Combining Graph and Text to Model Conversations: An Application to Online Abuse Detection

Noé Cecillon

AVIGNON UNIVERSITÉ

THÈSE DE DOCTORAT D'AVIGNON UNIVERSITÉ

École Doctorale n°536
Agrosciences & Sciences

Mention de doctorat :
Informatique

Laboratoire Informatique d'Avignon

Présentée par
Noé CÉCILLON

Combining Graph and Text to Model Conversations: An Application to Online Abuse Detection

---

Soutenue publiquement le 18 janvier 2024 devant le jury composé de :

Irina ILLINA	MCF HDR à Université de Lorraine, LORIA/INRIA	Rapporteuse
Julien VELCIN	PR à Université Lyon 2, ERIC	Rapporteur
Serena VILLATA	DR à Institut 3IA Côte d'Azur	Examinatrice
Harold MOUCHÈRE	PR à Nantes Université, LS2N	Président du jury
Vincent LABATUT	MCF HDR à Avignon Université, LIA	Directeur de thèse
Richard DUFOUR	MCF à Avignon Université, LIA	Co-directeur de thèse

LABORATOIRE INFORMATIQUE D'AVIGNON

ACKNOWLEDGEMENTS

This work would not have been accomplished without the help and encouragement of many people.

Firstly, I would like to sincerely thank Irina Illina and Julien Velcin for accepting to review my PhD thesis. I thank them for all the time and efforts they have committed to thoroughly read this manuscript. I also extend my thanks to Serena Villata and Harold Mouchère for participating in my thesis defense.

I would like to express my deepest gratitude to my advisors Vincent Labatut and Richard Dufour. I thank you for the trust you placed in me and all the help you have given me over the past 4 years. This entire thesis would not have been possible without your advice and expertise. I believe that the end of this thesis only marks a fresh start for future and fruitful collaborations with you.

Thanks to all my colleagues from the LIA for your welcome and for the numerous and various discussions.

I am also grateful to all the people involved in any way in this thesis. Particular thanks to Thomas without whom none of this would have happened.

I warmly thank all my friends for the great moments we spent together. Special thanks to Laure and Renzo, *we're going to hell, but we had a good laugh*.

Last but not least, I would like to heartily thank my parents for their support during my studies. Mom, you are the one that convinced me I was able to do this. I thank you for that, though you still do not fully understand what I am working on. Thanks to my sisters and brother for their continuous encouragements.

ABSTRACT

Online abusive behaviors can have devastating consequences on individuals and communities. With the global expansion of internet and the social networks, anyone can be confronted with these behaviors. Over the past few years, laws and regulations have been established to regulate this kind of abuse but the responsibility ultimately lies with the platforms that host online communications. They are asked to monitor their users in order to prevent the proliferation of abusive content. Timely detection and moderation is a key factor to reduce the quantity and impact of abusive behaviors. However, due to the sheer quantity of online messages posted every day, platforms struggle to provide adequate resources. Since this implies high human and financial costs, companies have a keen interest in automating this process. Although it may seem a relatively simple task, it turns out to be quite complex. Indeed, malicious users have developed numerous techniques to bypass the standard automated methods. Allusions or implied meaning are other examples of strategies that automatic methods struggle to detect. While usually performed on individual messages taken out of their context, it has been shown that automatic abuse detection can benefit from considering the context in which the message was posted.

In this thesis, we want to focus on the combination of content and structure of conversations to tackle the abuse detection task. Using the textual content of messages is the standard approach which was first developed in the literature. It has the advantage of being easy to set up, but on the other hand, it is vulnerable to text-based attacks such as obfuscation. The structure of the conversation which represent the context is less frequently used as it is more complicated to manipulate. Yet it allows to introduce a contextual aspect which helps detecting abuse occurrences when the text on its own is not sufficient. This context can be modeled as a contextual graph representing the conversation which includes the message. By comparing two methods based on feature engineering on a dataset of conversations extracted from a video games, we could show that a method relying exclusively on conversational graphs and ignoring the content was able to obtain better detection performance. The literature suggest that combining multiple modalities often result in a better detection of abusive messages. We propose multiple strategies to combine the content and structure of conversations and prove that their combination is indeed beneficial to the detection.

A limitation of feature-based methods is that they are costly in time and computational resources. Our study also highlights that only a fraction of the computed features are truly relevant for the task. Representation learning methods can be used to mitigate these issues by automati-

cally learning the representations of text and conversational graphs. For graphs, we demonstrated that using edge weights, signs and directions improved the performance. As no method exists for signed whole-graph embedding, we fill this gap in the literature by developing two such methods. We assess them on a newly constituted benchmark of three datasets of signed graphs and show that they perform better than their unsigned counterparts.

Lastly, we perform a comparative study of several lexical and graph-embedding method for abuse detection by applying them to our dataset of conversations. Our results show that they perform better than feature-based approaches on text and are slightly less effective on graphs. Still, they obtain promising results given that they are completely task independent, much more scalable and time-efficient than feature-based approaches.

RÉSUMÉ

Les comportements abusifs en ligne peuvent avoir des conséquences dramatiques sur les utilisateurs et les communautés. Avec l'avènement d'internet et des réseaux sociaux, personne n'est à l'abri de ce genre de comportement. Ces dernières années, de nombreux pays ont mis en place des lois visant à réduire ce type d'abus. Cependant, la responsabilité incombe principalement aux entreprises hébergant ces plateformes de discussion. Celles-ci se doivent de surveiller le comportement de ses utilisateurs afin d'éviter la prolifération de propos abusifs. Une détection et un traitement rapide des cas abusifs est un facteur important afin de réduire leur impact et leur nombre. Cependant, les plateformes en ligne ont souvent du mal à fournir les ressources nécessaires à cette détection à cause de la très grande quantité de messages postés chaque jour. Cette tâche de modération impliquant d'importants coûts humains et financiers, les entreprises ont un gros intérêt à l'automatiser. Bien que celà puisse paraître assez basique au premier abord, la détection automatique de contenu abusif se révèle assez complexe. En effet, les utilisateurs malveillants ont développé de nombreuses techniques pour tromper les méthodes automatiques. Par exemple, les propos implicites et l'utilisation de sous-entendus permettent souvent de ne pas se faire détecter par les méthodes automatiques standards. Pour contrer ce problème, il a été montré que prendre en compte le contexte dans lequel un message est posté permet d'améliorer la détection. Cependant la méthode la plus courante dans la littérature consiste à traiter des messages individuels, pris en dehors de leur contexte.

Dans ce manuscrit, on s'intéresse plus particulièrement à la combinaison du contenu et de la structure pour la détection de contenu abusif. Utiliser le contenu textuel des messages est l'approche la plus courante dans la littérature. Cette méthode présente l'avantage d'être facile à mettre en place, mais elle est aussi très vulnérable aux attaques basées sur le texte, notamment aux techniques d'obfuscation. La structure de la conversation, représentant le contexte, est beaucoup moins étudiée car elle est plus complexe à manipuler. Pourtant, elle permet d'introduire une notion de contexte qui permet de détecter des cas abusifs là où le texte seul n'en est pas capable. Ce contexte peut être modélisé sous la forme d'un graphe conversationnel représentant la conversation contenant le message étudié. En comparant deux méthodes construites à partir d'un procédé d'extraction de caractéristiques (feature engineering), nous avons montré qu'une méthode n'utilisant que des graphes conversationnels et ignorant le contenu textuel des messages était capable d'obtenir de meilleures performances. Des auteurs dans la littérature suggèrent que combiner plusieurs modalités d'information permet d'améliorer la détection de messages abusifs. À

cet effet, nous proposons plusieurs stratégies pour combiner le contenu et la structure des conversations et par nos expériences, nous prouvons que celà est en effet bénéfique pour la détection.

Une limitation de ces méthodes basées sur un ensemble de mesures est qu'elle sont assez coûteuses tant en ressources informatiques qu'en temps de conception. Notre étude montre également que seule une fraction des mesures calculées sont réellement importantes pour cette tâche. Les méthodes d'apprentissage de représentations peuvent être une solution à ce problème, en permettant d'apprendre automatiquement la représentation numérique d'un message ou d'un graphe conversationnel. Pour les graphes, nous avons démontré que considérer les attributs des liens, à savoir la direction, le poids et le signe, permet d'améliorer les performances. La littérature ne proposant aucune méthode de plongement de graphe entier signé, nous comblons ce vide en développant deux méthodes de ce type. Nous les évaluons sur un benchmark nouvellement créé et constitué de trois jeux de données de graphes signés, et prouvons qu'ils obtiennent de meilleurs résultats que leurs équivalents ne prenant pas en compte les signes.

Finalement, nous menons une étude comparative de plusieurs méthodes de plongement lexical et de graphes pour la détection de messages abusifs en les appliquant à un jeu de données de conversations. Nos résultats montrent qu'elles sont plus efficaces que les méthodes se basant sur un ensemble de mesures pour le texte, et légèrement moins efficaces pour les graphes. Cependant, ces résultats restent très encourageants car ces méthodes possèdent de nombreux autres avantages tels qu'être complètement indépendantes de la tâche, plus faciles à adapter à d'autres environnements d'utilisation, et beaucoup plus efficaces en termes de temps.

TABLE OF CONTENTS

Chapter 1

Introduction

Online abusive behaviors can harm a platform or community in the long term. From targeted personal attacks to major threats toward a group of persons, these behaviors can have a devastating effect. They may have a lasting impact on the mental health, confidence, and sense of safety of the victims [1]. In extreme cases, they can even trigger legal issues in some countries. In their 2021 study, Hinduja *et al.* [2] stated that 45.5% of the audited US middle and high schoolers had been cyberbullied during their lifetime. Because we focus exclusively on online abuse, we use the term *abuse* to refer to *online abuse* in the rest of this thesis. We also limit our study to text documents, although online abuse can be found in other media including audio, video, or animation. With the ever-growing quantity of text messages generated on the internet each year, the monitoring and detection of abusive content online has become a task of prime importance. This is a complex task, as abusive messages often make use of innuendos, irony, implicit statements or even refer to past events shared by the persons involved in the conversation. In these conditions, it is mandatory to understand the true meaning of a message before deciding whether to sanction its author.

The monitoring of online exchanges is usually done by human moderators to ensure the quality of the moderation. However, due to the increasingly global use of the Internet, it is often difficult for human moderators to treat all abusive comments on time. At the same time, online platforms are now responsible for the content that they host in multiple countries, which means that it can trigger legal issues if they fail to correctly moderate their content. Therefore, they have a strong interest in automating this moderation process. While a fully automated system can be difficult to implement, particularly because of the risk of censorship, a semi-automatic system can be used to assist human moderators for instance by pre-processing the comments and filtering out all the correct ones.

The standard approach to automatically detect abusive messages online is to do so by analyzing their content. Some markers of abuse, including swear words, and offensive or hateful expressions can easily be detected in this way. However, one can wonder if such expressions are necessarily representative of an abusive message. For instance, does quoting a previous message containing swear words make yours abusive? Does making a lame joke to a friend make you a harasser? The answer often depends on some general information, covering a wider scope than just the message itself. This is one of the reasons why it is difficult to automatically detect abusive comments. To overcome this, depending on the framework and application, multiple el-

ements can be considered such as the shared history between users, demographic data, or the conversation preceding the problematic message. This latter point is particularly important. Tweets taken without their conversational context are 50% more likely to be labeled as abusive by human annotators than the same tweets provided in their context [3]. While effective in many cases, a limitation of methods using the content to detect abuse is their vulnerability to text-based attacks such as obfuscation [4].

The structure of a conversation can reflect the presence of an abusive author. Indeed, they tend to participate significantly more in the discussions than others and to receive more replies than regular users [5]. It can be done deliberately by adopting a troll-like behavior to draw users into pointless discussions. It can also simply result from the fact that humans tend to react more to controversial subjects. Either way, the abusive author plays a central role in the conversation, and one can assume that the changes reflected in its structure can help discover an abuse case. Previous studies [6] showed that modeling conversations under the form of conversational graphs and characterizing them through various topological measures was efficient in detecting abusive messages. These graphs represent the structure of the conversation. This structure-based approach which completely ignores the content of the messages achieves even better results than a content-based method.

Combining multiple information modalities to analyze textual documents has proved effective on multiple tasks including sentiment analysis [7], named entity recognition [8], and recommendation [9]. For abuse detection, multimodal methods improve the performance of traditional text-oriented techniques. However, they are mainly centered around the combination of content and contextual data. The structure of the conversation is almost always overlooked in the literature. In a previous work conducted by our research team, Papegnies *et al.* [6] proposed a method based on the dynamics of conversation able to obtain strong results. The objective of this thesis is to go further and combine message content and conversation structure to build a representation that takes advantage of both and use this representation to improve the detection of abusive comments in online conversations.

Such combination can be conducted in several ways, which we explore in this manuscript. For any text classification task, the first step usually consists of transforming the raw input data into numeric representations. From the traditional feature-based strategies [6], [10]–[13] to more recent representation learning techniques [14]–[17], a lot of methods have been tested in the literature. By construction, they all capture different aspects of the input data. Furthermore, their efficiency is often closely related to the application and the data. We propose new methods and apply both traditional feature engineering-based approaches and more recent representation learning-based methods, in order to compare them to our abuse detection class and assess their complementarity.

1.1 Contributions

This thesis brings several contributions to the literature, the most prominent being three-fold. The first 2 arise in the context of abuse detection, while the third is specific to graph embeddings:

- **Combination of graph and text:** Many works in the literature combine different modalities of data to tackle the abuse detection task. It has been shown that including contextual information related to authors and conversations helps improve the classification compared to a content-only setting. The structure of the conversation, modeled by conversational graphs, is impacted by the behavior of users. This structural information on its own is able to obtain strong performance, and it has never been used in combination with other modalities. We propose multiple strategies to combine the text and the graphs to detect abusive messages in conversations.
- **Signed whole-graph embedding:** When leveraging graph-based approaches, we need to represent entire graphs. Whole-graph embedding methods are specifically designed to perform this task. As shown empirically, graph edges can include properties such as weights, directions, and signs, which are likely to improve the classification performance for abuse detection. However, all whole-graph embedding methods existing in the literature focus on unsigned graphs. Therefore, we propose 2 methods to fill this gap. The first is an adaptation of an *unsigned* whole-graph embedding approach to *signed* graphs, whereas the second is based on signed *vertex* representations, which we adapt to handle *whole* signed graphs.
- **Comparison of feature-based and embedding methods for abuse detection:** To combine content and structure, we develop methods to treat these aspects individually before combining them. We proceed in 2 successive steps.

  1. First, we study feature-based methods leveraging the content and the graphs (i.e. structure). We also perform a feature ablation study to determine the most important features.
  2. Second, we use several text and graph embedding methods of different natures, in order to automate the learning of representations. We determine the most efficient for the abuse detection task.

  We compare these 2 approaches in different aspects.

Besides these prominent contributions, this thesis also advances the field in 2 more minor aspects:

- **Network extraction:** To extract conversational graphs from conversation logs, we extend an existing network extraction method to obtain signed weighted directed graphs, as the original method is unable to produce signed graphs.

- **Data availability:** The development of abuse detection methods leveraging the structure of the conversation is limited by a lack of publicly available datasets. To tackle this issue, we made available the collection of conversational graphs extracted from our SpaceOrigin dataset and used throughout this thesis[1]. We also constituted the WAC dataset, that contains Wikipedia-based conversations annotated for 3 types of abuses [18]. Finally, we assembled a benchmark composed of 3 collections of signed networks annotated for diverse classification tasks.

1.2 Personal Bibliography

The following articles have been published in the framework of this PhD thesis:

- **International journal with peer review**
  - N. Cécillon *et al.*, "Graph embeddings for abusive language detection," *SN Computer Science*, vol. 2, no. 37, 2021. DOI: 10.1007/s42979-020-00413-7 (**Chapter 6**)
- **International conferences with peer review**
  - N. Cécillon *et al.*, "Abusive language detection in online conversations by combining content- and graph-based features," *Frontiers in Big Data*, vol. 2, p. 8, 2019, ISSN: 2624-909X. DOI: 10.3389/fdata.2019.00008 (**Chapter 4**)
  - N. Cécillon *et al.*, "WAC: A corpus of wikipedia conversations for online abuse detection," in *12th International Conference on Language Resources and Evaluation*, 2020. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2020/pdf/2020.lrec-1.173.pdf (**Chapter 2**)
- **National journal with peer review**
  - N. Cécillon *et al.*, "Approche multimodale par plongement de texte et de graphes pour la détection de messages abusifs," *Traitement Automatique des Langues*, vol. 62, no. 2, pp. 13–38, 2021. [Online]. Available: https://www.atala.org/sites/default/files/TAL_62_2_v2.pdf (**Chapter 6**)
- **National conference with peer review**
  - N. Cécillon *et al.*, "Tuning graph2vec with node labels for abuse detection in online conversations," in *11th MARAMI*, 2020. [Online]. Available: http://ceur-ws.org/Vol-2750/paper8.pdf (**Chapter 6**)

[1] https://figshare.com/articles/dataset/Conversational_Networks_For_Automatic_Online_Moderation/7442273?file=24681479

- **Submitted article**
  - N. Cécillon *et al.*, "Whole-graph representation learning for the classification of signed networks," *Submitted*, 2023 (**Chapter 5**)

1.3 Thesis Outline

The thesis is organized as follows.

- Chapter 2: We introduce the abuse detection task and review the literature associated with it.
- Chapter 3: We review the existing graph measures that can be used as classification features, to characterize the structure of a graph.
- Chapter 4: We describe our framework based on feature engineering to detect abusive messages using the textual content and the conversation structure. Using the measures previously described, we put it into practice on a conversation dataset, discuss its results, and perform a feature ablation study.
- Chapter 5: Because of the limitations detected in the previous approach, we want to use representation learning to automatically learn representations of messages and graphs. The previous chapter shows that edge attributes, including edge signs, bring useful information for the classification. As no signed whole-graph embedding methods currently exist in the literature, we propose 2 of them that we will later use to detect abusive messages. We evaluate the proposed methods on a new benchmark of 3 datasets and discuss their results.
- Chapter 6: We present our framework based on representation learning to detect abusive messages. We compare several text and graph embedding methods, including those proposed in Chapter 5, on the same conversation dataset as before. We discuss the results and perform a feature study.
- Chapter 7: We summarize our work, and identify its main perspectives.

Chapter 2

Survey of Online Abuse Detection

Online abuse includes malicious behaviors that have a threatening, intimidating or harassing effect on a group or an individual target. Detecting these behaviors is thus an important task, that researchers tackle with different approaches, including a majority based on the text and some based on different aspects of the conversation. The latter includes the use of conversational graphs to represent the dynamics of the conversation, as we will see in the remainder of this thesis. A key part in the development of an abuse detection method is to obtain a suitable dataset, but this often proves to be a complex task because of multiple factors that can affect the quality of human annotations in such datasets.

The following chapter is partially based on our work published at the *International Conference on Language Resources and Evaluation (LREC 2020)* [18]. It makes the following contributions:

1. We give a definition of what constitutes an online abuse based on the literature.
2. We survey the abuse detection methods and distinguish several categories depending on the information they use.
3. We review the publicly available datasets annotated in terms of abuse. This includes datasets of individual messages and datasets of messages in a flow of conversations. We also present

the 2 datasets related to this thesis, one is an original contribution and the other was previously developed in the same research team.

4. We formalize the abuse detection problem and present the standard evaluation settings for this task.

This chapter is organized as follows. In Section 2.1, we first define the concept of an online abuse. Second, we present and categorize the automatic abuse detection methods proposed in the literature in Section 2.2. In Section 2.3 we present the annotated datasets for abuse detection, including the 2 that are related to this thesis: one that we constituted and published, and the other that was previously used in our research team. We then formalize the problem of online abuse detection, and describe the standard evaluation methods in Section 2.4. Finally, we conclude this chapter in Section 2.5.

2.1 Definition of Online Abuse

Online abuse can be defined as the use of electronic media devices to cause harm to another person[1]. This wide concept points at a number of distinct abusive behaviors including hate speech, cyberbullying or impersonation. Some of these terms may be used interchangeably, which makes it difficult to have a consistent definition. To ease comparison, we curate the most accepted definitions from the literature of the main online abusive tactics.

Hate speech: Offensive discourse targeting a group or an individual based on inherent characteristics such as gender, religion, age, origin or sexual orientation [24], [25].

Cyberbullying: A repeated behavior, aimed at shaming or scaring an individual target. It includes posting embarrassing content of someone on social media and repeatedly sending hurtful remarks [26], [27].

Impersonation: The act of pretending to be another person on an online platform for the purpose of degrading or exploiting its public image [28].

Defamation: The action of posting comments with the intent to harm someone else's reputation, for instance by spreading lies [29], [30].

Doxing: The act of searching and publicly providing private information about a particular individual on the internet, typically with malicious intent [31], [32].

In this thesis, we use a general definition of *abuse*. We can describe it as all behaviors that do not respect the rules of the community in which the user is posting. Some of them can be specific to one community, but there is almost always a common core of rules prohibiting personal attacks, discrimination based on race, religion, or sexual orientation, as well as impersonation and doxing.

[1] https://cpdonline.co.uk/knowledge-base/safeguarding/online-abuse/

These various types of abuse are not without consequences and can have a negative impact on the people receiving these messages. Moderation is the process of monitoring user-generated content and taking actions against users who act abusively. This task is usually done by human moderators to ensure a high-level of moderation quality. Timely detection and moderation is a key factor to reduce the quantity and impact of abusive behaviors. However, due to the ever-growing quantity of content generated daily on the Internet, platforms struggle to provide adequate resources. Since this implies high human and therefore financial costs, companies have a keen interest in automating this process. Although it may seem a relatively easy task, it turns out to be quite complex. Indeed, over the years, malicious users have developed numerous techniques to bypass automated systems. In most situations, abuse comes as a text message, but it is not restricted to that. On platforms allowing to post multimedia content, abuse can come as a video, picture or audio record. However, this thesis focuses on text documents, so we will not consider such situations.

2.2 Abuse Detection Methods

2 main families of automatic approaches emerged in the abuse detection literature. The first and most widespread one is based on the textual content of the targeted message. Such methods can rely on a mix of standard Natural Language Processing (NLP) features, or be more machine learning-intensive with embedding-based approaches. The quality of these methods is however very often related to the training data used to learn abuse detection models. In the case of online social networks, the great variety of users, including very different language registers, spelling mistakes, as well as intentional obfuscation, makes it almost impossible to have models robust enough to be applied in all situations. Therefore, some authors consider contextual information to treat the abuse detection task. This may involve the messages occurring around the targeted message as well as information on the users themselves. Automatic abuse detection methods can be used in different settings. They can be used in a fully automated fashion, where they detect abusive instances and apply sanctions without any human intervention. The most common case, however, is a semi-automated framework where these methods are used to filter the potential abusive messages and human moderators review this small set of messages to take a decision. This pre-filter can greatly reduce the moderator's workload, and thus, the associated cost. In this section, we review the abuse detection literature, by first focusing on the textual methods (Section 2.2.1), then on the contextual methods (Section 2.2.2).

2.2.1 Textual Methods

The methods relying on the content of the targeted message to detect abuse or similar properties are the most widespread in the literature. We can divide them into 2 categories: standard

methods based on feature engineering, and more recent ones based on neural networks.

2.2.1.1 Feature Engineering-Based Methods

Spertus [33] was among the first to propose an automatic method for abuse detection. She hand-crafted rules over text to generate a feature vector composed of 47 elements representing the syntax and semantics of sentences. The rules are based on the author's observations of abusive and non-abusive messages. Such a manual feature engineering process was later used by Razavi *et al.* [10] who collected an *Insulting and Abusive Language Dictionary* composed of 2,700 words and expressions. Each entry is assigned a weight based on its potential abusive impact. Other works adopted such a lexicon-based approach later on [34]–[38]. Static rules and lexicon-based features are most of the time used as parts of systems, and combined with other techniques such as TF-IDF [11], [39], [40] and bag-of-words [11], [41].

The manual construction of an abuse lexicon is expensive, and results in a language-dependent dictionary. Wiegand *et al.* [42] propose a framework to automate this process. First, they build a small lexicon of negative polar expressions annotated via crowdsourcing. Then they extract a set of hand-crafted linguistic and semantic features to train an SVM which is used to automatically expand the lexicon. Mubarak *et al.* [43] propose a similar idea, but to expand a vocabulary of obscene words in Arab media. They perform a log odds ratio analysis to detect the words favoring documents categorized as obscene. This approach however requires a dataset with a wide variety of abusive and obscene words.

In sentiment analysis, the main objective is to detect the polarity of a message. In the context of abuse detection, a negative polarity can be seen as a marker of abuse. On other tasks such as authorship identification, sentiment analysis helped improve classification [44]. It was therefore adapted to detect various forms of abuse like hate speech [12], [13] and sexual harassment [45]. A limitation to this approach noted in [12] is that the words' meanings can change according to the context. For instance, the expression "You are stupid" is clearly offensive, whereas "This is stupid" is not. Hence, the aforementioned frameworks usually use sentiment-based features in combination with other syntactic, linguistic and lexicon-based features.

2.2.1.2 Neural Networks-Based Methods

With the advancements in computational capabilities, researchers have rapidly introduced abuse detection methods that rely on neural architectures. We first present those using distributed representations of messages, then those that don't but still use deep learning.

2.2.1.2.1 Distributed Representations

Djuric *et al.* [46] use the *paragraph2vec* [47] framework to learn low-dimensional representations of messages, and then train a logistic regression classifier to discriminate them based on this representation. With this method, they are able to outperform models trained on representations constructed from bag-of-words. An important observation of the authors is that text obfuscation in abusive comments leads to a high dimensionality and sparsity in bag-of-word representations, which has a negative impact on the classifier's performance. Nobata *et al.* [24] improve upon this work by training a regression model on a set of features derived from 4 different categories: word and character N-grams, linguistic features, syntactic features and distributional representations. They obtain the best results with all features combined, but note that character N-grams contribute significantly more to the performance than other features. Pavlopoulos *et al.* [48] and Mishra *et al.* [49] use 3 datasets annotated for toxicity, personal attacks and aggression to train Recurrent Neural Networks (RNN) operating on word embeddings and character N-gram features. Subsequently, numerous methods based on word embedding representations have been proposed to detect different types of abuse. This includes methods based on GloVe [50]–[54], FastText [50], [54], word2vec [52], [55], [56] and BERT [16], [17], [57]. Caselli *et al.* proposed HateBERT [58], a BERT model specifically designed for abusive language detection in English. To train HateBERT, they collect a dataset of approximately 1.5 million messages in English from various Reddit communities banned for being offensive, abusive, or hateful. They demonstrate better performance with HateBERT than a standard BERT model on multiple datasets. A number of studies are based on this model [59], [60].

2.2.1.2.2 Deep Learning

Badjatiya *et al.* [50] investigate different neural architectures trained for hate speech detection. They obtain their best performance with a Long-Short Term Memory (LSTM) model operating on word-level. Park *et al.* [61] classify sexist and racist abusive language by implementing 3 CNN-based models operating on different granularities of input features: characters, words or both. The model combining both characters and words outperforms the other models which use them separately and a logistic regression classifier based on N-grams used as a baseline. Akhter *et al.* [62] also show that deep learning models perform significantly better than machine learning models to detect abusive language in Urdu and Roman Urdu comments. They compare 4 architectures (Convolutional Neural Network, LSTM, Bidirectional-LSTM, and Contextual-LSTM) and obtain better results with one-layer than 2-layer architectures. The best performance being achieved by CNN. A tendency in recent studies is to develop hybrid deep learning models that combine multiple models instead of using them individually. This tendency is justified by the complexity of the task. The limitations of independent models can be complemented when used in combination with

one another, hence enhancing the overall performance of the system. We can cite CNN, LSTM, Bidirectional-LSTM and Gated Recurrent Units as some of the most widespread models in these hybrid architectures [63]–[67].

2.2.1.3 Discussion

There is a huge variety of methods relying on the content of the targeted message to detect abuse or related properties. Some of them use basic sets of hand-crafted rules while more sophisticated methods implement multiple deep learning architectures. While being generally more effective, the latter require large datasets for training. Since textual methods are usually language-dependent, they have to be trained on a dataset of the specific language, which can be difficult to find or constitute for low-resource languages. It is also common that a model trained on a particular type of text is not efficient on another type of text. Textual methods obtain very strong performance on multiple datasets and tasks. However, they have limitations when the abusive nature of a message is not only related to its content, but also to surrounding messages or contextual information. For instance, abuse can be spread over successive messages or can reference a shared history between users. This makes the detection more difficult. Furthermore, as noted in different studies [33], [42], they are vulnerable to text-based attacks. Abusive users can voluntarily obfuscate message content to deceive automatic methods based on text. Usual techniques include replacing letters (e.g. *f**k*), adding spaces or additional characters (e.g. *f u c k*, *f-u-c-k*) and more sophisticated methods such as encoding text in binary or hexadecimal. Hosseini *et al.* [4] demonstrate such an attack against the Perspective API[2], an API developed to mitigate toxicity online. For abusive comments, they are able to reduce the toxicity score given by the API to the level of benign phrases by slightly modifying the toxic words.

2.2.2 Contextual Methods

To address the aforementioned problems with the methods focusing on the content of the exchanged messages, some authors propose to consider the context of these messages. Menini *et al.* [3] explore the role of such context in abusive language annotation. They annotate 8,000 tweets in 2 conditions: with and without context. They find that almost half of the single tweets (i.e. tweets without context) labeled as abusive are later labeled as non-abusive when considering the context. This proves how context-dependent the offensiveness of swearing is. Yin *et al.* [39] also state that they often need to look at the context of a post to make a decision when manually labeling training data. In their comprehensive study of antisocial behavior in online discussion communities, Cheng *et al.* [5] add that instances of abusive messages generate significantly higher number of replies than regular messages. As the context can take multiple forms, we divide the context-based meth-

[2] https://perspectiveapi.com/

ods for abuse detection in 2 categories: those using the text surrounding a message and those using user-based features.

2.2.2.1 Surrounding Text-Based Features

Based on the previous observations, we can assume that the messages surrounding an abuse case can give precious information on the nature of this targeted message. In their model, Yin *et al.* [39] include features derived from the sentences neighboring the targeted message to detect harassment. Their objective is to spot conversations going off-topic, and use that as an indicator. In particular, they indicate that when a first abusive post appears, it often causes other users to respond with retaliatory harassment. Anuchitanukul *et al.* [68] propose a context-aware model that combines a representation of context (*history representation*) and target post. They encode the context as a sequence of sentences and learn the representations with BERT [69]. The concatenated representations of context and target post are then fed to a final classifier.

Vargas *et al.* [70] build on the many lexicon-based methods to propose a contextual lexicon approach. This is based on an offensive lexicon enhanced with contextual labels. They obtain better results for hate speech and offensive comment detection in Portuguese when using features built from this lexicon. Karan *et al.* [71] develop a preemptive toxic language detection system to assess if a comment is toxic based on the preceding messages. More formally, the model treats a sequence of comments including the one to classify and all of its ancestors. They show that considering the entire thread leads to better performance, in particular in threads which already contain a toxic comment. Almerekhi *et al.* [72] propose a related study. They investigate the causes leading to toxic discussion threads in Reddit. They define toxicity triggers, i.e. comments that incur direct toxic responses. These triggers can be used to detect threads where abusive messages are more likely to appear. The considered context can be even broader. Gao *et al.* [73] rely on the title of the news article that the message was posted for. They extract character and word N-gram features as well as lexicon features from the title to detect hate speech in Fox News user comments.

2.2.2.2 User-Based Features

Offensive authors tend to post abusive comments in multiple conversations. This suggests that considering user-level features could improve the detection of abusive messages. Researchers have proposed to incorporate different types of user-level context. Balci *et al.* [74] take advantage of user features such as gender, number of in-game friends, avatars or number of daily logins, to detect abuse in the community of an online game. Such demographic signals have been successfully leveraged in several works (e.g. [26], [75], [76]). The level of activity of users is especially interesting according to multiple studies. Cheng *et al.* [5] indicate that abusive authors are much

more involved in the conversations they participate in than regular users, and Xiang *et al.* [77] state that users who spend more time online are more likely to engage in antisocial behavior. Therefore, features such as the daily activity [74], account status [76], [78], [79] or number of messages posted [79]–[81] were used to include user-related information. Chatzakou *et al.* [78] also consider the inter-arrival time, i.e. the time between 2 consecutive messages, to study the waiting time in user's posting activity.

The fact that social media posts are short and noisy is a problem for abuse detection approaches. Qian *et al.* [82] mitigate this issue by collecting and analyzing the users' historical posts. They use the Twitter API to collect up to 400 tweets posted by each user and feed them into a pre-trained bi-LSTM to obtain an historical representation of the user, which is then used in combination with other representations based on text. Adding this historical representation of the user greatly improves the detection of abusive comments. While originally used to detect hate speech, this method could also be used to detect suspicious authors. Based on the same idea, Ziems *et al.* [79] estimate the historical language behavior of user based on a 4-year snapshot of its timeline. They consider that abusive comments are outside the norm. In this sense, they count the ratio of tokens in a post having zero occurrences in the timeline. According to their hypothesis, the higher this value, the greater the probability that the post is abusive.

2.2.2.3 Graph-Based Features

Modeling the users' social and conversational interactions via their respective graph showed promising results. A direct manner to use such structures is to describe them via a variety of graph measures such as Jaccard's similarity index [79], centrality scores, reciprocity or degree centrality [6], [78], [83], [84]. The latter method differs from the others in the sense that it relies exclusively on a conversational graph and ignores the content of messages, while other methods often use a mix of graph-based and linguistic features. Nonetheless, they achieve strong performance, even better than certain content-based method.

Another approach to leverage graph-based information, is to use graph embedding techniques. Mishra *et al.* [14] exploit community-based information from a graph representing authors of tweets to improve the detection of abusive comments. They create an undirected graph wherein vertices are authors and edges are the follower–followee relations between them. From this graph, they obtain a *user profile*, i.e. an embedding for each user, using node2vec [85]. This embedding emphasizes the structural role of the author in the community. They feed these user profiles to a gradient boosted decision tree classifier and prove that they greatly improve the system performances for detecting racist and sexist comments. In a subsequent work [86], they apply a Graph Convolutional Network (GCN) [87] to the same dataset, in order to obtain author representations. This approach allows producing input feature vectors for the author vertices which correspond to a representation of its entire set of tweets. This feature vector additionally incorporates the linguistic

behavior of authors in their learned profiles, which results in better performance. Ribeiro *et al.* [88] also apply a Graph Neural Network (GNN) using GraphSAGE [89], to learn user profiles. Nagar *et al.* [15] propose a framework that takes into account the content produced by authors and their profile information. They use a Variational Graph Auto-encoder to learn a joint representation of these 2 sources.

2.3 Datasets

This section is dedicated to the review of datasets for abuse detection. We only consider annotated datasets as we define our task as a supervised learning task. As mentioned previously in this chapter, abuse can take multiple forms, e.g. sexism, hate speech, racism, profanity or harassment. Here we consider abuse as a whole no matter its sub-category. Table 2.1 provides a summary of the recent datasets that are publicly available online, distinguishing between those that focus on independent messages (top part of the table) and whole conversations (bottom part). It is important to note that the majority of datasets in the literature use oversampling strategies to increase the proportion of abusive instances. Indeed, the typical prevalence of abusive tweets and personal attacks on Wikipedia talk pages is smaller than 1% [90], [91]. This oversampling can be done by considering specific sensitive topics and communities, or by using keywords related to abuse to filter messages.

This section is organized as follows. We first list and describe the existing datasets for abuse detection in Section 2.3.1. Then, we present the 2 datasets linked to this thesis in Section 2.3.2. Finally, we formalize the abuse detection problem and present how it is evaluated in Section 2.4.

2.3.1 Existing Datasets

Most of the datasets concern English messages, as this is the most common language on the Internet, but efforts are being made to provide datasets in other languages too. The last column of Table 2.1 highlights a recurring issue regarding the size of the datasets. Most of them consist of fewer than 10,000 messages, which is a limitation for deep learning approaches, as they often require a lot of data for training. Creating a large dataset is a long and expensive process, particularly to obtain high-quality human annotations. The challenging nature of this task is highlighted by the low inter-annotator agreement observed in the annotation process [90]. This is why some authors propose to automate the annotation of messages [18], [90] or even to automatically generate abusive instances with pre-trained language models (e.g. [94], [96]). We can distinguish 2 categories of abuse datasets: those constituted of individual messages, and those containing whole conversations.

Table 2.1: Summary of available abuse detection datasets.

Dataset	Ref.	Description	Language	Entries
SemEval-2023 Task 10	[92]	Class labels are - sexist, not sexist	English	20,000
Reddit context-aware dataset	[93]	Annotated with the parent post as context. Classes are hate, counter-hate or neutral	English	6,846
ToxiGen (Implicit Hate Speech)	[94]	Machine-generated dataset, labels are toxic and benign	English	274,186
Istanbul Convention dataset	[95]	5 classes: no hate speech, insult, exclusion, wishing harm, threatening harm. 2 datasets	Turkish	1,206 & 1,278
DynaHate	[96]	Machine-generated dataset, labels are hate or not hate + 5 secondary labels for hate type	English	41,255
Abuse is Contextual tweets dataset	[3]	Tweets enriched with preceding comment. Annotated for abuse	English	8,018
Civil Comments in Context	[97]	Comments annotated with and without context. Labels are non-toxic, unsure, toxic, very toxic	English	10,000
Hate Speech and Offensive Content Identification in Indo-European Languages (HASOC)	[98]	Class labels are - Non offensive, hate and offensive, hate, offensive, profane	English	5,335
			German	3,425
			Hindi	4,288
SemEval-2019 Task 5	[99]	Class labels are - Hateful, Non-Hateful	English	13,000
			Spanish	6,600
Wikipedia talk pages	[90]	Annotated for personal attack	English	115,737
Reddit Contextual Abuse Dataset	[100]	Contains conversation threads. Labels are 6 primary categories of abuse	English	27,494
CyberAgressionAdo	[101]	Full conversations. Collected through a role-playing game with high-schoolers	French	3,260
Ruddit	[102]	Annotation of full conversations. Real-valued scores ranging from supportive to offensive	English	6,000
Wikipedia Abuse Corpus	[18]	Provides conversations. Three datasets for personal attack, aggression, toxicity	English	382,665

2.3.1.1 Individual Message Datasets

The individual message datasets consider all messages independent of each other. They are listed in the top part of Table 2.1. They are the most common, as it is much easier to extract and annotate such individual messages (e.g. [94]–[96], [99]). We also include datasets such as [3], [92], [97] in this category. They provide the parent post or a few preceding posts in addition to the targeted post, but this context is too short to be considered as a real conversation. Indeed, it is mainly used to improve the quality of labels, by providing some context to the annotators, which has been proved beneficial.

