fixé*

Word2vec (Mikolov *et al.*, 2013) est une des approches les plus utilisées ces dernières années pour la représentation de mots. Word2vec permet d'apprendre des représentations tout en essayant de conserver une similarité sémantique et/ou syntaxique entre les mots. Cette méthode utilise un réseau de neurones à deux couches dans lequel la couche cachée constitue le plongement lexical. La couche de sortie permet d'implémenter une tâche de classification pour entraîner le modèle. Cette méthode propose

deux architectures symétriques. Le modèle SkipGram, qui prend un mot et cherche à prédire son contexte, c.-à-d. les quelques mots qui apparaissent avant et après lui dans le texte, et le modèle de sacs-de-mots continus (CBOW) qui prédit un mot à partir de son contexte.

Au contraire d'autres approches dont les modèles sont entraînés sur des fenêtres glissantes, comme Word2vec, *GloVe* (Pennington *et al.*, 2014) cherche à capturer directement des informations plus globales. Cette approche construit une matrice de cooccurences entre les mots du corpus et optimise la factorisation de ces informations pour générer des représentations vectorielles globales des mots. Puisqu'ils s'appuient sur les mots observés dans le corpus pour apprendre les représentations, Word2vec et GloVe ne sont pas capables de gérer les mots hors-vocabulaire. Pour la même raison, les mots peu courants ne sont pas toujours bien représentés par ces méthodes.

Parmi ces représentations fixées, nous pouvons aussi citer *fastText* (Bojanowski *et al.*, 2017). Cette approche est une extension de Word2vec qui, au lieu de traiter des mots, les décompose en n-grammes de caractères. Chaque mot se retrouve ainsi sous la forme d'un sac-de-mots de n-grammes de caractères (avec $3 \leq n \leq 6$ en général). Par exemple, avec $n = 3$, le mot *mode* serait représenté par (<mo,mod,ode,de>) où les signes « < » et « > » représentent le début et la fin du mot. Un modèle SkipGram apprend des représentations de tous ces sous-mots. Le vecteur associé à un mot est la somme de tous les n-grammes qui le composent. Ainsi, même les mots rares et hors-vocabulaire peuvent obtenir de bonnes représentations car il est très probable qu'au moins une partie de leurs n-grammes apparaissent dans d'autres mots.

La principale limitation des approches utilisant des représentations fixées est qu'elles génèrent une représentation unique par mot, qu'importe le contexte dans lequel il est utilisé. Ainsi le mot *droit* sera représenté de la même manière que l'on parle d'un « angle *droit* », de « *droit* international » ou de son « bras *droit* ».

2.1.2. *Représentation par un vecteur variant selon le contexte*

Les méthodes de plongement lexical dépendantes du contexte permettent de représenter des mots sous la forme d'un vecteur numérique tout en prenant en compte le fait qu'un mot peut avoir un sens différent selon le contexte dans lequel il est utilisé. Ainsi, avec ces méthodes, un même mot utilisé dans deux contextes différents obtiendra deux représentations vectorielles différentes.

Parmi ces approches, *ELMo* (Peters *et al.*, 2018) est une méthode qui traite une séquence de mots. Elle considère l'intégralité de cette séquence avant d'assigner un vecteur numérique à chaque mot qui la compose. ELMo utilise deux couches de réseaux de neurones récurrents à mémoire long et court terme (LSTM) pour créer ces représentations. Ces réseaux sont entraînés à prédire le mot suivant dans une séquence textuelle. ELMo utilise des réseaux qui sont bi-directionnels, ce qui permet au modèle d'avoir une information sur les mots qui se trouvent après, mais également sur ceux qui se trouvent avant. La représentation vectorielle associée à un mot est obtenue en

concaténant les états cachés des réseaux traitant la séquence de texte vers l'avant et vers l'arrière sur les deux couches.

Generative Pre-Training (Radford *et al.*, 2018) (GPT) est un modèle qui se fonde sur l'architecture *Transformer* proposée par Vaswani *et al.* (2017) et plus précisément sur la partie *décodeur*. Celle-ci est particulièrement adaptée pour entraîner un modèle de langage car elle est construite pour cacher les mots suivants dans une séquence. GPT empile douze couches de décodeurs qui sont entraînés à prédire le prochain mot d'une séquence. L'avantage de cette architecture, par rapport à des LSTM, est qu'elle permet de capter des relations entre mots à plus longue distance. En revanche, ce modèle est unidirectionnel. Il ne considère le contexte des mots que de gauche à droite.

BERT (Devlin *et al.*, 2019) propose d'associer différents concepts développés dans Transformer, ELMo et GPT. Comme GPT, BERT réutilise une partie de l'architecture Transformer mais, pour obtenir un modèle bi-directionnel, BERT utilise les *encodeurs* de Transformer et non pas les *décodeurs*. Pour entraîner ce modèle, BERT masque environ 15 % des mots de la séquence d'entrée, fait passer cette séquence à l'intérieur d'une pile d'encodeurs et prédit les mots masqués. Le fait de ne pas prédire le mot suivant mais plutôt de prédire des mots masqués à n'importe quel endroit de la séquence permet d'obtenir un modèle bi-directionnel. De plus, pour améliorer l'apprentissage des relations entre les phrases, le modèle est aussi entraîné à prédire si, pour deux phrases p_1 et p_2, p_2 est bien la phrase qui apparaît après p_1 ou alors si p_2 est simplement une phrase aléatoire.

RoBERTa (Liu *et al.*, 2019) est une amélioration de ce modèle qui propose notamment d'utiliser plus de données d'entraînement, d'augmenter la taille des lots de données et de ne traiter que la tâche de prédiction des mots masqués.

CamemBERT (Martin *et al.*, 2020b ; Martin *et al.*, 2020a) et *FlauBERT* (Le *et al.*, 2020b ; Le *et al.*, 2020a) sont deux modèles de plongement lexical en français directement adaptés de RoBERTa. Ils partagent donc beaucoup de similarités. Les différences entre ces deux approches portent principalement sur la manière de masquer les mots dans la tâche de prédiction des mots masqués et sur la quantité de données d'entraînement, FlauBERT utilisant moins de données.

Contrairement à la plupart des autres approches, *Flair* (Akbik *et al.*, 2018) propose des représentations qui sont construites au niveau des caractères et non pas des mots. Cette spécificité permet à Flair d'être performant pour traiter les mots rares, les mots mal orthographiés et de mieux modéliser des concepts linguistiques tels que les préfixes et suffixes. Cette architecture utilise un LSTM bi-directionnel qui traite des caractères. Pour chaque élément de la séquence à traiter, le réseau est entraîné à prédire le caractère suivant. Il possède donc deux représentations cachées pour chaque caractère de la séquence : l'une formée par le réseau traitant la séquence vers l'avant, et l'autre formée par le modèle la traitant vers l'arrière. La représentation finale d'un mot est obtenue à partir des représentations du modèle vers l'avant des caractères se trouvant avant la fin du mot ainsi que des représentations du modèle vers l'arrière des caractères se trouvant après le début du mot.

2.2. *Plongement de graphes*

Les plongements de graphes sont regroupés au sein d'une famille d'approches qui a pour but de représenter un graphe ou une partie de graphe sous la forme de vecteurs numériques de taille fixée tout en préservant au moins une partie de leurs propriétés structurelles (Cai *et al.*, 2018). Des graphes ayant certaines de ces propriétés en commun doivent avoir des représentations vectorielles assez proches (Yan *et al.*, 2007). En plus de ces propriétés structurelles, certaines méthodes sont capables d'intégrer des informations telles que le poids et la direction des liens ou des étiquettes de nœuds. Certaines méthodes de plongement de graphes sont adaptées directement de méthodes de plongement lexical. Les approches de plongement de graphes ne considèrent pas toutes les mêmes types d'objets. Dans cet article, nous nous concentrons sur les méthodes de plongement de *nœuds* et les méthodes de plongement de *graphes entiers*.