2.3.1.2 Conversation Datasets

On the other hand, there are datasets with full conversations, such as [18], [100]–[103]. They are listed at the bottom of Table 2.1. Some of them are fully annotated (i.e. all messages have individual annotation) while others provide conversations in which only one message is annotated and the rest serve as context. We discuss conversation datasets in more details because they best suit our research needs. As mentioned in the introduction, in this thesis, we want to combine text content and conversation structure to detect abuse. For the latter, we thus require some conversational context in order to exploit the conversation structure. *CyberAgressionAdo* [101] is composed of fully annotated conversations. They were collected through a role-playing game wherein scenarios were introduced and high-school students had to play a role such as conciliator, bystander or bully. The dataset however contains only 19 conversations of around 130 messages each, which is too few to train a machine learning algorithm. *Ruddit* [102] and the *Reddit Contextual Abuse Dataset* [100] both propose conversation threads from Reddit. They specifically include subreddits which are likely to contain higher-than-average levels of abuse. They contain 6,000 and 27,494 posts, respectively. However, we cannot use these datasets for our experiments because of the way the conversation threads work on Reddit. A thread groups comments under an initial post, while each comment can in turn answer to a previous comment. In this manner, threads often have a tree structure, with a large width and a small depth. As different branches of this tree are usually unrelated and concern different users, it is not possible to correctly use the structures of Reddit comment threads to build significant conversation graphs.

2.3.2 Datasets Related to this Thesis

We now discuss the datasets directly related to this thesis that aim at solving the issues observed with the other existing datasets. The first one, SpaceOrigin [11], was constituted in the same research team as this thesis and constitutes the main dataset for our experiments in the rest of this thesis. The second, Wikipedia Abuse Corpus [18] is an original dataset that we constituted and published.

The SpaceOrigin dataset [11] is a collection of annotated French messages enriched with their conversational context, i.e. the messages posted before and after in the same conversation. The messages were extracted from a database of users' in-game interactions on the Massively Multiplayer Online Role-Playing *SpaceOrigin*[3]. They were posted in the in-game chat used by all users to communicate, propose alliances or make strategies. The original database contains more than 4 million entries among which 655 have been flagged as being abusive by at least one user in the game, and confirmed as such by a human moderator. This ensures the high quality of annotations for the *Abuse* class. Among the messages never been flagged by a confirmed abuse report, 1,890 were chosen at random to compose the *Non Abuse* class. Each of the 2,545 messages in the constituted dataset, whatever its class, is associated to its surrounding context (i.e. messages posted in the same thread). It is composed of up to 2,000 messages on each side of the message (before and after). They are ordered from oldest to newest, usually with a gap of a few seconds between them. The annotated messages were selected in such a way that they do not appear in the context of each other. This is particularly important that no abusive message appear in the context of another message, as it could trigger reactions and bias the analysis. All the messages forming the context are unlabeled, they are only used to represent the conversation in which the labeled message was posted. To summarize, the SpaceOrigin dataset is composed of 2,545 distinct conversations, each containing one annotated message and the context before and after it. However, this dataset is based on proprietary data which means that we are not allowed to share the raw text publicly. This is the reason why it is not included in Table 2.1.

To counter this problem, we constituted the *Wikipedia Abuse Corpus (WAC)* [18], a dataset of Wikipedia comments, along with the conversation in which they were posted. The objective with WAC is to provide a large dataset of almost 400,000 messages with conversational context. Such a large dataset is non-existent in the literature, as it contains one order of magnitude more annotated messages than the second largest conversation dataset in Table 2.1. The conversations in WAC come from Wikipedia talk pages, i.e. the web pages associated to Wikipedia articles where editors can exchange. Typically, editors write a message explaining which change they made on the article and why. These comments are longer than other types of online posts such as tweets or chat messages, with an average length of more than 1,000 characters. The conversations, however, are relatively short, with a majority of them constituted of fewer than 20 messages. Abusive comments usually come from pages related to sensitive topics such as religion or politics. Abusive comments represent 13.31% of the total comments in the dataset. The conversations on Wikipedia talk pages are however particular, in the sense that the time gap between 2 consecutive comments on less-visited pages can be several years which makes them more like a succession of messages than actual conversations. To constitute this dataset, we used 2 other publicly available datasets: the Wikipedia Comment Corpus [90] containing more than 63 million individual machine-labeled

[3] https://www.spaceorigin.fr

Wikipedia comments for personal attack, and WikiConv [104], a large corpus of Wikipedia conversations. We combined these 2 datasets to retrieve the context of the annotated messages, and therefore build a new conversation-based dataset.

The fact that conversations in WAC are usually very short (a few messages), and the time gap between 2 consecutive comments make this dataset unusable for our methods in the remaining of this thesis: first, these require longer conversations, and second, the slow evolution is likely to exhibit very different dynamics compared to chat rooms. For this reason, the rest of the work presented in the following focuses on the SpaceOrigin corpus. Even though we could not take advantage of our own dataset to assess our proposed methods, it is worth stressing that it was used by others [105]–[108].

2.4 Formalization and Evaluation

The literature does not always refer to the exact same problem when talking about abuse detection. We first formalize this task (Section 2.4.1), before presenting different metrics used to assess the quality of abuse detection systems, including the one that we use in this thesis (Section 2.4.2).

2.4.1 Formalization of Abuse Detection

Abuse detection is usually defined as a classification task consisting in automatically determining whether a post is abusive given a set of guidelines [105]. It is almost always treated as a classification task of *individual* messages, where all messages to classify are considered independent from each other. Still, a few works propose to directly detect abusive authors instead of messages (e.g. [109]). For the classification, 2 approaches exist. The first one consists in treating abuse detection as a binary task, with an *Abuse* class covering all types of abuse listed in Section 2.1, and a *Non-Abuse* class (e.g. [11], [41], [48], [54]). Systems trained following this procedure are easier to adapt to different frameworks, as they use a global definition of abuse. In many cases, however, it is interesting to treat abuse detection as a multi-class problem with different types or degree of abuse (e.g. [50], [110], [111]). This is particularly interesting for systems that have to apply different sanctions depending on the category of abuse. In this thesis, we focus on the detection of abusive comments that we consider as such if they do not meet the guidelines of the platform they are posted on. This is therefore a binary classification task which involves classes *Abuse* and *Non abuse*. We adopt this approach for 2 reasons. First, the SpaceOrigin dataset that we use only provides this type of annotation and second, we want to build a general system which could be adapted to different settings and datasets. To this end, treating abuse as a single, global class encompassing all types of abuse seems to be the most suitable strategy.

2.4.2 Evaluation Metrics

To evaluate abuse detection, a number of different metrics are used in the literature. In the following, we focus on the 2 most widespread.

2.4.2.1 ROC-AUC

The **Receiver Operating Characteristic (ROC) curve** is another method used to evaluate abuse detection [78], [112]. Let us define true positives (TP) (resp. true negatives (TN)) as abusive (resp. non-abusive) messages that are correctly predicted as such by the system. Conversely, false positives (FP) (resp. false negatives (FN)) are non-abusive (resp. abusive) messages that are incorrectly predicted as such by the system. A false negative corresponds to an abusive message that was missed by the classifier, whereas a false positive is a message that was wrongly flagged.

The ROC curve is the plot of the true positive rate (TPR) against the false positive rate (FPR), at various threshold settings. In their most basic form, the former is defined as

$$TPR = \frac{TP}{TP + FN}, \tag{2.1}$$

and the latter as

$$TNR = \frac{TN}{TN + FP}. \tag{2.2}$$

In a binary classification setting, certain classifiers associate a continuous random variable X to each instance. X is a score representing the probability of the instance to be abusive. Given a threshold parameter T, the instance is abusive if $X > T$ and non-abusive otherwise. X follows the probability density $f_1(x)$ if the instance is abusive and $f_0(x)$ if it is non-abusive. It is therefore possible to compute both TPR and FPR as functions of T. For the former we get:

$$TPR(T) = \int_T^\infty f_1(x)\,\mathrm{d}x, \tag{2.3}$$

and for the latter:

$$FPR(T) = \int_T^\infty f_0(x)\,\mathrm{d}x. \tag{2.4}$$

The ROC curve is the plot of $TPR(T)$ versus $FPR(T)$ with T as the varying parameter. The performance of a classifier is usually assessed by calculating the **Area Under the Curve** (AUC) which can be seen as the probability of the classifier to correctly give a higher score to a random positive case than a random negative case. For a predictor f:

$$AUC(f) = \frac{\sum_{t_0 \in \mathcal{D}^0} \sum_{t_1 \in \mathcal{D}^1} \mathbf{1}[f(t_0) < f(t_1)]}{|\mathcal{D}^0| \cdot |\mathcal{D}^1|}, \tag{2.5}$$

where $\mathbf{1}[f(t_0) < f(t_1)]$ returns 1 if $f(t_0) < f(t_1)$ and 0 otherwise, $\mathcal{D}^0$ is the set of negative examples and $\mathcal{D}^1$ is the set of positive examples.

Several studies criticize the use of the ROC curve for assessing the performance of a classification system [113]–[115]. They criticize the use of some portions of area under the curve with low sensitivity which are rarely of interest and should therefore be excluded from the AUC computation. Furthermore, another possible limitation noted by [112] is related to the low prevalence of abusive messages in abuse datasets. The author states that precision-recall curves might be more appropriate than ROC for this task.

2.4.2.2 F-Measure

The most commonly used metric to evaluate abuse detection is the **F-measure** (e.g. [14], [34]–[36], [38], [39]). Let us define a binary classification problem of abusive messages with P positive instances (*Abuse* class) and N negative instances (*Non-Abuse* class). The Precision (Pre) (Equation 2.6) is the proportion of actual abusive messages among all messages detected as abusive by the method:

$$Pre = \frac{TP}{TP + FP}. \tag{2.6}$$

The Recall (Rec) is the proportion of abusive messages that are retrieved among all existing abusive messages:

$$Rec = \frac{TP}{TP + FN}. \tag{2.7}$$

The F-measure (Equation 2.8) is the harmonic mean of the precision and recall:

$$F1 = 2 \times \frac{Pre \times Rec}{Pre + Rec}. \tag{2.8}$$

The macro F-measure is the unweighted arithmetic mean of all the per-class F-measures. This metric handles all classes equally, even in case of unbalanced dataset. In a classification problem with n classes, it is defined as:

$$macro\ F1 = \frac{\sum_{c=1}^{n} F1_c}{n}. \tag{2.9}$$

The F-measure is by far the most commonly used metric to assess abuse detection. To ease the comparison with other studies, we decide to express all our performances in terms of macro F-measure in the remaining of this thesis. We use the macro averaged variant of this metric because the SpaceOrigin dataset that we use is unbalanced with only 25% of abusive messages.

2.5 Conclusion

In this chapter, we first defined what constitutes an abuse and the concepts related to their detection on online platforms. We then reviewed the abuse detection methods proposed in the literature. Afterwards, we gave an overview of the existing datasets of messages annotated for abuse detection. Almost all of them focus on individual messages rather than conversations, while

the literature points out that the conversational context is important to detect abusive comments. We presented the WAC corpus, a dataset of annotated conversations that we developed to mitigate this issue, but that exhibits certain limitations making it unsuitable in our case, and the SpaceOrigin corpus, which we use in the rest of the thesis. Finally, we defined the task and presented the metrics used in the abuse detection literature including the macro F-Measure, the metric that we use in the rest of this thesis.

As we saw in the survey of existing methods, leveraging multiple sources of information seems important in the abuse detection task. This motivates us to propose our own approaches, based on multiple modalities, i.e. text and graphs. As explained in Section 2.2, there are 2 main types of methods: those relying on textual content and those using contextual information. In this thesis, we focus on the conversational context, which we model through a graph representing the structure of the conversation. We propose to explore both aspects of this dual information by comparing and combining text- and graph-based methods. As a first step, we manually engineer features to train a classifier. On the one hand, this strategy is quite standard in NLP, so we directly present the selected text-related features when presenting our method in Chapter 4. On the other hand, using graph features is not that common when performing graph classification, especially when applied to abuse detection, as no method relying on conversational graphs currently exist. Hence, in the next chapter, we perform a review of the most popular topological metrics proposed in the literature to describe graphs, which we later leverage as classification features in Chapter 4.

Chapter 3

Graph metrics as graph features

Machine learning tasks such as classification and regression require numerical inputs of fixed size. Feature engineering is the process of extracting chosen features from raw data to obtain such numerical representations. Numerous topological measures were proposed over the years to characterize the structure and nature of graphs at different scales and scopes, and can be used as features to describe them. The scale depends on the nature of the characterized entity: vertex, subgraph or graph while the scope corresponds to the information used to characterize this entity: microscopic (interconnection between a vertex and its direct neighborhood), mesoscopic (structure of a subgraph and its direct neighborhood), and macroscopic (structure of the whole graph) [116]. Throughout this thesis, we are interested in 2 types of entity: the graphs, which are the elements that we want to classify, and the vertices, including some that are particularly interesting to represent, such as the author of the message. Furthermore, it is even possible to directly construct a representation of the graph by aggregating all the representations of the graph's vertices. Therefore, we focus only on the vertex and graph measures.

This chapter makes the following contribution:

1. We review the literature of graph metrics that can be used as graph features, focusing on the scales and scopes that are relevant for our abuse detection task. This review includes vertex and graph-focused measures.

It is organized as follows. In Section 3.1, we introduce concepts and notations that are used in the rest of the manuscript. Then, we survey the vertex-focused measures in Section 3.2, and the graph-focused measures in Section 3.3. For both, we distinguish topological measures in terms of scope.

3.1 Definitions and Notations

We start with the definition of a simple graph, i.e. undirected, unweighted, and unsigned.

Definition 3.1 (Simple graph)**.** *A* ***simple graph*** *is a pair $G = (V, E)$ composed of a set of vertices V and a set of edges $E \subseteq V^2$ connecting them. Each edge is a pair of vertices placed in an arbitrary but constant order (e.g. lexicographic order).*

We note $n = |V|$ the number of vertices and $m = |E|$ the number of edges. It is possible to add some information over the edges to enhance the graph structure.

Definition 3.2 (Directed graph)**.** *A* ***directed graph*** *is a pair $G = (V, E)$ composed of a set of vertices V and a set of directed edges $E \subseteq V^2$ connecting them. Unlike the undirected graph, both (u, v) and (v, u) can exist in such graphs.*

Definition 3.3 (Weighted graph)**.** *A* ***weighted graph*** *is a triple $G = (V, E, w)$ composed of a set of vertices V, a set of edges $E \subseteq V^2$ connecting them, and a function $w : E \to \mathbb{R}$ that associates a weight to each edge.*

For the sake of simplicity, we note w_{uv} the weight of edge (u, v). It is possible to have a graph that is both directed and weighted.

Definition 3.4 (Signed graph)**.** *A* ***signed graph*** *is a triple $G = (V, E, s)$ composed of a set of vertices V, a set of edges $E \subseteq V^2$ connecting them, and a function $s : E \to \{-, +\}$ that associates a sign to each edge. We note $E^- \subset E$ and $E^+ \subset E$ the subsets of negative and positive edges, respectively. Consequently, $E = E^- \cup E^+$.*

A signed graph can also be directed and/or weighted. The graphs can thus be just signed, but also signed directed, signed weighted, or signed directed weighted.

Definition 3.5 (Adjacency matrix)**.** *The* ***adjacency matrix*** *of a graph with vertex set $V = \{v_1, ..., v_n\}$ is a square $n \times n$ binary matrix $\mathbf{A}$ whose elements $\mathbf{A}_{ij}$ take the value 1 if an edge connects the vertex v_i to the vertex v_j and 0 otherwise.*

The adjacency matrix is symmetric in undirected graphs, but not necessarily in directed graphs.

Definition 3.6 (Neighborhood)**.** *The* ***neighborhood*** *$N(u)$ of a vertex $u \in V$ includes all vertices attached to this vertex: $N(u) = \{v \in V : (u, v) \in E\}$.*

In directed graphs, the neighborhood is split in 2: the *incoming* neighborhood and the *outgoing* neighborhood.

Definition 3.7 (Incoming and outgoing neighborhoods)**.** *In a directed graph, the* ***incoming*** N^{in} *and* ***outgoing*** N^{out} *neighborhoods of a vertex u are the sets of vertices connected to u through incoming and outgoing edges, respectively:* $N^{in}(u) = \{v \in V : (v,u) \in E\}$ *and* $N^{out}(u) = \{v \in V : (u,v) \in E\}$.

In undirected signed graphs, one can distinguish between 2 neighborhoods, depending on the edge signs:

Definition 3.8 (Positive and negative neighborhoods)**.** *The* ***positive*** $N^+(u)$ *and* ***negative*** $N^-(u)$ ***neighborhoods*** *of a vertex $u \in V$ focus each on a specific edge sign:* $N^{\pm}(u) = \{v \in V : (u,v) \in E^{\pm}\}$.

The unsigned neighborhood of vertex u is thus defined as the union of its positive and negative neighborhoods $N(u) = N^+(u) \cup N^-(u)$. In case of *directed* signed graph, the neighborhood of vertex u is split in 4 depending on edge sign and direction: $N(u) = N^{in+} \cup N^{in-} \cup N^{out+} \cup N^{out-}$.

Definition 3.9 (Shortest path)**.** *In an unweighted undirected graph, the* ***shortest path*** *between 2 vertices is the path that connects them with the fewest edges. In a weighted graph, it is the path of minimal total weight.*

Definition 3.10 (Distance)**.** *In an undirected graph, the* ***distance*** *between 2 vertices is the number of edges in the shortest path between them. In a directed graph, we consider the oriented paths. We note $d(u,v)$ the distance between u and v.*

3.2 Vertex-Focused Topological Measures

Vertex measures are defined relatively to a single vertex of the graph. We first describe the microscopic measures that characterize a vertex depending on its direct neighborhood (Section 3.2.1). Then, we explain the macroscopic measures which operate on the entirety of the graph (Section 3.2.2). Finally, we focus on the mesoscopic measures relying on an intermediate structure to characterize vertices (Section 3.2.3).

3.2.1 Microscopic Measures

The **Degree Centrality** measures the size of the direct neighborhood of the considered vertex v as:

$$k(v) = |N(v)|. \tag{3.1}$$

In directed and signed graphs, the **incoming** (Equation 3.2), **outgoing** (Equation 3.3), **positive** and **negative** (Equation 3.4) **degree centralities** of a vertex are defined as the cardinalities of the corresponding neighborhoods as:

$$k^{in}(v) = |N^{in}(v)|, \tag{3.2}$$

$$k^{out}(v) = |N^{out}(v)|, \tag{3.3}$$

$$k^{\pm}(v) = |N^{\pm}(v)|. \tag{3.4}$$

The **Strength Centrality** is the generalization of the degree to weighted graphs. It is based on the sum of the weights of the edges linked to the considered vertex. The strength of vertex v is defined as:

$$Str(v) = \sum_{u \in N(v)} w_{uv}. \tag{3.5}$$

Once again, it is possible to use the incoming (Equation 3.6) and outgoing (Equation 3.7) versions of that measure in a directed graph. Compared to the degree centrality, the strength takes into account the frequency or intensity of the interactions in the following way:

$$Str^{in}(v) = \sum_{u \in N^{in}(v)} w_{uv}, \tag{3.6}$$

$$Str^{out}(v) = \sum_{u \in N^{out}(v)} w_{vu}. \tag{3.7}$$

The **Local Transitivity** [117] (aka local clustering coefficient) corresponds to the proportion of the considered vertex's neighbors connected through an edge, relatively to the total number of connections possible among them. It is a score ranging from 0 (no inter-neighbor edge at all) to 1 (the vertex and its neighborhood are completely interconnected and form a clique). The measure is originally defined for unweighted undirected graphs:

$$LT(v) = \frac{2|\{(u,z) \in E : u, z \in N(v)\}|}{k(v)(k(v)-1)}, \tag{3.8}$$

where the degree can be reformulated as $k(v) = \sum_u \mathbf{A}_{uv}$. A weighted variant based on the strength centrality was proposed [118]:

$$LTw(v) = \frac{1}{Str(v)(k(v)-1)} \sum_{u,z} \frac{w_{uv} + w_{vz}}{2} \mathbf{A}_{uv}\mathbf{A}_{vz}\mathbf{A}_{uz}. \tag{3.9}$$

Burt's Constraint [119] measures how redundant the neighbors of a vertex are. Vertices with high constraint values are said to be constrained by their neighbors, i.e. they stand in a highly cohesive group of vertices. On the other hand, vertices with a low constraint value have a higher reach outside their direct neighbors pool. This measure is based on the proportion of the relations of some vertex v that are invested in connection with another vertex u, compared to the total weight that v shares with its neighbors. It is expressed as:

$$p_{vu} = \frac{\mathbf{A}_{vu} + \mathbf{A}_{uv}}{\sum_{z \in V \setminus \{v\}} (\mathbf{A}_{vz} + \mathbf{A}_{zv})}. \tag{3.10}$$

Burt's constraint (Equation 3.11) is composed of 2 parts: the direct influence p_{vu} and the indirect influence $\sum p_{vz} p_{zu}$ which measures the amount of weight indirectly shared by v and u through a mutual neighbor z. It is defined as:

$$BurtC(v) = \sum_{u \in V \setminus \{v\}} \Big(p_{vu} + \sum_{z \in V \setminus \{u,v\}} p_{vz} p_{zu} \Big)^2. \tag{3.11}$$

3.2.2 Macroscopic Measures

Macroscopic measures operate on the entirety of the graph structure. They allow characterizing the position of a particular vertex relatively to the whole graph. We can distinguish 3 families of macroscopic measures: the spectral measures (Section 3.2.2.1), the distance-based measures (Section 3.2.2.2) and the connectivity-based measures (Section 3.2.2.3).

3.2.2.1 Spectral Measures

The spectral measures are based on the Eigenvectors or Eigenvalues of the graph adjacency matrix, or any related matrix.

The **Eigenvector Centrality** [120] is another generalization of the degree, in which instead of just counting the direct neighbors, one also takes into account their own centrality:

$$EigC(v) = \frac{1}{\lambda} \sum_{u \in N(v)} EigC(u) = \frac{1}{\lambda} \sum_{u \in V} \mathbf{A}_{vu} EigC(u), \tag{3.12}$$

where $\lambda \neq 0$ is a constant. A central neighbor contributes more to the centrality than a peripheral one. A high Eigenvector value indicates a vertex that is connected to many vertices who themselves have high values. Using matrix notation, the Eigenvector centrality can be rewritten as the Eigenvector equation:

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}, \tag{3.13}$$

with $\mathbf{x} = \Big(EigC(v_1), ..., EigC(v_n) \Big)$, and λ is the largest Eigenvalue of A.

The **PageRank Centrality** [121] can be seen as a variant of the Eigenvector centrality originally designed for directed graphs, but that can also be applied to undirected graphs. It includes an additional normalization allowing to model the dilution of the influence of a vertex through all its outgoing edges. This is done through the damping factor d, which represents the probability to jump from one vertex to another without having an explicit connection between them. In practice, it is usually set to 0.85 based on empirical results [121]. At each time step t, the computation yields:

$$\mathbf{R}(t+1) = d\mathbf{M}\mathbf{R}(t) + \frac{1-d}{n}\mathbf{1}, \tag{3.14}$$

where $\mathbf{1}$ is the column vector of length n containing only ones and matrix $\mathbf{M}$ is defined as:

$$\mathbf{M}_{uv} = \begin{cases} 1/N^{out}(v) & \text{if } (v,u) \in E \\ 0 & \text{otherwise.} \end{cases} \tag{3.15}$$

This formula can be rewritten recursively as:

$$PR(v;t+1) = \frac{1-d}{n} + d \sum_{u \in N^{in}(v)} \frac{PR(u;t)}{N^{out}(u)}, \tag{3.16}$$

where the first term model the probability of randomly visiting a given vertex and in the second term, vertices equally dilute their influence to all their outgoing neighbors.

The **Hub** and **Authority Scores** [122] are 2 complementary measures for directed graphs, computed through the *Hyperlink-Induced Topic Search* (HITS) algorithm. They split the Eigenvector centrality value into 2 parts. The authority score focuses on incoming edges and the hub score on the outgoing edges. In matrix notation with constant α and β, authority and hub scores are respectively

$$\mathbf{a} = \alpha \mathbf{A}^T \mathbf{h}, \tag{3.17}$$

$$\mathbf{h} = \beta \mathbf{A}\mathbf{a}. \tag{3.18}$$

In their iterative definition, $Hub(v) = Auth(v) = 1$ for all vertices $v \in V$ in the initial step. Then, at each iteration, the authority of all vertices is updated as the sum of the hub score of all its incoming neighbors, divided by the square root of the sum squared authority scores:

$$Auth(v) = \frac{\sum\limits_{u \in N^{in}(v)} Hub(u)}{\sqrt{\sum\limits_{u \in V} Auth(u)^2}}. \tag{3.19}$$

This normalization is needed to obtain values that converge. The hub score of all vertices is updated in the same manner, except that it is the sum of all the authority scores of outgoing

neighbors divided by the square root of the sum of the squared hub scores:

$$Hub(v) = \frac{\sum\limits_{u \in N^{out}(v)} Auth(u)}{\sqrt{\sum\limits_{u \in V} Hub(u)^2}}. \tag{3.20}$$

The **Alpha Centrality**, or Katz Centrality [123], [124], relates to the same idea, but it considers all the walks between pairs of vertices instead of only the shortest paths in directed graphs. This measure computes the relative influence of a vertex within the graph by considering its direct and indirect neighbors, but the latter are penalized by an attenuation factor $\alpha \in (0, 1)$:

$$Alpha(v) = \sum_{\ell=1}^{\infty} \sum_{u=1}^{n} \alpha^{\ell} (\mathbf{A}^{\ell})_{uv}, \tag{3.21}$$

wherein the power of the adjacency matrix ($\mathbf{A}^{\ell}$) denotes the presence (or absence) of a path of length ℓ between 2 vertices.

The **Power Centrality** [125] generalizes both the Eigenvector and Alpha Centralities in directed unweighted graphs. It is based on the notion that the power of a vertex is recursively defined by the sum of the power of its neighbors. It uses an attenuation parameter $\beta \in (-1/\lambda_{\mathbf{A}1}; 1/\lambda_{\mathbf{A}1})$ (the reciprocal of the largest eigenvalue of the adjacency matrix $\mathbf{A}$) which controls the influence of distant vertices and a scaling parameter α. The power centrality is defined as follows:

$$PC = \alpha \left(\mathbf{I} - \beta \mathbf{A}\right)^{-1} \mathbf{A1}, \tag{3.22}$$

or in its iterative form:

$$PC(v) = \sum_{u \in N(v)} (\alpha + \beta PC_u) \mathbf{A}_{vu}. \tag{3.23}$$

The **SubGraph Centrality** [126] defines the notion of reachability based on closed walks in undirected graphs. The subgraph centrality of vertex v is computed as the sum of the closed walks of all lengths starting and ending at v. The contribution of the closed walks decreases as their length increases, making short walks more influential on the centrality of the vertex than longer ones. It is formally defined as:

$$SC(v) = \sum_{\ell=0}^{\infty} \frac{(\mathbf{A}^{\ell})_{vv}}{\ell!}, \tag{3.24}$$

where the numerator is the local spectral moments, which is defined as the v^{th} diagonal entry of the ℓ^{th} power of the adjacency matrix $\mathbf{A}$.

3.2.2.2 Distance-Based Measures

Another family of macroscopic measures is based on the notions of *shortest path* and *geodesic distance* between vertices.

The **Betweenness Centrality** [127] is related to the number of shortest paths going through the considered vertex. For every pair or vertices in a connected graph, there exists at least one shortest path connecting the vertices. In an undirected unweighted graph, the betweenness centrality of a vertex is the proportion of such shortest paths passing through it:

$$BetC(v) = \sum_{u \neq v \neq z} \frac{\sigma_{uz}(v)}{\sigma_{uz}}, \tag{3.25}$$

where σ_{uz} is the number of shortest paths between u and z, and $\sigma_{uz}(v)$ is the number of those passing through v. In directed graphs, this measure is defined as the proportion of *directed* shortest paths passing through v and in weighted graphs, it considers the paths of *minimal weight*.

The **Closeness Centrality** [128] is related to the reciprocal of the total geodesic distance between the vertex of interest and the other vertices. The more central a vertex is, the higher its closeness centrality. It is generally considered that it measures the efficiency of the vertex to spread a message over the graph, and its independence from the other vertices in terms of communication. The closeness centrality was originally defined as:

$$CloseC(v) = \frac{1}{\sum_u d(u,v)}, \tag{3.26}$$

where $d(u,v)$ is the distance between u and v. In directed and weighted graphs, this distance is the length of the shortest oriented path and the weight of the path of minimal weight, respectively. However, the closeness centrality often refers to its normalized form, where the previous formula is multiplied by the number of remaining vertices in the graph. This normalization eases the comparison of vertices in graphs of different sizes. The formula becomes:

$$CloseCNorm(v) = \frac{n-1}{\sum_u d(u,v)}. \tag{3.27}$$

This measure is generalized to directed graphs by considering the length of directed shortest paths, and to weighted graphs by considering the cost of the paths of minimal cost.

The **Eccentricity** [129] is related to the closeness centrality, but it is not a centrality measure. On the contrary, it quantifies how *peripheral* a vertex is, by considering the distance to its farthest neighbor in the graph. In directed and weighted graphs, this distance is computed on directed and weighted paths, respectively, in the following manner:

$$Ecc(v) = \max_{u \in V} d(u,v). \tag{3.28}$$

It is generalized to directed or weighted graphs using the same principle as for the closeness.

3.2.2.3 Connectivity-Based Measures

The last family of macroscopic measures is based on the notion of connectivity, i.e. whether or not a path exists between certain parts of the graph.

An **Articulation Point** is a vertex whose removal makes the graph disconnected, i.e. split into several separate components [129]. We define a binary feature indicating whether the vertex is an articulation point (1) or not (0).

3.2.3 Mesoscopic Measures

Mesoscopic measures rely on an intermediate structure to characterize a vertex, such as a subgraph.

The **Coreness Score** [130] is related to the k-core of the graph which is the maximal connected subgraph in which every vertex has at least degree k. The coreness score of vertex v is k if it belongs to the k-core but not to the $(k+1)$-core. Ingoing cores and outgoing cores can be used to compute this score in directed graphs.

The **Within-Module Degree** (or internal intensity) and **Participation Coefficient** [131] are a pair of complementary measures defined relatively to the community structure of the graph. They aim at characterizing the position of vertices at this intermediate level. They can be computed in undirected and directed graphs by considering the oriented paths in the latter [132]. The within-module degree measures the connectivity of a given vertex to other vertices in its community. Let $k_i(v)$ be the *community degree* of v for the i^{th} community, i.e. the number of edges that v shares with vertices in this community. The within-module degree is defined as the z-score (or statistical standardization) of $k_i(v)$:

$$WM(v) = \frac{k_i(v) - \mu_i[k_i(v)]}{\sigma_i[k_i(v)]}, \tag{3.29}$$

where $\mu_i[k_i(v)]$ is the average of k_i over all vertices in community C_i, and $\sigma_i[k_i(v)]$ is its standard deviation.

The participation coefficient measures the strength of a vertex's connections within its community. It is expressed as:

$$PC(v) = 1 - \sum_{1 \leq i \leq K} \left(\frac{k_i(v)}{k(v)}\right)^2, \tag{3.30}$$

where K is the number of communities in the graph.

A collection of 4 measures have been proposed to decompose community roles in a more precise way, and better capture the position of vertices in a modular graph [132]: **Internal Intensity**, **External Intensity**, **Diversity**, and **Heterogeneity**. They all rely on the notion of z-score shown in Equation 3.29 for the community degree $k_i(v)$.

The *internal intensity* is just another name for the within-module degree. The *external intensity* measures the amount of edges that a vertex have towards communities other than its own. It is based on the *external degree* $k_{ext}(v)$ of a vertex v, which corresponds to the total community degree over all communities except its own. The external intensity is the z-score of the external degree, which is obtained formally by replacing $k_i(v)$ by $k_{ext}(v)$ in Equation 3.29.

The *diversity* relies on the number of communities to which a vertex v is connected, outside of its own. Let $\epsilon(v)$ be this number. The diversity is defined as the z-score of $\epsilon(v)$, obtained by substituting $k_i(v)$ by $\epsilon(v)$ in Equation 3.29.

The *heterogeneity* of vertex v measures the variation of the number of edges a vertex has, from one community to another. It requires computing the standard deviation of the number of edges that u has to each community, denoted by $\delta(v)$. The heterogeneity is the z-score of δ, once again obtained by substituting $\delta(v)$ to $k_i(v)$ in Equation 3.29.

These measures can be extended to directed graphs by distinguishing incoming and outgoing links. This results in 2 versions of each measure, the in- and out- measures.

3.3 Graph-Focused Topological Measures

Though less numerous than vertex-focused measures, some measures have been specifically designed to characterize the graph structure as a whole. As for the vertex measures, one can distinguish them based on their scope.

3.3.1 Microscopic Measures

A number of very standard statistics exist to describe a graph using only local information. The **vertex count** or graph order, is the total number of vertices in the graph: $n = |V|$. The **edge count** or graph size is the total number of edges in the graph: $m = |E|$. In a signed graph, it can be splited in a positive count (m^+) and a negative count (m^-). The graph **density** represents the ratio between the number of edges present in a graph and the maximum number of edges that this graph can contain. It is defined as:

$$D = \frac{2m}{n(n-1)}. \tag{3.31}$$

The **Global Transitivity** [133], also known as global clustering coefficient, is the graph-focused counterpart of the local transitivity. It corresponds to the ratio of the count of triangles to connected triples (i.e. 3 vertices that are connected by either 2 or 3 edges in the graph). This measure is related to the proportion of vertices possessing a common neighbor that are directly connected.

The **Reciprocity** [134] measures the proportion of bilateral edges (i.e. edges pointing in both directions) over all pairs of vertices in a *directed* graph. Formally, the reciprocity of a graph G is written:

$$Rec = \frac{\sum_{(u,v)\in G} a_{uv}a_{vu}}{|E|}. \tag{3.32}$$

3.3.2 Macroscopic Measures

The **Weak Component Count** corresponds to the number of maximally connected subgraphs in an undirected graph. A subgraph is connected if there exists a path between any given pair of vertices. The **Strong Component Count** is its variant for directed graphs, which is based on directed paths. A directed graph is called strongly connected if there is a directed path in each direction between any given pair of vertices.

The **Cohesion** of a graph is the minimum number of vertices one needs to remove in order to make the graph disconnected [135]. The **Adhesion** is a similar measure, but for edges.

When describing the vertex-oriented measures, we defined an articulation point as a vertex whose removal makes the graph disconnected. The **Articulation Point Count** is the total number of articulation points in the graph:

$$APC = \sum_{v\in V} AP(v), \tag{3.33}$$

where $AP : V \to \{0, 1\}$ associates 1 to a vertex if it is an articulation point and 0 otherwise.

The **Diameter** is the length of the *longest shortest path* (i.e. the largest distance) over all pairs of vertices (indirectly) connected in the graph. It corresponds to the maximum eccentricity of any vertex in the graph:

$$dia = \max_{v\in V} Ecc(v). \tag{3.34}$$

The **Radius** is the minimum among all the maximum distances over all pairs of vertices (indirectly) connected in the graph. It corresponds to the minimal eccentricity of any vertex:

$$rad = \min_{v\in V} Ecc(v). \tag{3.35}$$

The **Average Distance** is the arithmetic mean of the lengths of the shortest paths processed over all pairs of (indirectly) connected vertices. Let $d(u, v)$ be the length of the shortest path

between u and v. The average distance is thus:

$$AvgDist = \frac{\sum_{(u,v)\in V^2} d(u,v)}{\frac{n(n-1)}{2}}. \tag{3.36}$$

3.3.3 Mesoscopic Measures

A clique is a subset of vertices such that every 2 distinct vertices in the clique are adjacent. In other words, a clique beloging to a graph G is a complete induced subgraph of G. The **Clique Count** is the total number of cliques in G.

The **Community Count** and the **Modularity** [136] are 2 measures related to the notion of community structure, i.e. of partition of vertex set reflecting the network mesoscopic organization. The former is the number of communities, while the latter is a measure of the quality of the community structure defined as:

$$Q = \sum_i (e_{ii} - a_i^2). \tag{3.37}$$

There are a very large number of different methods to detect such community structures, but their discussion is out of the scope of this thesis. See for instance [137] for a review.

3.4 Conclusion

With the graph measures presented throughout this chapter, it is possible to characterize the structure of a graph. Each one of them capture particular information and their multiplication allows to better characterize the entire graph. In the next chapter we want to use these measures in order to represent conversational graphs. This procedure known as *feature engineering* which has long been the standard approach allows us to obtain graph representations that have the benefit of being interpretable (i.e. based on known and identified characteristics). We can then feed them to a classifier in order to detect abusive messages in conversations, while identify the most discriminant feature in order to provide some interpretation to our results.

Chapter 4

Feature Engineering for Abuse Detection

In this chapter, we propose abuse detection methods based on text, graphs and an approach combining these 2 modalities of information. The methods using text and graphs individually through feature engineering are largely based on the work conducted in our research team by Papegnies *et al.* [6], [103]. The first one, called *content-based* method, relies on the content of the exchanged messages to perform abuse detection. The second one, called *graph-based* method, focuses on the interactions between participants of the conversation modeled by a conversational graph, and uses the graph measures described in Chapter 3. It significantly differs from the rest of the literature, as it completely ignores the content of the messages, and only focuses on the dynamics of the conversation.

These 2 types of approach perform reasonably well when taken separately, but given the different nature of text and graphs, one can assume that the content of the exchanged messages and the interactions between users convey different information which is at least partially complementary, hence improving the classification when combined. Mishra *et al.* [14] actually achieves better performances when combining these 2 modalities. To investigate this phenomenon, we propose 3 *Fusion* strategies to take advantage of both content- and graph-based methods.

To obtain numerical representations of text and graphs, both of our method use a standard approach adopted in the literature, which consists in selecting and extracting relevant measures from the raw data through a *Feature Engineering* process. For the methods based on the content of messages, this approach has been used in a number of works, e.g. [11], [33], [40]. Therefore, the content-based method uses a set of features that are quite standard in the literature of abusive language detection. On the other hand, there is no previous work that uses conversational graphs to handle this task. Hence, the adopted approach is exploratory and consists in selecting a huge range of widespread topological measures to describe the graphs. The objective is to have a large set of measures, including some that can handle edge signs, directions and weights, to increase the probability of capturing the important information carried by the graph structure. Finally, we propose to perform a feature ablation study on all the modalities and their combination to determine the best suited measures relatively to our task.

Our work differs from the original works proposed by Papegnies *et al.* [6], [103] on several points. First, we seek to compare the 2 sources of information and study their complementarity by combining them. Second, we extend the original graph extraction method to create signed graphs, which require us to include signed features in the graph-based method. Finally, we propose an extended analysis of our results.

This chapter is based on our work published at the *International workshop on Modeling and mining Social-Media-driven Complex Networks* [20]. The conversational graphs extracted from our proprietary dataset[1] and the source code[2] to reproduce our experiments are available online. This chapter makes the following contributions:

1. We develop a procedure to extract directed signed weighted conversational graphs from user-generated conversations.

2. We present and compare a *Content-based* and a *Graph-based* method on an abuse detection task. We propose 3 *Fusion* strategies to combine these methods and take advantage of both sources of information.

3. We perform a feature ablation study on text and graph modalities, and their combination, to identify the most important features for our task. Furthermore, we analysis how the temporal aspect and the edge attributes can impact the graph-based method.

The rest of this chapter is organized as follows. First, we present the methods that we use to extract conversational graphs from our dataset of conversations in Section 4.1. In Section 4.2, we introduce the methods and fusion strategies used through this chapter. In Section 4.3, we present

[1] https://figshare.com/articles/dataset/Conversational_Networks_For_Automatic_Online_Moderation/7442273?file=24681479

[2] https://github.com/CompNet/Alert

our experimental setup and put it into practice on the SpaceOrigin dataset. We extensively interpret the obtained results and determine the most important features on our task for each method. We further analyze some aspects related to graphs and discuss our findings regarding the context in Section 4.4. Finally, we review our main findings and identify some perspectives for future works in Section 4.5.

4.1 Graph Extraction from Conversations

Graphs can be used to represent conversations under the form of so-called conversational networks, which represent the flow of the conversation between persons. These persons are represented as vertices and the edges model the communication between them. While extracting conversational networks from well-structured conversations is straightforward, it can be far from trivial for unstructured conversations such as online chatrooms or IRC (Internet Relay Chat). Indeed, these multi-participant chats often have multiple distinct conversations that overlap. There can be a built-in mechanism to explicitly specify the recipient of a message, but it is rarely used by users, for time and practicality reasons. In this context, Mutton [138] proposes to use the temporal density and temporal proximity of the messages to build conversational networks. In other words, he considers that messages posted consecutively in the chat have a higher probability of being related than distant messages. We build an extraction framework based on this hypothesis.

In Chapter 2.3, we presented several datasets for abuse detection. For our experiments, we want to extract graphs from the selected dataset. Therefore, we require a dataset of conversations, with conversations long enough to have a variety of users and messages in each of them. The SpaceOrigin dataset is the one that best suits our requirements, so we use it in all the following experiments. As a reminder, it contains chat logs extracted from the in-game live conversation module of an online video game. It is composed of 2,545 messages labeled as *Abusive* or *Non-Abusive*, with each one being provided with its surrounding context. Since one of the method that we use in this chapter is based on conversational graphs to model the context of conversations, we have to extract such networks from the dataset. These networks are directed signed weighted graphs to exploit all the information available. Vertices and edges represent users and their communications, respectively. The following graph extraction procedure was first proposed by Papegnies *et al.* [6]. Their procedure is able to extract directed weighted graphs. We propose to extend it with a sentiment analyzer to infer the polarity of the interactions between users. This allows us to add signs to interactions and thus to extract directed signed weighted networks.

The key concepts of the graph extraction framework from a conversation are illustrated in Figure 4.1, in which each vertical rectangle represents a message. The messages are ordered from oldest to newest, and time flows from left to right in the figure. Each extracted network is defined relatively to a *targeted message*, i.e. a labeled message of the SpaceOrigin dataset that we use

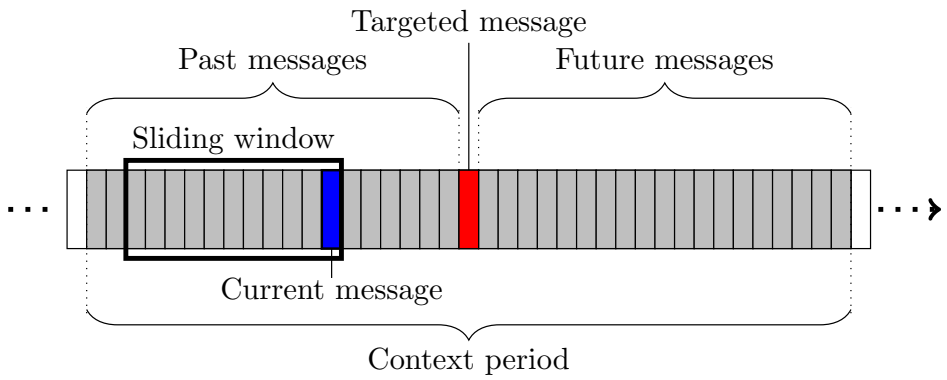

Figure 4.1: Illustration of the main concepts of our conversational network extraction process. Figure from [6].

in our classification task. Furthermore, labeled messages were originally extracted so that they do not appear in each other's context. There are 3 steps in the extraction framework:

- First, we select a subset of messages that we use to construct the network (the context period in Figure 4.1).
- Second, we identify the subset of users that are the likely receivers of each message (represented by the sliding window in Figure 4.1).
- Third, we compute the polarity of an exchange between 2 users based on the content of the exchanged message. We also compute the strength of the interaction depending on the proximity between the exchanged messages. We add edges and revise their weights and signs accordingly.

Context period. Each *targeted message* in the SpaceOrigin dataset is associated with its context that can be composed of thousands of messages. Therefore, we have to determine the subset of messages to use in order to extract the network associated to this *targeted message*. To this end, we restrict the extraction process to a so-called *context period*, a sub-sequence of messages. This sequence is centered around the *targeted message* which is represented in red in Figure 4.1. The *context period* spans symmetrically *before* (left side) and *after* (right side) the *targeted message*. Each participant posting at least one message during this period is modeled by a vertex in the produced conversational network.

Sliding window. Our dataset in composed of a series of messages. Users can write a direct reference to another user as `@user` to explicitly direct a message towards him. However, most of the messages do not contain such references. Therefore, we have to infer the recipients of messages.

We do so by sliding a mobile window over the whole *context period*, one message at a time. At each step, the network is updated either by creating new edges, or by updating the weights of existing ones. The *sliding window* has a fixed length expressed in number of messages, which is derived from ergonomic constraints relative to the online conversation platform from which the dataset was extracted. At a given time, the rightmost message of the window, i.e. the oldest one posted (in blue in Figure 4.1), is called *current message* and its author *current author*. This procedure allows focusing on a smaller part of the context period. Following the hypothesis that consecutive messages have a higher probability of being related than distant messages, we assume that the *current message* is aimed at the authors of the other messages posted right before it (i.e. the messages present in the sliding window). It is also possible that the current message contains direct references to users as `@user`. In the case of a direct reference to one or multiple users, we consider that the referenced users are the prime targets of the message. We connect the current author to the receivers (or strengthens their weights if the edge already exists) according to the process described next

Weight assignment. To assign weight to a connection between users, we list and order all the users that interact with the current author, excluding the current author himself. This list first contains the users directly referenced in the message, if there are any (step *a* in Figure 4.2). Then, we add all the authors of messages currently present in the *sliding window* except the ones that are already among the referenced users. They are ordered by their last posted message to take chronology into account (step *b* in Figure 4.2). Only the edges towards users in that list receive weight. We want to favor the first authors in the list as they are more likely to be the targeted receivers of the *current message*. The weights are adjusted depending on the proximity of the concerned receivers: the higher the proximity, the stronger the interaction. For this purpose, we use a recursive scoring function to assign weights to the connections. Each receiver is assigned a score which is a decreasing function of both his rank i in the ordered list of receivers and of the length N of this list. This function is defined in Equation 4.1:

$$f(i) = \begin{cases} 0.6 \times 0.4^{i-1}, & \text{if } 1 \leq i < N \\ 0.4^{i-1}, & \text{if } i = N. \end{cases} \tag{4.1}$$

The first receiver gets 60% of the total weight, and the rest of them share the remaining 40% using the same recursive 60-40% split scheme. This approach gives more importance to temporal proximity, with scores dropping fast when the receiver is not the author of the immediately preceding message. The resulting weights are represented on the right-hand side of Figure 4.2. Papegnies *et al.* [6] experimented with 2 other weight assignment strategies, but they were less effective than the recursive scoring function described above, so here we focus on this last one.

Relation polarity. In order to obtain signed edges, we additionally leverage a state-of-the-art

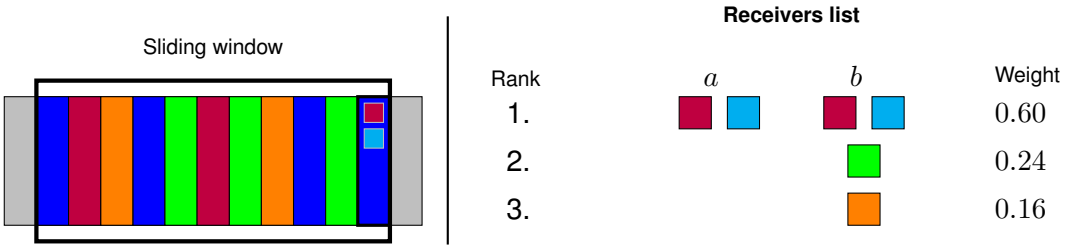

Figure 4.2: Example of sliding window (left) and computation of the corresponding receivers' weights (right). Each color represents a specific user. On the left, each message in the window is filled with the color of its author, whereas the small squares represent direct references to users. On the right, the a and b columns represent the different steps of the computation. See main text for details.

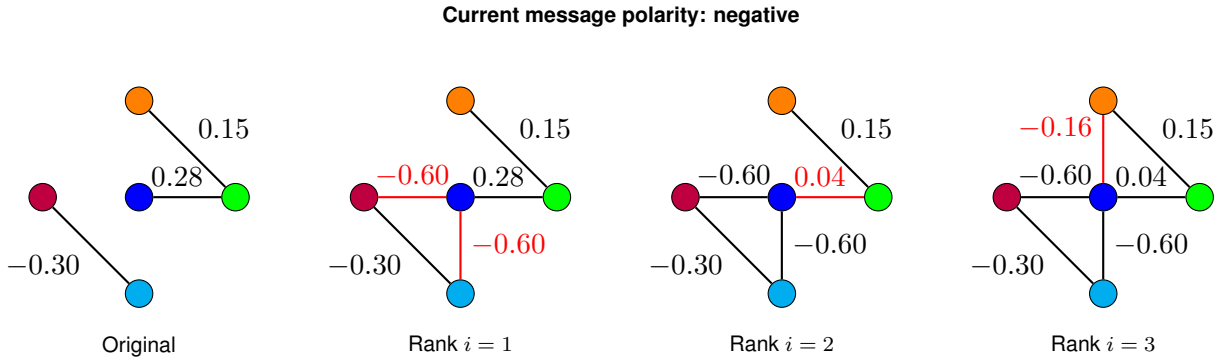

Figure 4.3: Update of the edges and weights of the conversational graph corresponding to the example in Figure 4.2. We consider that the sentiment analyzer determined a negative polarity for the current message. All the weights added at this step are therefore negative. The first graph displays the state before the update, and each remaining one corresponds to one rank in the receiver list.

BERT-like sentiment analyzer[3] to determine the polarity of the users' interactions, based on their exchanged textual content. More specifically, the analyzer is applied to the *current message*. The polarity thus computed can be either negative (hostile interaction) or positive (friendly) and is used as the sign of the weight computed at the current step. Figure 4.3 illustrates this weight update following the weights computed in Figure 4.2. In this illustration, we consider a negative polarity for the current message. The total weight of an edge is obtained by summing all its interactions over the whole context period, and can thus be negative or positive. The sign of this summed weight is used as the sign of the edge. This last part is an original proposition compared to the extraction process proposed by [6].

[3] https://github.com/TheophileBlard/french-sentiment-analysis-with-bert

Once this iterative framework has been applied over the whole *context period*, we obtain the *Full* network associated to one labeled message of the original dataset. We repeat this operation for each of the 2,545 labeled messages. Besides the *Full* network extracted over the whole context period, we additionally extract 2 smaller networks. We split the *context period* in 2, right in the middle, with only the *targeted message* included in both parts. We extract the *Before* network from the messages posted in the first half and the *After* network from the messages posted in the second half. In a live classification setting where only past information is available, one would only be able to use the *Before* network. However, in a more general setting we can use all 3 networks (*Full*, *Before*, *After*). Figure 4.4 shows an example of such networks obtained for a message of the corpus. The disconnected vertices in the *Before* and *After* networks correspond to users present in the context period, but active only before or after the *targeted message*.

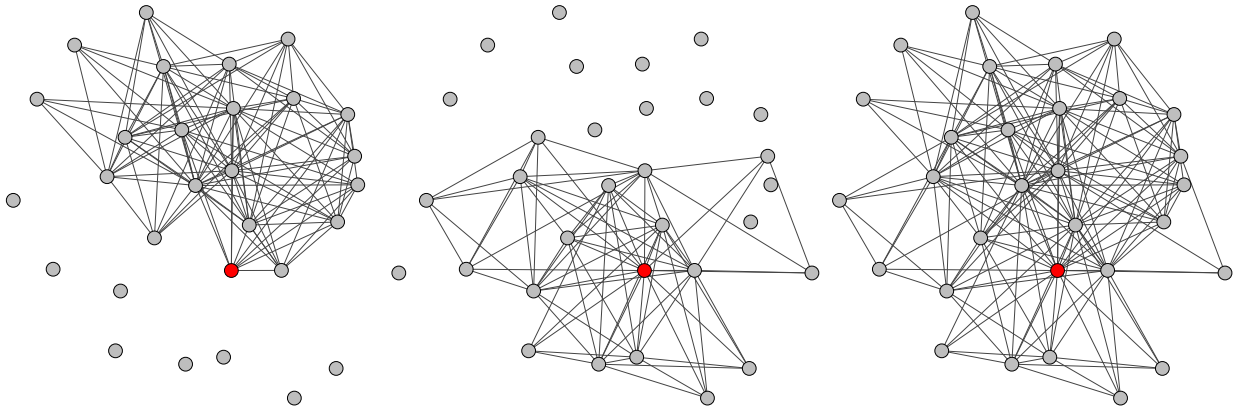

Figure 4.4: Example of the 3 types of conversational networks extracted for a given context period: *Before* (left), *After* (center), and *Full* (right). The author of the targeted message is represented in red.

4.2 Proposed Representation Methods

In this section, we describe the feature engineering approaches that we use to tackle the problem of detecting abusive messages in conversations. We first summarize the text- (Section 4.2.1) and graph-based (Section 4.2.2) approaches. We then describe the fusion methods that we propose, aiming at taking advantage of both sources of information (Section 4.2.3). Figure 4.5 shows the whole process discussed through this section.

4.2.1 Text-Based Features

One way of representing a word is to characterize it with appropriate measures selected through a *Feature engineering* process. This procedure is heavily task- and context-dependent, and must

therefore be performed specifically for each application. For instance, one can assume that the ratio of capital letters must be a key element to detect spam, while the presence of smileys is more important in sentiment analysis. With a carefully designed set of measures, it is possible to obtain representations of text that are suitable for downstream tasks such as text classification [139], keyword extraction [140] or text categorization [141]. There exist a huge range of features that can be used to characterize a word. Some of them are based on the morphology of words and some are based on the language. We use a selection of text-based features oriented for the representation of abusive messages, previously used in the literature [11], [142], [143]. We describe them in the remaining of this section.

4.2.1.1 Morphological Features

Morphological features are characteristics related to the shape of the message.

Message length: The length of the message expressed in number of characters. The intuition is that abusive messages are often either very short or very long which might be symptomatic of a massive copy/paste.

Word length: The average and maximum word length in the message expressed in number of characters.

Characters count: The number of characters in the message. We distinguish 5 classes of characters (letters, digits, punctuation, spaces, and others) and compute 2 features for each one: the number of occurrences and the proportion of that class of characters in the message.

Unique characters: The number of unique characters in the message. In conjunction with the previous features, it can allow to detect spam messages with a single character overly emphasized (e.g. "fuuuuuuuuuuuuuuuuuuuuuuuuuuuck").

Capital letters: The number and ratio of capital letters in the message. Abusive users tend to use a lot of capital letters to emphasize their messages.

Compression ratio: Abusive messages often contain a lot of copy/paste. To deal with such redundancy, we apply the Lempel-Ziv-Welch (LZW) compression algorithm [144] to the message and take the ratio of its raw to compress lengths, expressed in characters. This feature highlights users that tend to repeat exactly and multiple times the same text.

Collapsed characters: Abusive messages also often contain extra-long words, which can be identified by collapsing the message, i.e. extra occurrences of letters repeated more than 2 times consecutively are removed. For instance, looooooool would be collapsed to lool. We compute the difference between the raw and collapsed message lengths.

4.2.1.2 Language Features

Language features are associated to a higher granularity of text than morphological features. They are related to words in the message.

Number of words: Intuitively, a message with more words and diverse ones is more likely to be constructive. People that verbally abuse other users rarely take the time to elaborate. We compute the total number of words and the number of unique words in the message.

Bad words: Even though there are obfuscations techniques to hide abusive words, many abusive words can still be detected with a simple predefined list of insults and symbols considered as abusive. We count the number of occurrences in the message. We also count them in the collapsed version of the message. In this application, we use a list of French and English bad words as our dataset is in French. We created this list from dictionaries of insults and swear words.

Bag-of-Words (BoW): BoW is a model that uses the frequency of words in a corpus to convert a piece of text into fixed-length vectors. As the name suggests, it puts words in a "bag" and compute the frequency of every word. Put differently, a textual document is represented by all the words contained in it together with the frequency of each word. Each dimension of the vector corresponds to a word from the corpus, and the associated value is the number of times this word appears in the document. This means that the dimension of the vectors depends on the size of the vocabulary. This causes the same scalability issues as with the one-hot encoding. The frequency count of words used in this model helps compare and contrast documents. However, with such a "bag" representation, any information about the order of words in the original text is lost. Hence, the sentences "I love cats and hate dogs" and "I love dogs and hate cats" obtain similar BoW representations.

We lower-case the text and strip punctuation, in order to represent the message as a basic BoW. We then train a Naive Bayes classifier to detect abuse using this sparse binary vector (as represented in the very bottom part of Figure 4.5). The output of this simple classifier, in the form of a score, is then used as an input feature for the SVM classifier.

TF-IDF scores: Term Frequency-Inverse Document Frequency determines how relevant a word is to a document in a corpus. It adjusts for the fact that some words appear more frequently in general by applying a scaling factor to the term frequency. The TF-IDF value increases proportionally to the number of times a word appears in a document, and is offset by the number of documents in the corpus that contain that word. Formally, this approach comprises 2 metrics, Term Frequency (TF) and Inverse Document Frequency (IDF). TF is the ratio of the number of occurrences of a word in a document to the total number of words in that document. It is formalized as

$$TF(t,d) = \frac{f_{t,d}}{\sum\limits_{t' \in d} f_{t',d}}, \tag{4.2}$$

where $f_{t,d}$ is the count of a term t in document d, and the denominator is the total number of terms in d.

The IDF score measures the rarity of words in the text. It is the log ratio of the number of documents in the corpus N to the number of documents with the term in them D:

$$IDF(t, D) = log\frac{N}{D}. \tag{4.3}$$

The TF-IDF score of a document is the product of the 2 previous metrics (Equation 4.4). This method does not provide direct representations of words, but rather a score that indicates how important a term is in a document. This score can then be used among other measures in a feature engineering process to represent a word. It is defined as:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D). \tag{4.4}$$

We compute 2 overall TF-IDF scores corresponding to the sums of the standard TF-IDF scores of each individual word in the message. One is processed relatively to the *Abuse* class, and the other to the *Non-abuse* class. We proceed similarly with the collapsed message.

4.2.2 Graph-Based Features

To characterize the extracted conversational graphs, we use the measures formally defined in Section 3. We use the (un)weighted, (un)directed, (un)signed versions of all these measures when possible. Table 4.1 gives an overview of all the measures and their variants that we use. In this section, we give an interpretation of these measures in the context of abuse detection to understand why they can be useful for this task.

4.2.2.1 Vertex-Focused Topological Measures

We compute the vertex-focused measures for the vertex corresponding to the author of the targeted message (represented in red in Figure 4.4). The intuition is that measures related exclusively to this vertex could bring information that is not altered by other vertices. Hence, they could capture more precise information on the behavior of this particular author in the conversation.

Microscopic measures: In our application, microscopic measures are used to characterize the position of some user depending on its direct interlocutors. The **degree centrality** directly depicts the number of users that have exchanged (undirected version), received or sent (directed version) messages to the author. The **strength centrality** is the generalization of the degree to weighted graphs, which allows taking into account the frequency of the interactions. This is particularly important to describe authors with few interlocutors but strong interactions with them. Some users play a more central role than others in a community. As abusive users often operate on a short period of time, they are usually not very well embedded in the community. **Burt's**

constraint relates to this. The **local transitivity** gives information about the connections between the neighbors of the considered vertex. In our context, a high transitivity denotes that the author is implied in a single large conversation in which most users interact with each other. On the other hand, a low transitivity indicates that the user participates in several distinct conversations or that some of its interlocutors ignore each other.

Macroscopic measures allow characterizing the position of the targeted author relatively to the whole context period when dealing with the *Full* graph, or to one of its halves with the *Before* and *After* graphs. The **eigenvector centrality** helps to detect central vertices, which can be symptomatic of an abusive author. An important authority score can characterize a user which receives much attention, while a very important hub score can rather represent someone harassing others. In our conversational graph, we expect that a user participating a lot in the conversation will be central, and thus have a high **subgraph centrality**. The **betweenness centrality** can be interpreted as the level of control that the user has over information transmission. The goal of abusive authors is usually to reach as many people as possible with their messages. The **closeness centrality** precisely measures the efficiency of a given vertex to spread a message over the graph. A high closeness indicates an author that plays an important role in the discussion. The **eccentricity** also indicates how involved the considered user is in the conversation. The **articulation point** binary feature help detect users that play the role of a bridge between 2 separate groups of users in the conversation.

Mesoscopic measures rely on a subgraph that is likely to represent a conversation. The **coreness score** is related to the number of participants of the largest conversation in which the author of interest is involved. It can be seen as the potential targets of an abusive message. The **within-module degree** relates to how involved the user is in his current conversation, and the **participation coefficient** indicates if the user holds a mediation position in the conversation.

4.2.2.2 Graph-Focused Topological Measures

Although we suppose that the vertex corresponding to the author of the targeted message is the one with most important information for our task, it is still possible that other vertices bring additional information. A solution to obtain measures describing the entirety of the graph is to compute a vertex-focused measure over the whole vertex set, and average them. We apply this methodology with all the vertex features described in Section 4.2.2.1. In addition to these artificial aggregated features, we also leverage measures that are specifically defined at the graph scale.

We assume that in a standard conversation, all interactions tend to be bilateral, which is a property captured by the **reciprocity** measure. The **degree assortativity** measures the tendency for vertices to be connected with other similar vertices, and thus can differentiate groups of users based on their behaviors. Many users leave a conversation when an abusive author is active in it. Thus, users in healthy conversations interact with more people than users in conversations

containing abusive comments. The **weak component count** captures this phenomenon. The **cohesion** and **adhesion** denote the presence of several distinct conversations in the network, with users interacting in many of them. To compute the **community count** and the **modularity**, we extract the community structure through the InfoMap algorithm [145].

4.2.3 Combining the Textual and Structural Information

In the previous sections, we defined a set of text and graph features. We now propose a method seeking to take advantage of both to study the complementary nature of these 2 sources of information. Indeed, they are based on 2 completely distinct ways of representing messages in a conversation, i.e. the message itself and the conversational graphs. To check this complementarity, we propose and experiment with 3 different fusion strategies to combine the features extracted by both feature-based methods. Figure 4.5 shows the whole pipeline.

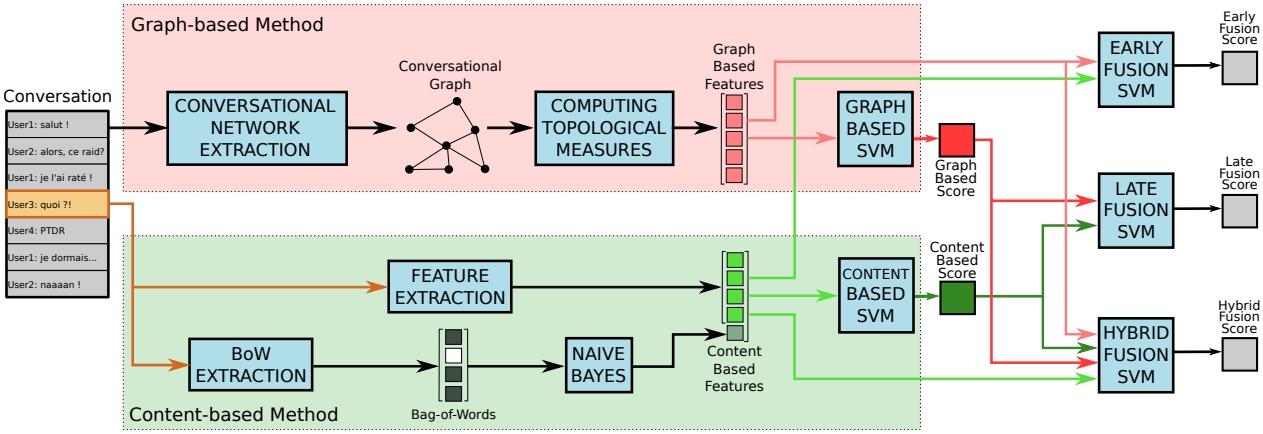

Figure 4.5: Representation of our processing pipeline. The top part (red) represent the graph-based method, the bottom part (green) represent the content-based method and the fusion strategies appear on the right side.

The left part represents the conversation in which the message to classify (in orange) was posted. This whole conversation is used by the graph-based method (top part, in red) to extract a conversational graph. Then, we compute all the topological measures described previously to represent this graph. Finally, this vector of features is fed to the graph-based SVM to obtain a predicted class. The bottom part (in green) represents the content-based method. The top part illustrates the extraction of text-based features directly from the message. In the bottom part, we first represent the message as a BoW, and train a Naive Bayes classifier to detect abuse using this sparse binary vector. The output of this classifier is added to the other features to form the message's representation which is then fed to the content-based SVM. We then use the features and scores produced by these 2 methods in our 3 fusion methods. We describe them as:

Scale	Scope	Name	Weight	Direction	Sign
Vertex	Microscopic	Degree Centrality	U	U/I/O	U/S
		Strength Centrality	W	U/I/O	-
		Local Transitivity	U/W	U	-
		Burt's Constraint	U/W	-	-
	Macroscopic	Eigenvector Centrality	U/W	U/D	-
		PageRank Centrality	U/W	U/D	-
		Hub / Authority scores	U/W	D	-
		Alpha Centrality	U/W	D	-
		Power Centrality	U	D	-
		SubGraph Centrality	U	U	-
		Betweenness Centrality	U/W	U/D	-
		Closeness Centrality	U/W	U/I/O	-
		Eccentricity	U	U/I/O	-
		Articulation point	-	U	-
	Mesoscopic	Coreness Score	-	U/I/O	-
		Within-module Degree	U	U/I/O	-
		Participation Coefficient	U	U/I/O	-
		External Intensity	U	U/I/O	-
		Diversity	U	U/I/O	-
		Heterogeneity	U	U/I/O	-
Graph	Microscopic	Vertex count	-	-	-
		Edge count	-	-	U/S
		Density	-	-	-
		Global Transitivity	U	U	-
		Reciprocity	-	D	-
		Degree Assortativity	-	U/D	-
	Macroscopic	Weak Components Count	-	U	-
		Strong Components Count	-	D	-
		Adhesion/Cohesion	-	D	-
		Articulation Points	-	U	-
		Diameter	U/W	U/D	-
		Radius	U	U/I/O	-
		Average Distance	U	U/D	-
	Mesoscopic	Clique Count	-	-	-
		Communities	U	D	-
		Modularity	U/W	U	-

Table 4.1: Summary of the graph measures used in the graph-based method. The letters in the *Weight*, *Direction* and *Sign* columns stand for: *Unweighted*, *Undirected* or *Unsigned* (U), *Weighted* (W), *Directed* (D), *Incoming* (I), *Outgoing* (O) and *Signed* (S).

Early Fusion: The first strategy consists in constituting a global feature set containing all content- and graph-based features from Sections 4.2.1 and 4.2.2 respectively, then training an

SVM classifier directly using these features. This is illustrated in the top-right corner of Figure 4.5. The rationale with this method is that the classifier has access to the whole set of features, independently of its source (text or graph). Thus, it can determine which ones are the most relevant for the classification task and put the focus on them to improve the performance.

Late fusion: The second strategy is processed in 2 steps. First, we apply separately both text and graph methods. We feed the respective features of each method to an SVM that outputs a score corresponding to the probability of each message to be abusive given by the content- and graph-based methods, respectively. Then, we fetch these 2 scores to a new classifier trained to determine if a message is abusive or not. This third classifier thus only takes 2 input variables and outputs the final predicted class of the message. This is illustrated in the middle-right part of Figure 4.5. This approach relies on the assumption that the 2 initial classifiers could act as an extreme dimensionality reduction system which compact the important information into a single score. Hence, these scores contain all the information that the final classifier needs, and not the noise present in the raw features.

Hybrid fusion: Finally, the third fusion strategy seeks to combine both previous proposed ones. We create a feature set containing features from both sources, like with *Early Fusion*, but also both scores used in *Late Fusion*. This whole set is used to train a new classifier. It corresponds to the bottom-right corner of Figure 4.5. The idea with this strategy is to check if the scores convey additional information compared to the raw features, in which case combining scores and features should lead to better results.

4.3 Experiments

In the previous section, we described the various text and graph features that we extract, and the approaches proposed to combine them. We now assess the performances of these methods. We first describe the experimental protocol followed in our experiments regarding the automatic detection of abusive messages in conversations (Section 4.3.1). Then, we present and discuss our results, in terms of classification performance (Section 4.3.2). Finally, we perform an in-depth analysis of our features to determine the most important ones for the classification and gain a better comprehension of our dataset and task (Section 4.3.3).

4.3.1 Experimental Protocol

We conduct our experiments on the SpaceOrigin dataset previously described in Section 2.3 composed of 2,545 annotated messages. We are in a posterior classification setup which implies that we have access to the context before and after the annotated messages. To handle all classes equally in this unbalanced setting, we express the performance in terms of macro F-measure, as defined in Section 2.4.

For the graph-based method, we extract conversational networks following the methodology described in Section 4.1. There are 2 important parameters for this extraction process, the *context period* size and the *sliding window* length. We use the values matching best performances on the test set, obtained during the greedy search of the parameter space performed in [6]. We fix the length of the sliding window to 8 messages and the size of the context period to 1,600 messages. Half of which were posted before the annotated message and the other half after. Implementation-wise, we use the iGraph library [146] to extract the conversational networks and process the corresponding features. We use the Sklearn toolkit [147] to get the text-based features.

We use an SVM for the classification, as it is well-suited for small datasets such as ours. We also experimented with other standard classifiers, but that did not change the order in which the methods are ranked. Therefore, we use an SVM as our classification system in the rest of this thesis. For the same reason of dataset dimension, we set up our experiments with a 10-fold cross-validation. We use a 70%-train / 30%-test split, which means that for each run of the cross-validation, 7 folds are used for the train and the remaining 3 compose the test set.

4.3.2 Classification Results

Table 4.2 presents the macro F-measure scores obtained for the independent text and graph-based methods, and all 3 fusion strategies described previously (*Early Fusion*, *Late Fusion*, *Hybrid Fusion*). It also shows the number of features used to perform the classification. Note that *Late Fusion* has only 2 direct inputs (the scores generated by content- and graph-based SVMs), but these in turn have their own inputs, which explains the values displayed in the table.

Table 4.2: Comparison of the performances obtained with the methods *Content-based*, *Graph-based*, and the 3 *Fusion* strategies. The total runtime is expressed as *h*:*min*:*s*.

Method	**Number of features**	**Total runtime**	F**-measure**
Content-Based	29	0:41	75.21
Graph-Based	477	3:51:25	83.40
Early Fusion	506	3:55:17	84.51
Late Fusion	506 (2)	3:54:57	84.17
Hybrid Fusion	508	3:57:44	**84.97**

The first interesting result is that the graph-based method clearly outperforms the content-based one. Intuitively, one could think that the content of a message is the main factor to determine its degree of abusiveness. However, this result shows otherwise. Relying on the interactions between users on a short period of time before and after a particular message is posted, seems

to be much more efficient for this task. Though surprising at first sigh, this result is in line with previous studies [6], [11]. We suppose that when an abusive message is posted in a conversation, it tends to change the way users interact with each other. It can be either by generating a massive spike in reactions toward the author of that message, or by *killing* the conversation with people not wanting to interact with the abusive author and ultimately leaving the discussion. Thus, the abusive author plays a central role in the conversational graph that can be captured by our features. On the other hand, non-abusive messages usually go unnoticed and are mixed in with all the others messages, therefore not playing such a central role in the graph.

This result also emphasizes the importance of using conversational context to correctly detect abuse. Indeed, the content-based method might easily be tricked by spelling mistakes (intentional or not) or by obfuscation techniques (e.g. "f..k 0ff"). This can lead to abusive messages not being detected when using the content, while the graph-based method is unaffected by such techniques. The modeling of conversations and interactions can also reduce the false positive rate. For instance, in the case of a borderline message which some would consider abusive, knowing that the 2 authors share a long history drastically reduces the likelihood that the message is truly abusive. It might be more of a joke between friends.

Although showing many benefits, the graph-based method is computationally very expensive due to the high number of computed features and the complex nature of some of them. It took almost 4 hours to compute the graph-based features for the full dataset, while it took less than a minute on the same machine for the content-based approach. This difference is also due to the number of features for each method: only 29 for the text and 477 for the graphs. One could argue that the performance gap between these 2 approaches is mainly caused by this major difference in the size of the features set. We address this question in the next section and show that this is not the case. The content-based method also suffers from the fact that there is no clear standard in the features used to describe a message. On the other hand, there are many well established topological measures to capture various properties of graphs.

With the fusion methods, we try to further improve the classification performance by combining the 2 modalities. Our first observation is that we get higher F-measure values compared to both individual methods when performing the fusion, no matter the fusion strategy. This confirms what we expected, i.e. both sources are at least partly complementary, since the performance increases when merging them. The content can bring useful information for the messages that are clearly abusive in their form, while the conversational graphs can bring information based on the context.

Next, when comparing the fusion strategies, it appears that *Hybrid Fusion* performs better than the others, with an F-measure of 84.97%. This is what we were expecting, since this strategy is the combination of the other 2, benefiting from both. The *Late Fusion* is the least efficient fusion strategy but by just a few tenths of a point. This method only gets 2 features, which are themselves the outputs of 2 other classifiers. This means that these classifiers do a good work in

summarizing their inputs, without losing much of the information necessary to efficiently perform the classification task. We could see these intermediate classifiers as a pre-processing step, which reduces the dimensionality of data. The *Early Fusion* produces better results than the *Late Fusion* with its combination of all the available features. Regarding runtime, the fusion methods are a few minutes longer than graph-based method since these approaches require computing both content- and graph-based features.

4.3.3 Feature Study

We developed our content- and graph-based methods based on an exploratory approach which consists in selecting a very wide range of widespread features in the literature. We adopted this methodology because we had no idea of the features that were important for this specific task. The goal with such an exploratory approach was not to miss any important features. This resulted in large sets of 29 content features and 477 graph features that are computationally expensive to calculate, especially the latter. Furthermore, we assume that some of them are useless for the classification, i.e. they do not bring any additional information compared to the rest of the features.

In this section, we try to estimate the discriminative power of our features with regard to our classification task. The objective is to identify the best features for all our methods and fusion strategies. In order to do so, we apply an iterative approach based on the *Sklearn* toolkit, which allows us to fit a linear kernel SVM to the dataset and provide a ranking of the input features reflecting their importance in the classification process. Using this ranking, we identify the least discriminant feature, remove it from the dataset, and train a new model with the remaining features. The impact of this deletion is measured by the performance loss, in terms of F-measure. We reiterate this process until only one feature remains. We call *Best Features* (BF) the minimal subset of features allowing to reach 97% of the original performance (when considering the complete feature set). This threshold was chosen arbitrarily to have a limited number of BF, while preserving good performance.

We apply this process to our content- and graph-based methods and all 3 fusion strategies. We then repeat the classification process using only their respective BF. The results are shown in Table 4.3 with a comparison to the original results obtained on the complete feature sets. Note that the *Late Fusion BF* performance is obtained using the scores produced by the SVMs trained on *Content-based BF* and *Graph-based BF*. These are also used as features when computing the BF for *Hybrid Fusion BF* (together with the raw content- and graph-based features). In terms of classification performance, by construction, the methods are ranked exactly like when considering all available features.

The BF obtained for each method are listed in Table 4.4. The last 4 columns indicate which variants of the graph-based features are concerned. Indeed, as explained in Section 4.2.2, most of these topological measures can handle/ignore edge weights and/or edge directions, can be

Table 4.3: Comparison of the performances obtained with the methods (*Content-based*, *Graph-based*, *Fusion*) and their subsets of *Best Features* (BF). The total runtime is expressed as *h*:*min*:*s*.

Method	**Number of features**	**Total Runtime**	F**-measure**
Content-Based	29	0:41	75.21
Content-Based BF	3	0:27	73.17
Graph-Based	477	3:51:25	83.40
Graph-Based BF	10	7:59	81.08
Early Fusion	506	3:55:17	84.51
Early Fusion BF	4	9:49	82.29
Late Fusion	506 (2)	3:54:57	84.17
Late Fusion BF	13	9:31	81.89
Hybrid Fusion	508	3:57:44	**84.97**
Hybrid Fusion BF	4	10:29	82.66

vertex- or graph-focused, and can be computed for each of the 3 types of networks (*Before*, *After* and *Full*).

There are 3 *Content-Based BF*. The first is the *Naive Bayes* prediction, which is not surprising as it comes from a fully fledged classifier processing BoW. This system on its own could be used as the main abuse detection method in some scenarios. In our application context, the naive Bayes classifier alone obtains a F-measure score of 62.01. The second BF is the *TF-IDF score* computed over the *Abuse* class, which shows that considering term frequencies indeed improve the classification performance. The fact that this is the score computed over the *Abuse* class is particularly interesting because it suggests that there are words or family of words which are characteristic of abusive messages. Of course, we can think of insults and swear words. This information should be captured by our *Bad words* feature, but since it is not among the BF, we can assume that its information is redundant with the *TF-IDF score*. The third BF is the *Capital Ratio* (proportion of capital letters in the comment), which is likely to be caused by abusive messages tending to be shouted, and therefore written in capitals.