2.2.1. *Plongement de nœuds*

Le plongement de nœuds est la forme de plongement de graphe la plus courante dans la littérature. Ces méthodes prennent un graphe en entrée et produisent un vecteur de taille fixée pour chacun de ces nœuds. En suivant la classification proposée par Goyal et Ferrara (2018), on différencie trois types d'approches en fonction de la méthode utilisée pour créer les représentations : la factorisation de matrices, les réseaux de neurones et les marches aléatoires. Cette dernière catégorie peut également utiliser des réseaux de neurones, mais avec une manière différente d'échantillonner le graphe.

2.2.1.1. Factorisation de matrices

Un graphe peut aisément être représenté sous la forme d'une matrice, par exemple avec une matrice d'adjacence, de transition, ou laplacienne. Les premières études portant sur le plongement de nœuds ont proposé de décomposer ces matrices en produits de matrices plus petites et de dimensions fixées pour obtenir des représentations vectorielles des nœuds. C'est ce qu'on appelle la *factorisation de matrices*.

L'approche la plus directe est d'utiliser des méthodes existantes de réduction de dimension et de les appliquer à des matrices représentant des graphes. C'est ce que font Roweis et Saul (2000) avec leur méthode *Locally Linear Embedding*. Les auteurs considèrent que chaque nœud du graphe est une combinaison linéaire pondérée de ses voisins. En premier lieu, la méthode estime les poids qui permettent de reconstruire le plus précisément les caractéristiques originales d'un nœud à partir de ses voisins et utilise ensuite ces poids pour générer les représentations vectorielles des nœuds. Belkin et Niyogi (2002) proposent une méthode appelée *Laplacian Eigenmaps* qui apprend les représentations en calculant les vecteurs propres de la matrice laplacienne du graphe. Avec cette méthode, les nœuds fortement connectés sont proches dans l'espace de représentation. Ou *et al.* (2016) proposent une méthode spécialement adaptée pour traiter des graphes dirigés. Un inconvénient majeur de ces approches est leur grande complexité temporelle qui les rend difficiles à utiliser sur de très grands graphes, notamment des graphes pour des applications de la vie réelle. *Graph Factori-*

zation (Ahmed *et al.*, 2013) optimise le coût temporel de son apprentissage et permet de traiter des graphes avec plusieurs centaines de millions de nœuds.

2.2.1.2. Réseaux de neurones

Les approches neuronales ont été adaptées avec succès à de nombreux domaines, notamment le plongement de graphes. La méthode *Graph Convolutional Networks* proposée par Kipf et Welling (2017) propose un processus itératif dans lequel chaque itération capture le voisinage local d'un nœud et la répétition des itérations permet de capturer son voisinage global. À chaque itération, la nouvelle représentation d'un nœud est obtenue en combinant celles de ses voisins et la sienne à l'itération précédente.

Les réseaux antagonistes génératifs ont aussi été adaptés au plongement de nœuds. *GraphGAN*, proposé par Wang *et al.* (2018), utilise deux modèles. Premièrement, un générateur $G(v|v_c)$ fait une approximation de la connectivité réelle entre les nœuds v et v_c et détermine les nœuds qui ont la plus forte probabilité d'être connectés à v_c. Ensuite, un discriminateur $D(v, v_c)$ calcule la probabilité qu'un lien existe entre v et v_c. Le générateur tente de générer de fausses paires de nœuds connectés aussi indiscernables que possible. Le discriminateur quant à lui tente de discerner les paires réelles des fausses paires.

2.2.1.3. Marches aléatoires

Les marches aléatoires ont été adaptées au plongement de graphes pour essayer de reproduire certaines méthodes de plongement lexical telle que Word2vec. Elles permettent de représenter le graphe sous forme séquentielle, à l'instar des mots dans une phrase. Elles sont utilisées pour échantillonner le graphe et permettent d'avoir une représentation au moins partielle de sa structure. À partir d'un nœud de départ, les méthodes utilisant des marches aléatoires génèrent des séquences de nœuds en sélectionnant un voisin et en répétant cette procédure jusqu'à obtenir une séquence d'une certaine longueur. Un avantage des marches aléatoires est qu'elles peuvent permettre de traiter des graphes qui sont trop gros pour être explorés dans leur intégralité.

DeepWalk (Perozzi *et al.*, 2014) utilise ce procédé. La méthode commence par générer des séquences de nœuds grâce à des marches aléatoires. Elle utilise ensuite un modèle SkipGram développé pour Word2vec pour générer les représentations des nœuds. Le modèle considère un nœud en entrée et cherche à prédire son contexte, c.-à-d. ses voisins. Les mêmes auteurs proposent aussi une autre méthode de génération des marches aléatoires (Perozzi *et al.*, 2017). *Node2vec* (Grover et Leskovec, 2016) est une évolution de DeepWalk. La différence principale est que Node2vec utilise des marches aléatoires biaisées pour intégrer une notion plus flexible de voisinage. Ce biais permet de contrôler la manière dont les marches aléatoires vont explorer le graphe. Des marches aléatoires restant dans le voisinage proche du nœud vont permettre de capturer l'équivalence structurelle entre les nœuds alors que des marches aléatoires explorant des nœuds distants vont permettre de capturer une structure plus large. Ces marches aléatoires sont données à un modèle SkipGram pour apprendre les

plongements. Node2vec initialise aléatoirement ses représentations de nœuds, ce qui peut provoquer un blocage sur un optimum local durant le calcul des représentations. Chen *et al.* (2018) proposent une stratégie d'initialisation des poids optimisée pour éviter ce problème.

GraphWave (Donnat *et al.*, 2018) apprend des représentations robustes aux légères perturbations dans la topologie du graphe, et qui préservent le rôle structurel des nœuds. Cette méthode s'inspire d'un procédé physique réel qui consiste à propager de l'énergie, sous la forme d'ondes de chaleur, à travers le graphe, en partant du nœud que l'on souhaite représenter. Les auteurs considèrent que la structure du nœud et de son voisinage peuvent être modélisées selon la manière dont cette énergie se diffuse au sein du graphe. Formellement, c'est la distribution des coefficients d'ondes de chaleur qui est utilisée pour obtenir les représentations vectorielles des nœuds.

2.2.2. *Plongement de graphes entiers*

Les plongements de nœuds sont les méthodes les plus courantes dans la littérature. Cependant, certaines tâches requièrent des informations à un niveau de granularité plus élevé. Dans ce cas, on peut utiliser des méthodes de plongement de graphes entiers. Elles permettent de représenter l'intégralité d'un graphe sous la forme d'un seul vecteur numérique. Elles prennent une collection de graphes et produisent une représentation vectorielle pour chacun d'entre eux.

Graph2vec (Narayanan *et al.*, 2017) est conçu analogiquement aux méthodes de plongement lexical développées en TAL. Ces méthodes s'appuient sur le fait qu'un document est formé d'une séquence de mots. Les auteurs de Graph2vec considèrent qu'un graphe est formé d'une séquence de sous-graphes enracinés autour de chaque nœud, c.-à-d. le voisinage du nœud à un certain ordre. L'algorithme prend l'ensemble des graphes à représenter et génère leurs représentations en suivant deux étapes. D'abord, il extrait les sous-graphes enracinés autour de chaque nœud du graphe. Ensuite, cet ensemble de sous-graphes, considéré comme le vocabulaire du graphe, est donné à un modèle SkipGram qui permet de générer les représentations. Cette méthode est capable de capturer l'*équivalence structurelle*. Des graphes dont les nœuds possèdent ce type de similarité obtiendront des représentations proches. *Spectral Features* (de Lara et Pineau, 2018) a été développé pour effectuer de la classification de graphes non orientés et non pondérés. La première étape de cet algorithme est assez standard et consiste à calculer le spectre de la matrice laplacienne normalisée du graphe et d'en conserver uniquement les k plus petites valeurs propres positives. Ces valeurs triées dans l'ordre croissant forment la représentation du graphe. k est un paramètre qui permet de contrôler la dimension des représentations. Dans le cas où le graphe contient moins de k nœuds (ce qui résulte en moins de k valeurs propres), la représentation est complétée par la droite avec des zéros. D'autres méthodes s'inspirent de concepts provenant du traitement d'images (Mousavi *et al.*, 2017), ou de la physique (Tsitsulin *et al.*, 2018).

3. Approche suivie pour la détection de messages abusifs

Dans ce travail nous traitons une tâche de classification binaire consistant à déterminer si un message est abusif ou non. Nous traitons l'aspect textuel et conversationnel des messages. Pour ce dernier, nous avons besoin de transformer un contexte de messages en un graphe conversationnel. Pour cela, nous utilisons la méthode proposée par Papegnies *et al.* (2019) que nous présentons brièvement dans la section 3.1. Nous utilisons des méthodes de plongement pour apprendre automatiquement des représentations des messages et graphes. Tous les modèles que nous employons sont présentés dans la section 3.3. Dans nos expérimentations, nous les comparons à deux méthodes de référence (Papegnies *et al.*, 2019 ; Cécillon *et al.*, 2019) que nous présentons dans la section 3.2.

3.1. *Extraction des graphes*

Intuitivement, on pourrait imaginer que le contenu échangé dans une conversation en ligne est l'information la plus déterminante pour détecter les évènements importants tels que l'envoi d'un message abusif. Cependant, il a été montré dans (Papegnies *et al.*, 2019) que la structure de la conversation et la manière dont elle évolue au fil des messages échangés, est aussi porteuse d'informations importantes qui permettent de très bonnes performances de classification. Cette information peut être capturée en modélisant les interactions entre utilisateurs sous la forme d'un *graphe conversationnel*. Dans ce travail, nous suivons la méthodologie présentée par Papegnies *et al.* (2019) pour extraire ces graphes depuis des conversations. La figure 1 illustre ce processus dont nous expliquons les points essentiels dans la partie qui suit.

La méthode d'extraction est conçue pour traiter un historique de messages provenant d'une messagerie instantanée en ligne. Elle extrait un graphe qui modélise le contexte conversationnel dans lequel apparaît un message qui est appelé le *message ciblé* (en rouge dans la figure 1). Dans le cadre d'une tâche de classification, ce message est celui que l'on cherche à classifier. La méthode définit une *période de contexte* centrée autour du message ciblé qui contient un nombre k fixé de messages postés juste avant et après celui-ci. La période de contexte est donc une séquence de $2k + 1$ messages consécutifs. Le graphe représente tous les évènements qui se déroulent à l'intérieur de cette période. Les nœuds représentent chacun un utilisateur actif, c.-à-d. qui a posté au moins un message durant la période de contexte. Les liens sont dirigés et pondérés, et modélisent les interactions entre ces utilisateurs à l'intérieur de la période de contexte. La direction des liens indique le sens de l'échange (c.-à-d. qui envoie un message à qui ?) et les poids représentent l'intensité globale de l'échange entre deux utilisateurs. Le système de messagerie instantanée est une suite continue de messages qui ne permet pas d'adresser un message spécifiquement en réponse à un autre. Ainsi, la méthode d'extraction s'appuie sur la supposition qu'un message s'adresse à tous les utilisateurs actifs à ce moment de la conversation et pas à un seul. Des liens dirigés de l'auteur du message vers chaque utilisateur actif sont créés. Leurs

poids sont toutefois modulés, en supposant que plus deux messages sont proches dans la conversation (en nombre de messages les séparant), et plus l'intensité de l'interaction est forte. Plus de détails sur la méthode pour déterminer le poids et la direction des liens, ainsi que les différents paramètres qui peuvent être ajustés pour contrôler cette extraction, sont donnés dans (Papegnies *et al.*, 2019).

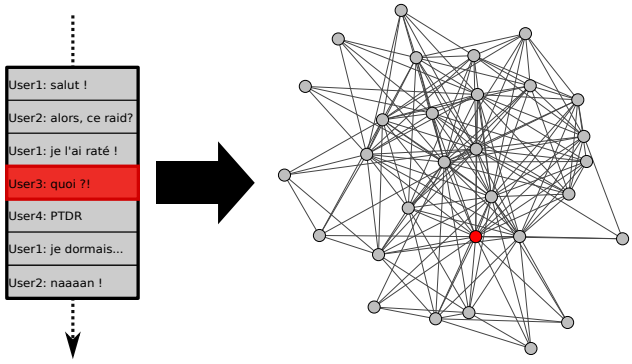

Figure 1. *Processus d'extraction des graphes conversationnels à partir d'une séquence de messages. La partie de gauche est un extrait de conversation avec le message ciblé en rouge, et la partie de droite est le graphe correspondant avec l'auteur du message ciblé en rouge .*

3.2. *Performances de référence*

Les performances de référence auxquelles nous comparons nos résultats proviennent de travaux traitant la même tâche que nous, c.-à-d. la détection de messages abusifs (Papegnies *et al.*, 2019 ; Cécillon *et al.*, 2019).

Cécillon *et al.* (2019) proposent deux approches. La première s'appuie sur le contenu textuel des messages uniquement. Un ensemble de caractéristiques est calculé pour représenter chaque message. On retrouve notamment des caractéristiques basiques portant directement sur la forme du message : par exemple sa longueur, la taille moyenne des mots qui le composent, le type de caractères utilisés, la proportion de majuscules. D'autres caractéristiques un peu plus spécifiques aux messages abusifs capturent le fait qu'un morceau de texte soit copié-collé en boucle ou qu'une lettre soit exagérément répétée, comme par exemple dans *noooooooon*. En plus de ces caractéristiques morphologiques, d'autres portant sur le langage sont aussi utilisées. On a par exemple le nombre total de mots, le nombre de mots distincts ainsi que le nombre de mots considérés comme abusifs au moyen d'un dictionnaire prédéfini. De plus, deux scores tf-idf sont calculés, chacun correspondant à la somme des scores tf-idf standards obtenus pour chaque mot du message. Le premier est calculé relativement à la classe *Abus* et le second par rapport à la classe *Non Abus*. Un autre élément important est obtenu en transformant le message en un sac-de-mots. Pour cette étape, la ponctua-

tion est retirée et toutes les lettres sont transformées en minuscules. Un classificateur bayésien naïf est ensuite entraîné à détecter des messages abusifs à partir de ces représentations. La sortie de ce classificateur est utilisée comme une caractéristique pour représenter le message. Au total, la méthode fondée sur le contenu textuel regroupe 29 caractéristiques.

La seconde approche traite des interactions entre utilisateurs et ignore complètement le contenu textuel. Il est à noter que le fait d'ignorer le contenu textuel des messages rend cette approche indépendante de la langue. La méthode utilise des graphes conversationnels extraits selon la procédure présentée dans la section 3.1. L'objectif est d'obtenir une représentation du graphe complet. Pour ce faire, la méthode extrait un ensemble de mesures topologiques. Ces mesures sont sélectionnées pour représenter le graphe à différentes échelles. Ainsi, certaines mesures caractérisent le graphe dans sa globalité (p. ex. le diamètre ou la densité), tandis que d'autres caractérisent des nœuds individuellement (p. ex. le degré ou la centralité de proximité). Les mesures relatives aux nœuds sont généralement calculées pour l'intégralité des nœuds du graphe et permettent de créer deux types de caractéristiques. La première est la valeur obtenue pour le nœud correspondant à l'auteur du message ciblé. La seconde est la moyenne des valeurs obtenues pour tous les nœuds du graphe. Les mesures caractérisent également différents niveaux du graphe. Ainsi, les mesures de transitivité et de réciprocité ne caractérisent qu'une partie locale du graphe. Les mesures d'excentricité et de centralité intermédiaire concernent le graphe dans son ensemble alors que le score de modularité et le coefficient de participation permettent de caractériser des sous-structures intermédiaires. La liste complète des 459 caractéristiques de cette méthode est détaillée dans (Papegnies *et al.*, 2019).

3.3. *Modèles*

Dans cette partie, nous présentons les modèles que nous utilisons dans nos expériences et leurs paramètres. Pour les plongements de graphes, nous nous concentrons sur les méthodes obtenant les meilleurs résultats présentés dans le travail de Cécillon *et al.* (2021).