We obtain 10 features for the *Graph-Based BF*. Among them, 4 are computed at the vertex-scale (i.e. only for the targeted vertex) and 6 are computed over the whole graph. Most of these features seem to confirm our previous assumption that abusive messages tend to generate more reactions than non-abusive messages. The *Coreness* score, which is closely related to the number of users interacting with the author of a message, is the best topological measure for this task. We find 2 variants of this measure in the BF, one computed over the *Full* graph and the other over the *Before* graph. The presence of the *Authority* score among BF can be explained by the same

Table 4.4: Best features obtained for our 5 methods. The letters in the *Graph* column stand for *Before* (B), *After* (A) and *Full* (F). Those in the *Weights* and *Directions* columns stand for: *Unweighted* or *Undirected* (U), *Weighted* (W), *Directed* (D), *Incoming* (I) and *Outgoing* (O). Those in the *Signs* column mean *Unsigned* (U) or *Signed* (S). Those in the *Scale* column stand for *Graph-scale* (G) or *Vertex-scale* (N).

Method	Best Features (BF)	Graph	Weights	Directions	Signs	Scale
Content-Based	Naive Bayes	–	–	–	–	–
	TF-IDF Abuse Score	–	–	–	–	–
	Character Capital Ratio	–	–	–	–	–
Graph-Based	Coreness Score	F	–	I	–	G
	Authority Score	B	W	D	–	G
	Closeness Centrality	B	W	O	–	G
	PageRank Centrality	A	U	D	–	N
	Closeness Centrality	B	W	O	–	N
	Coreness Score	B	–	I	–	G
	Degree Centrality	F	U	O	U	N
	Vertex Count	F	–	–	–	G
	Reciprocity	A	–	D	–	G
	Closeness Centrality	A	W	U	–	N
Early Fusion	Coreness Score	A	–	O	–	G
	Strength Centrality	B	W	O	–	N
	Naive Bayes	–	–	–	–	–
	Eccentricity	B	–	I	–	G
Late Fusion	*Content-Based* BF ∪ *Graph-Based* BF	–	–	–	–	–
Hybrid Fusion	Graph-based output	–	–	–	–	–
	Coreness Score	B	–	I	–	G
	Content-based output	–	–	–	–	–
	Hub Score	B	U	D	–	N

reasons as it also relates to how central a vertex is in the graph. There are 3 variants of the *Closeness* centrality in the BF subset. 2 of them focus on the targeted vertex while the third is averaged over all vertices to represent the graph scale. This measure is considered as the efficiency of the vertex to spread a message over the graph. Once again, this directly relates to the tendency of abusive message to spread wider and faster than other messages. The *PageRank* captures how influential a vertex is in the graph and the *Degree* centrality can be interpreted as

the number of users that have exchanged with the author. All these features can be interpreted as different measures of our assumption. However, they all capture different information since removing any of the BF has a noticeable negative impact on the classification performance. The last 2 BF are the *Reciprocity*, which captures the tendency of certain users to respond or not to others. Finally, the *Vertex count* is the number of users participating in the conversation. The fact that an abusive author interact in the discussion can affect one way or the other the number of active users in the discussion.

There are 4 features for *Early Fusion BF*. One is the *Naive Bayes* feature (content-based), and the other 3 are topological measures (graph-based features). The *Coreness* score, though with different variants, was also among the graph-based BF. The *Strength* centrality is a generalization of the degree centrality, which was also among them. The *Eccentricity* of the graph reflects important changes in the interactions between users. It is likely caused by angry users piling up on the abusive user after he has posted some inflammatory remark. Note that eccentricity is related to the closeness centrality. Hence, the 4 *Early Fusion BF* are either directly among the BF of one of our 2 methods or, related to a feature that is part of them.

For *Hybrid Fusion BF*, we also get 4 features but those include both scores produced by the classifiers from the content- and graph-based methods. Those are completed by 2 graph-based features, the *Coreness* score (also found in the *Early Fusion BF*) and Hub score. It is interesting to note that the *Hybrid Fusion BF* and *Early Fusion BF* are a mix of features based on text and graphs. The complementary nature of these 2 methods was already proved by the results in Table 4.2, but this also demonstrates that both methods provide some measures that cannot be replaced by features of the other type.

This in-depth study of our features gives us a better understanding of our dataset and classification process. We now know which text and graph measures are the most important to characterize abusive messages of our dataset. In addition, we demonstrated that a small subset of features can be enough to train a classifier almost as good as the original classifier trained on the complete set of features. We can retain 97% of the performance while using only 3 out of the 29 features (10.34%) based on the content. This effect is even stronger with graphs, as only 10 out of 477 (2.10%) features are needed to retain this degree of performance. This is a very interesting result, since the feature computation is by far the most computationally expensive step of the framework. Computing only a handful of features instead of hundreds allows drastically decreasing the computational cost of the classification. This is true for all methods. For instance, a classifier trained with the *Graph-Based BF*, only requires 3.5% of the total *Graph-Based* runtime.

4.4 Analysis and Discussion

We now discuss the results of the feature engineering methods aiming to detect abusive messages. Our 2 points concern the graph-based system. First, we explore its temporal aspect (Section 4.4.1). Second, we investigate how considering the edge weights, directions and signs in graphs impact the classification performance (Section 4.4.2).

4.4.1 Temporal Aspect

As explained in the graph extraction process (Section 4.1), there are actually 3 networks extracted for each message to classify: the *Full*, *Before* and *After* networks. The graph-based results shown until now are all obtained using the *All* feature set, i.e. the features resulting from the calculation of all topological measure variants for all 3 individual graphs (*Full*, *Before* and *After*).

However, the *After* network is only available in a post-classification context where users have had time to respond to the message of interest. This is the case in our work, and more generally with most of the research work. However, real life applications often require taking a decision as soon as the message is posted. For instance, to block it before it is published if there is a suspected abuse. In this case, the detection method can only rely on the context posted *Before* the message. In this section, we study the performance of our framework in a real-time classification context. For the sake of completeness, we also consider the opposite case, where only the context *After* the message is available. This could correspond to an application which logs messages only after one of them has been reported.

To obtain a better understanding of the role of each part of the context, we train and test a classifier on each of the 3 feature sets (*Full*, *Before* and *After*). Each feature set is composed of the graph-based features extracted from the corresponding graph. Table 4.5 shows the classification performance obtained. As a comparison, we also include the *All* set, which is the combination of the other 3. This is the set of features that was used in all the previous experiments. Aside from the input graphs, the experimental protocol is exactly the same as in the previous section.

Table 4.5: Comparison of the performances obtained with the feature sets *Before*, *After*, *Full* and *All*. The latter is the combination of the first 3.

Temporality	**Number of features**	F**-measure**
Before	159	79.08
After	159	78.97
Full	159	82.89
All	477	**83.40**

The performance of the *Before* and *After* feature sets are almost on par. They obtain acceptable performance level considering that they have access to only half of the *Full* context. This shows that the context before is as important as the context after to detect an abusive message. This is surprising, as one could think that the reactions after a message give more information about its nature than its predecessors.

For the context *After*, we already mentioned the fact that abusive messages tend to modify the way authors interact with each other and thus change the structure of the graph. Regarding the context *Before* the message, we can assume that it is possible to predict an abusive message before it is even posted thanks to the build-up leading to that message. The abusive author starts to stand out from the other users and draws attention before posting its abusive comment. This result is very promising as it shows that building a real-time classifier using only the past messages is possible and can give decent performance. This is the type of application that could typically be used on online forums or social networks.

The *Full* feature set logically obtains a better F-measure as it covers the same context period as the previous 2 combined. It is worth noting that this feature set obtain noticeably higher performance than the *Before* and *After* sets, which means that there is information allowing to discriminate abusive messages in both sets and that their information do no completely overlap. As expected, the *All* feature set performs best overall, as it is the union of all the other considered feature sets. However, it is only slightly better than *Full*. This shows that the *Full* feature set already conveys almost all the information. This is interesting as using the *Full* set instead of *All* means computing only one third of the features and thus greatly decreases the runtime without impacting heavily the classification performance. Furthermore, the *All* set requires extracting 3 graphs, while the *Full* set only requires one. The time saving is negligible compared to the previous point, but it is still important as it could have a bigger impact for very large graphs. These 2 options are worth exploring in a future work, and especially in combination with the best features.

4.4.2 Impact of Edge Attributes

Edge directions and weights are graph-specific information that cannot be found in the content-based method. The weight represents the intensity of an interaction between 2 users, the direction its reciprocity (unilateral or bilateral). On the other hand, the edge sign, which represents its polarity (hostile or friendly), is obtained through a sentiment analysis of the text, and is therefore directly related to it. Some of the measures used to represent graphs are able to deal with these properties, while others ignore them. To investigate in more details how using or not these properties affects the classification performance, we set up the following experiments. We define 8 variants of our conversational graphs, all the possible combinations of *(Un)weighted-(Un)directed-(Un)signed* graphs. Note that the *Weighted-Directed-Signed* graph is the standard configuration used in all our previous experiments. We then compute our graph-based features for each graph and compare

their respective classification performance. The results are shown in Table 4.6.

Table 4.6: Comparison of the F-measure obtained when considering the 8 different types of graphs. Bottom right is the standard configuration used in the rest of this section.

Graph type		**Unsigned**	**Signed**
Unweighted	Undirected	72.26	72.39
	Directed	72.51	72.60
Weighted	Undirected	74.94	74.92
	Directed	75.18	**75.21**

It appears that all weighted methods obtain better performance than their unweighted counterparts. This indicates that edge weights help the classifier to identify abusive messages. If we think of abusive authors as persons who draw a lot of attention because of their behavior, this is reflected in the edge weights that are higher than other users overall. As a reminder, most of the weights in our graphs are computed based on an assumption that consecutive messages respond to each other. While generally true, this assumption can lead to incorrect weight assignation in some cases. Consequently, we think that weights can have an even stronger impact in applications where they are perfectly computed.

The directions also bring a slight improvement, but very limited compared to the weights. We can think of some cases where edge weights used in conjunction with directions can highlight an abusive behavior. For instance, a very strong unilateral link between 2 users can be symptomatic of harassment.

Edge signs appear to have a small impact on the performance. This impact is probably limited due to the very small number of signed features. Edge signs are particularly well suited for conversational graphs as they can model friendships and enmities, so we think that in other configurations, it could have a lot more impact. Nonetheless, the best overall performance is reached with the *Weighted – Directed – Signed* graphs showing that this is worth including measures that account for these graph-specific properties.

4.5 Conclusion

In this chapter, we have presented several approaches to automatically detect online abusive messages using feature engineering from different sources of information. We extended this framework by applying 2 methods: the first one based on the content and using NLP features. The second, a *graph-based* approach, integrates all the contextual aspects of conversions through conversational graphs. We manually constituted a set of 29 textual features to create a representation of the message in the *content-based* method, and a set of 477 topological measures to character-

ize the graph in the *graph-based* approach. We applied these methods to the dataset extracted from the in-game chat of SpaceOrigin, an online video game from Chapter 2.3.2. Our experiments showed that the method relying exclusively on feature engineering from graphs to model the interactions between users (i.e. the context in which a message is posted), obtain better classification performance than the method relying on the content and ignoring context of the targeted message. This highlights how important the context is when dealing with abuse detection.

We also proposed 3 fusion strategies to combine the 2 proposed methods and showed that their combination improve the performance, suggesting that text and graphs are 2 complementary sources of information for this task. Then, we studied in details the features of all our methods and identified the ones that are the most important in the classification process. We showed that using only a very small subset of so-called *best features* was enough to obtain 97% of the original performance. Furthermore, we explored the temporal aspects of our *graph-based* method. Though it is better to use all the context available around the message to classify, we showed that using only the context *before* or *after* the message still leads to satisfying performance. Finally, we demonstrated how using the edge directions, edge signs and most importantly the edge weight improve the performance.

We believe that this work can be extended in several ways. First, one important limitation of our methodology based on feature engineering is the high computational time required to extract the graph features. However, we show that using a small subset of relevant features can dramatically reduce the processing time (by more than 96%) while keeping more than 97% of the original performance. Second, our graph-based method currently uses all the context available. An extension could be to adapt it to a real-time classification setting by considering only the messages which were posted before the message to process. Third, we showed that edge weights, directions and signs were important parameters which improve the classification performance of the graph-based method. They provide additional information regarding the nature and the intensity of the interactions between users. Finally, one main finding of this chapter is that combining content and graphs works fine, but it is difficult and computationally expensive in a feature engineering framework. It is also possible to miss some important features with this approach. Therefore, in the rest of this manuscript, we propose to use representation learning to automatically learn representations of messages and graphs. For text, we focus on existing methods [148]–[154] in Chapter 6. There are also many graph representation learning techniques, however they do not cover the full range of attributes available in our conversational networks. In the next chapter, we review the graph representation learning literature, and propose methods to fill this methodological gap.

Chapter 5

Signed Whole-Graph Embedding

In Chapter 4, we experimentally showed that edge directions, weights and signs help improve the detection of abusive messages when using feature-based approaches. Because of the important computational cost of feature engineering methods for graphs, and the complexity for a human to create a set of discriminant features, we propose to use representation learning as a solution to automatically learn graph representations. It is even possible that such representations also improve the classification performance because they are dense representations automatically learned without having to redefine them. Graph representation learning methods can operate at different granularity levels, such as vertex and whole graph. Vertex embedding is the task consisting in learning a representation for each vertex of the graph, while whole-graph embedding seeks to learn a single representation of the graph as a whole. In our context, the abuse detection task consists in classifying graphs, therefore we would like to learn representations of *whole*

graph, while taking into account edge attributes. On the one hand, a number of graph embedding approaches allow, or can easily be tweaked, to take weights and directions into account (e.g. [85], [155]–[157]). There are weighted and/or directed methods that are designed for vertex and for whole-graph embedding. On the other hand, signs are much more complicated to incorporate in a graph embedding pipeline, as the duality between positive and negative edges involves complex concepts. Therefore, several recent methods only handle *vertex* representations in signed graphs (e.g. [158]–[160]), and the literature offers *no method* for signed *whole-graph* representations.

In this chapter, we first review the literature of graph representation learning. Then, we propose 2 original approaches to learn whole-graph representations of signed graphs. The first is a generalization of the *unsigned* whole-graph embedding method Graph2vec [161] to *signed* whole graphs. The second one is a generalization of the signed *vertex* embedding method Signed Graph Convolutional Networks (SGCN) [160] to signed *whole* graphs. We propose several variants of these models and assess them experimentally. The SpaceOrigin dataset that we used in the previous chapter is relatively small, so we constitute a benchmark composed of 3 distinct collections of graphs, including SpaceOrigin for the experimental evaluation of our methods. This benchmark also allows us to improve the diversity of graphs, both in terms of structure and properties.

The following chapter is based on our work [23]. Our source code is publicly available online[1]. It makes the following contribution:

1. We review the graph representation learning literature, focusing on the vertex- and graph-level methods.

2. We propose 2 adaptations of existing methods to learn whole-graph representations of signed graphs, for which we define several variants.

3. We constitute and share a benchmark annotated for signed graph classification, and composed of 3 distinct collections.

4. We present extensive computational experiments established for the proposed methods, on our benchmark.

The rest of this chapter is organized as follows. In Section 5.1 we introduce the main concepts and notations used later in the chapter. Next, we survey graph embedding techniques in Section 5.2. Then, we present our signed graph classification benchmark in Section 5.3, and we describe the methods that we propose to handle signed whole-graph representation learning in Section 5.4. We present and discuss our results in Section 5.5. Finally, we review our main findings and describe how our work can be extended in Section 5.6.

[1] https://github.com/CompNet/SWGE

5.1 Definitions and Notations

The presence of signs associated to edges allows defining the notion of path sign:

Definition 5.1 (Sign of a path)**.** *The **sign of a path** or cycle corresponds to the product of its constituting edge signs. Consequently, this sign is negative if the path or cycle contains an odd number of negative edges, and positive otherwise.*

Edge signs can also be used to define additional properties, in particular the reachable set, which we use later:

Definition 5.2 (Reachable set of a vertex)**.** *The positive (resp. negative) **reachable set** of a vertex $u \in V$ is the subset of vertices that are connected to u through positive (resp. negative) shortest paths.*

Structural Balance (SB) is a fundamental property of signed graphs [162], [163], stating that the graph can be partitioned in 2 clusters such that all positive edges are internal, i.e. they connect vertices from the same cluster, whereas all negative edges are external, i.e. they lie in-between both clusters.

Definition 5.3 (Structural balance (SB))**.** *A graph $G = (V, E, s)$ is said to be **structurally balanced** if it is possible to find a bisection $C = \{C_1, C_2\}$ of V such that $E[C_1, C_2] = E^-$ and $E[C_1] \cup E[C_2] = E^+$, where $E[X]$ denotes the edges connecting 2 vertices from $X \in V$, and $E[X, Y]$ denotes the edges connecting 2 vertices from X and $Y \in V$.*

This definition is the strict version proposed by Cartwright *et al.* [164], and illustrated in Figure 5.1.a). Equivalently, all the cycles of a structurally balanced graph are positive.

In real-world networks, though, graphs are rarely *perfectly* balanced, and no bisection exists that respects the SB definition. In this case, one may want to measure the *amount* of imbalance of the graph, through the notion of *Frustration*. The positive edges located in-between the clusters and the negative edges located inside them are said to be *frustrated*, as they do not respect SB. The Frustration (aka *Line Index* or *Imbalance*) of this bisection is the number of such edges.

Definition 5.4 (Frustration of a bisection)**.** *For a graph $G = (V, G, s)$, let $C = \{C_1, C_2\}$ be an arbitrary bisection of V. The **frustration** of this bisection is the total number of external positive edges and internal negative edges : $F(G, C) = |E^+[C_1, C_2]| + |E^-[C_1]| + |E^-[C_2]|$.*

The Frustration of a graph corresponds to the minimal Frustration over all possible bisections of the graph.

Definition 5.5 (Frustration of a graph)**.** *The **frustration** of a graph $G = (V, G, s)$ is the minimal frustration over all possible bisections: $F(G) = \min_{C \in \mathcal{C}} F(G, C)$, where $\mathcal{C}$ is the set of all possible bisections of V.*

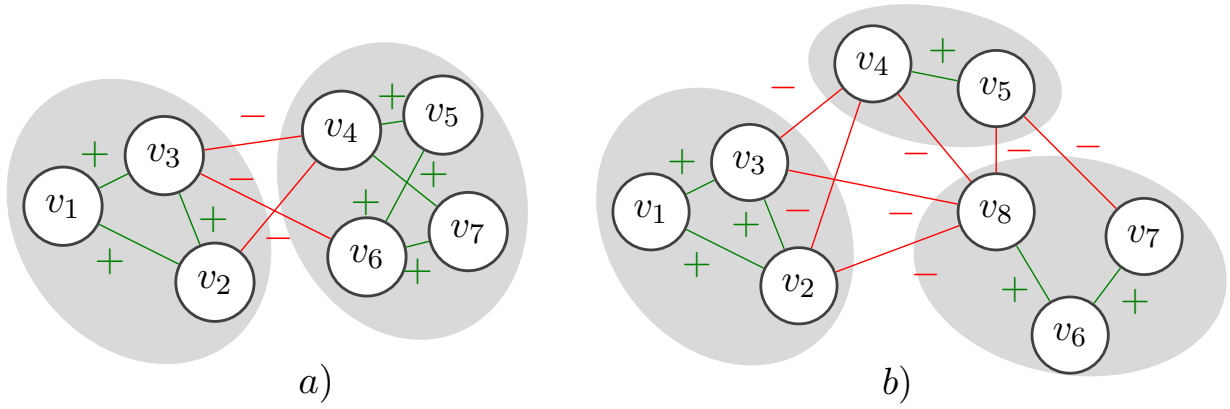

Figure 5.1: Examples of balanced graphs according to a) Structural Balance (SB); and b) Generalized Balance (GB) ($k = 3$).

Put differently, the Frustration is the minimal number of edges whose sign must be switched to reach perfect structural balance. Computing the frustration requires solving a combinatorial optimization problem which was proposed by Doreian *et al.* [165], and later called the *Correlation Clustering* (CC) problem [166]. One of the datasets presented in Section 5.3 is directly related to this CC problem.

SB was later generalized to allow partitions constituted of more than 2 antagonistic clusters [167], i.e. clusters where all positive edges are internal, and all negative edges are external. A graph which can be split into k antagonistic clusters is said to be k-balanced.

Definition 5.6 (Generalized balance (GB))**.** *A graph $G = (V, E, s)$ is said to enforce the* ***general balance*** *property if it is possible to find a partition $C = \{C_1, ..., C_k\}$ of V such that $\bigcup_{1 \leq i < j \leq k} E[C_i, C_j] = E^-$ and $\bigcup_{1 \leq i \leq k} E[C_i] = E^+$.*

Equivalently, a graph respecting the *generalized balance* has no cycle with exactly one negative edge. Figure 5.1.b) illustrates the notion of GB. Note that the Frustration remains a valid imbalance measure for this generalization of the SB.

5.2 Graph Representation Learning

Graph representation learning methods have to learn representations of graphs that preserve at least some part of their topological properties [168]. 2 graphs with similar properties must sit close in the embedding space [169]. Graph embedding is a very popular research topic, and the literature provides numerous methods allowing to automatically train models into representing various *types* of graphs (directed, weighted, signed, multiplex, etc.), and various *parts* of graphs (vertices,

edges, subgraphs, whole graphs, etc.). Each category better fits the needs of different applications and problems. In this thesis, we are interested in abuse detection, which we formulated as a graph classification task. Therefore, this chapter mainly describes whole-graph representation methods. However, as shown in Chapter 4, it is alternatively possible to agglomerate vertex representations to create graph representations. For this reason, we also include vertex-level methods in our review. Furthermore, representation learning provides representations which can be used for various tasks. We explore this possibility with a benchmark constituted of 3 tasks, including abuse detection, later in this chapter. We first summarize the main approaches for representing the vertices of graphs (Section 5.2.1), then we review the methods that directly handle graphs as a whole (Section 5.2.2). Finally, edge signs are difficult to integrate in an embedding framework due to their nature. We dedicate the last section of this review to the main approaches designed to deal with signed graphs, which all handle only vertex representation, and not whole graphs (Section 5.2.3).

5.2.1 Vertex Embedding

Vertex embedding methods take a graph as an input and output a fixed-length vector representation for each of its vertices. It is the most common form of graph embedding in the literature. Following the taxonomy proposed in [170], we distinguish 3 categories of methods depending on the general approach used to perform the transformation: Matrix Factorization (Section 5.2.1.1), Random Walks (Section 5.2.1.2) and Graph Neural Networks (GNNs) (Section 5.2.1.3). Note that random walks also use neural networks but leverage a different strategy to sample the graph.

5.2.1.1 Matrix Factorization

There are various ways to represent a graph in a matrix form, such as the adjacency, Laplacian or transition matrices. The pioneering studies on vertex embedding propose to map them into low-dimensional vectors by decomposing such matrices into products of smaller matrices of the desired dimension, a process called *Matrix Factorization* (MF). Depending on the matrix properties, there are multiple approaches to decompose it.

The most straightforward approach is to leverage existing dimensionality reduction techniques, originally designed for tabular data, and apply them to a graph matrix. Doing so with the **Locally Linear Embedding** method proposed by Roweis *et al.* [171] amounts to considering that the representation of each vertex in the graph is a weighted linear combination of its neighbors'. The method first estimates weights that best reconstruct the original characteristics of a vertex from its neighbors, and then uses these weights to generate vector representations. This method has been used in the literature to perform face recognition [169]. Belkin *et al.* propose **Laplacian Eigenmaps** [156], a method aiming at keeping strongly connected vertices close in the representation space. Representations are obtained by computing the Eigenvectors of the graph Laplacian.

Typical applications for this method include vertex classification and link prediction [156], [170]. Both these approaches preserve the first order proximity, i.e. vertices which are directly connected have close representations. A major drawback of these 2 methods is that their time complexity is in O($|E|d^2$) with d being the number dimensions in the learned representation. This makes them poorly scalable and impossible to use on very large real-world graphs.

Graph Factorization [157] proposed by Ahmed *et al.* is much more time efficient and can handle graphs with several hundred million vertices. It uses stochastic gradient descent to optimize the matrix factorization. To improve its scalability, this approach uses some approximation strategies, which can introduce noise in the generated representations. Furthermore, the authors focus on preserving only the first-order proximity. Hence, the global graph structure is not necessarily well-preserved by this method. Ahmed *et al.* use this method to partition graphs and to predict the volume of e-mail exchanges between pairs of users [157].

Ou *et al.* introduce a matrix factorization method called **High-Order Proximity preserved Embedding (HOPE)** [155]. The similarity matrix is obtained using centrality measures like Rooted PageRank, Katz measure and Adamic-Adar score. HOPE is specifically designed to preserve asymmetric transitivity in directed graphs. To this end, 2 vector representations are learned for each vertex to capture asymmetric edges, a *source* vector and a *target* vector. Applications of this method include link prediction, proximity approximation and vertex recommendation [172]. However, once again the time complexity of this factorization method is high and does not allow the processing of very large graphs.

Li *et al.* present **BoostNE** [173]. This multi-level graph embedding framework learns multiple graph representations at different granularity levels. Inspired from boosting, it is built on the assumption that multiple weak embeddings can lead to a stronger and more effective one. It applies an iterative process to a closed form vertex connectivity matrix. This process successively factorizes the residual obtained from the previous factorization, to generate increasingly finer representations. The sequence of representations produced is then assembled to create the final embedding. Li *et al.* apply their method to a multi-label vertex classification task.

Matrix factorization was the first strategy developed to learn representations of vertices. However, it has a major limitation. Their time complexity is very high and depends on the number of edges in the graph. Hence, it makes them unusable on large real-world graphs. To prevent this issue, authors proposed a more efficient strategy to model the structure of the graphs: random walks.

5.2.1.2 Random Walks

Random walks have first been adopted by graph embedding approaches trying to mimic word-embedding methods such as *word2vec* [174]. Since vertices in graphs have no natural ordering,

random-walks allow representing the graph structure under a sequential form, analogous to sentences in a text. They are used to sample the graph, and can be seen as a proxy allowing to obtain a partial representation of its structure. They also have the advantage of being able to deal with graphs too large to be explored in their entirety. Given a starting vertex, random-walk-based methods generate vertex sequences by selecting a neighbor and repeating this procedure until the sequence reaches a certain length. Different strategies can generate random walks that sample the graph in completely distinct ways, allowing to capture various properties.

Perozzi *et al.* propose **DeepWalk** [175]. It is among the first vertex embedding methods based on random-walks. First, DeepWalk samples vertex sequences using uniform random walks and then applies the standard *SkipGram* model [174] to generate the representations. This model takes a vertex as input and aims at predicting its context, i.e. the vertices in its neighborhood. With this method, vertices with similar contexts share similar representations. Typical applications of this approach include vertex classification [176], [177] and link prediction [155]. However, a limitation is that 2 vertices can be structurally similar (i.e. they play the same role in the graph) but be distant in the graph, hence, not share any common neighbors. Their representations might thus be completely different.

The **Node2vec** [85] method proposed by Grover *et al.* was developed following the idea of *DeepWalk*. The main difference is that *Node2vec* uses *biased* random-walks to provide a more flexible notion of a vertex's neighborhood and better integrate the notion of structural equivalence. More specifically, it uses 2 parameters to bias the transition probabilities between vertices. The *return parameter* controls the likelihood of immediately revisiting a vertex in the random walk. The *in-out parameter* can restrict the walks to a local neighborhood or conversely, increase the probability of visiting vertices which are further away from the current one. Node2vec has been used to predict links in a biomedical context [172], and to classify vertices [176]. However, the model randomly initializes the embeddings, which can result in being stuck in a local optima during the computation of embeddings. Chen *et al.* propose an improved weight initialization strategy to avoid such problems in their **Hierarchical Representation Learning** method [178]. In **Walklets** [179], Perozzi *et al.* introduce a new random walk strategy. Traditional random-walk methods select the next vertex from the current vertex's neighbors. Instead, *Walklets* proposes to skip over vertices to obtain sequences which are not constituted of direct neighbors. This strategy allows modeling and preserving higher order relationships between vertices and can be used in multi-label classification problems [179].

5.2.1.3 Graph Neural Networks

Modern neural approaches have been successfully adapted to many fields including graph representation learning. The proposed methods were first direct adaptation of standard neural network

approaches to graphs, before developing a new family of neural networks that directly operates on graph data: Graph Neural Networks (GNNs).

Wang *et al.* propose the **Structural Deep Network Embedding** (SDNE) framework [177]. This method learns representations based on first and second order proximities in the graph. These 2 properties are jointly optimized using a deep autoencoder and a variation of Laplacian Eigenmaps, applying a penalty when similar vertices are mapped far from each other in the embedding space. This allows a good representation of both the local and global structure of the graph. This method has been used on vertex classification and link prediction tasks [170], [177].

Generative Adversarial Networks (GANs) have also been adapted to vertex embedding. Wang *et al.* [180] propose **GraphGAN**, which works through 2 models. First, a generator $G(v|v_c)$ tries to approximate the true connectivity between vertices v and v_c and selects the most likely connected vertices to v_c. Second, a discriminator $D(v, v_c)$ computes the probability of an edge to exist between v and v_c. The generator tries to fit the distribution of vertices as much as possible to generate the most indistinguishable fake pairs of connected vertices. The discriminator tries to distinguish between ground truth and the fake pairs created by the generator. This method is however only able to capture the local structure. Wang *et al.* apply GraphGAN to vertex classification, link prediction and movie recommendation tasks.

Graph Convolutional Networks (GCNs) [87] are a family of neural networks adapting traditional Convolutional Neural Networks (CNNs) so that they can process graph data instead of standard tabular data [181]. Shallow embedding methods such as Graph2vec may miss complex patterns in the graphs, and deep learning methods like CNNs are likely to solve this limitation [182]. GCNs generalize the convolution operation by considering graph neighborhood instead of linear or grid neighborhoods, as in standard CNN used in NLP and image processing. Kipf *et al.* leverage their method to perform document and entity classification.

GraphSAGE [89] aggregates feature information from the local neighborhood of a vertex. At each iteration, it samples a neighborhood of fixed size for each vertex and aggregate them through an aggregator function. Thus, vertices aggregate information from their local neighbors at each iteration and propagate it through the following layers of the network. By this way, vertex representations incrementally gain more and more information from further neighbors as the process iterates.

5.2.2 Whole-Graph Embedding

As mentioned before, vertex embedding methods are the most widespread in the literature. But some tasks require information at a higher granularity, in which case one would turn to whole-graph embedding. These methods allow representing a whole graph as a single vector of fixed length. They take a collection of graphs, and output a representation for each of them. Some strategies

used for vertex embedding can be transposed to whole-graph embedding matrix factorization and random-walks.

Lara *et al.* [183] propose the **Simple and Fast** algorithm based on the spectral factorization of the graph Laplacian. It computes the k smallest positive Eigenvalues of normalized Laplacian of the input graph in ascending order, to form the representation of the whole graph. Lara *et al.* use their approach to predict the properties of chemical compounds [183].

Mousavi *et al.* [184] introduce a whole-graph embedding hierarchical framework called **Pyramidal Graph Embedding**, based on some ideas originating from image processing algorithms. Important global information from images can be extracted by recursively analyzing local information. In the context of graphs, this means that the overall graph structure can be modeled by analyzing substructures at different scales. To this end, a graph pyramid is formed with subgraphs of different scales. Every graph is embedded into vector representations, which are all concatenated to form the global graph embedding. The representations are obtained by factorizing an affinity matrix. This pyramidal approach is especially designed for large graphs, since they potentially contain more different scales. Mousavi *et al.* used it for graph classification tasks.

Verma *et al.* propose a **Family of Graph Spectral Distances** [185] to represent a whole-graph. This method is built on the assumption that the graph atomic structure is encoded in the multiset of all vertex pairwise distances. It computes the Moore-Penrose Pseudoinverse spectrum of the graph Laplacian. A vector representation of the whole graph is constructed from the histogram of this spectrum. Typical tasks include graph classification in various fields such as bioinformatics and social networks [185], [186].

Tsitsulin *et al.* introduce **NetLSD** [186], a permutation- and size-invariant, scale-adaptive embedding method. Like the aforementioned vertex embedding method Laplacian Eigenmaps [156], NetLSD operates on the Laplacian matrix of the graph. It relies on a physical analogy consisting in simulating a heat diffusion process on the graph to preserve its structure. The method processes the amount of heat transferred between vertices at different times scales. These heat traces at different time scales are then used to compute the heat trace signature of the graph, i.e. the vector representation of the graph. Tsitsulin *et al.* use NetLSD for graph classification and for community detection.

Another popular approach involves leveraging models from the field of NLP, such as in **Graph2vec** [161] and **Graph Classification via Graph Structure Learning (GC-GSL)** [187]. Both are based on an analogy with the Doc2vec approach defined for text [47]. Graph2vec, one of the earliest methods for whole-graph representation learning, is an unsupervised and task-agnostic approach designed to learn whole-graph representations. Its principle is to consider graphs (documents, in the analogy) as collections of subgraphs (words that compose the document). The procedure enumerates rooted subgraphs around all vertices of the considered graphs. Each one represents the neighbor-

hood of a vertex (the so-called root) at a certain order.

Another family of approaches relies on graph autoencoders to learn the representation in an unsupervised way, e.g. **Permutation-Invariant Graph-level Autoencoder (PIGAE)** [188], or the **Denoising Autoencoder (DAE-based)** method from [189]. Such neural networks are constituted of 2 parts. First, the encoder receives a raw representation of the graph, which is compressed in order to remove redundant information and superfluous variability, and get a fixed-sized and compact representation. Second, the decoder is in charge of reconstructing the original input based on the compressed representation. The autoencoder is trained by minimizing the reconstruction error.

Deep Divergence Graph Kernels (DDGK) [190] is an unsupervised method that adopts a different approach based on attention. It relies on an encoder to learn a first compact representation of the graphs. Pairs of graphs are then fed to a cross-graph attention network, using this representation. This attention network is trained into reconstructing one representation based on the other. Based on the latter, DDGK is able to compute a divergence measure between the input graphs, which is then used to build the embedding space.

The literature contains another family of approaches, which adapts CNNs from the field of image processing to handle graphs, resulting in supervised methods able to learn whole-graph representations for specific classification tasks, e.g. **Patchy-San Convolutional Network (PSCN)** [191] and **NgramCNN** [192]. PSCN adapts the notion of convolution to the context of graphs, which allows applying the same principle as for image processing. The graph is represented by a collection of subgraphs, on which PSCN performs convolutions. These are then aggregated to create higher-level representations of the graph. NgramCNN operates in a similar fashion, but it first decomposes the graph into a sequence of n-gram blocks connected through overlapping regions, before applying convolutions on them.

Finally, another strategy is to modify vertex-oriented Graph Neural Networks (GNNs) in order to produce whole-graph representations, e.g. **Message Passing Neural Network (MPNN)** [193] or **Virtual Column Network (VCN)** [194]. This is conducted by the addition of a so-called master node, which is connected to all the other vertices. At the end of the training, the vector associated to this master node can be used as a representation of the graph.

5.2.3 Signed Graph Embedding

Signed graphs are commonly used in social sciences to represent friendly relations and enmity between individuals (modeled by vertices). It is much more difficult to handle signs than other attributes such as weights and directions in embedding methods, because of the complex relationships induced by the positive and negative edges. Although less common than for *unsigned* graphs, a number of graph embedding methods have been developed for *signed* graphs. However, they are all designed to operate at vertex-level and no signed whole-graph embedding method ex-

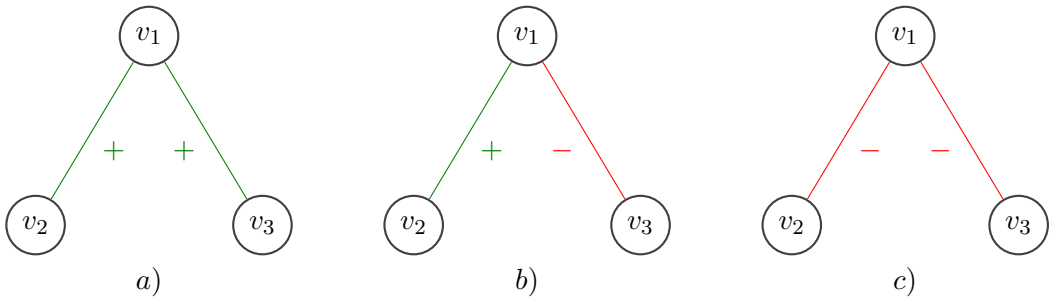

Figure 5.2: 3 types of open triads.

ists. Therefore, we dedicate this section to the review of the literature on signed *vertex* embedding.

Historically, the first type of such methods relies on random walks, e.g. **Signed Network Embedding (SNE)** [158] or **Signed Directed Embeddings (SIDE)** [159]. The general idea is to sample the graph using random walks, and feed them to a standard neural network, which learns a representation that preserves simultaneously graph structure and edge signs. Put differently, the representations of 2 well-connected vertices tend to be close in the embedding space, whereas those of 2 vertices connected by a negative edge tend to be distant.

Signed Network Embedding (SiNE) is a deep learning framework for vertex embedding in signed graphs [195]. Following [196], it is based on the assumption that the representation of a vertex should be similar to its positive neighbors, and dissimilar to its negative ones. To model signed networks based on this principle, SiNE proceeds at a local level by focusing on a very specific type of subgraph. It extracts the set of all open triads (i.e. 3 vertices connected by 2 edges) present in the graph. Figure 5.2 shows the 3 types of possible open triads. SiNE focuses on those containing one positive and one negative edges (Figure 5.2.b). This set of triads represents the graph, and is fed to a deep learning framework composed of 2 neural networks sharing the same weights. In accordance with the principle mentioned earlier, this model is trained in order to minimize the similarity between the representation of the vertex located at the center of the triad and its negative neighbors, while maximizing its similarity with its positive ones. One neural network within the framework is dedicated to the positive neighbors, while the other network handles the negative ones.

A limitation of this approach is that it ignores vertices whose neighbors are all positive or all negative. To handle this case, SiNE introduces a dummy vertex that is connected through a negative edge to each vertex with an all-positive neighborhood. This allows creating as many dummy triads containing one positive and one negative edges, which can be processed by the framework. The principle is the same for all-negative neighborhoods, except with positive dummy edges.

Most GCNs are designed to produce *vertex*-level representations. The general principle is as follows. Each vertex is initially represented by a vector, that can be generated randomly or

based on some vertex features. In a convolution layer, the representation of a given vertex is combined with those of its neighbors. The result is then fed to a generally non-linear function (e.g. multilayer perceptron) in order to get the updated vertex representation. The information is propagated through multiple layers to incorporate information from multi-hop neighbors. The maximal number of hops corresponds to the number of convolution layers in the network. GCNs achieve state-of-the-art performances on many tasks [160] such as vertex classification, edge prediction, and community detection.