3.3.1. *Plongement lexical*

Word2vec (Mikolov *et al.*, 2013) génère des plongements fixés de mots. Dans ce travail nous utilisons un modèle[1] entraîné sur du texte en français provenant d'articles Wikipedia. Chaque mot est associé à un plongement de dimension 300. Le modèle fastText[2] (Bojanowski *et al.*, 2017) que nous utilisons est lui aussi entraîné sur du texte français provenant de Wikipedia. Toutes les représentations sont de dimension 300. Les auteurs de CamemBERT (Martin *et al.*, 2020b) proposent un modèle *BASE* composé de 12 couches d'encodeurs, 768 dimensions cachées et 12 têtes d'attention, et

1. `https://github.com/Kyubyong/wordvectors`
2. `https://github.com/flairNLP/fasttext`

un modèle *LARGE* composé de 24 couches d'encodeurs, 1 024 dimensions cachées et 16 têtes d'attention. Selon l'étude des auteurs (Martin *et al.*, 2020b), le second donne des performances supérieures pour des tâches de reconnaissance d'entités nommées et de reconnaissance d'implication textuelle, et similaires sur des tâches d'étiquetage morphosyntaxique et d'analyse syntaxique. Nous utilisons donc le modèle *LARGE* dans nos expérimentations. Ce modèle[3] est pré-entraîné sur le corpus *CCNet* (Wenzek *et al.*, 2020) contenant 135GB de textes en français provenant de divers sites web. Les plongements sont des vecteurs de dimension 1 024. Comme pour CamemBERT, FlauBERT (Le *et al.*, 2020b ; Le *et al.*, 2020a) propose un modèle *BASE* composé de 12 couches d'encodeurs, 768 dimensions cachées et 12 têtes d'attention, et un modèle *LARGE* composé de 24 couches d'encodeurs, 1 024 dimensions cachées et 16 têtes d'attention. Ces modèles sont entraînés sur un corpus regroupant 71GB de textes en français agrégés à partir de plusieurs sources. Nous utilisons le modèle *LARGE*[4] qui propose des plongements de 1 024 dimensions. Pour Flair (Akbik *et al.*, 2018), nous utilisons les modèles vers l'avant et vers l'arrière[5] entraînés sur des articles Wikipedia en français. Chaque modèle crée des plongements de longueur 1 024 qui sont concaténés pour obtenir le plongement final du mot d'une longueur de 2 048.

3.3.2. *Plongement de graphes*

Pour les plongements de graphes, nous utilisons à la fois des méthodes traitant des nœuds (Node2vec, GraphWave) et des graphes entiers (Spectral Features, Graph2vec). Tous les modèles sont entraînés sur nos propres données, en suivant les paramètres optimaux identifiés par Cécillon *et al.* (2021).

Node2vec (Grover et Leskovec, 2016) permet de biaiser la génération des marches aléatoires en ajustant les paramètres p et q. Nous utilisons une valeur de $0,95$ pour p et $1,00$ pour q qui permettent d'avoir des représentations qui capturent la structure globale du graphe ainsi que l'équivalence structurelle entre nœuds. Les représentations de nœuds sont de longueur 128. Pour GraphWave (Donnat *et al.*, 2018), qui préserve le rôle des nœuds dans la structure du graphe, nous fixons la taille des plongements de nœuds à 100. Pour Spectral Features (de Lara et Pineau, 2018), le seul paramètre à fixer est le nombre de valeurs propres k à utiliser, qui correspond aussi à la dimension des plongements appris. Dans nos expérimentations, nous fixons cette valeur à 128.

Graph2vec (Narayanan *et al.*, 2017) permet d'associer une étiquette à chaque nœud qui est utilisée lors de l'extraction des sous-graphes. Deux sous-graphes contenant des nœuds avec des étiquettes identiques seront considérés comme similaires. Cela permet d'introduire une notion supplémentaire de similarité entre les nœuds. Par défaut, Graph2vec utilise le degré d'un nœud comme son étiquette. Dans ce travail, nous reprenons l'idée de Cécillon *et al.* (2020) qui proposent trois types alternatifs d'étiquettes et évaluons leur impact sur la performance de classification. La première,

3. `https://camembert-model.fr/`
4. `https://github.com/getalp/Flaubert`
5. `https://github.com/flairNLP/flair`

Graph2vec-auteur, consiste à utiliser l'identifiant unique associé à l'auteur du message. Ainsi, on peut identifier qu'un même auteur est actif dans différentes conversations. L'intuition est que le fait d'identifier les auteurs peut permettre de prendre en compte un contexte à plus grande échelle qu'une simple conversation. Par exemple, cette approche peut prendre en compte le fait que des utilisateurs ont des interactions dans différentes conversations et qu'il y a donc déjà un historique entre eux. Dans *Graph2vec-distance*, nous calculons la distance qui sépare le nœud du nœud ciblé, et l'utilisons comme étiquette. L'objectif est de regrouper différents utilisateurs sous la même étiquette en fonction de leur proximité avec l'auteur représenté par le nœud ciblé. Ces regroupements peuvent correspondre aux personnes proches de l'auteur pour ceux ayant une faible distance, et aux personnes ayant de brefs contacts avec l'auteur pour ceux ayant une distance plus importante. Cette approche permet de réduire le nombre total d'étiquettes différentes dans le graphe, et pourrait potentiellement faciliter la détection de similitudes structurelles. Enfin, *Graph2vec-cible* est une extension de l'approche précédente qui vise à réduire encore plus le nombre d'étiquettes. Ici, on utilise une étiquette binaire qui indique si le nœud est celui ciblé ou non. Ces étiquettes se basant sur les utilisateurs et leur proximité par rapport à l'auteur correspondant au nœud ciblé, elles permettent d'introduire une information directement liée à notre tâche. Pour toutes ces stratégies, les graphes sont représentés par un vecteur de dimension 128.

4. Validation expérimentale

Dans cette section, nous présentons notre jeu de données (sous-section 4.1), notre protocole expérimental (sous-section 4.2), ainsi que les performances obtenues par les méthodes de plongement lexical et de graphes (sous-section 4.3). Enfin, nous combinons ces plongements et évaluons leur complémentarité (sous-section 4.4).

4.1. *Jeu de données*

Notre jeu de données se compose de 1 320 messages annotés comme étant *abusifs* ou *non abusifs*. Ces messages en français ont été postés par des utilisateurs du jeu en ligne SpaceOrigin[6] dans l'espace de discussions textuelles instantanées du jeu. Les messages considérés comme abusifs sont ceux qui ont été signalés par les autres utilisateurs et confirmés comme étant abusifs par les modérateurs. Ils constituent la classe *Abus* et représentent la moitié de notre jeu de données. La classe *Non abus* est constituée de messages sélectionnés aléatoirement parmi tous ceux n'ayant jamais été signalés. De plus, deux messages de cette classe ne peuvent pas apparaître dans la même conversation, c.-à-d. ils ne peuvent pas avoir été postés à des instants proches. À Chacun des 1 320 messages est associé son contexte, c.-à-d. les messages postés juste avant et après, pour permettre l'extraction de graphes conversationnels, comme

6. `https://play.spaceorigin.fr/`

indiqué dans la section 3.1. Une difficulté de notre tâche est qu'elle nécessite des conversations complètes et pas de simples messages annotés. A notre connaissance, aucun jeu de données de conversations avec des messages annotés pour de l'abus n'est disponible publiquement.

4.2. *Protocole expérimental*

Nous effectuons nos expériences sur une tâche de classification binaire qui consiste à déterminer si un message appartient à la classe *Abus* ou *Non abus*. Nous utilisons le jeu de données présenté dans la section 4.1 et les graphes conversationnels relatifs à chacun des messages qui sont disponibles en ligne[7]. Ces graphes sont obtenus en suivant la méthode d'extraction de la section 3.1 en suivant la configuration optimale proposée par Papegnies *et al.* (2019), à savoir : une fenêtre glissante de 10 messages, une période de contexte de 850 messages et une stratégie linéaire pour le calcul des attributs des nœuds. Pour les plongements de graphes, nous utilisons les quatre méthodes les plus efficaces présentées par Cécillon *et al.* (2021) (deux avec plongement de nœuds et deux avec plongement de graphes entiers) en plus de notre méthode de référence décrite dans la section 3.2 et des trois variantes de Graph2vec que nous décrivons dans la section 3.3.2. Ces méthodes sont utilisées pour représenter chaque graphe conversationnel sous la forme d'un seul vecteur numérique. Pour les méthodes de plongement de graphes entiers, nous utilisons cette représentation complète du graphe ; et pour les méthodes de plongement de nœuds, nous utilisons uniquement la représentation du nœud de l'auteur du message ciblé. Pour les plongements lexicaux, nous utilisons les cinq modèles décrits dans la section 3.3.1 en plus de la méthode de référence (voir section 3.2). Ces cinq méthodes permettent de générer des représentations individuelles de mots. Puisque notre objectif est de représenter un message dans son entièreté, nous faisons la moyenne des représentations de tous les mots le composant pour obtenir sa représentation globale. De plus, les méthodes s'appuyant sur l'architecture BERT, à savoir CamemBERT et FlauBERT, apprennent directement une représentation globale de la séquence d'entrée. Nous utilisons cette représentation globale en plus de la représentation moyenne pour ces deux méthodes.

Nous utilisons toutes ces représentations comme valeurs d'entrée d'une machine à vecteurs de support (SVM) pour effectuer la classification. Nous utilisons l'implémentation de l'outil *Sklearn* (Pedregosa *et al.*, 2011) sous le nom de SVC. Du fait de la taille relativement petite de nos données, nous effectuons toutes nos expériences avec une validation croisée en 10 parties. Chaque partie a une répartition égale des classes *Abus* et *Non abus*. Nous utilisons 70 % des données pour l'entraînement, les 30 % restants étant dédiés au test. La performance présentée est une moyenne calculée sur les 10 modèles issus de la validation croisée, et est exprimée en F-mesure, la moyenne harmonique de la Précision et du Rappel. Nos classes étant équilibrées, les micro et macro F-mesures sont équivalentes dans nos expériences. Le terme F-

7. DOI : 10.6084/m9.figshare.7442273

mesure que nous utilisons dans la suite de ce papier réfère à ces 2 mesures. À moins que le contraire ne soit mentionné, toutes les différences de performance que nous évoquons sont statistiquement significatives (pour un test de Student avec $p < 0.05$). De plus, nous avons conduit ces expériences avec un perceptron multicouche (MLP) pour évaluer l'apport d'une telle méthode. Cependant, les performances obtenues sont très comparables au SVM, donc nous avons décidé de ne pas les présenter ici. Nous supposons que la taille de nos données impacte négativement l'efficacité d'une telle approche neuronale pour notre tâche.

4.3. *Performances de classification*

Le tableau 1 présente les valeurs de F-mesure obtenues par notre méthode de référence et nos cinq méthodes de plongement lexical considérées. La première colonne dénote l'échelle des représentations générées par les méthodes : *Mot* pour les méthodes qui génèrent des représentations de mots que nous moyennons pour obtenir une représentation du message, et *Message* pour les méthodes qui apprennent directement une représentation globale du message. Le tableau indique également la dimension des représentations apprises par chaque méthode. Nous conservons les dimensions originales de tous les modèles pré-entraînés que nous utilisons.

Pour la majorité des modèles étudiés, le type d'erreur de classification le plus fréquent est de classer un message non-abusif comme abusif. Ce phénomène représente environ 60 % des erreurs totales. Certains messages non-abusifs sont relativement ambigus et difficiles à classifier.

Échelle	**Méthode**	**Dimension**	**F-mesure**
-	Référence-Texte	29	79,89
Mot	Word2vec	300	75,21
	fastText	300	81,24
	Flair	2 048	82,56
	CamemBERT-W	1 024	89,79
	FlauBERT-W	1 024	85,12
Message	CamemBERT-M	1 024	89,62
	FlauBERT-M	1 024	79,79

Tableau 1. *F-mesures obtenues par notre méthode textuelle de référence et les 5 méthodes de plongement lexical. La différence entre CamemBERT-W et CamemBERT-M n'est pas statistiquement significative.*

Notre première observation est que presque toutes les méthodes de plongement lexical obtiennent une meilleure F-mesure que la méthode de référence s'appuyant sur des caractéristiques choisies *a priori*. Word2vec et fastText sont parmi les méthodes les moins performantes sur notre tâche. Ce n'est pas surprenant car ces deux méthodes

produisent des représentations fixées, c.-à-d. une représentation unique par mot. Elles ne peuvent donc pas faire la différence entre des homographes. La différence de performance entre ces deux méthodes peut s'expliquer par le fait que Word2vec n'est pas capable de traiter les mots non observés lors de l'apprentissage alors que fastText en est capable. Dans le contexte de messages en ligne, on peut considérer que c'est quelque chose d'essentiel car les fautes d'orthographe et l'utilisation de termes très spécifiques sont courants.

Les représentations dépendantes du contexte de Flair obtiennent des performances légèrement meilleures, mais qui sont inférieures aux autres méthodes mettant en pratique cette approche. L'utilisation de représentations au niveau des caractères ne semble donc pas être la mieux adaptée à une tâche de détection d'abus, du moins sur ces données.

La méthode FlauBERT-M, qui apprend une représentation globale du message, obtient des performances inférieures à FlauBERT-W qui construit cette représentation à partir des mots qui composent le message. On peut supposer que le caractère abusif d'un message ne repose souvent que sur quelques mots et qu'un traitement au niveau des mots permet de mieux les détecter. Toutefois, on ne retrouve pas cette tendance avec CamemBERT puisque les deux approches obtiennent des performances qui ne sont pas significativement différentes, CamemBERT-W obtenant toutefois la meilleure F-mesure (89,79 %) de toutes les approches de plongement lexical. Cela constitue une amélioration de dix points par rapport à notre méthode de référence.

Ces résultats démontrent tout l'intérêt des approches de plongement lexical. Pour notre tâche, elles sont capables d'apprendre des représentations de messages qui capturent beaucoup plus d'informations pertinentes pour la classification qu'un simple ensemble de caractéristiques morphologiques calculées directement sur le message. De plus, ces méthodes n'ayant pas toutes les mêmes propriétés et objectifs, on peut supposer qu'elles sont complémentaires. Le *et al.* (2020b) montrent par exemple que la combinaison de CamemBERT et FlauBERT obtient de meilleurs performances que les deux méthodes utilisées séparément sur une tâche d'étiquetage morpho-syntaxique. Nous explorons ces possibilités dans la section 4.4.

Le tableau 2 présente les F-mesures obtenues par notre méthode de référence sur les graphes et les modèles de plongement de graphe décrits dans la section 3.3.2. Pour les quatre premières méthodes, nous avons refait les expériences en utilisant les modèles et configurations de Cécillon *et al.* (2021). Les trois dernières sont les versions modifiées de Graph2vec que nous proposons pour inclure des informations supplémentaires sur les nœuds. La première colonne du tableau indique s'il s'agit d'une méthode de plongement de *nœuds* ou de *graphes entiers*.

Les résultats que nous obtenons avec les plongements non-modifiés sont très similaires à ceux obtenus par Cécillon *et al.* (2021) et confirment leurs observations. Tout d'abord, puisque le graphe conversationnel représente le message et tout son contexte, on pourrait penser qu'utiliser un plongement de graphe entier permettrait de capturer un maximum d'information. Cependant, les résultats montrent que résumer le graphe

Échelle	**Méthode**	**Dimension**	F**-mesure**
-	Référence-Graphe	459	88,08
Nœuds	Node2vec	128	83,15
	GraphWave	200	82,89
Graphe entier	Spectral Features	128	79,59
	Graph2vec	128	81,91
	Graph2vec-auteur	128	82,21
	Graph2vec-cible	128	81,86
	Graph2vec-distance	128	84,30

Tableau 2. *F-mesures obtenues par notre méthode graphe de référence, quatre méthodes de plongement de graphes et les trois variantes de Graph2vec proposées. Les performances de Node2vec et GraphWave, ainsi que celles de Graph2vec, Graph2vec-auteur et Graph2vec-cible, ne sont pas significativement différentes. En revanche, celle de Graph2vec-distance l'est.*

à la simple représentation du nœud de l'auteur du message ciblé est aussi efficace. La position de ce nœud spécifique et son rôle dans le graphe semblent suffisants pour représenter le graphe conversationnel et les méthodes de plongement de nœuds sont capables de capturer ces informations. À la différence de ce qui a été observé sur le texte seul, aucune méthode de plongement de graphe n'obtient de meilleure performance que la méthode de référence. On peut supposer que ces plongements sont trop généraux pour cette tâche, par comparaison à la méthode de référence qui intègre un très grand nombre de descripteurs divers.