Standard GCNs are only able to handle *unsigned* networks, though. Leveraging the information conveyed by edge signs mainly requires adapting the message passing rules used when computing the graph-based convolution. **Signed Graph Convolutional Networks** (SGCN) [160] allow doing so. This method relies on a dual hidden representation of a vertex, corresponding to its positive vs. negative reach sets, respectively noted $\mathbf{h}^+$ and $\mathbf{h}^-$. It uses balance theory to aggregate and propagate vertex representations across layers. Formally, the hidden representations $\mathbf{h}$ at layer $\mathbf{t}$ are updated as follows:

$$\mathbf{h}_t^+(u) = \sigma\Big(\mathbf{W}_t^+\Big[\sum_{v\in N^+}\frac{\mathbf{h}_{t-1}^+(v)}{k^+(u)}, \sum_{v\in N^-}\frac{\mathbf{h}_{t-1}^-(v)}{k^-(u)}, \mathbf{h}_{t-1}^+(u)\Big]\Big), \tag{5.1}$$

$$\mathbf{h}_t^-(u) = \sigma\Big(\mathbf{W}_t^-\Big[\sum_{v\in N^+}\frac{\mathbf{h}_{t-1}^-(v)}{k^+(u)}, \sum_{v\in N^-}\frac{\mathbf{h}_{t-1}^+(v)}{k^-(u)}, \mathbf{h}_{t-1}^-(u)\Big]\Big), \tag{5.2}$$

where σ is a non-linear activation function, $[\,]$ denotes the concatenation, and the $\mathbf{W}$ matrices are learnable weights. Ultimately, the dual hidden representations are concatenated to obtain a single vertex representation.

In addition, methods such as **Signed Graph Attention Networks (SiGAT)** [197] or **Signed Network Embedding via Graph Attention (SNEA)** [198] introduce attention to GNNs, in order to give more importance to relevant neighbors during the message passing step.

5.3 Datasets

A bottleneck in the development of whole-graph oriented methods for signed graphs is the lack of data. Indeed, these methods require collections of signed graphs, with each graph representing an instance in the treated task. However, the literature is mostly built around single, very large graphs, appropriate to test vertex representation learning techniques. The abuse detection task that we treat throughout this manuscript does not provide enough diversity to allow a correct assessment of our proposed methods. Therefore, we propose to extend it with 2 additional collections of signed networks annotated for diverse classification tasks. They come from various sources and are different in their structures and sizes. The benchmark that we constitute is thus composed of 3 collections of graphs. The first one is the *SpaceOrigin* collection extracted in Chapter 4.1 from

the raw dataset described in Section 2.3. It contains exactly the same graphs as before, so we do not further describe it in this section. The second one (Section 5.3.1) is an existing collection of artificially generated graphs, and the third one (Section 5.3.2) is an existing collection of real-world vote networks. We leverage the metadata associated to these collections, in order to repurpose them and define proper classification tasks. Global statistics describing the 3 constituted datasets are available in Table 5.1.

5.3.1 Correlation Clustering Instances

This dataset is proposed in [199], originally as a means to study the space of optimal solutions of the *Correlation Clustering* (CC) problem [166], described in Section 5.1. This graph partitioning problem consists in detecting internally collaborative but mutually hostile communities, which amounts to identifying the vertex partition that minimizes the total number of frustrated edges. The authors want to study the multiplicity and the diversity of the optimal solutions to CC. For this purpose, they define a random model and generate a collection of artificial graphs with planted partition, applying various levels of noise to control the problem difficulty. They use an exact method to identify all possible optimal solutions for each graph in this collection, and study how certain graph characteristics relate to the number of solutions.

Due to the NP-hard nature of CC, they focus on relatively small graphs, with a maximal order (number of vertices) of $n = 50$. They produce a total of 24,660 unweighted signed graphs, including 22,560 completely connected graphs (i.e. every pair of vertices is connected), while the remaining 2,100 graphs are not completely connected, with a density ranging from 0.25 to 0.75.

In order to use this dataset in the present work, we define a classification problem by associating each graph of the collection to a label. This problem, noted *Correlation Clustering Solutions* (CCS), consists in predicting whether there is a single vs. several optimal CC solutions for the graph of interest. It is thus a binary classification task.

5.3.2 European Parliament Roll-Calls

The third and last dataset is based on a collection of signed graphs extracted in [200] from a description of the voting activity at the European Parliament (EP). The raw data corresponds to roll-call votes cast individually by Members of the EP (MEPs) during plenary sessions, in the course of the 7th term (2009–2014). Such votes can take one out of 3 values: FOR (MEP supporting the proposition), AGAINST (MEP opposing the proposition) or ABSTENTION (MEP not taking a stand despite being present). MEPs can also be absent, and consequently not take part at all to a roll-call. Some MEPs resigned from their position in the course of their term of office, and some MEPs were appointed after the beginning of the term. In such cases, we consider them absent for all the roll-call that they missed.

Each network extracted by the authors corresponds to a specific roll-call, using vertices to model MEPs and edges to represent agreement between them: a positive sign models an identical vote, and a negative one a disagreement. Note that ABSTENTION is an expressed opinion, so a FOR *vs* ABSTENTION or AGAINST *vs* ABSTENTION is considered as a disagreement. Contrariwise, an absent MEP is not considered as expressing an opinion and no edge is associated with him. The goal of the authors is to study the polarization of the EP, and more specifically its voting patterns, and how these are affected by various criteria such as the topic of the voted proposition. For this purpose, they first identify factions of similarly voting MEPs in each roll-call network. Next, they compare the resulting vertex partitions to identify types of situations that result in a comparable voting pattern.

The raw data contains 6,595 roll-calls, from which the authors extract much more networks by leveraging the metadata associated to the voting activity. Since each MEP belongs to a member state, and to a European political group, they extract overall networks containing all MEPs, but also state- and group-specific networks, containing only the MEPs from a given member state or political group, respectively. In total, they produce 244,015 networks. Some of them are too small or too sparse (almost empty) to be interesting in a classification context, though. We constitute our own dataset by first filtering out these unusable instances. We consider state- and group-specific networks with 20 or more MEPs. Then, we randomly sample 6,000 networks among all the networks matching the previous criteria.

Like for the previous dataset, the initial collection of networks considered here is not originally used to perform any prediction task. We leverage the clusters of networks exhibiting similar voting patterns identified by the authors, and define a classification task consisting in predicting the number of factions identified in a network. There are 3 classes (i.e. 1 faction, 2 factions, 3 factions) in this task. We call the resulting dataset *European Parliament Factions* (EPF). We present the statistics of the 3 datasets in the next section.

5.3.3 Brief Comparison

Table 5.1 provides a few descriptive statistics for all 3 datasets, in order to help to interpret the classification results in Section 5.5.1. CCS is by far the largest dataset in number of graphs (*# Graphs*), however these are smaller. Moreover, the size of graphs in terms of vertices is not as uniform in both SO and EPF, covering 2 orders of magnitude. The number of edges is quite variable in all 3 dataset. In EPF, a few graphs have no positive or no negative edges at all. Overall, the graphs tend to be rather dense compared to unsigned real-world networks.

The Frustration, which we compute for both types of considered balance (SB vs. GB) is expressed as a proportion over the total number of edges in the graph, in order to have comparable values. The top right columns contain the number of classes in each dataset, and the Gini index showing how imbalanced they are.

Table 5.1: Statistics describing the 3 datasets. Notations ± and [ ] denote the standard deviation, and minimum & maximum.

Dataset	# Graphs	Avg. # Vertices	# Classes	Gini Index
SO	2,545	47.74 ±20.34 [2;214]	2	0.74
CCS	24,660	27.31 ±7.44 [16;50]	2	0.66
EPF	6,000	67.34 ±59.21 [20;274]	3	0.44

Dataset	Avg. # Negative Edges	Avg. # Positive Edges	% Positive Edges
SO	166.1 ±22.96 [1;1,692]	245.9 ±30.41 [1;2,323]	59.61 ±24.73 [0.0;100.0]
CCS	220.6 ±130.45 [25;833]	131.0 ±80.11 [24;392]	37.54 ±11.99 [20.1;79.0]
EPF	333.9 ±877.20 [0;15,933]	2,552.2 ±5,761.46 [0;33,153]	78.30 ±22.22 [0.0;100.0]

Dataset	Avg. SB Frustration	Avg. GB Frustration	Avg. density
SO	0.30 ±0.04 [0.01;0.48]	0.25 ±0.04 [0.01;0.46]	0.48 ±0.16 [0.10;1.00]
CCS	0.37 ±0.05 [0.03;0.51]	0.33 ±0.04 [0.01;0.49]	0.95 ±0.17 [0.19;1.00]
EPF	0.28 ±0.04 [0.01;0.46]	0.22 ±0.03 [0.00;0.45]	0.70 ±0.19 [0.07;1.00]

5.4 Proposed Methods

In this section, we describe the 3 families of methods which we propose to handle signed whole-graph representation learning. The first can be considered as a baseline, and relies on the aggregation of signed *vertex* embeddings (Section 5.4.1). The second is an adaptation of an *unsigned* whole-graph embedding method to **signed** graphs (Section 5.4.2). The third is based on a Graph Convolutional Network able to learn signed *vertex* representations, which we adapt to handle **whole** signed graphs (Section 5.4.3).

5.4.1 Signed Network Embedding

As described in Section 5.2.3, SiNE learns a compact representation for each vertex in the input graph, based only on its direct neighborhood. Since our tasks consist in classifying graphs and not vertices, our proposition is to aggregate the representations of all the vertices in the graph in order to get a graph-level representation. This approach is quite standard in the literature [201], [202]. It relies on the assumption that individual vertex representations each contain some part of information that describes the whole graph. Thus, aggregating all the vertex representations allows obtaining a graph-level representation. For the sake of completeness, we consider 2 approaches for the aggregation process: averaging the vertex representations, and summing them.

SiNE exhibits some limitations, though. First, it does not really handle vertices possessing only negative edges. The authors argue that this case is very uncommon in real-world data, but

such vertices exist in our datasets. Second, SiNE is based only on local information, while we can suppose that global information might be important to represent the graph in its entirety. Therefore, this method can be considered as a baseline to assess whether it is beneficial to use whole-graph embedding methods to learn representations for the classification of signed networks, compared to more standard vertex embedding methods.

5.4.2 Signed Graph2vec

Graph2vec [161] enumerates rooted subgraphs to represent the neighborhood of a vertex. These rooted subgraphs are named using labels obtained with the relabeling procedure of the Weisfeiler–Lehman isomorphism test [203] (WL for short). Starting with the degree as the initial vertex label, this procedure goes through 2 phases to iteratively update these labels. First, each vertex is described by a tuple constituted of its previous label, and a sorted multiset containing those of its neighbors. Second, each unique tuple is replaced by a new label, to be used in the next iteration. 2 identical tuples are replaced by the same label, but 2 different one get distinct labels. This phase allows keeping a compact representation of the vertices. At the end of the process, each rooted subgraph is represented by its root's label. More formally, the label update rule is:

$$\ell_t(u) = f\Big(\ell_{t-1}(u), \{\ell_{t-1}(v) : v \in N(u)\}\Big), \tag{5.3}$$

where $\ell_t(u)$ is the label of the subgraph rooted in u at iteration t, $N(u)$ is the neighborhood of u, and f is an injective function used to replace the tuples by new labels. The number of iterations corresponds to the desired order of the neighborhood covered by the rooted subgraph. A higher number of iterations leads to more distant vertices being considered during the relabeling phase.

At the end of this process, 2 isomorphic rooted subgraphs should get the same label. The obtained labels are then used to train the standard Doc2vec SkipGram model. Graph2vec has proven its effectiveness in many tasks involving the classification of *unsigned* graphs [204]–[206].

This method is not able to take advantage of the additional information present in signed graphs (i.e. edge signs), though. For this purpose, we define Signed Graph2vec, an adaptation of Graph2vec that relies on 2 variants of the WL relabeling procedure able to handle edge signs. The first one, noted `SG2Vn` (`n` for neutral), is straightforward and does not assume that the network has any form of structural balance. Regarding label initialization, instead of using the degree, we use both positive and negative degrees. For each vertex u, we first define a tuple $(k^+(u); k^-(u))$, and then replace each unique pair by a unique label using f. For the rest of the iterations, we proceed as in Graph2vec, except that we append the sign of the concerned edge in front of each neighbor when building the labels:

$$\ell_t(u) = f\Big(\ell_{t-1}(u), \{[s(u,v), \ell_{t-1}(v)] : v \in N(u)\}\Big), \tag{5.4}$$

where $s(u,v)$ is the sign of edge (u,v), and $[\,]$ denotes string concatenation. This allows distinguishing vertices holding the same label, but connected to the root with edges of opposed signs.

The second relabeling method, noted `SG2Vsb`, is the one proposed (for a different purpose) by Zhang *et al.* [207], and it assumes that the network is structurally balanced. Each vertex is represented by 2 labels, based on its positive and negative reachable sets, respectively (see Section 5.1). The authors do not explain how they perform their initialization, so we use the positive (resp. negative) degree for the positive (resp. negative) label. The method requires one update rule for each label:

$$\ell_t^+(u) = f\Big(\ell_{t-1}^+(u), \{\ell_{t-1}^+(v) : v \in N^+(u)\}, \{\ell_{t-1}^-(v) : v \in N^-(u)\}\Big) \tag{5.5}$$

$$\ell_t^-(u) = f\Big(\ell_{t-1}^-(u), \{\ell_{t-1}^-(v) : v \in N^+(u)\}, \{\ell_{t-1}^+(v) : v \in N^-(u)\}\Big), \tag{5.6}$$

At the end of the process, f is applied to tuples formed by the positive and negative labels of each vertex, resulting in the final rooted subgraph labels.

5.4.3 Signed Graph Convolutional Networks

This method is based on SGCN, whose implementation is conveniently available online[2], and can be modified to fit our needs. Similarly to `SG2Vsb` (Section 5.4.2), this method uses the balance theory to aggregate and propagate vertex representations across layers.

SGCN effectively learns representations of vertices in signed graphs. However, our objective is to handle whole graphs. A solution proposed for unsigned graphs in the literature [193] consists in adding a *master node* (also called *virtual* or *super* node), which is connected to all other vertices in the graph. One then uses the representation of this master node as the representation of the whole graph. The intuition is that, as the master node is connected to all parts of the graph, its representation is able to aggregate all its information.

In the context of signed graphs, connecting a master node to the rest of the graph is not trivial, though, as there are 2 different types of edges. We propose 5 interconnection schemes. Figure 5.3 shows an example for each one of them. The first 3 ones do not respect structural balance, and can be considered as baselines: `SGCN+` (Figure 5.3.a) and `SGCN-` (Figure 5.3.b) consist in connecting a single master node to the rest of the graph using only positive and only negative edges, respectively. With `SGCN±` (Figure 5.3.c), we use 2 distinct master nodes (one positive and one negative), which allows us to combine both previous schemes at once. The whole-graph representation is obtained by summing both master node representations. The fourth scheme, `SGCNsb` (Figure 5.3.d), is based on the (strict) structural balance. Using the *signnet* library[3], we detect the optimal graph bisection. Then, we connect one distinct master node to each cluster

[2] https://github.com/benedekrozemberczki/SGCN
[3] https://github.com/schochastics/signnet

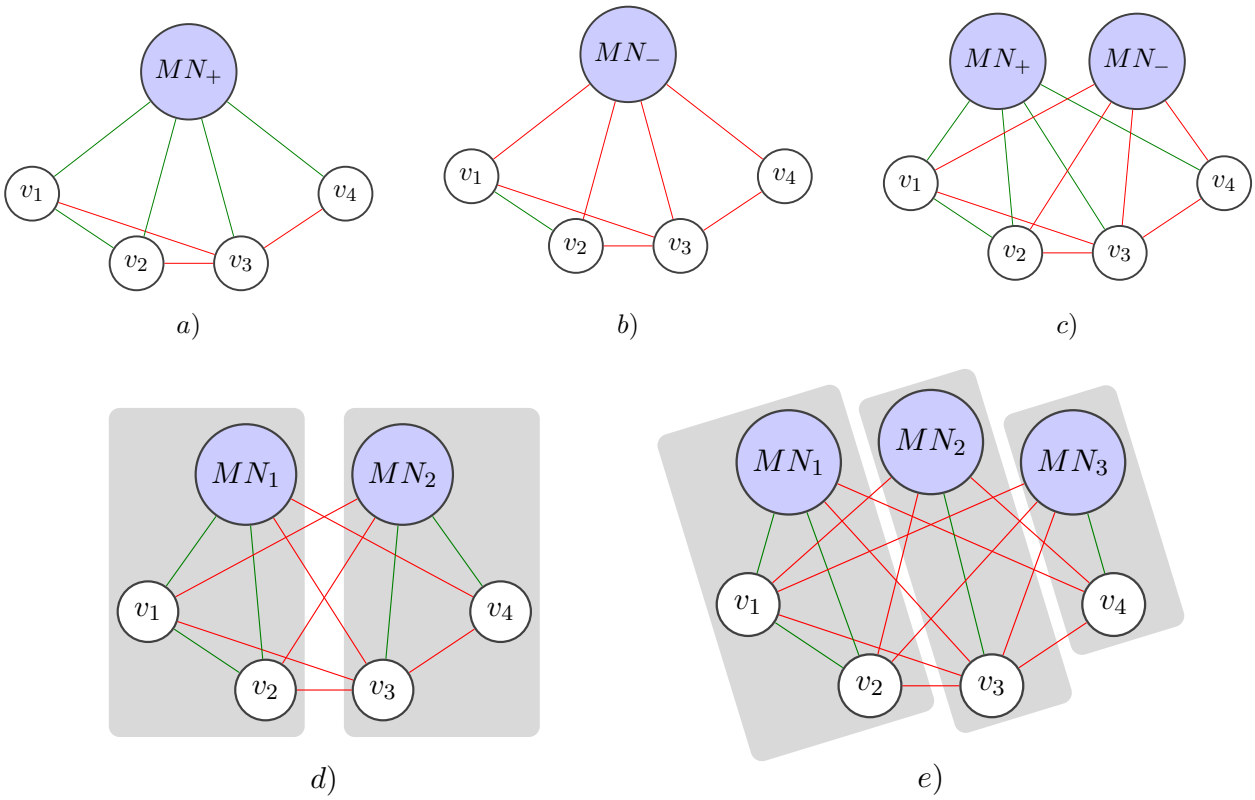

Figure 5.3: Examples of the 5 proposed interconnection schemes: SGCN+ (a), SGCN- (b), SGCN±(c), SGCNsb (d) and SGCNgb (e). *MN* stands for Master Node. Green and red edges represent positive and negative connections, respectively.

while respecting structural balance, i.e. positive edges within the cluster and negative ones with the other cluster. Like before, the whole-graph embedding is obtained by summing the representations of both master nodes. Finally, the fifth scheme, `SGCNgb` (Figure 5.3.e), relies on the *generalized* balance. Like before, we find the optimal graph partition, but this time there may be more than 2 clusters. We add one master node for each cluster, and sum their representations to get the whole-graph embedding.

5.5 Experiments

Our experimental protocol consists in assessing all the methods described in Section 5.4 on the datasets and tasks presented in Section 5.3. As a reminder, there are 3 tasks each based on a distinct dataset of signed graphs. The SO and CCS tasks contain 2 classes while EPF has 3. For all the tasks and methods, we first produce the representations and then train an SVM to perform the classification, using a 10-fold cross-validation. For the sake of consistency, this is the

Table 5.2: Results in terms of macro F-measure obtained with SiNE. Each line focuses on a task while columns focus on the type of aggregation functions for vertex representations.

Task	Sum	Average
SpaceOrigin (SO)	**55.42**	50.21
Correlation Clustering Solutions (CCS)	**50.48**	48.62
European Parliament Factions (EPF)	**69.81**	67.58

same classifier and the same procedure as in Section 4. For all tasks, we experiment with 1 to 5 iterations or layers. We fix the limit to 5 because it is the maximum average diameter among our 3 datasets. Furthermore, increasing this parameter requires a lot of computational resources for some methods. We conduct our experiments on an Nvidia RTX 2080 Ti GPU. We first present and discuss our detailed results for each method in Section 5.5.1. Then, we compare the 3 families of methods in Section 5.5.2.

5.5.1 Results

In this section, we discuss each family of methods separately: SiNE (Section 5.5.1.1), Signed Graph2vec (Section 5.5.1.2), and Signed Graph Convolutional Networks (Section 5.5.1.3). All the performance values are expressed in terms of macro F-measure, i.e. by computing the F-measure for each class separately, then averaging them to get the overall performance. This allows giving the same importance to all classes, even in imbalanced datasets as explained in Section 2.4.2.2.

5.5.1.1 Signed Network Embedding

We first present the results obtained with SiNE, the signed vertex embedding method that we consider as our baseline. The performance scores are shown in Table 5.2 for all 3 datasets.

On SO and CCS, the performances are close to a random classifier, which would get an expected 50% macro F-measure. The method is completely unable to learn correct representations of signed graphs for these 2 datasets. SiNE performs better on EPF with the best result reaching 69.81%. We assume that for this task, the local information available is often sufficiently discriminative. Moreover, vertex-oriented methods such as SiNE are usually designed to operate on very large graphs. This is not the case for the first 2 datasets. The graphs in EPF tend to be larger both in terms of vertices and edges. Hence, SiNE can extract a greater number of training triads, contributing to the improved model performance. Furthermore, the choice of the aggregation function has a notable impact on performance, with the sum operation yielding better results across all tasks. Averaging the representations has a tendency to smooth them which ultimately result in a loss of information.

5.5.1.2 Signed Graph2vec

Second, we present the results obtained with our 2 proposed variants of the *Signed Graph2vec* method: `SG2Vn` (which does not enforce SB) and `SG2Vsb` (which does). As a reference, we also include the original Graph2vec method (noted `G2V`) in the study, which simply ignores all edge signs. The performances are shown in Table 5.3 for various numbers of iterations. As a reminder, this parameter controls the number of iterations performed with the WL algorithm. In other terms, it controls the order of the rooted subgraphs extracted to describe the graph, and therefore the range taken into account when characterizing vertex neighborhoods. In the following, we discuss each dataset separately.

5.5.1.2.1 SpaceOrigin Conversations

The top part of Table 5.3 presents the performances of the 3 Graph2vec variants on the *Signed SpaceOrigin* dataset. It appears that they are affected diversely by the number of iterations. On the one hand, `G2V` and `SG2Vn` get their best score with a single iteration, and increasing iterations tend to reduce the performance. This is a tendency that we already observed with this method in a previous work [19]. On the other hand, `SG2Vsb` starts low but increases consistently with the number of iterations. In the end, it reaches an F-measure of 77.44% and outperforms both other variants. The average diameter is 5.47 in this dataset, which means 5 iterations correspond to an almost complete graph coverage.

`G2V` and `SG2Vn` obtain very similar performances. From this observation, we can assume that a lot of information is already conveyed by the unsigned graph structures, in this dataset. This is consistent with the results obtained in the article that published the original data [6], and with our experiments in Section 4, where we had some success performing a similar task with the unsigned version of this dataset. Nevertheless, `SG2Vsb` obtain better performances than the other 2 variants which indicates that the signs bring some additional discriminative power, which `SG2Vsb` is able to leverage. By comparison, this is not the case of *SG2Vn*. We infer that the relatively low level of SB Frustration in this dataset (0.30) favors methods relying on SB.

5.5.1.2.2 Correlation Clustering Solutions

The results obtained for this dataset are shown in the middle part of Table 5.3. The performance is clearly lower, for all 3 variants, and even similar to the score expected from a random classifier (50%), with a maximum F-measure of 52.57% obtained by `SG2Vn`. As explained previously, the dataset contains 2 subsets: some graphs are completely connected, and the rest are not. Assuming that one type of graph might be more difficult to handle than the other, we try training separately on these subsets. However, we do not get any significant difference between the obtained results,

Table 5.3: Results in terms of macro F-measure obtained with `Graph2vec` and our 2 proposed signed adaptations. Each column focuses on a specific number of iterations used when extracting rooted subgraphs.

Task	Method	1 it.	2 it.	3 it.	4 it.	5 it.
SpaceOrigin (SO)	G2V	75.09	71.32	72.62	73.96	73.77
	SG2Vn	74.85	71.44	72.15	72.88	72.37
	SG2Vsb	67.29	72.01	74.88	76.98	**77.44**
Correlation Clustering Solutions (CCS)	G2V	48.43	48.12	45.97	51.85	52.07
	SG2Vn	49.70	49.25	48.37	**52.57**	52.12
	SG2Vsb	49.84	51.81	49.70	51.37	51.62
European Parliament Factions (EPF)	G2V	45.63	49.76	52.31	60.43	63.14
	SG2Vn	75.63	81.44	84.18	86.16	88.68
	SG2Vsb	80.83	82.90	86.31	87.99	**89.98**

which are similar to those of Table 5.3.

We assume that either this classification task is too hard, in the sense that the information available in the graphs is not sufficient to perform the prediction, or that none of the 3 Graph2vec variants are able to capture the relevant information. The value predicted in this task is directly related to the distribution of edge signs (see Section 5.3.1), so we know with certainty that the information conveyed by signs is essential. The fact that `G2V` has similar performance to its signed counterparts hints at the second assumption (methods unable to capture relevant information). In addition, the higher level of SB Frustration (0.37) may hinder the performance of `SG2Vsb`, compared to `SG2Vn`.

Increasing the number of iterations eventually improves the performance, but the effect is not as strong and as stable as for the previous dataset. The graphs are more compact in this dataset, with an average diameter of only 3.60, which may partially explain this observation. Indeed, a few iterations are enough to retrieve all available information, and increasing their number does not bring any new neighbors.

5.5.1.2.3 European Parliament Factions

The results for this dataset are shown in the bottom part of Table 5.3. It appears that both our signed adaptations perform drastically better than the original unsigned method. The difference in F-measure is the largest of the 3 datasets: 63.14 (`G2V` vs. 88.68 (`SG2Vn`) and 89.98 (`SG2Vsb`). Edge signs thus appear to be even more important for this task than it was for the *Signed SpaceOrigin* dataset. The SB-based variant `SG2Vsb` is slightly above `SG2Vn`, which seems to indicate that this

type of structure may be relevant to this classification task. The average SB Frustration is 0.28 for this dataset.

This dataset contains larger graphs than the others, with an average of 67.34 vertices (vs. 47.74 and 27.31 previously) and a mean diameter of 4.12. This may explain the very strong effect of the number of iterations on the performance, even for G2V, the unsigned variant. It seems that even a small increase (proportionally to the network size) of the part of the graph covered when extracting rooted subgraphs, is enough to greatly improve the quality of the classification.

5.5.1.3 Signed Network Embedding

In this section, we present the results obtained with the 5 variants that we proposed for the SGCN method. Each one relies on the addition of one or several master nodes, through different interconnection schemes. The first 3 (SGCN+, SGCN−, SGCN±) ignore any type of structural balance, whereas the others enforce respectively SB (SGCNsb) and GB (SGCNgb). We consider 2 methods to extract a graph representation: using only the last layer vs. the sum of all layers. Preliminary experiments show that the former performs better, so we only focus on this approach in the following discussion.

We also include SGCN as a reference in our study, i.e. the original method without any master node. To get a graph-level representation, we proceed like for SiNE, and sum the representations of all vertices. We alternatively experimented with averaging them, but got lower performance, which explains why we focus only on sum, here. The results are shown in Table 5.4, for various numbers of convolution layers. Each additional layer allows considering neighbors that are located 1-hop farther away from the vertex. In the following, we discuss each dataset separately.

5.5.1.3.1 Signed Space Origin

The top part of Table 5.4 shows the results on the *Signed SpaceOrigin* dataset. Increasing the number of layers in the convolutional network results in better performances for all variants, but to different extents. For instance, when going from 1 to 5 layers, the performance gain is +7,1 F-measure points for SGCNgb but only +0.11 for SGCN±. With an average diameter of 5.47 in this dataset, using 5 layers allows an almost complete graph coverage.

All 3 variants that ignore any form of balance (SGCN+, SGCN−, SGCN±) are below the unsigned baseline (SGCN), which indicates that using signs improperly is counterproductive as it decreases the classification performance. SGCN+ largely outperform both other variants, probably because there are many more positive than negative edges in this dataset.

Among the variants that take balance into account, SGCNgb consistently outperforms SGCNsb: 55.67 vs. 73.69 with 5 layers. This shows that the type of balance selected when learning the representation must match the structural properties of the considered graphs. Interestingly, SGCN

Table 5.4: Results in terms of macro F-measure obtained with SGCN and our 5 proposed interconnection schemes. Each column focuses on a specific number of convolution layers.

Task	Method	1 lay.	2 lay.	3 lay.	4 lay.	5 lay.
SpaceOrigin (SO)	SGCN	66.48	67.21	68.06	68.87	69.54
	SGCN+	65.12	66.78	68.42	68.98	69.29
	SGCN−	54.19	54.89	55.56	55.59	55.89
	SGCN±	49.28	48.95	49.01	49.25	49.39
	SGCNsb	52.69	54.99	55.49	55.08	55.67
	SGCNgb	66.59	68.85	71.21	72.28	**73.69**
Correlation Clustering Solutions (CCS)	SGCN	70.29	70.65	71.27	71.43	71.89
	SGCN+	70.50	70.86	71.13	71.27	71.35
	SGCN−	70.14	70.55	70.84	71.02	71.18
	SGCN±	70.76	71.02	71.00	71.20	71.37
	SGCNsb	69.59	70.12	70.86	71.21	71.46
	SGCNgb	71.75	72.20	72.98	73.24	**73.49**
European Parliament Factions (EPF)	SGCN	90.16	90.87	91.63	92.04	92.65
	SGCN+	88.56	89.30	90.49	91.11	91.87
	SGCN−	88.61	89.30	90.49	91.09	91.80
	SGCN±	86.32	88.45	90.01	90.78	91.56
	SGCNsb	91.11	92.09	93.31	94.17	94.99
	SGCNgb	92.36	93.65	95.29	96.04	**96.43**

is the second-best method, which illustrates the methodological importance of the interconnection scheme when using a master node approach. It is on par with SGCNgb when using a single layer, because they are equivalent for this specific parameter value, however the difference quickly grows with the number of layers.

5.5.1.3.2 Correlation Clustering Solutions

The middle part of Table 5.4 shows the F-measure scores for the *Correlation Clustering Solutions*. On this dataset, all methods yield quite similar results, except for SGCNgb that once again obtains the best performances. In particular, this method is able to capture more information at the whole-graph level than SGCN at the vertex level. Increasing the number of layers still improves the results, but the effect is much weaker than for the previous dataset. The graphs are smaller there, which may explain this, as more layers do not bring more information after a certain point. These results also show that this task is not as challenging as assumed when discussing *Graph2vec* re-

sults, since it is possible to get scores much higher than the expected performance of a random classifier.

5.5.1.3.3 European Parliament Factions

The bottom part of Table 5.4 shows the results obtained for the EPF graphs. The behavior on this dataset is similar enough to that on *Signed SpaceOrigin*: the 3 variants that ignore balance (`SGCN+`, `SGCN-`, `SGCN±`) are below the original `SGCN` method (the latter being the worst, again), whereas `SGCNgb` gets the best results, peaking at 96.43% when using 5 layers. There are 3 differences, though: first, the overall performance is much better, with a minimal F-measure of 86.32. This is in line with the behavior exhibited by Graph2vec on the same datasets, and could be explained by the low level of Frustration (0.20). Second, `SGCN+` and `SGCN-` perform similarly. Third, `SGCNsb` is above `SGCN`, which could mean that many graphs have a 2-cluster structure in this dataset.

5.5.2 Comparison

We now compare and analyze the results of the 3 families of methods. Our baseline based on SiNE is consistently the least efficient method on all tasks. The gap of F-measure reaches up to 22.02 points with the best method, on the *Signed SpaceOrigin* dataset. This could be explained in part by the size of the graphs that constitute our benchmark: these are relatively small, whereas SiNE was designed to handle large graphs, with hundreds or thousands of vertices. Nevertheless, it remains that our results indicate that applying a method designed to handle whole graphs directly leads to much better classification results than simply aggregating multiple vertex representations.

Signed Graph2vec, through its `SG2Vsb` variant, obtains the best performance on the *Signed SpaceOrigin* dataset, while SGCN, through its `SGCNgb` variant, dominates on both other datasets. Overall, the variants that respect some form of structural balance obtain better results than the ones that do not. This confirms the interest of including this notion in the design of representation learning methods for signed graphs, in order to preserve this property in the representation space. However, it is worth noticing that datasets do not equally favor the 2 types of balance considered here: *Signed Space Origin* leads to better results with SB, whereas both others favor GB.

Increasing the number of iterations in Graph2vec, or that of layers in SGCN, has a positive impact on performances for all datasets, and for almost all variants, including the unsigned ones. This effect is generally stronger when the appropriate type of balance is leveraged to aggregate the representation of direct and indirect neighbors, though, which confirms the importance of this concept when dealing with signed graphs. For the SGCN variants, it also shows that the method does not suffer from oversmoothing on the considered datasets.

Signed methods consistently obtain better performances than the unsigned Graph2vec method

and all datasets. This proves that signs are indeed relevant information for the considered graph classification tasks. Including them directly in the representation learning process improves the quality of the learned representations for all tasks.

Graph Convolutional Networks are the best performing method, overall. However, this comes at a cost: they also are the most expensive in terms of computational runtime. Learning the representation of a graph with SGCN variants takes more than 25 seconds in average. As a comparison, Graph2vec variants take an average of 0.15 second per graph, and SiNE 0.65 second.

5.6 Conclusion

In this chapter, we have presented 2 methods for signed whole-graph embedding. The first is an adaptation to signed whole graphs of Graph2vec, an *unsigned* whole-graph embedding method. We generalize the original Weisfeiler-Lehman relabeling procedure in order to handle edge *signs*. Our second proposed method is based on SGCN, a Graph Convolutional Network method designed to learn signed *vertex* representations, which we adapt to handle signed *whole graphs*. Our proposition is to integrate one or several master nodes connected to all parts of the graph, and to use their representation as the representation of the whole graph. The intuition is that, as the master node is connected to all parts of the graph, this representation is able to aggregate all its information. This idea was first proposed for unsigned graphs in the literature, and we propose solutions to adapt it to signed graphs. The challenge is to determine how to connect the master node to the rest of the graph, since there are 2 different types of edges. We propose 5 interconnection schemes.

We constituted a benchmark for signed graphs classification containing artificially generated and real-world graphs. By applying our methods to 3 graph classification tasks included in this benchmark, we could show that both of them are more efficient on all tasks than our baseline based on the aggregation of vertex representations. In particular, methods respecting the structural balance obtain better results than the ones that do not. Therefore, this is a property that must be considered when developing an embedding method for signed graphs. To return to the 2 questions that we wanted to answer throughout this chapter, our experimental results show that methods specifically designed to handle whole graphs are indeed able to learn better representations of such objects compared to vertex oriented methods, especially for our dataset. This is an important result, since this is a part of the literature that is currently rather neglected. The second question was about the interest of using signed representations of graphs. We demonstrated that the signed methods were more effective than unsigned ones on all the datasets, confirming the benefit of considering signs.

We believe that this work opens new directions for future research. First, our proposed benchmark of 3 graph classification tasks allows evaluating and comparing methods designed for signed

whole-graph representation learning. This benchmark could be expanded with new collections of graphs, especially larger ones. Second, the master node approach could be applied to other types of signed GNNs such as Graph Attention Networks. Third, it would be interesting to generalize our methods so that they can use edge weights, as we showed in Section 4 that this brings some additional discriminative power when available: we propose such an extension in Chapter 6.1.3. Finally, another interesting future direction is to experiment with other master node interconnection schemes, based on other variants of Structural Balance. The concept of *Relaxed Balance* [208], [209], in particular, is very interesting, as it allows inter-cluster (resp. intra-cluster) edges to be positive (resp. negative). With these signed whole-graph embedding methods, we now have all the tools available to apply representation learning to abuse detection. This is the subject of our next chapter.

Chapter 6

Representation Learning for Abuse Detection

In Chapter 4, we noted 2 main limitations of feature-based approaches. On the one hand, one has to manually search the existing measures and select the most appropriate ones to constitute the feature set. To be exhaustive, it often has to include a few hundred features. On the other hand, the computation of these feature sets can be resource-intensive, in particular that of graph-based methods. Representation learning can solve this problem by automatically learning appropriate embedding representations of text and graphs. These methods can rely on edge attributes such as signs, weights, and directions to improve the representations. However, no signed whole-graph embedding method existed in the literature. In Chapter 5, we filled this gap by proposing 2 such approaches and showed that they performed better than their unsigned counterparts. In this chapter, we leverage these approaches, together with other existing representation learning methods, to tackle the same task as in Chapter 4, and thus study how automatically learned representations compare with feature engineering.

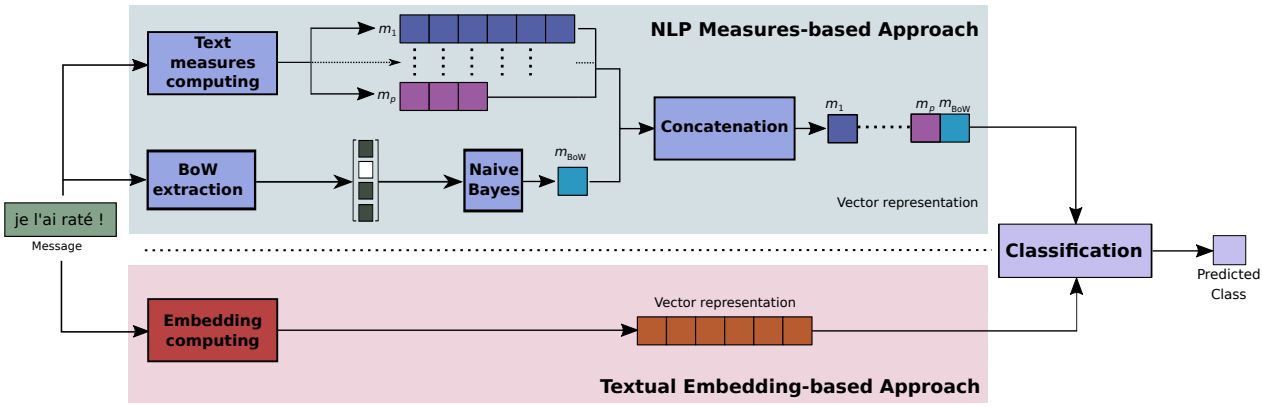

Figure 6.1: Illustration of the differences between the feature-based method presented in Section 4.2.1 (in orange) and text embedding methods used throughout this chapter (in red).

Lexical embeddings transform words into vectors that preserve their semantic and syntactic information. These methods offer robust representations that can help overcome the limitations of standard NLP techniques. Different methods can learn representations of text at different levels of granularity. Some of them provide representations of words, while others provide representations of sentences or documents. Textual embeddings have been used in a wide range of applications, including the detection of sexist and racist comments [49] and the offensive or hate speech detection [83], [210].

Graph embedding methods learn representations of various parts of graphs (i.e. vertices, edges, subgraphs, whole graphs, etc.). By construction, different methods are assumed to capture different aspects of the graph structure or properties, but it is difficult to compare them directly, for the same reason. One way to assess the appropriateness of a graph embedding method for a task is to do so empirically [170].