Node2vec obtient un tout petit avantage par rapport à GraphWave, l'autre méthode de plongement de nœuds. Au niveau du graphe entier, Spectral Features est plus de deux points en dessous de toutes les variantes de Graph2vec. Son approche fondée sur la matrice laplacienne du graphe ne réussit pas à capturer toute l'information utile pour la classification. Parmi les trois variantes de Graph2vec que nous proposons, deux obtiennent des performances quasiment similaires à celles du modèle original. Identifier les auteurs (Graph2vec-auteur) ou le message ciblé (Graph2vec-cible) permet de créer des classes de nœuds plutôt que de traiter les nœuds comme des éléments individuels. Cependant, cela n'apporte pas d'information supplémentaire permettant de mieux identifier les messages abusifs. On peut supposer que l'approche Graph2vec-cible, qui ne distingue que deux classes de nœuds, ne propose pas une information suffisamment riche pour apporter un gain de performances. Au contraire, Graph2vec-auteur utilise une étiquette différente pour chaque auteur. Graph2vec n'arrive pas à tirer profit d'une telle approche, probablement parce qu'il est difficile de repérer des similitudes structurelles de graphes avec autant d'étiquettes de nœuds différentes. En revanche, l'approche qui introduit la distance entre chaque nœud et le nœud ciblé apporte une amélioration qui en fait la méthode de plongement de graphe la plus efficace avec une F-mesure de $84,30$. Cette distance est une donnée relative à notre tâche,

car elle permet de catégoriser les utilisateurs selon leur proximité avec le nœud ciblé (p. ex.ami proche, connaissance, inconnu). Cette proximité est un élément de contexte pouvant être important pour détecter un abus. Cela montre qu'ajouter une information spécifique au problème lors de l'apprentissage des représentations est bénéfique pour la qualité des représentations apprises.

La meilleure approche par plongement est donc moins performante que la référence d'environ quatre points. Toutefois, ces approches conservent des avantages par rapport à la référence. D'abord, elles requièrent beaucoup moins de ressources computationnelles. Il a fallu plus de huit heures pour calculer toutes les mesures topologiques de la méthode de référence alors que quelques minutes sont suffisantes sur la même machine pour Node2vec, GraphWave, Spectral Features et Graph2vec. Ces approches automatiques sont ainsi mieux adaptées pour traiter des corpus et graphes plus grands. De plus, ces méthodes ne sont pas conçues spécialement pour la détection d'abus. Elles pourront s'adapter plus facilement à d'autres environnements d'utilisation.

4.4. *Fusion*

La littérature montre que combiner différentes sources d'information peut se révéler efficace pour détecter des messages abusifs (Mishra *et al.*, 2018a ; Cécillon *et al.*, 2019). Dans cette section, nous proposons de fusionner certaines méthodes de plongement lexical et de graphes que nous utilisons dans la section 4.3. L'intuition est que ces méthodes s'appuient sur des modalités très différentes (c.-à-d. le texte et les graphes conversationnels) et que leurs plongements doivent donc capturer des informations différentes, possiblement complémentaires. Si tel est le cas, leur combinaison devrait permettre d'améliorer les performances de classification. En plus de combiner des méthodes portant sur le texte et les graphes, nous considérons également des méthodes traitant la même modalité. Comme indiqué par Le *et al.* (2020b), cette combinaison peut amener une amélioration des performances.

Nous utilisons les trois stratégies de fusion proposées par Cécillon *et al.* (2019) pour combiner les représentations apprises par nos méthodes.

– La *fusion précoce* consiste à former une unique représentation en concaténant celles des méthodes à fusionner. Avec cette stratégie, le classificateur a accès à l'intégralité des informations.

– La *fusion tardive* est décomposée en deux étapes. D'abord, un classificateur est appliqué à chaque méthode à fusionner séparément. Il est utilisé pour calculer un score qui est la probabilité d'être abusif de chaque message du corpus. Ensuite, ces scores sont concaténés et utilisés comme entrées d'un nouveau classificateur pour effectuer la classification. L'intuition est que ces scores peuvent représenter toute l'information importante pour ce dernier classificateur en éliminant les parties inutiles que l'on pourrait retrouver dans les représentations originales.

– La *fusion hybride* est une combinaison des deux autres stratégies qui consiste à concaténer la représentation unique de la fusion précoce avec les scores de la fusion

tardive.

Le tableau 3 donne les valeurs de F-mesure obtenues en appliquant ces trois méthodes de fusion aux méthodes de plongement les plus performantes identifiées dans la section 4.3.

	Précoce		**Tardive**		**Hybride**	
Méthodes	**Dim.**	**F-m.**	**Dim.**	**F-m.**	**Dim.**	**F-m.**
Réf. Graphe + Réf. Texte	488	90,44	2	89,07	490	90,92
Cam-W + Cam-M	2 048	90,12	2	89,72	2 050	90,15
Cam-W + FlauBERT-W	2 048	89,69	2	89,86	2 050	90,09
Graph2vec-dist. + Node2vec	256	*87,07	2	*85,12	258	*84,14
Graph2vec-dist. + Cam-W	1 152	93,52	2	92,83	1 154	93,62
Node2vec + Cam-W	1 152	92,86	2	92,65	1 154	93,02
Réf. Graphe + Cam-W	1 483	94,23	2	93,54	1 485	94,43

Tableau 3. *F-mesures obtenues par les trois types de fusion en combinant les méthodes de plongement les plus performantes. Le tableau est décomposé en quatre parties en fonction des types de méthodes fusionnées ; la première partie pour les fusions des références, la deuxième pour les fusions entre plongements lexicaux, la troisième pour les fusions entre plongements de graphes et la quatrième pour les fusions entre plongements lexicaux et graphes. Les méthodes CamemBERT-W, CamemBERT-M sont abréviées en Cam-W et Cam-M, Graph2vec-distance en Graph2vec-dist. et les références en Réf. Texte et Réf. Graphe. Les résultats significativement différents des deux autres types de fusion sur la même ligne sont marqués d'une étoile (*).*

En premier lieu, nous remarquons que les trois types de fusion donnent des résultats assez similaires. En général, la fusion hybride obtient les meilleures performances, devant la fusion précoce puis la fusion tardive. C'est un résultat logique puisque la fusion hybride est une combinaison des deux autres. Toutes les caractéristiques utilisées dans la fusion précoce sont résumées en seulement deux scores dans la fusion tardive. Malgré cela, la fusion tardive obtient des performances qui ne sont que légèrement inférieures à la fusion précoce.

La combinaison hybride des deux méthodes de référence apporte une augmentation de près de trois points par rapport à la référence sur les graphes utilisée seule. Cela confirme l'intuition que le texte et les graphes intègrent des informations qui sont au moins partiellement complémentaires, et que les combiner est bénéfique pour la classification. La fusion des deux approches de CamemBERT (CamemBERT-W et CamemBERT-M) n'apporte pas une augmentation significative des performances. On peut donc supposer qu'elles capturent des informations similaires. Il en est de même pour la fusion de CamemBERT et FlauBERT, qui n'améliore pas la performance de CamemBERT utilisé seul. Ces plongements lexicaux capturent donc des informations

redondantes, ce qui peut s'expliquer par le fait que les deux approches CamemBERT et FlauBERT sont dérivées de la même architecture BERT.

Au niveau des graphes, la fusion précoce de Graph2vec-distance et Node2vec ($87,07$ %) permet de se rapprocher à un point de la méthode de référence sur les graphes ($88,08$ %). Cela veut dire que ces méthodes traitant les graphes entiers pour la première et les nœuds pour la seconde capturent des informations différentes et que celles-ci sont en partie complémentaires. En revanche, les fusions tardive et hybride donnent de moins bonnes performances. Ainsi, nous supposons que toutes les informations complémentaires capturées par ces deux méthodes ne peuvent pas être résumées en seulement deux scores.

Enfin, la combinaison de la meilleure méthode s'appuyant sur le contenu textuel (CamemBERT-W) avec chacune des méthodes fondées sur les graphes donne systématiquement de meilleures performances que la combinaison des deux méthodes de référence ($90,92$ %). Les combinaisons hybrides de CamemBERT-W avec Graph2vec-distance et Node2vec donnent respectivement une F-mesure de $93,62$ % et $93,02$ %. La meilleure performance est obtenue en combinant CamemBERT-W et la méthode de référence sur les graphes ($94,43$ %). Il est intéressant de voir que cette combinaison n'obtient qu'un point de mieux que les deux combinaisons précédentes, alors que lorsqu'elles sont utilisées seules, Graph2vec-distance et Node2vec sont quatre et cinq points derrière la méthode de référence.

Le tableau 4 liste la meilleure performance que nous obtenons pour chacun des types d'approches étudiés. Pour les plongements lexicaux, c'est l'approche CamemBERT appliquée au niveau des mots qui est la plus efficace. Pour les graphes, c'est la méthode de plongement de graphes entiers Graph2vec dans sa version utilisant la distance comme étiquette des nœuds. Il est à noter qu'elle reste tout de même moins performante que la méthode de référence sur les graphes. Pour la fusion entre méthodes utilisant le texte, c'est la fusion hybride des approches CamemBERT appliquées au niveau des mots et du message qui obtient la meilleure F-mesure, sans pour autant apporter une amélioration conséquente par rapport aux méthodes utilisées individuellement. La fusion précoce entre un plongement de nœuds et un plongement de graphes entiers (Graph2vec et Node2vec) obtient les meilleurs résultats pour une fusion entre deux approches utilisant les graphes. Enfin, la meilleure performance générale est obtenue en fusionnant les graphes et le texte. La fusion hybride de la référence sur les graphes avec CamemBERT obtient une F-mesure de $94,43$ % avec une Précision de $93,62$ % et un Rappel de $95,27$ %. Ce résultat est la preuve que ces différentes modalités ne contiennent pas les mêmes types d'informations et que les combiner apporte un réel gain par rapport à une utilisation séparée des deux.

5. Conclusion

Dans ce travail, nous utilisons des plongements lexicaux et des plongements de graphes pour détecter automatiquement des messages abusifs en ligne. Nous compa-

Type	**Méthode(s)**	**Dim.**	F**-m.**
Référence	Référence-Graphe	459	88,08
	Référence-Texte	29	79,89
Texte	CamemBERT-W	1 024	89,79
Graphe	Graph2vec-distance	128	84,30
Fusion texte-texte	CamemBERT-W + CamemBERT-M	2 050	90,15
Fusion graphe-graphe	Graph2vec-dist + Node2vec	256	87,07
Fusion graphe-texte	Référence-Graphe + CamemBERT-W	1 485	94,43

Tableau 4. *Résumé de la méthode la plus performante en termes de F-mesure, pour chaque type d'approche étudié dans ce travail.*

rons cinq méthodes de plongement lexical sur du texte en français pour identifier la plus efficace pour cette tâche. Les plongements dépendant du contexte se montrent beaucoup plus performants que les plongements fixés. La représentation CamemBERT obtient la meilleure F-mesure avec $89,79$ %, améliorant largement le score obtenu par notre méthode de référence s'appuyant sur des descripteurs sélectionnés manuellement. Nous comparons également quatre méthodes de plongement de graphes dont l'efficacité pour cette tâche a déjà été montrée. La moitié d'entre elles génère des plongements au niveau des nœuds alors qu'il s'agit de graphes entiers pour l'autre moitié. Additionnellement, nous explorons une possibilité offerte par l'une de ces méthodes : Graph2vec. Nous considérons trois stratégies d'étiquetage des nœuds permettant d'intégrer différentes informations relatives à notre tâche directement lors de l'apprentissage des plongements. La version de Graph2vec utilisant la distance entre un nœud et le nœud ciblé comme étiquette est la méthode de plongement de graphes qui obtient la meilleure performance avec une F-mesure de $84,30$ %. Cela reste inférieur au score de $88,08$ % obtenu par notre méthode de référence, mais ce sont des résultats très prometteurs considérant le fait que la quantité de données utilisées pour apprendre ces plongements de graphes est faible.

Enfin, nous avons expérimenté trois stratégies de fusion pour combiner les différents plongements. Nos résultats montrent que fusionner des plongements lexicaux et de graphes permet de tirer profit des deux sources d'information et de sensiblement améliorer la détection des messages abusifs. La combinaison de CamemBERT et Graph2vec obtient une F-mesure de $93,62$ %, juste en dessous de notre meilleure performance absolue ($94,43$ %) obtenue en combinant CamemBERT et notre méthode de référence sur les graphes utilisant un ensemble de mesures topologiques prédéfinies. Dans ce travail, nous combinons des plongements lexicaux et de graphes qui sont appris séparément. Une piste de développement est de combiner ces deux modalités directement lors de l'apprentissage pour obtenir une seule représentation incluant toutes les informations. Les messages étant postés dans un certain ordre, on pourrait aussi penser à intégrer un aspect temporel. Au niveau des graphes par exemple, nous

envisageons d'extraire un graphe conversationnel *dynamique*, c.-à-d.une séquence de graphes reflétant l'évolution de la conversation au cours du temps, et notamment au moment où le message qui nous intéresse a été posté.

6. Bibliographie

Ahmed A., Shervashidze N., Narayanamurthy S., Josifovski V., Smola A. J., « Distributed Large-Scale Natural Graph Factorization », *22nd International Conference on World Wide Web*, p. 37-48, 2013.

Akbik A., Blythe D., Vollgraf R., « Contextual String Embeddings for Sequence Labeling », *27th International Conference on Computational Linguistics*, p. 1638-1649, 2018.

Badjatiya P., Gupta S., Gupta M., Varma V., « Deep Learning for Hate Speech Detection in Tweets », *26th International Conference on World Wide Web Companion*, p. 759-760, 2017.

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.

Belkin M., Niyogi P., « Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering », *Advances in Neural Information Processing Systems 14*, p. 585-591, 2002.

Bojanowski P., Grave E., Joulin A., Mikolov T., « Enriching Word Vectors with Subword Information », *Transactions of the ACL*, vol. 5, p. 135-146, 2017.

Cai H., Zheng V. W., Chang K. C.-C., « A Comprehensive Survey of Graph Embedding : Problems, Techniques, and Applications », *IEEE TKDE*, vol. 30, n° 9, p. 1616-1637, 2018.

Chatzakou D., Kourtellis N., Blackburn J., De Cristofaro E., Stringhini G., Vakali A., « Mean Birds : Detecting Aggression and Bullying on Twitter », *2017 ACM on Web Science Conference*, p. 13-22, 2017.

Chen H., Perozzi B., Hu Y., Skiena S., « HARP : Hierarchical Representation Learning for Networks », *32nd AAAI Conferenceon Artificial Intelligence*, 2018.

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 Conference on Social Computing*, p. 71-80, 2012.

Cécillon N., Dufour R., Labatut V., Linarès G., « Tuning Graph2vec with Node Labels for Abuse Detection in Online Conversations », *11th MARAMI*, 2020.

Cécillon N., Labatut V., Dufour R., Linarès G., « Abusive Language Detection in Online Conversations by Combining Content- and Graph-Based Features », *Frontiers in Big Data*, vol. 2, p. 8, 2019.

Cécillon N., Labatut V., Dufour R., Linarès G., « Graph Embeddings for Abusive Language Detection », *SN Computer Science*, 2021.