In this chapter, we apply embedding techniques to the abuse detection task. We follow the same procedure as in Chapter 4 with the exception that representations of text messages and conversational graphs are automatically learned through embedding methods instead of features. The differences between these 2 methodologies are illustrated by Figure 6.1 (text) and Figure 6.2 (graphs). We perform extensive experiments to compare multiple textual and graph embedding methods, including the signed whole-graph embedding approaches proposed in Chapter 5, to assess their appropriateness for this specific task. Our objective here is to determine how embedding methods perform compared to feature-based ones. Since the learned representations are not directly interpretable, it is not straightforward to understand exactly which information is captured by the embeddings. Therefore, we also perform an in-depth analysis of the methods to detect which properties are captured by which methods.

The following chapter is based on our works published in the international journal *SN Computer*

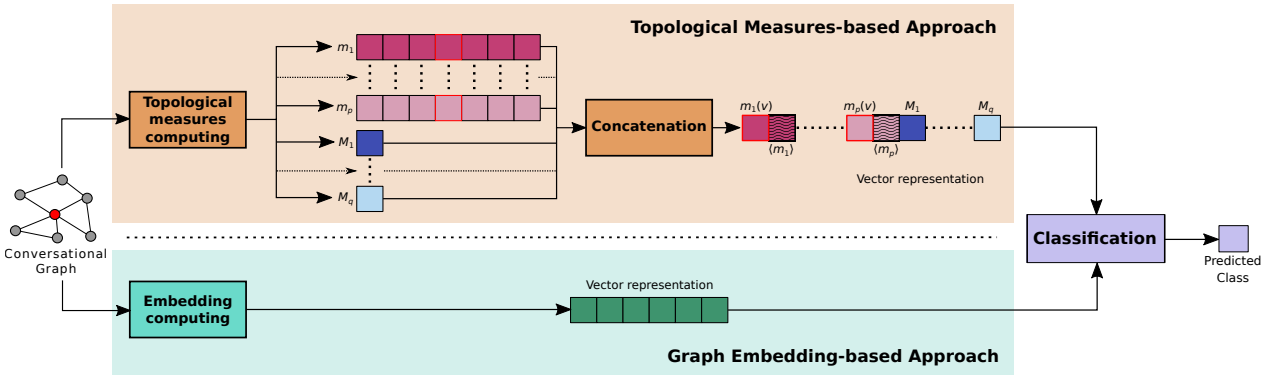

Figure 6.2: Illustration of the differences between the feature-based method presented in Section 4.2.2 (in orange) and graph embedding methods used throughout this chapter (in green).

Science (2021) [19], the French journal *Traitement Automatique des Langues* (2021) [21], and the *MARAMI* 2020 conference [22]. Some of the results on edge attributes and directions have not yet been published. This chapter makes the following contributions:

1. We propose extensions of the SG2V and SGCN models from Chapter 5 that can use edge weights and directions in addition to the signs

2. We propose 3 variants of Graph2vec, a graph embedding method, that integrate information relative to our task.

3. We experiment and compare 6 lexical embedding methods and 11 graph embedding methods designed to operate at different scales of the graph (vertex and whole graph). We assess these methods on the abuse detection task on the SpaceOrigin dataset and estimate their complementarities by combining them following the 3 fusion strategies presented in Section 4.2.3.

The rest of this chapter is organized as follows. In Section 6.1, we present all the lexical embedding methods that we use, the parameters of graph embedding approaches, the weighted directed extensions of the 2 signed whole-graph embedding methods developed in the previous chapter and the variants of Graph2vec that we propose to incorporate task-specific information. Next, we put all these methods into practice on our abuse detection task and discuss their results in Section 6.2. Then, we put a focus on determining which information is captured by every method in Section 6.3. Finally, we review our main findings in Section 6.4, and identify some perspectives for this work.

6.1 Embedding Methods

In this section, we present the lexical models from the literature, which we use in our experiments (Section 6.1.1). We then summarize the existing methods from the literature that we selected to produce graph embeddings (Section 6.1.2). Finally, we present some extensions of graph embedding approaches designed to fit our needs (Section 6.1.3.3).

6.1.1 Lexical Embedding Methods

Representation learning for text data is the task that consists of transforming textual input into vector representations which can then be used by machine learning algorithms for various tasks. Word embedding methods learn dense vectors of fixed size that represent a set of words, one vector per token in the vocabulary. They efficiently encode the semantic and syntactic information of words. Those that are used in similar ways obtain very close representations in the vector space, naturally capturing their meaning. However, representing polysemy and homonymy is difficult. We can distinguish 2 categories of word embedding approaches: the ones that are context-invariant and output a *fixed* representation for each word, and the *contextualized* ones that can generate different embeddings for a given word depending on the context in which it is used. Here, we experiment with 5 methods, the first 2 are context-invariant while the other 3 are contextualized. One constraint of our dataset is that it is constituted of French messages. Therefore, we selected models for which a pre-trained French version is publicly available.

Word2vec [174] generates fixed representations of words. It is one of the most popular word embedding approaches in recent years. In particular, it proposes the SkipGram model architecture which has been reused in numerous methods since. The model learns representations while trying to keep the semantic and/or syntactic similarity of words. Word2vec is based on a distributional hypothesis and learns the meaning of words from a large corpus of texts. Technically, it uses a neural network that has an input layer, an output layer, and a projection layer. The latter constitutes the word embedding. The output layer is used to perform a classification task allowing to train the model. There are 2 variants of Word2vec with symmetrical architectures: the Continuous Bag of Words (CBOW) model that predicts a word based on its context fed as input, and the SkipGram model that aims to predict the surrounding context words given the center word. Word2vec models are trained with a sliding window process, i.e. at each iteration, the model targets a word before moving one word forward. The size of that window determines how many words before and after a given word are included in its context. In this application, we set it to 10 words. We use a model[1] trained on French texts extracted from Wikipedia. Each word is associated with a representation of dimension 300.

[1] https://github.com/Kyubyong/wordvectors

Since the representation process of Word2vec is based on the words observed in the training corpus, this method is unable to deal with out-of-vocabulary words. For the same reason, uncommon words that appear only a few times in the training phase often obtain poor-quality representations. **fastText** [211] is an extension of Word2vec that was developed to mitigate the aforementioned limitation. It breaks words down to N-grams of characters instead of treating full words as Word2vec does. In this way, each word can be represented as a bag-of-characters N-grams (with 3 $\leq N \leq$6, generally). For instance, with $N = 3$, the word *mode* would be represented by (<mo,mod,ode,de>) where the " < " and " > " symbols correspond to the beginning and the end of the word. Once the character N-grams are extracted, a SkipGram model learns their representations. The representation of a word is obtained by summing all its N-grams. With this approach, rare and out-of-vocabulary words can still have quality representations since it is very likely that at least a portion of their N-grams appear in other words. We use a model[2] [211] trained on French texts from Wikipedia. All the representations are of dimension 300.

CamemBERT [212] and **FlauBERT** [213] are 2 French word embedding models directly adapted from the RoBERTa [214] architecture. Therefore, they share a lot of similarities. The main difference between them lies in the way that they mask tokens in the masked language model. CamemBERT also uses more training data. They both provide contextualized representations of words.

CamemBERT [212] is trained on a masked language modeling task. The authors follow the original architecture configurations of BERT. They propose a $\text{CamemBERT}_{\text{BASE}}$ model with 12 layers of encoders, 768 hidden dimensions, 12 attention heads and 110M parameters, and a $\text{CamemBERT}_{\text{LARGE}}$ model with 24 layers of encoders, 1,024 hidden dimensions, 16 attention heads and 335M parameters. According to the authors' study [212], the latter gives better results on named entity recognition but similar results on part-of-speech and dependency parsing tasks. Thus, we use the $\text{CamemBERT}_{\text{LARGE}}$ model in the remainder of this chapter. This model[3] is pre-trained on the *CCNet* [215] corpus containing 135GB of French text extracted from various websites. All the representations are of size 1,024.

Similar to CamemBERT, **FlauBERT** [213] proposes a $\text{FlauBERT}_{\text{BASE}}$ model composed of 12 layers of encoders, 768 hidden dimensions and 12 attention heads, and a $_{\text{LARGE}}$ model with 24 layers of encoders, 1,024 hidden dimensions and 16 attention heads. These models are trained on a corpus of 71GB of French text aggregated from multiple online sources. We use the $_{\text{LARGE}}$ model[4] that provides embeddings of dimension 1,024.

Unlike the 4 previous approaches, **Flair** [216] learns representations that are built at the character level and not the word level. This feature allows Flair to be particularly efficient when dealing with rare or misspelled words and to better model language-related concepts such as prefixes

[2] https://github.com/flairNLP/fasttext
[3] https://camembert-model.fr/
[4] https://github.com/getalp/Flaubert

and suffixes. This architecture uses a bi-directional LSTM operating on characters. The model is trained to predict the next character for each element in the sequence to process. Therefore, it learns 2 hidden representations for each character of the sequence: one for the forward network and one for the backward network. The embedding of a word is obtained by combining the forward representations of the characters located before the end of the word and the backward representations of the characters located after the beginning of the word. We use the forward and backward models[5] trained on French Wikipedia articles. Each model generates representations of size 1,024 that we concatenate to obtain the final embedding of a word, with 2,048 dimensions.

6.1.2 Selected Graph Embedding Methods

In this chapter, we compare several graph embedding techniques applied to abuse detection. This task consists of classifying conversational graphs. However, our approach to represent conversations through conversational graphs is particular. Indeed, vertices represent authors and the objective is to classify one message posted by a specific author, given the context represented by the whole graph. Therefore, one could assume that the vertex corresponding to this specific author is more important than other vertices and that using the representation of that single vertex could be enough to characterize the whole conversation. Furthermore, it is possible to construct a representation of the graph by combining vertex-level features. For this reason, we include both vertex and whole-graph embedding methods in this study. The methods from the literature that we selected use different strategies and focus on preserving various aspects of the graph, to include as much diversity as possible. All implementations are from the *Karate club toolkit* [201] except Node2vec, which was implemented by E. Cohen[6].

In the remainder of this section, we list the methods selected for our experiments, half of them treating the graph at the vertex scale, and the other half treating the whole graph. We selected them to represent a diversity of techniques. For each, we indicate their parameter settings, which we experimentally determined in a previous study [19]. In the following description, the names of the parameters correspond to those used in these toolboxes. Table 6.1 summarizes the exact parameter settings for all the graph embedding methods, which are described in more detail in the previous chapter (Section 5.2). Table 6.2 shows a summary of the edge attributes (weights, directions, signs) that the graph embedding methods are able to handle.

6.1.2.1 Vertex Embedding Methods

In the conversational graph extracted from the context of the targeted message, all vertices are not equal. One of them represents the author of the targeted message (represented in red

[5] https://github.com/flairNLP/flair
[6] https://github.com/eliorc/node2vec

Table 6.1: Parameters of the graph embedding methods.

Parameter	FGSD	SF	NGNN	G2V	SG2V	SGCN	DW	WL	N2V	BNE	GW	KH
dimensions	200	128	64	128	128	128	128	32	128	8	100	64
hist_range	10	-	-	-	-	-	-	-	-	-	-	-
wl_iterations	-	-	-	1	5	-	-	-	-	-	-	-
layers	-	-	8	-	-	5	-	-	-	-	-	-
down_sampling	-	-	-	10^{-4}	10^{-4}	-	-	-	-	-	-	-
learning_rate	-	-	10^{-3}	0.06	0.06	-	0.05	0.05	-	-	-	10^{-3}
epochs	-	-	300	12	12	-	-	-	-	-	-	200
min_count	-	-	-	1	-	-	1	1	-	-	-	-
window_size	-	-	-	-	-	-	10	4	10	-	-	-
walk_number	-	-	-	-	-	-	5	5	10	-	-	-
walk_length	-	-	-	-	-	-	80	80	20	-	-	-
p	-	-	-	-	-	-	-	-	0.95	-	-	-
q	-	-	-	-	-	-	-	-	1.0	-	-	-
iterations	-	-	-	-	-	-	-	-	-	16	-	-
order	-	-	-	-	-	-	-	-	-	1	-	-
alpha	-	-	-	-	-	-	-	-	-	0.01	-	-
step_size	-	-	-	-	-	-	-	-	-	-	0.2	-
heat_coefficient	-	-	-	-	-	-	-	-	-	-	0.5	-
approximation	-	-	-	-	-	-	-	-	-	-	100	-
switch	-	-	-	-	-	-	-	-	-	-	1,000	-

in Figure 4.4), which we assume plays a particular role if an abuse is occurring at this moment of the conversation. We can suppose that this vertex is the most important in the graph, and that characterizing it individually could be enough to represent the full conversation. The vertex embedding methods presented in this section allow learning the representation of this vertex.

DeepWalk [175] (DW) samples a certain number of uniform random walks starting from each vertex to represent the graph structure. There are multiple parameters to tweak the generation of random walks. We extract 5 random walks starting at each vertex of the graph with a maximum length of 80. We use a neighborhood of 10 vertices and consider all vertices appearing at least once in the graph. The representations are of dimension 128.

Node2vec [85] (N2V) also uses random walks to characterize the graph structure. When drawing these random walks, it introduces a bias controlled by 2 parameters. The return parameter `p` controls the likelihood of immediately revisiting a vertex during the walk, and the in-out parameter `q` controls the balance between the breadth-first and depth-first strategies. We use values 0.95 and 1.00 respectively in order to obtain representations that capture the global structure of the graph and the structural equivalence between vertices. We extract 10 random walks of length 20 starting from each vertex with a `window_size` of 10. The representations are of size 128.

Walklets [179] (WL) is an extension of DeepWalk which aims at explicitly modeling multi-scale

relationships, i.e. combining distinct views of vertex relationships at different granularity levels. The key change is in the random walk sampling algorithm, as the walks can now skip some vertices to reach farther parts of the network. It creates a representation for each size of *skip*) and the output representation is the result of their concatenation. We use the same parameters as DeepWalk, except for the `window_size` that denotes the size of *skips* in random walks. We use a value of 4 and 32 dimensions for each representation. The final representation is thus of size 128, the product of the values of these 2 parameters.

BoostNE [173] (BNE) also learns multiple graph representations at different granularity levels, but unlike Walklets, it relies on matrix factorization. It successively factorizes a vertex connectivity matrix to obtain representations corresponding to an increasingly finer granularity. The final embedding is obtained by concatenating these representations. We consider 16 `iterations` that each generates a representation of size 8, for a final embedding of size 128.

GraphWave [217] (GW) mimics a physical process consisting in propagating some energy through the graph structure, starting from the vertex of interest. The way this energy is diffused over the graph is assumed to characterize the vertex and its neighborhood. The `heat_coefficient`), which corresponds in terms of graph structure to the radius of the considered vertex neighborhood, is set to 0.5. We set the size of the embeddings to 100.

k-hop Graph Neural Networks [218] (KH) aggregates the direct neighbors of a vertex and its k-hop neighborhood through a graph neural network. We use a learning rate of 0.001 with 200 epochs to obtain representations of dimension 64.

6.1.2.2 Whole-Graph Embedding Methods

Using a description of the whole graph amounts to consider the entire conversation at once when performing the classification. It can help better capturing the global dynamics of the exchanges between users in the graph.

Family of Graph Spectral Distances [185] (FGSD) discretizes the distribution of the vertex pairwise distances through a histogram. We can control the way this histogram is computed. We set the range covered by our histogram to 10 and generate representations of dimension 200.

Spectral Features [183] (SF) computes the spectrum of the normalized graph Laplacian, keeping only the k smallest positive Eigenvalues. These Eigenvalues, in ascending order, form the representation of the graph. Parameter k, called `dimensions` in the implementation, therefore directly controls the size of the representation. We set it to 128.

Nested Graph Neural Networks [219] (NGNN) extracts rooted subgraphs and pool their representations into a single one to represent the whole graph. We use a learning rate of 0.001, with 300 epochs and 8 GNN layers to learn representations of dimension 64.

Graph2vec [161] (G2V) considers a graph as a collection of subgraphs, which are then used to train a SkipGram model. More precisely, it looks for so-called *rooted* subgraph, i.e. vertex

neighborhoods of a certain order. We extract rooted subgraphs of degree 1 (`wl_iterations`), with an embedding size of 128.

Signed Graph2vec (SG2V) is a method that we proposed in Chapter 5.4.2. It is an adaptation of the original Graph2vec approach that is able to learn representation of signed graphs. Following the results in Chapter 5.5.1, we extract rooted subgraphs of degree 5 (`wl_iterations`) with a representation size of 128. We use the variants which enforce structural balance (SG2Vsb).

Signed Graph Convolutional Networks (SGCN) is the other signed whole-graph embedding method that we developed (Chapter 5.4.3). It includes *master* vertices to connect some parts of the graph and uses the representations of such global vertices to represent the whole graph. We use 5 layers in our graph convolutional network and generate representations of size 128. We use the best performing variant (SGCNgb), which enforces the generalized balance.

Scale	Method	Reference	Weighted	Directed	Signed
Vertices	DeepWalk	[175]	-	-	-
	Node2vec	[85]	✓	✓	-
	Walklets	[179]	-	-	-
	BoostNE	[173]	✓	-	-
	GraphWave	[217]	-	-	-
	KH	[218]	-	-	-
Whole graph	FGSD	[185]	✓	-	-
	Spectral Features	[183]	-	-	-
	NGNN	[219]	-	-	-
	SG2V	Section 5.4.2	-	-	✓
	WD-SG2V	Section 6.1.3.1	✓	✓	✓
	SGCN	Section 5.4.3	-	-	✓
	WD-SGCN	Section 6.1.3.2	✓	✓	✓
	Graph2vec	[161]	-	-	-
	Graph2vec-author	Section 6.1.3.3	-	-	-
	Graph2vec-target	Section 6.1.3.3	-	-	-
	Graph2vec-distance	Section 6.1.3.3	-	-	-

Table 6.2: Summary of the edge attributes handled by the selected graph embedding methods.

6.1.3 Proposed Whole-Graph Embedding Methods

In Chapter 5, we proposed SG2V and SGCN as whole-graph embedding methods able to operate on undirected unweighted signed graphs. However, in Chapter 4.4.2 we found that consid-

ering edge attributes and directions has a positive impact on the detection performance of abusive messages. Consequently, we think that including edge weights and directions directly in the representation learning process of SG2V and SGCN could improve the quality of their learned representations of whole graphs. We first introduce a Weighted Directed variant of SG2V (WD-SG2V) in Section 6.1.3.1 that generalizes the Weisfeiler-Lehman (WL) relabeling procedure in order to handle these supplementary edge attributes. Then, we propose a Weighted Directed variant of SGCN (WD-SGCN) in Section 6.1.3.2. Finally, we propose an extension of the Graph2vec model in Section 6.1.3.3, which includes certain task-related information into the embedding.

6.1.3.1 Weighted Directed SG2V

In Section 5.4.2, we proposed 2 signed adaptations of the Graph2vec model relying on 2 variants of the WL relabeling procedure able to handle edge signs. To incorporate supplementary edge attributes in the learned representations, we follow the same procedure and propose a new variant able to handle edge weights and directions. The original Graph2vec label update rule is an iterative process where a vertex is described by a tuple constituted of its previous label, and a sorted multiset containing those of its neighbors. It is formally defined in Equation 5.3. In order to incorporate both other edge properties, we first consider the set of *outgoing* neighbors of a vertex u and split them into 4 quartiles based on the *weight* of the concerned edges. Based on this breakdown, we can attribute a quartile number (i.e. 1, 2, 3 or 4) to each edge. This allows distinguishing vertices holding the same label but connected to u with different weight levels. We experimentally determined that splitting the weights into quartiles is the best option, as using fewer quartiles does not provide enough possibilities to distinguish edges, and using more of them leads to too many possible labels, hence reducing the efficiency of the method.

The first variant that we proposed in Section 5.4.2, SG2Vn, uses the original label update rule except that it appends the sign of the concerned edge in front of each neighbor (Equation 5.4). Our proposition for **WD-SG2Vn** is to further append the quartile number in front of each neighbor when building the labels in the following manner:

$$\ell_t(u) = f\Big(\ell_{t-1}(u), \{[q(u,v), s(u,v), \ell_{t-1}(v)] : v \in N(u)\}\Big), \tag{6.1}$$

where $q(u,v)$ and $s(u,v)$ are the quartile number and the sign of edge (u,v), respectively, and $[\,]$ denotes string concatenation.

The second variant proposed in Section 5.4.2, SG2Vsb, creates 2 labels for each vertex, based on its positive and negative reachable sets. Our proposition for **WD-SG2Vsb** is the same as above, i.e. adding the quartile number in front of each positive and negative neighbor when building the labels. The update rules for the positive and negative labels are:

$$\ell_t^+(u) = f\Big(\ell_{t-1}^+(u), \{\Big[q(u,v), \ell_{t-1}^+(v)\Big] : v \in N^+(u)\}, \{\Big[q(u,v), \ell_{t-1}^-(v)\Big] : v \in N^-(u)\}\Big) \quad (6.2)$$

$$\ell_t^-(u) = f\Big(\ell_{t-1}^-(u), \{\Big[q(u,v), \ell_{t-1}^-(v)\Big] : v \in N^+(u)\}, \{\Big[q(u,v), \ell_{t-1}^+(v)\Big] : v \in N^-(u)\}\Big). \quad (6.3)$$

Once all the iterations are completed, f is applied to tuples formed by the positive and negative labels of each vertex, resulting in the final rooted subgraph labels.

6.1.3.2 Weighted Directed SGCN

SGCN is a graph embedding method that originally learns representations of vertices in signed graphs. In Section 5.4.3, we adapted it to handle whole graphs by adding master vertices to the graph and using their representations as the representation of the whole graph. To make this model able to use edge weights and directions, we propose a modification of its aggregation step. The original model consists of collecting features from neighboring vertices and aggregating them. In **WD-SGCN**, to obtain the representation of each vertex u, we first consider only their set of *ingoing* neighbors as they are the ones that have influence on the vertex. Then, to integrate the fact that all these neighbors have different degrees of importance depending on the weight of the edge shared with the vertex of interest, we perform a weighted aggregation of the neighbors. In other words, we weigh the features from neighboring vertices by the edge weight that we normalize to obtain a sum of 1. Then we aggregate them. This gives more importance to neighbors with a strong relationship in the resulting representation. We apply this procedure with all the 5 interconnection schemes that we proposed for SGCN.

6.1.3.3 Tuning Graph2vec

In addition to the standard Graph2vec [161] model presented in the previous section, we use a feature of that method to propose 3 Graph2vec variants specifically designed for our abuse detection task. Graph2vec allows labeling each vertex used in the rooted subgraph extraction process. In this way, 2 subgraphs containing vertices with identical labels are considered similar. This introduces an additional concept of similarity between vertices in addition to the graph structure. If no label is given, Graph2vec uses the vertex degree as the default label. We propose and assess the performances of 3 alternative labeling procedures that are designed for our task, and illustrated in Figure 6.3.

The first one, called **Graph2vec-author** (Figure 6.3.a), consists of using the unique identifier associated with the author of a post in our dataset. Therefore, we can identify if a same author is active in several conversations. Our intuition is that identifying the authors could allow us to consider context at a larger scale than a simple conversation. For instance, the fact that 2 authors

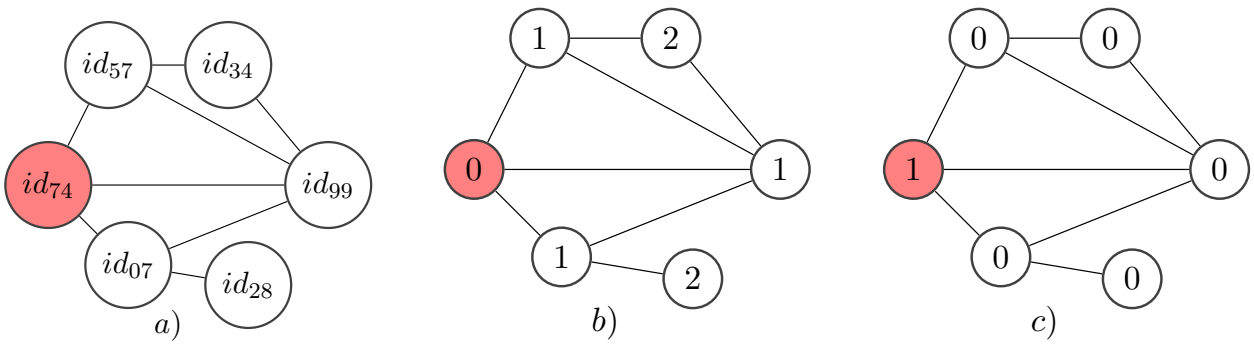

Figure 6.3: Examples of the 3 labeling procedures: Graph2vec-author (a), Graph2vec-distance (b), Graph2vec-target (c). The targeted vertex is represented in red.

interact in multiple conversations and that there is a history between them could be an important information.

With **Graph2vec-distance** (Figure 6.3.b), we compute the distance, between the vertex and the targeted vertex, and use it as its label. The purpose is to group different vertices (i.e. users) with the same label depending on their proximity with the author represented by the targeted vertex. The groups with low distances could correspond to the persons close to the author while the groups with high distances represent the users that only have brief contact with him. This strategy reduces the number of distinct labels in the graph and could make it easier to detect structural equivalence.

Finally, **Graph2vec-target** (Figure 6.3.c) is an extension of the previous strategy that further reduces the number of labels in the graph down to 2. With *Graph2vec-target*, we use a binary label that indicates whether the vertex is the targeted one or not. The intuition is that the targeted vertex plays a particular role in the graph, thus it could be important to differentiate it from others.

These labels generated by the 3 strategies are all based on the users and their proximity to the author of the targeted message. Therefore, they introduce information that is directly related to our task. For all these methods, we use representations of size 128.

6.2 Experiments

In this section, we present our experimental protocol (Section 6.2.1) and the performances obtained by the studied lexical and graph embedding methods (Section 6.2.2). Finally, we combine these embeddings (Section 6.2.3) as we did in Chapter 4 and summarize our findings (Section 6.2.4).

6.2.1 Experimental Protocol

The objective of the following experiments is to draw a comparison between embedding methods and feature-based approaches studied in Section 4. To this end, we use the same experimental protocol as the one described in Section 4.3.1. We use the SpaceOrigin dataset and conduct the experiments on the binary classification task consisting of detecting instances of *Abusive* and *Non abusive* messages. We separate the experiments into 2 distinct aspects: textual and conversational.

The textual aspect relies on the content of the messages to perform the detection. We use the 5 lexical embedding models described previously in Section 6.1.1. As a baseline, we also include the feature-based method developed in Section 4. The 5 approaches generate embeddings at the word level. As our objective is to represent the message as a whole and not individual words, we use the standard technique of averaging the representations of all words in a message to obtain its global embedding [220]. Furthermore, the methods based on the BERT architecture, i.e. CamemBERT and FlauBERT, directly learn a global representation of the sequence through a particular token added at the very beginning of the sequence. We use this global representation as a second technique to represent the messages, in addition to the averaged representations.

The conversational aspect focuses on the use of contextual information. It relies on the conversational graphs extracted from the SpaceOrigin dataset. They represent the full conversations (i.e. context) in which messages were posted. In the following experiments, we use the models described in Section 6.1.2. We also include the feature-based baseline developed in Section 4.2.2 for reference. Additionally, we experiment with the proposed weighted directed variants of SG2V and SGCN, and the 3 strategies proposed to tune Graph2vec for this specific task. Half of the selected methods learn vertex embeddings and the other half focus on the whole graph. The latter directly learns representations of the graph as a whole, we use these representations. For the former, we use 2 strategies. First, we use only the representation of the vertex corresponding to the author of the targeted message. Second, we average the representations of all vertices in the graph into a single global one.

All these methods are used to learn representations of messages which are ultimately used as input values of an SVM. As in Chapter 4.3, we perform a 10-fold cross-validation with 70%-train / 30%-test split. The performance is expressed in terms of macro F-measure.

6.2.2 Classification Results

In this section, we present the results of our experiments with lexical and graph embeddings on the abuse detection task. We also include the content- and graph-based original methods developed through feature engineering (Chapter 4), as a reference. In the following, we discuss each aspect separately.

6.2.2.1 Lexical Embedding

Table 6.3 shows the results obtained by the baseline and the 5 lexical embedding methods that we consider. The first column denotes the scale of the generated representations: *Word* for the methods that generate representations of words that we average to obtain the representation of a message, and *Message* for BERT-based methods able to directly learn a global representation of the message. The third column shows the dimension of the learned representations. We keep the original dimension of all the pre-trained models used.

For most models, the most frequent classification error is to label a *non-abusive* message as *abusive*. This phenomenon represents approximately 60% of the total classification errors. Some non-abusive messages are quite ambiguous and therefore, difficult to classify.

Scale	**Method**	**Dimension**	F**-measure**	**Total Runtime**
-	Baseline-Text	29	75.21	0:41
Word	Word2vec	300	71.14	04:14
	fastText	300	76.01	04:29
	Flair	2,048	77.33	05:25
	CamemBERT-W	1,024	**81.02**	06:56
	FlauBERT-W	1,024	78.46	07:11
Message	CamemBERT-M	1,024	80.96	04:59
	FlauBERT-M	1,024	74.99	05:08

Table 6.3: F-measures obtained by our baseline approach (based on feature engineering) and the 5 word embedding methods.

Our first observation is that all the word embedding approaches but one obtain a better F-measure than the baseline. Word2vec and fastText are among the least effective approaches for this classification task. This is not surprising since they are the 2 methods that produce fixed representations of words, i.e. a single representation for each word, independent of the context. Therefore, they are unable to distinguish homographs which can lead to confusion on the meaning of a message. The gap in terms of performance between these 2 methods can be explained by the fact that Word2vec is completely unable to treat words that were not seen during training (i.e. out-of-vocabulary words) while fastText, by construction, is able to deal with them. In the context of online messages, this feature is essential since spelling mistakes, rare and very specific terms, or even intentional obfuscation, are more than common.

The contextualized embeddings learned by Flair yield better performances, but they are lower than all other contextual methods. It seems that the character-level representation of Flair is not well suited for the abuse detection task, or at least, for our dataset.

FlauBERT-M, which learns a global representation of the message, gets lower performances than its counterpart Flaubert-W, which creates the embedding by averaging the representations of words composing the message. We assume that the abusive nature of a message is often based on just a few words, and that operating directly on words allows improving their detection. However, we do not find this situation with CamemBERT since its 2 variants obtain performances that are not statistically different. CamemBERT-W, nonetheless, gets the best F-measure (81.02%) of all the lexical embedding methods. This constitutes an improvement of almost 6 points compared to our baseline.

These results highlight all the potential of word embedding methods. On this classification task, they are able to learn representations that capture much more discriminative information compared to the baseline based on a handcrafted set of morphological and language features computed on the message. This directly leads to an improvement of up to 6 points in terms of F-measure on the abuse detection task. Furthermore, these methods are not built following the same concepts and objectives, therefore, we can suppose that they could be complementary. For instance, Le *et al.* [213] show that the combination of CamemBERT and FlauBERT yields better performances than the 2 methods used separately on a POS tagging task. We explore these options in Section 6.2.3.

On the other hand, word embedding methods require much more time than the baseline to obtain the representations of messages. This could be a limitation in a real-world application where the time to classify a message is limited.

6.2.2.2 Graph Embedding

The results of our graph embedding methods and the baseline are shown in Table 6.4. The last 3 rows are the Graph2vec variants tuned to include additional information through vertex labels. The first column indicates whether it is a *vertex* or *whole-graph* embedding method. With these results, we want to answer 3 research questions.

- First, which scale is the most efficient for learning representations of conversational graphs (Section 6.2.2.2.1)?
- Second, which edge attributes (weights, signs or directions) should be considered in the representation learning process of graphs (Section 6.2.2.2.2)?
- Third, which vertex attribute brings more information to the representation (Section 6.2.2.2.3)?

6.2.2.2.1 Representation Scale

We evaluated 3 different strategies in Table 6.4: learning a representation of the whole graph, using the representation of the targeted vertex, and averaging the representations of all vertices in

Scale	Method	Dimension	F-measure	Total Runtime
-	Baseline-Graph	477	83.40	03:51:25
Vertices	DeepWalk	128	73.72	14:29
	DeepWalk-avg	128	73.54	
	Node2vec	128	74.59	15:01
	Node2vec-avg	128	74.44	
	Walklets	128	74.84	16:21
	Walklets-avg	128	74.21	
	BoostNE	136	71.53	16:00
	BoostNE-avg	136	71.51	
	GraphWave	200	80.34	15:12
	GraphWave-avg	200	79.58	
	k-hop GNN	64	73.96	24:55
	k-hop GNN-avg	64	73.55	
Whole graph	FGSD	200	71.13	06:35
	Spectral Features	128	74.23	06:57
	NGNN	64	74.51	16:58
	SG2V	128	79.13	06:24
	WD-SG2V	128	79.31	06:53
	SGCN	128	74.74	14:14
	WD-SGCN	128	75.02	14:50
	Graph2vec	128	78.17	04:54
	Graph2vec-author	128	78.24	05:10
	Graph2vec-target	128	78.05	05:08
	Graph2vec-distance	128	**80.47**	05:04

Table 6.4: F-measures of our baseline and graph embedding methods, including the 3 proposed variants of Graph2vec. In the top part, avg denotes a method averaging all vertices in the graph.

the graph. For these last 2 strategies, the representation of the targeted vertex always gives better performances than averaging all the vertices' representations. We can assume that the targeted vertex is the most important for our classification task and considering the other vertices does not bring additional information, and even add noise to the learned representation.

Between vertex and whole-graph approaches, there is no clear distinction in terms of F-measure. Since the graph in its entirety represents the message and its associated conversation, one could assume that embedding the whole graph could allow capturing more information than a single ver-

tex embedding. However, it seems that these graphs can be well-characterized by focusing on the vertex of the author of the *targeted message*. This could mean that the relative position of this vertex in the graph is enough to characterize the whole conversation and that vertex-level embeddings are able to capture such information.

6.2.2.2.2 Edge Attributes

To assess the impact of edge attributes on the learned representations, we compare the performances of SG2V (Table 6.5) and SGCN (Table 6.6) on all 8 possible combinations of edge weights, directions, and signs.

Table 6.5: Comparison of the F-measure obtained when considering the 8 different types of graphs with SG2V.

Graph type		**Unsigned**	**Signed**
Unweighted	Undirected	79.13	79.21
	Directed	79.15	79.24
Weighted	Undirected	79.21	79.29
	Directed	79.23	**79.31**

Table 6.6 shows the results for SG2V. We can observe that each attribute brings a bit of information, leading to a better representation. The best performance is obtained when considering edge weights, directions, and signs.

Table 6.6: Comparison of the F-measures obtained when considering the 8 different types of graphs with SGCN.

Graph type		**Unsigned**	**Signed**
Unweighted	Undirected	74.74	74.85
	Directed	74.76	74.88
Weighted	Undirected	74.86	74.92
	Directed	74.91	**75.02**

Table 6.6 shows the results for SGCN. The observation is similar to SG2V. All the attributes slightly improve the representation, and the best performance is obtained when considering all of them. These results confirm what we already showed in Chapter 4, that edge properties contain information that improves the quality of the representations.

6.2.2.2.3 Vertex Attributes

Regarding the Graph2vec variants that we propose, 2 of them obtain performances almost equivalent to the one of the original model. Identifying the author (Graph2vec-author) or the targeted message itself (Graph2vec-target) allows to create classes of vertices rather than treating vertices as individual components. However, this does not bring additional information useful for the detection of abusive messages. We suppose that the Graph2vec-target approach, which only distinguishes 2 classes of vertices, does not bring information discriminant enough to improve the detection performance. On the other hand, Graph2vec-author creates a distinct label for each author. Graph2vec is unable to benefit from this, probably because it is difficult to detect and capture structural equivalence in graphs with so many different vertex labels. Graph2vec-distance, the method that uses the distance between each vertex and the targeted one, obtains better results that make it the best-performing graph embedding method with a F-measure of 80.47. This distance is closely related to our task since it can categorize users based on their proximity to the targeted author (e.g. close friend, acquaintance, stranger). This proximity can be a key contextual element to detect an abuse. This shows that adding a task-specific information during the learning of the embeddings can improve their quality.

6.2.2.2.4 General Observations

Unlike what we observed on the textual embeddings, no graph embedding method is able to outperform the baseline. This is not surprising, because this set of features was specifically designed for this task and dataset and includes a large number of various measures, whereas the embedding methods are somewhat generic. The text baseline also includes much less features than the graph baseline, which probably made it easier to beat.

GraphWave (vertex scale approach), Graph2vec, and its signed variant SG2V (whole-graph scale) yield the best performances among graph embedding approaches. The other methods, obtain correct performances with an F-measure approximately ranging from 71% to 78%. Spectral Features and FGSD, which operate on the whole graph, might be penalized by the small size of our dataset and by the fact that graphs have approximately the same size and thus, possibly similar structures. DeepWalk is less efficient than Node2vec, which is in line with other studies [85], [170]. The Walklets algorithm learns multi-scale relationships in the graph. However, such relationships might not be very developed in our graphs, which could explain its lower performance. This observation could also be the reason for the very poor performance of BoostNE, which also operates on different granularity levels. Signed Graph2vec (SG2V) is one point above Graph2vec, which confirms once again that signs are important for the representation.

In the end, the best graph embedding method (Graph2vec-distance) is less efficient than our baseline by approximately 3 points. However, embedding-based approaches still have 2 major

advantages compared to the feature-based baseline. First, they are not specifically designed for this task or dataset and are hence more likely to be efficient in other settings. For instance, the textual embedding models that we use are not specifically pre-trained on abusive datasets, which illustrates this ability to be applied in various contexts. Second, embedding methods are more scalable than hand-crafted sets of features. Computing the topological measures used in our baseline is computationally very expensive, with a total runtime of almost 4 hours. On the other hand, it only takes a few minutes to deal with the embedding methods on the same machine, which makes them a lot more time-efficient.

6.2.3 Fusion of Embeddings

Past work in the literature [14] and the study conducted in Chapter 4.2.3 show that combining multiple types of information can be effective in the context of abusive message detection. In this chapter, we propose to combine some of the lexical and graph embedding methods that we used in the previous experiments. Our intuition is that these methods are based on completely different modalities, namely the text and the conversational graphs, and that their embeddings must therefore capture different information, possibly complementary. In this case, their combination should improve the classification performance. Furthermore, we also consider the combination of methods that operate on the same modality. As mentioned by Le *et al.* [213], the combination of multiple methods can lead to improved performance.

We use the 3 fusion strategies that we previously proposed in Chapter 4.2.3 to combine content- and graph-based features.

In this chapter, we focus on the embedding methods which obtained the best performances in our experiments. For the textual embeddings, we select Camembert-W, FlauBERT-W (word level), and CamemBERT-M (message level). For the graphs, we use GraphWave (vertex-scale approach) and Graph2vec-distance (whole-graph scale approach). We also consider the baselines for both text and graph-based methods. Table 6.7 shows the F-measures obtained when applying the 3 fusion strategies to the best-performing methods.