Dadvar M., Trieschnigg D., Ordelman R., de Jong F., « Improving Cyberbullying Detection with User Context », *35th European Conference on IR Research*, vol. 7814, March, 2013.

de Lara N., Pineau E., « A Simple Baseline Algorithm for Graph Classification », *arXiv*, 2018.

Devlin J., Chang M.-W., Lee K., Toutanova K., « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding », *NAACL-HLT*, p. 4171-4186, 2019.

Dinakar K., Reichart R., Lieberman H., « Modeling the detection of Textual Cyberbullying », *5th ICWSM / Workshop on the Social Mobile Web*, p. 11-17, 2011.

Djuric N., Zhou J., Morris R., Grbovic M., Radosavljevic V., Bhamidipati N., « Hate speech detection with comment embeddings », *24th WWW*, p. 29-30, 2015.

Donnat C., Zitnik M., Hallac D., Leskovec J., « Learning Structural Node Embeddings via Diffusion Wavelets », *24th ACM SIGKDD*, p. 1320–1329, 2018.

Founta A. M., Chatzakou D., Kourtellis N., Blackburn J., Vakali A., Leontiadis I., « A Unified Deep Learning Architecture for Abuse Detection », *10th ACM WebSci*, p. 105-114, 2019.

Goyal P., Ferrara E., « Graph embedding techniques, applications, and performance : A survey », *Knowledge-Based Systems*, vol. 151, p. 78 - 94, 2018.

Grover A., Leskovec J., « Node2vec : Scalable Feature Learning for Networks », *22nd ACM SIGKDD*, p. 855-864, 2016.

Kipf T. N., Welling M., « Semi-Supervides Classification With Graph Convolutional Networks », *ICLR*, 2017.

Le H., Vial L., Frej J., Segonne V., Coavoux M., Lecouteux B., Allauzen A., Crabbé B., Besacier L., Schwab D., « FlauBERT : des modèles de langue contextualisés pré-entraînés pour le français », *31ème JEP, 27ème CTALN, 22ème RECITAL*, p. 268-278, 2020a.

Le H., Vial L., Frej J., Segonne V., Coavoux M., Lecouteux B., Allauzen A., Crabbé B., Besacier L., Schwab D., « FlauBERT : Unsupervised Language Model Pre-training for French », *12th Language Resources and Evaluation Conference*, p. 2479-2490, 2020b.

Liu Y., Ott M., Goyal N., Du J., Joshi M., Chen D., Levy O., Lewis M., Zettlemoyer L., Stoyanov V., « Roberta : A robustly optimized bert pretraining approach », *arXiv*, 2019.

Martin L., Muller B., Javier Ortiz Suárez P., Dupont Y., Romary L., Villemonte de la Clergerie E., Sagot B., Seddah D., « Les modèles de langue contextuels Camembert pour le français : impact de la taille et de l'hétérogénéité des données d'entrainement », *33ème JEP, 27ème CTALN, 22ème RECITAL*, p. 54-65, 2020a.

Martin L., Muller B., Ortiz Suárez P. J., Dupont Y., Romary L., de la Clergerie E., Seddah D., Sagot B., « CamemBERT : a Tasty French Language Model », *ACL*, p. 7203-7219, 2020b.

Mikolov T., Chen K., Corrado G., Dean J., « Efficient estimation of word representations in vector space », *ICLR Workshop Track Proceedings*, 2013.

Mishra P., Del Tredici M., Yannakoudakis H., Shutova E., « Author Profiling for Abuse Detection », *27th International Conference on Computational Linguistics*, p. 1088-1098, 2018a.

Mishra P., Yannakoudakis H., Shutova E., « Neural Character-based Composition Models for Abuse Detection », *2nd Workshop on Abusive Language Online*, vol. 2nd Workshop on Abusive Language Online (ALW2), p. 1-10, October, 2018b.

Mousavi S. F., Safayani M., Mirzaei A., Bahonar H., « Hierarchical Graph Embedding in Vector Space by Graph Pyramid », *Pattern Recognition*, vol. 61, p. 245–254, 2017.

Narayanan A., Chandramohan M., Venkatesan R., Chen L., Liu Y., Jaiswal S., « graph2vec : Learning Distributed Representations of Graphs », *13th MLG*, 2017.

Nobata C., Tetreault J., Thomas A., Mehdad Y., Chang Y., « Abusive language detection in online user content », *25th International Conference on World Wide Web*, p. 145-153, 2016.

Okky Ibrohim M., Budi I., « A Dataset and Preliminaries Study for Abusive Language Detection in Indonesian Social Media », *Procedia Computer Science*, vol. 135, p. 222 - 229, 2018.

Ou M., Cui P., Pei J., Zhang Z., Zhu W., « Asymmetric Transitivity Preserving Graph Embedding », *22nd ACM SIGKDD*, p. 1105-1114, 2016.

Padilla Montani J., Schüller P., « TUWienKBS at GermEval 2018 : German Abusive Tweet Detection », *GermEval 2018 Workshop*, p. 45-50, 2018.

Papegnies E., Labatut V., Dufour R., Linarès G., « Conversational Networks for Automatic Online Moderation », *IEEE Trans. Comput. Social Systems*, vol. 6, n° 1, p. 38-55, 2019.

Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E., « Scikit-learn : Machine learning in Python », *Journal of Machine Learning Research*, vol. 12, p. 2825-2830, 2011.

Pennington J., Socher R., Manning C. D., « GloVe : Global Vectors for Word Representation », *Empirical Methods in Natural Language Processing*, p. 1532-1543, 2014.

Perozzi B., Al-Rfou R., Skiena S., « DeepWalk : Online Learning of Social Representations », *20th ACM SIGKDD*, p. 701-710, 2014.

Perozzi B., Kulkarni V., Skiena S., « Don't Walk, Skip ! Online Learning of Multi-scale Network Embeddings », *IEEE/ACM ASONAM*, p. 258-265, 2017.

Peters M., Neumann M., Iyyer M., Gardner M., Clark C., Lee K., Zettlemoyer L., « Deep Contextualized Word Representations », *NAACL-HLT*, p. 2227-2237, 2018.

Radford A., Narasimhan K., Salimans T., Sutskever I., « Improving Language Understanding by Generative Pre-Training », 2018, .

Roweis S. T., Saul L. K., « Nonlinear Dimensionality Reduction by Locally Linear Embedding », *Science*, vol. 290, n° 5500, p. 2323-2326, 2000.

Salminen J., Almerekhi H., Milenković M., Jung S., An J., Kwak H., Jansen B. J., « Anatomy of Online Hate : Developing a Taxonomy and Machine Learning Models for Identifying and Classifying Hate in Online News Media. », *ICWSM*, June, 2018.

Tsitsulin A., Mottin D., Karras P., Bronstein A., Müller E., « NetLSD : Hearing the Shape of a Graph », *24th ACM SIGKDD*, p. 2347-2356, 2018.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I., « Attention is All you Need », *NIPS*, vol. 30, p. 6000-6010, 2017.

Wang H., Wang J., Wang J., Zhao M., Zhang W., Zhang F., Xie X., Guo M., « GraphGAN : Graph Representation Learningwith Generative Adversarial Nets », *32nd AAAI Conference on Artificial Intelligence*, p. 2508-2515, 2018.

Warner W., Hirschberg J., « Detecting Hate Speech on the World Wide Web », *Second Workshop on Language in Social Media*, p. 19-26, 2012.

Waseem Z., Hovy D., « Hateful symbols or hateful people ? predictive features for hate speech detection on twitter », *NAACL Student Research Workshop*, p. 88-93, 2016.

Wenzek G., Lachaux M.-A., Conneau A., Chaudhary V., Guzmán F., Joulin A., Grave E., « CCNet : Extracting High Quality Monolingual Datasets from Web Crawl Data », *12th Language Resources and Evaluation Conference*, p. 4003-4012, 2020.

Xiang G., Fan B., Wang L., Hong J., Rose C., « Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus », *21st ACM CIKM*, p. 1980-1984, 2012.

Yan S., Xu D., Zhang B., Zhang H., Yang Q., Lin S., « Graph Embedding and Extensions : A General Framework for Dimensionality Reduction », *IEEE PAMI*, vol. 29, p. 40-51, 2007.

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.