Our first observation is that all 3 fusion strategies yield close fairly similar results. Generally speaking, the hybrid fusion achieves top performance, ahead of the early fusion and the late fusion. This is an expected result since the hybrid fusion is a combination of the other 2. All the information available in the early fusion process is summarized in just 2 scores in the late fusion. One could think that this is not enough to preserve all the information, but despite this process, the late fusion performs only slightly worse than the early fusion.

As we previously showed in Chapter 4, the hybrid fusion of the 2 baselines improves the performance by more than 2 points compared to the graph baseline used alone. The fusion of the 2 CamemBERT approaches (CamemBERT-W and CamemBERT-M) does not bring a significant performance boost. Therefore, we can suppose that they both capture similar information. The same

		Early		Late		Hybrid	
Methods	**Base Perf.**	**Dim.**	**F-m.**	**Dim.**	**F-m.**	**Dim.**	**F-m.**
Graph Base. + Text Base.	83.40	506	84.51	2	84.17	508	84.97
CamemBERT-W + CamemBERT-M	81.02	2,048	81.32	2	80.21	2,050	81.17
CamemBERT-W + FlauBERT-W	81.02	2,048	79.85	2	80.10	2,050	80.34
Graph2vec-dist. + GraphWave	80.47	256	82.45	2	82.07	258	81.19
Graph2vec-dist. + CamemBERT-W	81.02	1,152	85.66	2	85.10	1,154	86.07
GraphWave + CamemBERT-W	81.02	1,152	85.99	2	85.80	1,154	86.11
Graph Base. + CamemBERT-W	83.40	1,501	86.77	2	86.21	1,503	**87.06**
Graph2vec-dist + Text Base	80.47	157	82.24	2	82.21	159	82.36

Table 6.7: F-measures obtained by the fusion of the best performing embedding methods following the 3 fusion strategies. The second column indicates the performance of the best method among the 2 combined ones. The table is split into 4 parts depending on the nature of the methods combined. The first part is for the baselines, the second for the fusion of lexical embeddings, the third for the fusion between graph embeddings, and the fourth for the fusion of text and graphs.

applies to the fusion of CamemBERT and FlauBERT, which stays on par with the performance of CamemBERT used on its own. These lexical embeddings thus capture redundant information, which can be explained by the fact that they are both based on the same RoBERTa architecture.

For the graph-based methods, the early fusion of GraphWave and Graph2vec-distance (82.45%) reaches a performance only one point lower than the graph baseline (83.40%). On the one hand, this result acknowledges the assumption that graph embedding methods that operate on different granularity levels (vertices for GraphWave, whole-graph for Graph2vec-distance), can capture complementary information. On the other hand, the late and hybrid fusions perform worse. Thus, we hypothesize that all the complementary information captured by these 2 methods cannot be summarized in only 2 scores.

Finally, the combination of the best text-based method (CamemBERT-W) with each approach based on graphs always achieves better performance than the combination of the 2 baseline methods (84.97%). The hybrid fusions of CamemBERT-W with Graph2vec-distance and GraphWave obtain a F-measure of 86.07% and 86.11% respectively. The overall best performance is achieved by combining CamemBERT-W with the graph baseline (87.06%). Interestingly, this combination is only one point above both previous combinations, while when used on their own, Graph2vec-distance and GraphWave are 5 points behind the baseline. The best combination method is 7 points above the best embedding method used alone.

6.2.4 Results Summary

To summarize our results in this chapter, Table 6.8 shows the best performance we obtain for each type of approach studied. For lexical embeddings, the best method is CamemBERT applied to the words (CamemBERT-W) and averaged over the set of words composing a message. For graphs, this is the whole-graph embedding method Graph2vec in its variant using the distance as a vertex label (Graph2vec-distance). It is worth noting that Graph2vec is still less effective than our graph-based baseline. For fusion between text-based methods, this is the hybrid combination of CamemBERT methods applied on word and message level that outputs the best F-measure. The improvement is, however, very small compared to the methods used on their own. The early fusion between GraphWave (vertex-level) and Graph2vec (whole-graph level) is the best combination of graph embedding methods. Finally, the overall best performance is achieved by the fusion of text and graphs. The hybrid combination of CamemBERT with the graph-based baseline achieves a 87.06% F-measure, with a 86.21% precision and 87.93% recall. This result shows that, for the task of abuse detection, different modalities contain different types of information and that combining them brings real benefits compared to using them separately.

Type	**Method(s)**	**Dim.**	**F-m.**
Baseline	Baseline-Graph	477	83.40
	Baseline-Text	29	75.21
Lexical embedding	CamemBERT-W	1,024	81.02
Graph embedding	Graph2vec-distance	128	80.47
Fusion text-text	CamemBERT-W + CamemBERT-M	2,048	81.32
Fusion graph-graph	Graph2vec-dist + GraphWave	256	82.45
Fusion graph-text	Baseline-Graph + CamemBERT-W	1,501	**87.06**

Table 6.8: Summary of the best-performing method in terms of F-measure for each type of approach studied in this chapter.

6.3 Feature Study

In Chapter 4, we identified, among the measures used in the baselines, the most discriminative features for our classification task, which we called *Best Features*. We were able to define subsets of 4 features for the text-based approach and of 10 features for the graph-based method which were sufficient to reach 97% of the original performance when considering the complete feature set. Since embedding methods provide vector representations in which dimensions are not directly interpretable, it does not make sense to apply the same method here. Instead, we propose to study

whether the best features from our baselines are well captured by the embedding methods used in this chapter. We perform this analysis for both lexical and graph embedding approaches.

To this end, We compare the F-measure score obtained by each embedding method on its own, with the score obtained by using a representation composed of the same embedding completed by one of the best features. If the performance significantly increases, we conclude that the topological measure was not captured by the embedding. These cases are represented in red in Figures 6.4 and 6.5. If the performance stays the same or increases by less than 0.50 point with the additional feature, we conclude that the structural property corresponding to this topological measure is well captured by the embedding (represented in green). If the performance increase is higher than 0.50 point but not statistically significant, we conclude that the property is only partially captured by the embedding method (represented in orange). If the performance increase is higher than 0.50 point and statistically significant, we conclude that the property is not captured by the embedding method (represented in red). We first study text-related features (Section 6.3.1), then graph-related measures (Section 6.3.2).

6.3.1 Text Features

Results for text best-features addition are shown in Figure 6.4. The ratio of capital letters in the message seems to be well captured by all the embedding methods. The same observation applies to the TF-IDF score computed over the *Abuse* class except for Word2vec, which only partially captures this measure. In contrast, no method completely captures the information conveyed by the *Naive Bayes* score. Word2vec is even completely unable to capture it. This feature comes from a fully-fledged classifier and is, by far, the most important one from the content-based baseline. Therefore, we can suppose that this information is too complex to be modeled by our embedding methods. Furthermore, it integrates several types of information which might be difficult for an embedding to capture. However, almost all these methods yield a better F-measure than the baseline. Therefore, we can suppose that these methods might be able to capture other properties of the message that are not represented by any text-related feature, but improve the overall performance when all combined through the embedding. Another interesting observation of this study is that we can understand why Word2vec performs worse than the baseline and is the least efficient embedding method, as it fails to capture the 2 most important features of that baseline.

6.3.2 Graph Features

An interesting result shown by Figure 6.5 is that some topological measures seem to be well captured by almost all the embedding methods (e.g. Authority score at graph level, PageRank centrality, Degree centrality, Vertex count, and Reciprocity). Contrariwise, the Coreness score at graph level on the Full and Before graphs are partially captured by GraphWave and Spectral

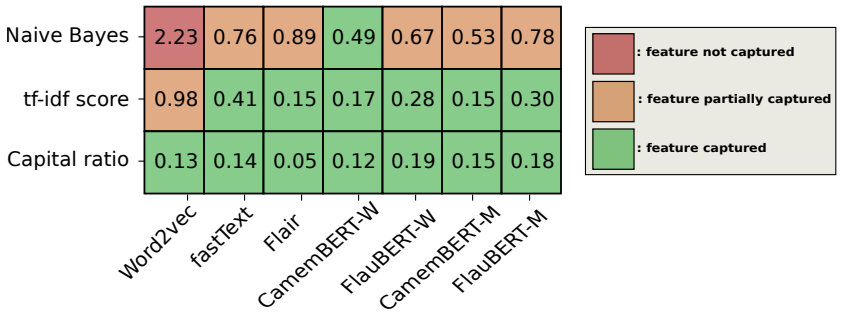

Figure 6.4: Text measures captured (green), partially captured (orange), or not captured (red) by the word embedding approaches. Each value is the difference between the F-measure score obtained by the embedding method on its own and the score obtained by the embedding method completed by the corresponding Best Feature.

Features, and not captured at all by all other methods. The Closeness Centrality at the graph level and the vertex level on the Before and After graphs are well captured by some methods and partially captured by others. BoostNE fails to capture the authority at the graph level, which might explain why this method obtains the worst results among vertex embeddings. SG2V and Graph2vec, 2 whole-graph embedding methods, are the only ones to correctly capture all the features except for the 2 mentioned above. There is no surprise that they are 2 of the best-performing methods, together with GraphWave. The latter, considered as a vertex embedding method, is the only approach to capture, at least partially, all the best features. However, these 3 methods obtain lower F-measure scores than the baseline. We assume that they might not capture other properties of the graph which are less important but improve the performance when all combined. A possibility to combine text and graphs is to directly learn representations from these 2 aspects. We developed an experimental work in this direction (see Appendix A) based on the architecture of the textual embedding Flair [216]. It seeks to learn combined representations of text and graphs to improve the quality of the representations.

An interesting result of this study is that there is no clear difference in the type of information captured by vertex and whole-graph embedding approaches. Vertex embedding methods can capture certain graph-scale topological measures, and whole-graph embedding methods can capture some vertex-scale measures. This property may result from the relatively small size of our graphs, as the second-order neighborhood of a vertex might include the majority of the graph. Thus, differences between vertex and whole-graph embedding methods are not as important as they could be on larger graphs. Furthermore, our graphs are built around a specific vertex. This specificity might help the whole graph embeddings to capture better vertex-level information.

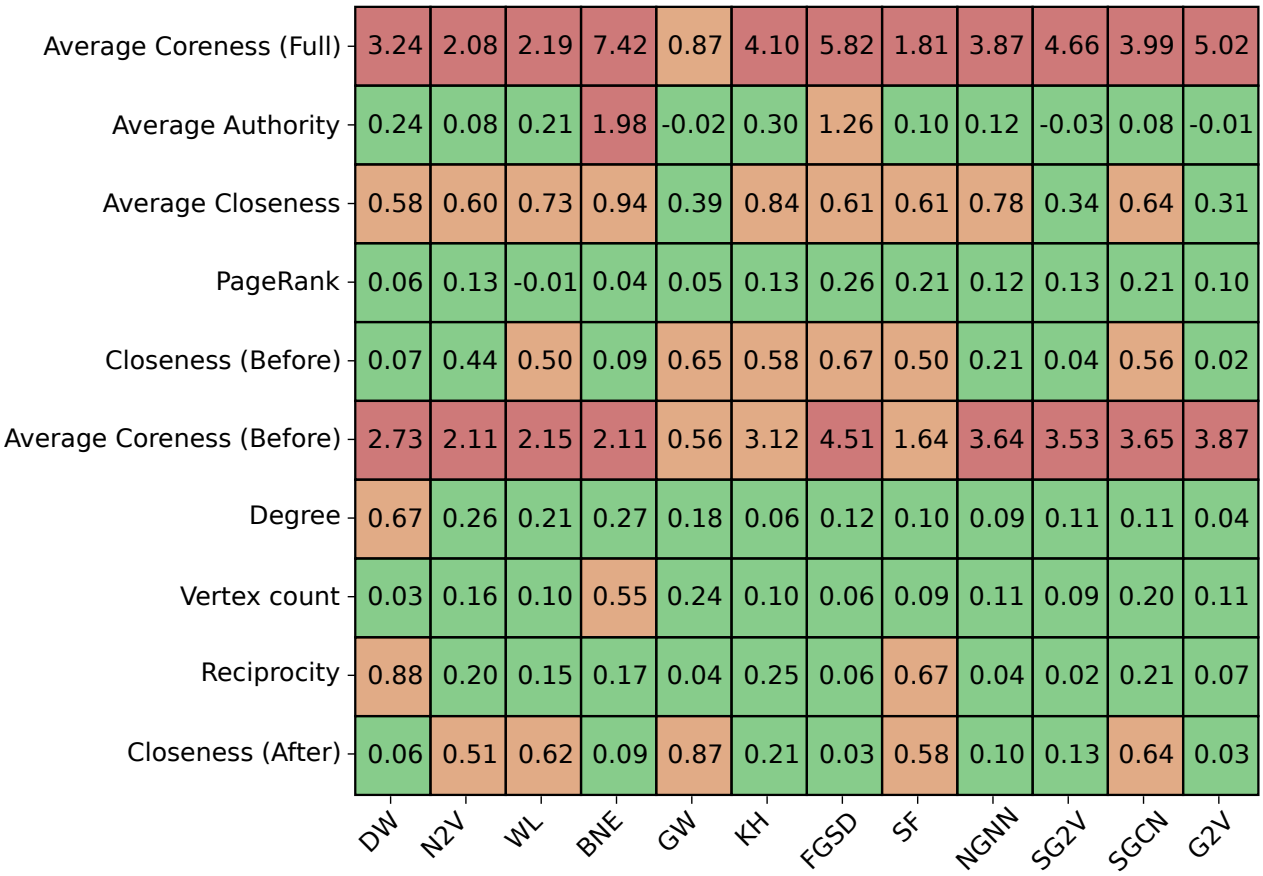

Figure 6.5: Topological measures captured (green), partially captured (orange), or not captured (red) by the embedding approaches. The first 4 topological measures are computed at the *Graph* level and the last 5 topological measures are computed at the *Vertex* level. Each value is the difference between the F-measure score obtained by the embedding method on its own and the score obtained by the embedding method completed by the corresponding Best Feature.

6.4 Conclusion

In this chapter, we have experimented with embedding methods to automate the representation learning process as part of the automatic abuse detection task. Our experiments cover the textual and contextual aspects of that task. We first compared 5 lexical embedding methods to learn representations of messages in French. The contextualized embeddings were much more efficient than the fixed ones. The best approach, CamemBERT, achieved up to 81.02% F-measure. This is a massive improvement over our baseline based on a manually crafted set of features. We then compared 11 graph embedding methods, half of them generating embeddings at the vertex scale, and the other half treating the whole graph. We proposed variants of the SG2V and SGCN models able to treat weighted, directed graphs. Both variants obtain better results than the original model

showing that edge attributes convey important information for this task. In addition, we explored a functionality offered by Graph2vec, one of these methods. We consider 3 vertex labeling strategies allowing us to integrate various information related to our task directly into the learning process. The variant using the distance between a vertex and the targeted vertex as a label was the best graph embedding method with a F-measure of 80.47%. Finally, we experimented with the 3 fusion strategies from Chapter 4.2.3 to combine different types of embeddings. We could show that the combination of textual and graph embeddings takes advantage of both sources of information and significantly improves the detection of abusive messages, as already observed with the features in Chapter 4.

We think that these results are very promising in multiple aspects. The textual embeddings are much more efficient than the feature engineering approaches on the abuse detection task. The embeddings are much more robust and thus, less likely to be negatively impacted by spelling mistakes and other special language features that are common in online conversations. Graph embeddings, though less efficient than the feature-engineered baseline, still demonstrate great results. This is particularly interesting since these methods are completely task-independent, much more scalable, and time-efficient.

We believe that our work opens new directions for future research. In this work, we combine lexical and graph embeddings which are learned separately. An idea could be to combine these aspects directly during the representation learning process to obtain a single embedding including all these types of information. As the messages are posted in a specific order, we could also think of an approach including a temporal aspect, for instance with a dynamic conversational graph reflecting the evolution of conversation over time.

Chapter 7

Conclusion and Perspectives

7.1 Conclusion

In this thesis, we tackled the problem of combining 2 modalities, text and graphs, representing conversation content and structure, for the analysis of text-based documents. We applied it to the detection of abusive messages in online conversations. We observed that the literature was largely focused on text-based approaches because this seems to be the most straightforward way of detecting an abuse case. While effective for flagrant instances of abuse, this approach shows its limitations when it comes to detecting more intricate occurrences such as innuendos and allusions. The conversation in which messages are posted represents important contextual information that can greatly improve the detection in such situations. As is the case in many other domains, this multimodal approach combining the text and the graphs provides even better detection performances. As they each focus on different aspects of the conversation, they are able to capture different but complementary information.

To compare the content and structural aspects, we divided our studies into 2 parts. The first focuses on the textual content of the message itself, whereas the second deals with conversational graphs built to represent the conversation in which the message was posted. The key part of an abuse detection system is the way it represents messages and conversations. We first used a methodology based on feature engineering. We used a set of measures that characterize the messages and the graphs. For the content-based method, we used a mix of NLP features including morphological and topological ones. For the graph-based method, we used a wide range of topological measures defined for different scales and scopes. The representations of the messages are thus directly related to the nature of the information considered by these 2 methods. By applying them to a dataset of real-world conversations extracted from a video game, and feeding the representations to a classifier, we showed that the graph-based method obtained better results than those of the content-based method. This is a particularly important result since this aspect of the literature is largely overlooked. We then proposed a method seeking to take advantage of both the text and the graphs by combining them. Through several variants, we could confirm that this combination is beneficial to the detection of abusive messages. We were also able to show that edge attributes (i.e. weight, direction, sign) bring more discriminative power to the graph-based method. Furthermore, while it was originally designed to do the classification afterward, this

method could be adapted to a live setting as the performance is still convincing when considering only the messages posted prior to the targeted message. The main limitation of this feature-based framework is its important computational cost due to the large number of features, and the algorithmic complexity to compute some of them. This motivated us to determine subsets of the most important features for each method. They drastically reduce the processing time of the methods while preserving a high performance.

Another solution to obtain representations of messages and graphs is to automatically discover them through representation learning. More precisely, embedding techniques present several advantages over feature engineering, including being more flexible and time-efficient. As our graph-based approach for abuse detection roughly corresponds to a graph classification task, we proposed to use whole-graph embedding, the specific type of embedding methods dedicated to learning representations of the whole graph at once. Motivated by the fact that edge directions and weights improve the classification, we proposed to consider a third sort of edge attribute in the form of signs. As there is no signed whole-graph embedding method in the literature, we developed our own 2 methods. Signed Graph2vec (SG2V), a generalization of the unsigned whole-graph embedding method Graph2vec to *signed* whole graphs, and a generalization of the signed vertex embedding method SGCN to signed *whole* graphs. There is no benchmark to evaluate such methods, so we compiled our own benchmark composed of 3 datasets of annotated signed graphs. It contains real vote networks, conversation networks, and artificially generated graphs. We empirically showed that our 2 proposed methods obtain better classification results than standard unsigned whole-graph and signed vertex-oriented methods.

To complete our comparison between text and graphs for the analysis of text-based documents, we extended our previous framework to replace feature-based methods with automatic embedding approaches. We compared several approaches from the literature. For the aspect focused on the content, we selected methods operating on different granularities of text. For the graph-based system, the selected methods include our 2 signed ones, also operating on different scales. We showed that, for the content, textual embeddings were much more effective than the feature-based system. This observation did not stand for the graphs, as the graph embeddings are slightly less effective. However, they still have numerous advantages in terms of computational cost and adaptability to different environments. We concluded the comparison by combining the text and graph aspects and have once again found that they are complementary, as their joint use improved the classification performance on top of the gain already brought by text representation learning. Embedding methods provide vector representations in which dimensions are not directly interpretable. To overcome this issue and provide a way to interpret our results, we proposed a framework to detect which properties are captured by the embeddings. It consists of adding a dimension to the representation vector containing a single one of the previously selected features. We then compared the performance obtained with and without this additional dimension. If the performance increased

as the feature is used, we concluded that the property associated with it was not properly captured by the considered embedding method. Otherwise, the information brought by the feature is redundant and does not impact the performance. This experiment showed that all the textual embedding methods were able to capture or partially capture the important text-based features, which is in line with the fact that textual embeddings are more effective than the feature-based system. On the other hand, most of the graph embedding methods are unable to capture variants of an important feature (i.e. the coreness centrality) which might be the reason why they are less effective than the feature-based system.

7.2 Perspectives

The different studies conducted in this thesis suggest diverse opportunities that remain open to extend this work. We already discussed some of them in general terms at the end of each chapter. In the following, we discuss them more precisely.

Task reformulation: In the literature, abuse detection is always formulated as a classification problem. Be it binary or multi-class, such a framework usually suffers from an imbalanced distribution of classes. In real-world online discussions, the prevalence of abusive messages is very low, inferior to 1% in general [90], [91]. That is a reason why most authors oversample abusive occurrences when creating datasets, by using comments from specific communities and topics with a higher probability of abuse. Hence, one could argue that these datasets are not representative of reality. As abuse occurrences are relatively rare, one could consider them as anomalies that deviate from the majority of the data. For this reason, an interesting research line would be to reformulate the abuse detection problem as an anomaly detection task, instead of a classification task, where the goal is to identify abusive comments (anomalies) that appear inconsistent with the rest of the conversation. By construction, anomaly detection methods are suitable for highly imbalanced datasets.

Conversation datasets: The abuse detection literature mainly focuses on content and overlooks structure, even though it is highly efficient, as we showed. One reason for this trend is the lack of experimental datasets providing suitable data. Indeed, annotated datasets of conversations are rare, and for the few that exist, the conversations are often too short to exploit the full potential of structure-based approaches. We constituted a conversation dataset [18] to mitigate this issue (see Appendix B.2), but it is also limited to rather short conversations. An interesting work would be to extend this effort to create a large-scale dataset of lengthy conversations annotated for abuse. This would open opportunities for future research based on the structure of conversations. An idea could be to use recent text generation models to create large conversations. However, such models are usually tweaked to avoid generating abusive comments. Another interesting point could be to work on a multilingual corpus, to assess the possibilities of transfer learning from one language

to another.

Temporal aspect: When working on the structure, we used static graphs to represent conversations. However, conversations evolve through time. An interesting research line would be to directly integrate the temporal aspect of conversations, either by using representation techniques able to simultaneously embed structural and temporal information [221] or by extracting a dynamic graph (i.e., a sequence of graphs to represent the evolution of conversations over time). We started to experiment with different types of context around messages based on temporality: the context posted before, after, and the combination of both. However, this is still ongoing research, and we believe that integrating the temporal aspect would improve the classification performance.

Combining text and graphs: Finally, in this thesis, we showed that it is efficient to combine the text and graphs to detect abuse. However, we performed this combination only after the representation step, i.e. we combined the text and graph representations learned by our different systems. A possible improvement could be to simultaneously learn from both in order to obtain a global representation of a message including text and graph information. We experimented with this idea by extending the Flair [216] framework, a textual embedding framework, to include text and graphs. We briefly present this preliminary work in Appendix A. However, the preliminary results and difficulties with this approach prevented us from developing it further, and it remains an open work.

.

Appendices

Appendix A

Text and Graph Joint Embedding

This appendix presents a preliminary work of an embedding method designed to jointly learn from text and graphs. It is based on the Flair embedding architecture [216], a textual embedding model. The preliminary results with this approach were negative, but the methodology can still be interesting so we present it. We first explain this original architecture (Section A.1), then we explain how we propose to transform this model to include both graphs and text (Section A.2).

A.1 Flair Embedding

Flair [216] is a textual embedding model that learns contextualized representations at the character level. An input sentence is treated as a character sequence and input into a bi-directional LSTM operating on characters. Working at the character level has two major advantages over word-level models: first, it is particularly efficient when dealing with rare or misspelled words and second, it is able to better model language-related concepts such as prefixes and suffixes. The model is trained to predict the next character for each element in the sequence to process. As it uses a bi-directional LSTM, the Flair model learns 2 hidden representations for each character of the sequence: one for the forward network and one for the backward network. The contextual embedding of a word in a sentence is obtained by concatenating the forward representations of the characters located before the end of the word and the backward representations of the characters located after the beginning of the word. Figure A.1 shows an example of this process.

A.2 Proposed Architecture

Our objective is to develop a model able to jointly learn from graphs and text. To this end, we propose a two-stepped architecture that we call Flair^2. The first step is similar to the standard Flair model, and is used to learn representations of words based on their character-level decomposition. The outputs of this first step are thus word-level representations. We introduce the graph representations in the second step of our proposed model. These representations can be learned from any graph embedding model presented in Section 6.1.2. The second step of Flair^2 uses a similar architecture as the first one, but its inputs are the word-level representations from the first step and the representation of the graph representing the corresponding conversation. Our objective with

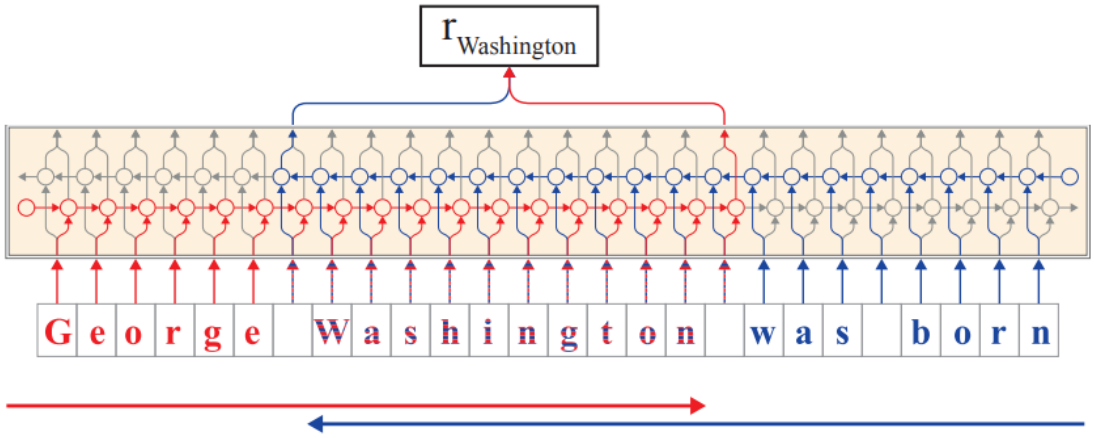

Figure A.1: Extraction of the embedding for word "Washington" in the sentence "George Washington was born". The forward model is shown in red and the backward model in blue. Figure from [216].

this second part of our model is two-fold. First, as a message is a sequence of words, in Chapter 6 we proposed to average the representations of all the words in a sentence to obtain its representation. Here, we think that this step could automate this aggregation process and directly learn a better manner to combine the words' representations to obtain a representation of a message. Second, as we input both text and graph representations to the model, we think that the resulting representations could integrate information from both modality.

Combining graph and text right from the learning phase, instead of treating them separately and combining their learned representations could help better capture the complementary information of both modality.

Appendix B

Datasets

As mentioned in Chapter 2, we use the SpaceOrigin dataset throughout this thesis. We tried to experiment with some others, but we always faced limitations related to the specific needs for our task. This includes the fact that our methods require full conversations with a certain diversity in the messages and authors. In this appendix, we detail the two datasets that we tried to experiment with. First, we present Ruddit (Section B.1), a dataset of conversations from the platform Reddit[1], which is annotated for abuse, and explain its limitations relatively to our task. Then, we present Wikipedia Abusive Conversations (WAC) (Section B.2), a corpus that we constituted to overcome the limitations of the other existing datasets.

B.1 Ruddit

Ruddit [102] is a dataset of 6,000 English language Reddit comments annotated for offensiveness. Reddit is a social news aggregation and discussion website where users can can create a *post* to share various types of content such as links, text posts, images, and videos. Users can also *comment* an existing post and reply to a comment to start a discussion. A *post* can thus be viewed as the root of a discussion (called *thread*) divided into multiple sub-discussions.

Reddit is divided into forums called *subreddits*, each dedicated to specific topics. To constitute the Ruddit dataset, Hada *et al.* sampled 808 posts from particular subreddits. This selection was made to increase the proportion of offensive comments. 50% of the selected posts are from a selection of subreddits covering diverse generic themes, 25% are from *ChangeMyView*, a subreddit with posts on controversial topics and the final 25% are from random subreddits. The authors extracted the first and last 25 comments from these 808 posts and sampled 6,000 comments from this set to constitute the Ruddit dataset. All these comments have then been annotated with a real-valued score between -1 (maximally supportive) and 1 (maximally offensive). The annotation was done using crowdsourcing.

The Ruddit dataset thus has quality annotations for offensiveness and provides comments that are not completely independent from each others as some of them were posted in the same *thread*. However, a number of limitations prevented us from using this dataset in our experiments. We

[1] https://www.reddit.com/

describe them in the following section.

B.1.1 Limitations of Ruddit

Conversation structure: The first limitation of Ruddit is inherent to Reddit. Indeed, each *comment* on a *post* is considered as the strat of a new conversation, and each conversation can then be divided into multiple sub-conversations. A general observation is that posts with a lot of comments, often tend to generate many sub-conversations, with only very few different users interacting in each of them. This phenomenon results in conversations that are relatively short and with only a limited number of participants, which is the opposite of what our proposed methods require. Furthermore, we could observe that comments posted under the same *post* but in different sub-conversations are often completely unrelated. This is due to the fact that while the original post is the same, conversations tend to quickly deviate into other subjects.

Missing Data: Another limitation of this dataset is that it does not provide full conversations. Indeed, Ruddit contains multiple annotated comments from the same conversation but there are missing comments between them. As the authors provide the *id* of the reddit posts, our idea was to retrieve all the missing comments to obtain full conversations. However, this showed poor results as a lot of the original comments were missing either because they were deleted by their author or because they were removed by moderators. The same problem applied with authors whose account had been deleted, making it impossible for us to correctly associate comments with their author.

The aforementioned limitations prevented us from using Ruddit in our experiments and inspired us to constitute our own dataset of annotated conversations. We discuss this dataset in the next section.

B.2 Wikipedia Abusive Conversations

The Wikipedia Abusive Conversations (WAC) [18] is a large dataset of Wikipedia comments that we constituted. The particularity of this dataset is that each annotated comment is provided along with the conversation in which it was posted. Our objective with this work was to encourage further development of context- and thread-based methods in the area of abusive content detection by providing the first large scale corpus of contextualized abusive messages. WAC contains roughly 193,000 conversations and 383,000 messages annotated as being abusive or not. In Section B.2.1, we give details on the dataset and how we contructed it. In Section B.2.2, we explains the limitations of this dataset that prevented us from using it in our experiments.

B.2.1 Proposed Corpus

The conversations in WAC come from Wikipedia talk pages, i.e. the web pages associated to Wikipedia articles where editors can exchange. Typically, editors write a message explaining which change they made on the article and why. Every Wikipedia article and user has a related talk page. To constitute WAC, we aggreagted data from 2 existing datasets.

B.2.1.1 Wikipedia Comment Corpus

The Wikipedia Comment Corpus (WCC) [90] is a corpus of discussion comments from English Wikipedia talk pages which are extracted using the revision history of each considered talk page. In January 2016, a dump of more than 63M Wikipedia talk page comments was made public. From this dump, Wulczyn *et al.* sampled 3 smaller datasets that were annotated by humans for different type of abuse:

- *personal attack*: abusive content directed at somebodys person rather than providing evidence;
- *aggression*: malicious remark to a person or group on characteristics such as religion, nationality or gender;
- *toxicity*: comment that can make other people want to leave the conversation.

The *Personal attack* and *Aggression* datasets contains exactly the same 115k comments while the Toxicity dataset contains more comments (159k). Among them, 77k appear in all 3 datasets. Table B.1 shows the prevalence of abusive comments in each dataset.

Dataset	**Comments**	**Percentage abusive**
Personal attack	115,864	13.4 %
Aggression	115,864	14.7 %
Toxicity	159,686	11.5 %

Table B.1: Main properties of the three datasets constituting the Wikipedia Comment Corpus (WCC).

This dataset thus provides high quality annotations for 383k messages. However, all these messages are unrelated and we cannot use conversation-based techniques directly on them. To obtain this conversation structure, we use a second dataset.

B.2.1.2 WikiConv

WikiConv [104] is a large public corpus also based on Wikipedia talk pages. This corpus contains full conversations, and not only isolated comments like WCC. It was created by analyzing the history of Wikipedia talk pages. However, Wikipedia does not provide a standard post system for these pages such as those commonly used in online forums. Instead, the talk page is similar to a regular Wikipedia article page, or a wiki page in general: in theory, users have the ability to edit it by adding, modifying or removing text anywhere. However, in practice, a set of writing and formatting conventions[2] allow giving structure to the various conversations taking place on the talk page. For instance, when a user adds his own post, he indents it so as to indicate its hierarchical level in the conversation tree. Figure B.1 shows an example of Wikipedia conversation under the form of the rendered talk page and the corresponding Wikicode (Wikipedia markup language).

Displayed page	**Raw Text**
Edgar don't do massive revert You are deleting sections that were added to make this article more comprehensive. Moe 03:15, 10 November 2019 (UTC) I am returning to previous version that was more stable longer and was a Featured Article Edgar 04:23, 10 November 2019 (UTC) Ask rest of the editors, if that's what they want first. Ask first and then get a consensus period. Moe 04:26, 10 November 2019 (UTC) When you're all done, please fix the rugby disambiguation problem that I fixed in the middle of this edit war. Rev 14:29, 10 November 2019 (UTC)	== Edgar don't do massive revert == You are deleting sections that were added to make this article more comprehensive. [[User:Moe|Moe]] 03:15, 10 November 2019 (UTC) : I am returning to previous version that was more stable longer and was a Featured Article [[User:Edgar|Edgar]] 04:23, 10 November 2019 (UTC) ::Ask rest of the editors, if that's what they want first. Ask first and then get a consensus period. [[User:Moe|Moe]] 04:26, 10 November 2019 (UTC) :::When you're all done, please fix the rugby disambiguation problem that I fixed in the middle of this edit war. [[User:Rev|Rev]] 14:29, 10 November 2019 (UTC)

Figure B.1: Part of the *Japan* Wikipedia article talk page: rendered page (left) and corresponding Wikicode (right).

Therefore, Hua *et al.* defined heuristics to retrieve all the messages in a talk page. Each message is identified by an *id*. WikiConv thus provides large conversations but without high quality annotations. On objective in the next section is to combine the annotations from WCC with the conversations of WikiConv in order to obtain annotated messages along with the conversation in which it was posted.

B.2.1.3 Combining WCC and WikiConv

Comments in WCC and WikiConv are identified by an *id* that, in most cases, can be used to link messages from both datasets. To create our WAC dataset, we retrieve the WikiConv's conversation associated with each comment in WCC. After this retrieval, all the annotated message are associated with their corresponding conversation. Note that multiple annotated messages can

[2] https://en.wikipedia.org/wiki/Help:Talk_pages#Replying_to_an_existing_thread

occur in the same conversation. Because of missing data, some messages from the original WCC dataset can not be retrieved in WikiConv and are discarded. In total, the WAC corpus contains more than 2.2M unique messages split into 168,827 unique conversations. Among the 2.2M, 382,665 messages are annotated and the rest are messages that only appear in the conversation around an annotated one. The number of annotated messages and the division of annotations in all three datasets is summarized in Table B.2.

Dataset	**Annotated messages**	**Abuse**	**Non-abuse**
Personal attack	113,174	14,934 (13.20%)	98,240 (86.80%)
Aggression	113,174	16,331 (14.43%)	96,843 (85.57%)
Toxicity	156,317	19,700 (12.60%)	136,617 (87.40%)
Total	382,665	50,965 (13.31%)	331,700 (86.69%)

Table B.2: Number of annotated messages and distribution of annotations in the proposed *Wikipedia Abusive Conversations* (WAC) corpus.

The Wikipedia Abusive Conversations is thus a large scale dataset of messages annotated for 3 types of abuse and provided along with the conversation in which they were posted. However, we were unable to benefit from this dataset in our experiments because of the limitations detailed in the following section.

B.2.2 Limitations of WAC

Conversation Structure: As for Ruddit (Section B.1), WAC conversations are not structured as a linear sequence of messages. A conversation is usually composed of multiple sub-conversations with fery few different users interacting in each of them. In the context of this thesis, his is an important limitation as our proposed methods benefit from having a diversity of users in the conversations.

Length of conversations: The conversations in WAC are also relatively short, with a majority of them constituted of fewer than 20 messages. Figure B.2 shows the distribution of the conversations length in the dataset. Note that the *y*-axis scale is logarithmic for readability reasons. We can observe that some conversations contain more than 1,000 messages but for a large majority, conversations are only 1- to 20-message long.

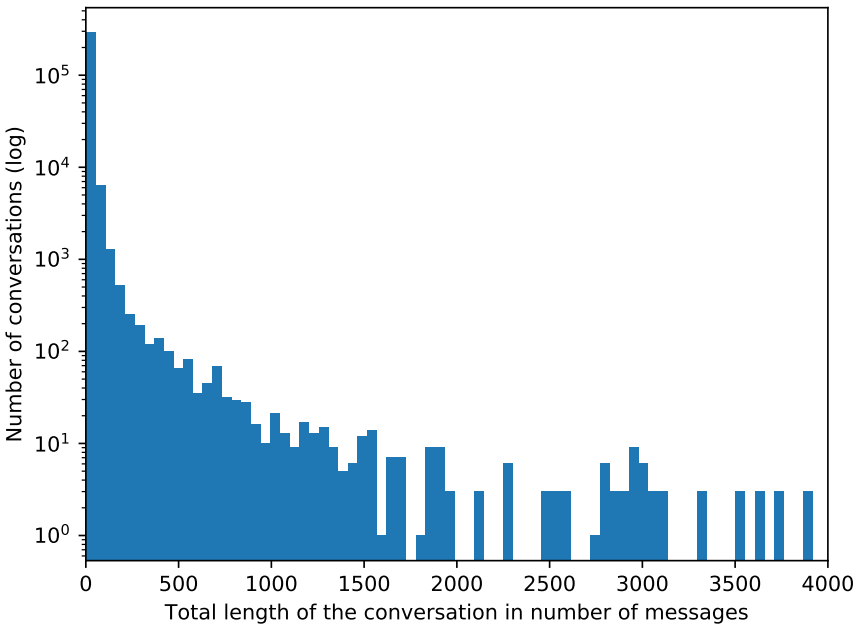

Figure B.2: Distribution of the conversation lengths in *Wikipedia abusive Conversations*, expressed in number of messages. The *y*-axis scale is logarithmic.

For these reasons and after preliminary experiments that were inconclusive, we decided to not use the WAC dataset for our experiments.

REFERENCES

[1] M. O'Reilly, N. Dogra, N. Whiteman, J. Hughes, S. Eruyar, and P. Reilly, "Is social media bad for mental health and wellbeing? exploring the perspectives of adolescents," *Clinical Child Psychology and Psychiatry*, vol. 23, no. 4, pp. 601–613, 2018. DOI: 10.1177/1359104518775154 (cited on p. 11).

[2] S. Hinduja and J. W. Patchin, "Cyberbullying: Identification, prevention, and response," [Online]. Available: https://cyberbullying.org/Cyberbullying-Identification-Prevention-Response-2023.pdf (cited on p. 11).

[3] S. Menini, A. Palmero Aprosio, and S. Tonelli, "Abuse is contextual, what about nlp? the role of context in abusive language annotation and detection," in *ArXiv Preprint*, 2021. [Online]. Available: https://arxiv.org/pdf/2103.14916.pdf (cited on pp. 12, 22, 26, 27).

[4] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, "Deceiving google's perspective api built for detecting toxic comments," *CoRR*, vol. cs.LG, p. 1702.08138, 2017. [Online]. Available: https://arxiv.org/abs/1702.08138 (cited on pp. 12, 22).

[5] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, "Antisocial behavior in online discussion communities," in *9th International AAAI Conference on Web and Social Media*, 2015, pp. 61–70. [Online]. Available: https://cs.stanford.edu/people/jure/pubs/trolls-icwsm15.pdf (cited on pp. 12, 22, 23).

[6] E. Papegnies, V. Labatut, R. Dufour, and G. Linarès, "Conversational networks for automatic online moderation," *IEEE Trans. Comput. Social Systems*, vol. 6, no. 1, pp. 38–55, 2019. DOI: 10.1109/TCSS.2018.2887240 (cited on pp. 12, 24, 45–50, 59, 60, 88).

[7] A. Gandhi, K. Adhvaryu, S. Poria, E. Cambria, and A. Hussain, "Multimodal sentiment analysis: A systematic review of history, datasets, multimodal fusion methods, applications, challenges and future directions," *Information Fusion*, vol. 91, pp. 424–444, 2023. DOI: https://doi.org/10.1016/j.inffus.2022.09.025 (cited on p. 12).

[8] Z. Wu, C. Zheng, Y. Cai, J. Chen, H.-f. Leung, and Q. Li, "Multimodal representation with embedded visual guiding objects for named entity recognition in social media posts," in *28th ACM International Conference on Multimedia*, 2020, pp. 1038–1046. DOI: 10.1145/3394171.3413650 (cited on p. 12).

[9] Z. Huang, X. Xu, J. Ni, H. Zhu, and C. Wang, "Multimodal representation learning for recommendation in internet of things," *IEEE Internet of Things Journal*, vol. 6, pp. 10 675–10 685, 2019. DOI: 10.1109/JIOT.2019.2940709 (cited on p. 12).

[10] A. H. Razavi, D. Inkpen, S. Uritsky, and S. Matwin, "Offensive language detection using multi-level classification," in *Canadian Conference on Artificial Intelligence*, 2010, pp. 16–27. DOI: 10.1007/978-3-642-13059-5_5 (cited on pp. 12, 20).

[11] E. Papegnies, V. Labatut, R. Dufour, and G. Linares, "Impact of content features for automatic online abuse detection," in *International Conference on Computational Linguistics and Intelligent Text Processing*, 2017, pp. 404–419. DOI: 10.1007/978-3-319-77116-8_30 (cited on pp. 12, 20, 27–29, 46, 52, 60).

[12] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection," *IEEE Access*, vol. 6, pp. 13 825–13 835, 2018. DOI: 10.1109/ACCESS.2018.2806394 (cited on pp. 12, 20).

[13] R. Martins, M. Gomes, J. J. Almeida, P. Novais, and P. Henriques, "Hate speech classification in social media using emotional analysis," in *7th Brazilian Conference on Intelligent Systems*, 2018, pp. 61–66. DOI: 10.1109/BRACIS.2018.00019. (cited on pp. 12, 20).

[14] P. Mishra, M. Del Tredici, H. Yannakoudakis, and E. Shutova, "Author profiling for abuse detection," in *27th International Conference on Computational Linguistics*, 2018, pp. 1088–1098. [Online]. Available: https://www.aclweb.org/anthology/C18-1093 (cited on pp. 12, 24, 31, 45, 113).

[15] S. Nagar, S. Gupta, C. S. Bahushruth, F. A. Barbhuiya, and K. Dey, "Hate speech detection on social media using graph convolutional networks," in *International Conference on Complex Networks and Their Applications*, 2022, pp. 3–14. DOI: 10.1007/978-3-030-93413-2_1 (cited on pp. 12, 25).

[16] R. Ali, U. Farooq, U. Arshad, W. Shahzad, and M. O. Beg, "Hate speech detection on twitter using transfer learning," *Computer Speech & Language*, vol. 74, 2022. DOI: 10.1016/j.csl.2022.101365 (cited on pp. 12, 21).

[17] H. Saleh, A. Alhothali, and K. Moria, "Detection of hate speech using bert and hate speech word embedding with deep model," *Applied Artificial Intelligence*, vol. 37, 2023. DOI: 10.1080/08839514.2023.2166719 (cited on pp. 12, 21).

[18] N. Cécillon, V. Labatut, R. Dufour, and G. Linarès, "WAC: A corpus of wikipedia conversations for online abuse detection," in *12th International Conference on Language Resources and Evaluation*, 2020. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2020/pdf/2020.lrec-1.173.pdf (cited on pp. 14, 17, 25–28, 123, 130).

[19] ——, "Graph embeddings for abusive language detection," *SN Computer Science*, vol. 2, no. 37, 2021. DOI: 10.1007/s42979-020-00413-7 (cited on pp. 14, 88, 97, 100).

[20] ——, "Abusive language detection in online conversations by combining content- and graph-based features," *Frontiers in Big Data*, vol. 2, p. 8, 2019, ISSN: 2624-909X. DOI: 10.3389/fdata.2019.00008 (cited on pp. 14, 46).

[21] N. Cécillon, R. Dufour, and V. Labatut, "Approche multimodale par plongement de texte et de graphes pour la détection de messages abusifs," *Traitement Automatique des Langues*, vol. 62, no. 2, pp. 13–38, 2021. [Online]. Available: https://www.atala.org/sites/default/files/TAL_62_2_v2.pdf (cited on pp. 14, 97).

[22] N. Cécillon, R. Dufour, V. Labatut, and G. Linarès, "Tuning graph2vec with node labels for abuse detection in online conversations," in *11th MARAMI*, 2020. [Online]. Available: http://ceur-ws.org/Vol-2750/paper8.pdf (cited on pp. 14, 97).

[23] N. Cécillon, N. Arnk, V. Labatut, and R. Dufour, "Whole-graph representation learning for the classification of signed networks," *Submitted*, 2023 (cited on pp. 15, 70).

[24] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *25th International Conference on World Wide Web*, 2016, pp. 145–153. DOI: 10.1145/2872427.2883062 (cited on pp. 18, 21).

[25] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Computing Surveys*, vol. 51, pp. 1–30, 2018. DOI: 10.1145/3232676 (cited on p. 18).

[26] M. Dadvar, D. Trieschnigg, R. Ordelman, and F. de Jong, "Improving cyberbullying detection with user context," in *35th European Conference on IR Research*, vol. 7814, Mar. 2013. DOI: 10.1007/978-3-642-36973-5_62 (cited on pp. 18, 23).

[27] R. N. M. Mercado, H. F. C. Chuctaya, and E. G. C. Gutierrez, "Automatic cyberbullying detection in spanish-language social networks using sentiment analysis techniques," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 7, pp. 228–235, 2018. DOI: 10.14569/IJACSA.2018.090733 (cited on p. 18).

[28] O. Goga, G. Venkatadri, and K. P. Gummadi, "The doppelgänger bot attack: Exploring identity impersonation in online social networks," in *2015 Internet Measurement Conference*, 2015, pp. 141–153. DOI: 10.1145/2815675.2815699 (cited on p. 18).

[29] S. R. Sangwan and M. Bhatia, "Denigration bullying resolution using wolf search optimized online reputation rumour detection," *Procedia Computer Science*, vol. 173, pp. 305–314, 2020. DOI: 10.1016/j.procs.2020.06.036 (cited on p. 18).

[30] R. A. Monteiro, R. L. S. Santos, T. A. S. Pardo, T. A. de Almeida, E. E. S. Ruiz, and O. A. Vale, "Contributions to the study of fake news in portuguese: New corpus and automatic detection results," in *International Conference on Computational Processing of the Portuguese Language*, 2018, pp. 324–334. DOI: 10.1007/978-3-319-99722-3_33 (cited on p. 18).

[31] Y. Karimi, A. Squicciarini, and S. Wilson, "Automated detection of doxing on twitter," *ACM Human-Computer Interactaction*, vol. 6, pp. 1–24, 2022. DOI: 10.1145/3555167 (cited on p. 18).

[32] D. M. Douglas, "Doxing: A conceptual analysis," *Ethics and Information Technology*, vol. 18, pp. 199–210, 2016. DOI: 10.1007/s10676-016-9406-0 (cited on p. 18).

[33] E. Spertus, "Smokey: Automatic recognition of hostile messages," in *14th National Conference on Artificial Intelligence and 9th Conference on Innovative Applications of Artificial Intelligence*, 1997, pp. 1058–1065. [Online]. Available: http://dl.acm.org/citation.cfm?id=1867616 (cited on pp. 20, 22, 46).

[34] N. D. Gitari, Z. Zuping, H. Damien, and J. Long, "A lexicon-based approach for hate speech detection," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 4, pp. 215–230, 2015. DOI: 10.14257/ijmue.2015.10.4.21 (cited on pp. 20, 31).

[35] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? predictive features for hate speech detection on twitter," in *NAACL Student Research Workshop*, 2016, pp. 88–93. [Online]. Available: http://www.aclweb.org/anthology/N16-2013 (cited on pp. 20, 31).

[36] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *International AAAI Conference on Web and Social Media*, 2017, pp. 512–515. DOI: 10.1609/icwsm.v11i1.14955 (cited on pp. 20, 31).

[37] E. Abdelzaher, "Lexicon-based detection of violence on social media," *Cognitive Semantics*, vol. 5, no. 1, pp. 32–69, 2019. DOI: 10.1163/23526416-00501002 (cited on p. 20).

[38] M. Fortunatus, P. Anthony, and S. Charters, "Combining textual features to detect cyberbullying in social media posts," *Procedia Computer Science*, vol. 176, pp. 612–621, 2020. DOI: 10.1016/j.procs.2020.08.063 (cited on pp. 20, 31).

[39] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, "Detection of harassment on Web 2.0," in *WWW Workshop: Content Analysis in the Web 2.0*, 2009, pp. 1–7. [Online]. Available: http://www.cse.lehigh.edu/~brian/pubs/2009/CAW2/ (cited on pp. 20, 22, 23, 31).

[40] K. Dinakar, R. Reichart, and H. Lieberman, "Modeling the detection of textual cyberbullying," in *5th ICWSM / Workshop on the Social Mobile Web*, 2011, pp. 11–17. [Online]. Available: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/view/3841 (cited on pp. 20, 46).

[41] J. Salminen, H. Almerekhi, M. Milenkovi, S. Jung, J. An, H. Kwak, and B. J. Jansen, "Anatomy of online hate: Developing a taxonomy and machine learning models for identifying and classifying hate in online news media.," in *ICWSM*, Jun. 2018. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/15028 (cited on pp. 20, 29).

[42] M. Wiegand, J. Ruppenhofer, A. Schmidt, and C. Greenberg, "Inducing a lexicon of abusive words a feature-based approach," in *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 1046–1056. DOI: 10.18653/v1/N18-1095 (cited on pp. 20, 22).

[43] H. Mubarak, K. Darwish, and W. Magdy, "Abusive language detection on arabic social media," in *Proceedings of the First Workshop on Abusive Language Online*, 2017, pp. 52–56. DOI: 10.18653/v1/W17-3008 (cited on p. 20).

[44] R. Martins, J. Almeida, P. Henriques, and P. Novais, "Increasing authorship identification through emotional analysis," in *World Conference on Information Systems and Technologies*, 2018, pp. 763–772. DOI: 10.1007/978-3-319-77703-0_76 (cited on p. 20).

[45] E. Alawneh, M. Al-Fawa'reh, J. M. T., and F. M. A., "Sentiment analysis-based sexual harassment detection using machine learning techniques," in *International Symposium on Electronics and Smart Devices*, 2021, pp. 1–6. DOI: 10.1109/ISESD53023.2021.9501725 (cited on p. 20).

[46] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate speech detection with comment embeddings," in *24th WWW*, 2015, pp. 29–30. DOI: 10.1145/2740908.2742760 (cited on p. 21).

[47] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *31st International Conference on International Conference on Machine Learning*, vol. 32, 2014, pp. II1188II1196 (cited on pp. 21, 77).

[48] J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos, "Deep learning for user comment moderation," in *Workshop on Abusive Language Online*, vol. First Workshop on Abusive Language Online, Aug. 2017, pp. 25–35. DOI: 10.18653/v1/W17-3004 (cited on pp. 21, 29).

[49] P. Mishra, H. Yannakoudakis, and E. Shutova, "Neural character-based composition models for abuse detection," in *2nd Workshop on Abusive Language Online*, vol. 2nd Workshop on Abusive Language Online (ALW2), Oct. 2018, pp. 1–10. DOI: 10.18653/v1/W18-5101 (cited on pp. 21, 96).

[50] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *26th International Conference on World Wide Web Companion*, 2017, pp. 759–760. DOI: 10.1145/3041021.3054223 (cited on pp. 21, 29).

[51] R. Kshirsagar, T. Cukuvac, K. McKeown, and S. McGregor, "Predictive embeddings for hate speech detection on twitter," in *2nd Workshop on Abusive Language Online*, 2018, pp. 26–32. DOI: 10.18653/v1/W18-5104 (cited on p. 21).

[52] R. Cao, R. K.-W. Lee, and T.-A. Hoang, "Deephate: Hate speech detection via multi-faceted text representations," in *12th ACM Conference on Web Science*, 2020, pp. 11–20. DOI: 10.1145/3394231.3397890 (cited on p. 21).

[53] O. E. Ojo, T. H. Ta, A. Gelbukh, H. Calvo, G. Sidorov, and O. O. Adebanji, "Automatic hate speech detection using deep neural networks and word embedding," *Computacion y Sistemas*, vol. 26, no. 2, pp. 1007–1013, 2022. DOI: 10.13053/CyS-26-2-4107 (cited on p. 21).

[54] N. Badri, F. Kboubi, and A. H. Chaibi, "Combining fasttext and glove word embedding for offensive and hate speech text detection," *Procedia Computer Science*, vol. 207, pp. 769–778, 2022. DOI: 10.1016/j.procs.2022.09.132 (cited on pp. 21, 29).

[55] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *First Workshop on Abusive Language Online*, 2017, pp. 85–90. DOI: 10.18653/v1/W17-3013 (cited on p. 21).

[56] H. Faris, I. Aljarah, M. Habib, and P. A. Castillo, "Hate speech detection using word embedding and deep learning in the arabic language context," in *9th International Conference on Pattern Recognition Applications and Methods*, 2020, pp. 453–460. DOI: 10.5220/0008954004530460 (cited on p. 21).

[57] A. Mazari, N. Boudoukhani, and A. Djeffal, "Bert-based ensemble learning for multi-aspect hate speech detection," *Cluster Computing*, 2023. DOI: 10.1007/s10586-022-03956-x (cited on p. 21).

[58] T. Caselli, V. Basile, J. Mitrovi, and M. Granitzer, "HateBERT: Retraining BERT for abusive language detection in english," in *5th Workshop on Online Abuse and Harms*, ACL, 2021, pp. 17–25. DOI: 10.18653/v1/2021.woah-1.3 (cited on p. 21).

[59] M. A. Khan, N. Yadav, M. Jain, and S. Goyal, "International conference on learning representations," in *The Art of Embedding Fusion: Optimizing Hate Speech Detection*, 2023. [Online]. Available: https://arxiv.org/pdf/2306.14939.pdf (cited on p. 21).

[60] N. Zampieri, I. Illina, and D. Fohr, "Improving hate speech detection with self-attention mechanism and multi-task learning," in *10th Language & Technology Conference: HumanLanguage Technologies as a Challenge for Computer Science and Linguistics*, 2023. [Online]. Available: https://hal.laas.fr/INRIA/hal-04017250v1 (cited on p. 21).

[61] J. H. Park and P. Fung, "One-step and two-step classification for abusive language detection on twitter," in *First Workshop on Abusive Language Online*, 2017, pp. 41–45. DOI: 10.18653/v1/W17-3006 (cited on p. 21).

[62] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. AbdelMajeed, and T. Zia, "Abusive language detection from social media comments using conventional machine learning and deep learning approaches," *Multimedia Systems*, vol. 28, no. 6, pp. 1925–1940, 2022. DOI: 10.1007/s00530-021-00784-8 (cited on p. 21).

[63] S.-J. Bu and S.-B. Cho, "A hybrid deep learning system of cnn and lrcn to detect cyberbullying from sns comments," in *International Conference on Hybrid Artificial Intelligence Systems*, 2018, pp. 561–572. DOI: 10.1007/978-3-319-92639-1_47 (cited on p. 22).

[64] R. Beniwal and A. Maurya, "Toxic comment classification using hybrid deep learning model," in *Sustainable Communication Networks and Application*, 2021, pp. 461–473. DOI: 10.1007/978-981-15-8677-4_38 (cited on p. 22).

[65] M. Alotaibi, B. Alotaibi, and A. Razaque, "A multichannel deep learning framework for cyberbullying detection on social media," *Electronics*, vol. 10, 2021. DOI: 10.3390/electronics10212664 (cited on p. 22).

[66] M. Raj, S. Singh, K. Solanki, and R. Selvanambi, "An application to detect cyberbullying using machine learning and deep learning techniques," *Springer Nature Computer Science*, vol. 3, 2022. DOI: 10.1007/s42979-022-01308-5 (cited on p. 22).

[67] B. A. H. Murshed, J. Abawajy, S. Mallappa, M. A. N. Saif, and H. D. E. Al-Ariki, "Dea-rnn: A hybrid deep learning approach for cyberbullying detection in twitter social media platform," *IEEE Access*, vol. 10, pp. 25 857–25 871, 2022. DOI: 10.1109/ACCESS.2022.3153675 (cited on p. 22).

[68] A. Anuchitanukul, J. Ive, and L. Specia, "Revisiting contextual toxicity detection in conversations," *Journal of Data and Information Quality*, vol. 15, pp. 1–22, 2022. DOI: 10.1145/3561390 (cited on p. 23).

[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423 (cited on p. 23).

[70] F. Vargas, F. Rodrigues de Góes, I. Carvalho, F. Benevenuto, and T. Pardo, "Contextual-lexicon approach for abusive language detection," in *International Conference on Recent Advances in Natural Language Processing*, 2021, pp. 1438–1447. [Online]. Available: https://aclanthology.org/2021.ranlp-1.161 (cited on p. 23).

[71] M. Karan and J. najder, "Preemptive toxic language detection in wikipedia comments using thread-level context," in *Third Workshop on Abusive Language Online*, 2019, pp. 129–134. DOI: 10.18653/v1/W19-3514 (cited on p. 23).

[72] H. Almerekhi, H. Kwak, B. J. Jansen, and J. Salminen, "Detecting toxicity triggers in online discussions," in *30th ACM Conference on Hypertext and Social Media*, 2019, pp. 291–292. DOI: 10.1145/3342220.3344933 (cited on p. 23).

[73] L. Gao and R. Huang, "Detecting online hate speech using context aware models," in *International Conference Recent Advances in Natural Language Processing*, 2017, pp. 260–266. DOI: 10.26615/978-954-452-049-6_036 (cited on p. 23).

[74] K. Balci and A. A. Salah, "Automatic analysis and identification of verbal aggression and abusive behaviors for online social games," *Computers in Human Behavior*, vol. 53, pp. 517–526, 2015. DOI: 10.1016/j.chb.2014.10.025 (cited on pp. 23, 24).

[75] M. Dadvar, F. de Jong, R. Ordelman, and D. Trieschnigg, "Improved cyberbullying detection using gender information.," in *12th Dutch-Belgian Information Retrieval Workshop*, 2012, pp. 23–26. [Online]. Available: http://dir2012.intec.ugent.be/system/files/proceedings/DIR2012_04_Maral_Dadvar.pdf (cited on p. 23).

[76] M. A. Al-Garadi, K. D. Varathan, and S. D. Ravana, "Cybercrime detection in online communications: The experimental case of cyberbullying detection in the twitter network," *Computers in Human Behavior*, vol. 63, pp. 433–443, 2016. DOI: 10.1016/j.chb.2016.05.051 (cited on pp. 23, 24).

[77] G. Xiang, B. Fan, L. Wang, J. Hong, and C. Rose, "Detecting offensive tweets via topical feature discovery over a large scale twitter corpus," in *21st ACM CIKM*, 2012, pp. 1980–1984. DOI: 10.1145/2396761.2398556 (cited on p. 24).

[78] D. Chatzakou, N. Kourtellis, J. Blackburn, E. De Cristofaro, G. Stringhini, and A. Vakali, "Mean birds: Detecting aggression and bullying on twitter," in *2017 ACM on Web Science Conference*, 2017, pp. 13–22. DOI: 10.1145/3091478.3091487 (cited on pp. 24, 30).

[79] C. Ziems, Y. Vigfusson, and F. Morstatter, "Aggressive, repetitive, intentional, visible, and imbalanced: Refining representations for cyberbullying classification," in *14th International AAAI Conference on Web and Social Media*, 2020. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/7345/7199 (cited on p. 24).

[80] E. F. Unsvåg and B. Gambäck, "The effects of user features on twitter hate speech detection," in *2nd Workshop on Abusive Language Online*, 2018, pp. 75–85. DOI: 10.18653/v1/W18-5110 (cited on p. 24).

[81] H. Hosseinmardi, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Prediction of cyberbullying incidents in a media-based social network," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2016, pp. 186–192. DOI: 10.1109/ASONAM.2016.7752233 (cited on p. 24).

[82] J. Qian, M. ElSherief, E. Belding, and W. Y. Wang, "Leveraging intra-user and inter-user representation learning for automated hate speech detection," in *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018, pp. 118–123. DOI: 10.18653/v1/N18-2019 (cited on p. 24).

[83] A. M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis, "A unified deep learning architecture for abuse detection," in *10th ACM WebSci*, 2019, pp. 105–114. DOI: 10.1145/3292522.3326028 (cited on pp. 24, 96).

[84] M. Saveski, B. Roy, and D. Roy, "The structure of toxic conversations on twitter," in *Web Conference*, 2021, pp. 1086–1097. DOI: 10.1145/3442381.3449861 (cited on p. 24).

[85] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *22nd ACM SIGKDD*, 2016, pp. 855–864. DOI: 10.1145/2939672.2939754 (cited on pp. 24, 70, 75, 101, 103, 112).

[86] P. Mishra, M. Del Tredici, H. Yannakoudakis, and E. Shutova, "Abusive language detection with graph convolutional networks," in *Conference of the North American Chapter of the ACL*, 2019, pp. 2145–2150. DOI: 10.18653/v1/N19-1221 (cited on p. 24).

[87] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017 (cited on pp. 24, 76).

[88] M. Ribeiro, P. Calais, Y. Santos, V. Almeida, and M. J. W., "Characterizing and detecting hateful users on twitter," in *International AAAI Conference on Web and Social Media*, 2018. [Online]. Available: https://arxiv.org/pdf/1803.08977.pdf (cited on p. 25).

[89] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1025–1035. [Online]. Available: https://dl.acm.org/doi/10.5555/3294771.3294869 (cited on pp. 25, 76).

[90] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *26th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, Feb. 2017, pp. 1391–1399. DOI: 10.1145/3038912.3052591 (cited on pp. 25, 26, 28, 123, 131).

[91] A. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis, "Large scale crowdsourcing and characterization of twitter abusive behavior," in *12th International AAAI Conference on Web and Social Media*, 2018. DOI: 10.1609/icwsm.v12i1.14991 (cited on pp. 25, 123).

[92] H. Kirk, W. Yin, B. Vidgen, and P. Röttger, "Semeval-2023 task 10: Explainable detection of online sexism," in *17th International Workshop on Semantic Evaluation*, 2023, pp. 2193–2210. DOI: 10.18653/v1/2023.semeval-1.305 (cited on pp. 26, 27).

[93] X. Yu, E. Blance, and L. Hong, "Hate speech and counter speech detection: Conversational context does matter," in *Conference of the NAACL: Human Language Technologies*, 2022, pp. 5918–5930. DOI: 10.18653/v1/2022.naacl-main.433 (cited on p. 26).

[94] T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar, "Toxigen: A large-scale machine-generated dataset for adversarialand implicit hate speech detection," in *60th Annual Meeting of the ACL*, 2022, pp. 3309–3326. [Online]. Available: https://aclanthology.org/2022.acl-long.234.pdf (cited on pp. 25–27).

[95] F. Beyhan, B. Çark, . Arn, A. Terziolu, B. Yanikoglu, and R. Yeniterzi, "A turkish hate speech dataset and detection system," in *13th Language Resources and Evaluation Conference*, 2022, pp. 4177–4185. [Online]. Available: https://aclanthology.org/2022.lrec-1.443 (cited on pp. 26, 27).

[96] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela, "Learning from the worst: Dynamically generated datasets to improve online hate detection," in *59th Annual Meeting of the ACL and the 11th International Joint Conference on NLP*, 2021, pp. 1667–1682. DOI: 10.18653/v1/2021.acl-long.132 (cited on pp. 25–27).

[97] A. Xenos, J. Pavlopoulos, and I. Androutsopoulos, "Context sensitivity estimation in toxicity detection," in *5th Workshop on Online Abuse and Harms*, 2021, pp. 140–145. DOI: 10.18653/v1/2021.woah-1.15 (cited on pp. 26, 27).

[98] T. Mandl, S. Modha, G. K. Shahi, A. K. Jaiswal, D. Nandini, D. Patel, P. Majumder, and J. Schäfer, "Overview of the hasoc track at fire 2020: Hate speech and offensive content identification inindo-european languages," in *CEUR Workshop*, 2020, pp. 87–111. [Online]. Available: https://ceur-ws.org/Vol-2826/T2-1.pdf (cited on p. 26).

[99] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, and M. Sanguinetti, "Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter," in *13th International Workshop on Semantic Evaluation*, 2019, pp. 54–63. DOI: 10.18653/v1/S19-2007 (cited on pp. 26, 27).

[100] B. Vidgen, D. Nguyen, H. Margetts, P. Rossini, and R. Tromble, "Introducing cad: The contextual abuse dataset," in *2021 Conference of the NA Chapter of the ACL: Human Language Technologies*, 2021, pp. 2289–2303. DOI: 10.18653/v1/2021.naacl-main.182 (cited on pp. 26, 27).

[101] A. Ollagnier, E. Cabrio, S. Villata, and C. Blaya, "Cyberagressionado-v1: A dataset of annotated online aggressions in french collected through a role-playing game," in *13th Language Resources and Evaluation Conference*, 2022, pp. 867–875. [Online]. Available: https://aclanthology.org/2022.lrec-1.91 (cited on pp. 26, 27).

[102] R. Hada, S. Sudhir, P. Mishra, H. Yannakoudakis, S. M. Mohammad, and E. Shutova, "Ruddit: Norms of offensiveness for english reddit comments," in *ACL-IJCNLP 2021*, 2021, pp. 2700–2717. DOI: 10.18653/v1/2021.acl-long.210 (cited on pp. 26, 27, 129).

[103] E. Papegnies, V. Labatut, R. Dufour, and G. Linarès, "Detection of abusive messages in an on-line community," in *CORIA 2017 - Conférence en Recherche d'Informations et Applications- 14th French Information Retrieval Conference*, Mar. 2017, pp. 153–168. DOI: doi:10.24348/coria.2017.16 (cited on pp. 27, 45, 46).

[104] Y. Hua, C. Danescu-Niculescu-Mizil, D. Taraborelli, N. Thain, J. Sorensen, and L. Dixon, "Wikiconv: A corpus of the complete conversational history of a large online collaborative community," in *Conference on Empirical Methods in Natural Language Processing*, vol. 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2818–2823. DOI: 10.18653/v1/d18-1305 (cited on pp. 29, 132).

[105] S. Kiritchenko, I. Nejadgholi, and K. C. Fraser, "Confronting abusive language online: A survey from the ethical and human rights perspective," *Journal of Artificial Intelligence Research*, vol. 71, pp. 431–478, 2021. DOI: 10.1613/jair.1.12590 (cited on p. 29).

[106] N. Babakov, V. Logacheva, and A. Panchenko, "Beyond plain toxic: Building datasets for detection of flammable topics and inappropriate statements," *Language Resources and Evaluation*, 2023. DOI: 10.1007/s10579-023-09682-z (cited on p. 29).

[107] Z. Li, M. Rei, and L. Specia, "Multimodal conversation modelling for topic derailment detection," in *EMNLP*, 2022, pp. 5115–5127. DOI: 10.18653/v1/2022.findings-emnlp.376 (cited on p. 29).

[108] F. Syeda Faizan, L. Seemab, and L. Rabia, "Fine tuning bert for unethical behavior classification," in *International Conference on Digital Futures and Transformative Technologies*, 2021, pp. 1–6. DOI: 10.1109/ICoDT252288.2021.9441540 (cited on p. 29).

[109] T. Xu, G. Goossen, H. K. Cevahir, S. Khodeir, Y. Jin, F. Li, S. Shan, S. Patel, D. Freeman, and P. Pearce, "Deep entity classification: Abusive account detection for online social networks," in *30th USENIX Security Symposium*, 2021, pp. 4097–4114. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/xu-teng (cited on p. 29).

[110] S. Subramani, S. Michalska, H. Wang, J. Du, Y. Zhang, and H. Shakeel, "Deep learning for multi-class identification from domestic violence online posts," *IEEE Access*, vol. 7, pp. 46 210–46 224, 2019. DOI: 10.1109/ACCESS.2019.2908827 (cited on p. 29).

[111] M. Jorgensen, M. Choi, M. Niemann, J. Brunk, and J. Becker, "Multi-class detection of abusive language using automated machine learning," in *15th International Conference on Wirtschaftsinformatik*,, 2020, pp. 1763–1775. DOI: 10.30844/wi_2020_r7-jorgensen (cited on p. 29).

[112] A. Garcia-Recuero, "Discouraging abusive behavior in privacy-preserving online social networking applications," in *25th International Conference Companion on World Wide Web*, 2016, pp. 305–309. DOI: 10.1145/2872518.2888600 (cited on pp. 30, 31).

[113] J. M. Lobo, A. Jimenez-Valverde, and R. Real, "Auc: A misleading measure of the performance of predictive distribution models," *Global Ecology and Biogeography*, vol. 17, pp. 145–151, 2008. DOI: 10.1111/j.1466-8238.2007.00358.x (cited on p. 31).

[114] S. Halligan, D. Altman, and S. Mallett, "Disadvantages of using the area under the receiver operating characteristic curve to assess imaging tests: A discussion and proposal for an alternative approach," *European Radiology*, vol. 25, pp. 932–939, 2015. DOI: 10.1007/s00330-014-3487-0 (cited on p. 31).

[115] J. Muschelli, "Roc and auc with a binary predictor: A potentially misleading metric," *Journal of Classification*, vol. 37, pp. 696–708, 2019. DOI: 10.1007/s00357-019-09345-1 (cited on p. 31).

[116] K. Orman, "Contribution to the interpretation of evolving communities in complex networks : Application to the study of social interactions," PhD thesis, LIRIS, 2014. [Online]. Available: https://www.theses.fr/2014ISAL0072 (cited on p. 33).

[117] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998. DOI: 10.1038/30918 (cited on p. 36).

[118] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani, "The architecture of complex weighted networks," *Proceedings of the National Academy of Sciences*, vol. 101, no. 11, pp. 3747–3752, 2004. DOI: 10.1073/pnas.0400087101 (cited on p. 36).

[119] R. S. Burt, "Structural holes and good ideas," *American Journal of Sociology*, vol. 110, no. 2, pp. 349–399, 2004. DOI: 10.1086/421787 (cited on p. 37).

[120] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *Journal of Mathematical Sociology*, vol. 2, no. 1, pp. 113–120, 1972. DOI: 10.1080/0022250X.1972.9989806 (cited on p. 37).

[121] S. Brin and L. E. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, vol. 30, pp. 107–117, 1998. DOI: 10.1016/S0169-7552(98)00110-X (cited on p. 38).

[122] J. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the Association for Computing Machinery*, vol. 46, no. 5, pp. 604–632, 1999. DOI: 10.1145/324133.324140 (cited on p. 38).

[123] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953. DOI: 10.1007/bf02289026 (cited on p. 39).

[124] P. Bonacich and P. Lloyd, "Eigenvector-like measures of centrality for asymmetric relations," *Social Networks*, vol. 23, no. 3, pp. 191–201, 2001. DOI: 10.1016/S0378-8733(01)00038-7 (cited on p. 39).

[125] P. F. Bonacich, "Power and centrality: A family of measures," *American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987. DOI: 10.1086/228631 (cited on p. 39).

[126] E. Estrada and J. A. Rodriguez-Velazquez, "Subgraph centrality in complex networks," *Physical Review E*, vol. 71, no. 5, p. 056 103, 2005. DOI: 10.1103/PhysRevE.71.056103 (cited on p. 39).

[127] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977. DOI: 10.2307/3033543 (cited on p. 40).

[128] A. Bavelas, "Communication patterns in task-oriented groups," *Journal of the Acoustical Society of America*, vol. 22, no. 6, pp. 725–730, 1950. DOI: 10.1121/1.1906679 (cited on p. 40).

[129] F. Harary, *Graph Theory*. Addison-Wesley, 1969. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/705364.pdf (cited on pp. 40, 41).

[130] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983. DOI: 10.1016/0378-8733(83)90028-X (cited on p. 41).

[131] R. Guimerà and L. A. N. Amaral, "Functional cartography of complex metabolic networks," *Nature*, vol. 433, pp. 895–900, 2005. DOI: 10.1038/nature03288 (cited on p. 41).

[132] N. Dugué, V. Labatut, and A. Perez, "Identifying the community roles of social capitalists in the Twitter network," in *IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, IEEE, 2014, pp. 371–374. DOI: 10.1109/ASONAM.2014.6921612 (cited on p. 41).

[133] R. D. Luce and A. D. Perry, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, 1949. DOI: 10.1007/BF02289146 (cited on p. 42).

[134] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, ser. Structural Analysis in the Social Sciences. Cambridge, UK: Cambridge University Press, 1994, vol. 8. [Online]. Available: http://www.cambridge.org/zw/academic/subjects/sociology/sociology-general-interest/social-network-analysis-methods-and-applications (cited on p. 43).

[135] D. R. White, F. Harary, M. Sobel, and M. Becker, "The cohesiveness of blocks in social networks: Node connectivity and conditional density," *Sociological Methodology*, vol. 31, no. 1, pp. 305–359, 2001. DOI: 10.1111/0081-1750.00098 (cited on p. 43).

[136] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026 113, 2004. DOI: 10.1103/PhysRevE.69.026113 (cited on p. 44).

[137] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, Community detection in graphs, 2010. DOI: https://doi.org/10.1016/j.physrep.2009.11.002 (cited on p. 44).

[138] P. Mutton, "Inferring and visualizing social networks on internet relay chat," in *8th International Conference on Information Visualisation*, 2004, pp. 35–43. DOI: 10.1109/IV.2004.1320122 (cited on p. 47).

[139] R. Dzisevi and D. eok, "Text classification using different feature extraction approaches," in *Open Conference of Electrical, Electronic and Information Sciences*, 2019, pp. 1–4. DOI: 10.1109/eStream.2019.8732167 (cited on p. 52).

[140] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt, "A text feature based automatic keyword extraction method for single documents," in *Advances in Information Retrieval*, 2018, pp. 684–691. DOI: https://doi.org/10.1007/978-3-319-76941-7_63 (cited on p. 52).

[141] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proceedings of the Workshop on Speech and Natural Language*, 1992, pp. 212–217. [Online]. Available: https://aclanthology.org/H92-1041.pdf (cited on p. 52).

[142] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, "Detecting offensive language in social media to protect adolescent online safety," in *International Conference on Privacy, Security, Risk and Trust and International Conference on Social Computing*, 2012, pp. 71–80. DOI: 10.1109/SocialCom-PASSAT.2012.55 (cited on p. 52).

[143] H. Chen, S. Mckeever, and S. J. Delany, "Presenting a labelled dataset for real-time detection of abusive user posts," in *International Conference on Web Intelligence*, 2017, pp. 884–890. DOI: 10.1145/3106426.3106456 (cited on p. 52).

[144] L. V. Batista and M. M. Meira, "Texture classification using the lempel-ziv-welch algorithm," in *Brazilian Symposium on Artificial Intelligence*, 2004, pp. 444–453. DOI: 10.1007/978-3-540-28645-5_45 (cited on p. 52).

[145] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, p. 1118, 2008. DOI: 10.1073/pnas.0706851105 (cited on p. 56).

[146] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. 1695, 2006. [Online]. Available: http://igraph.sf.net (cited on p. 59).

[147] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://www.jmlr.org/papers/v12/pedregosa11a.html (cited on p. 59).

[148] K. Babi, S. Martini-Ipi, and A. Metrovi, "Survey of neural text representation models," *Information*, vol. 11, no. 11, 2020. DOI: 10.3390/info11110511 (cited on p. 68).

[149] T. Schomacker and M. Tropmann-Frick, "Language representation models: An overview," *Entropy*, vol. 23, no. 11, 2021. DOI: https://doi.org/10.3390/e23111422 (cited on p. 68).

[150] P. Siebers, C. Janiesch, and P. Zschech, "A survey of text representation methods and their genealogy," *IEEE Access*, vol. 10, pp. 96 492–96 513, 2022. DOI: 10.1109/ACCESS.2022.3205719 (cited on p. 68).

[151] A. Mohamed, H.-y. Lee, L. Borgholt, J. D. Havtorn, J. Edin, C. Igel, K. Kirchhoff, S.-W. Li, K. Livescu, L. Maaløe, T. N. Sainath, and S. Watanabe, "Self-supervised speech representation learning: A review," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, pp. 1179–1210, 2022. DOI: 10.1109/JSTSP.2022.3207050 (cited on p. 68).

[152] R. Patil, S. Boit, V. Gudivada, and J. Nandigam, "A survey of text representation and embedding techniques in nlp," *IEEE Access*, vol. 11, pp. 36 120–36 146, 2023. DOI: 10.1109/ACCESS.2023.3266377 (cited on p. 68).

[153] F. Incitti, F. Urli, and L. Snidaro, "Beyond word embeddings: A survey," *Information Fusion*, vol. 89, pp. 418–436, 2023. DOI: https://doi.org/10.1016/j.inffus.2022.08.024 (cited on p. 68).

[154] Z. Liu. and M. Sun, "Representation learning and nlp," in *Representation Learning for Natural Language Processing*, Springer Nature Singapore Singapore, 2023, pp. 1–27. DOI: https://doi.org/10.1007/978-981-99-1600-9_1 (cited on p. 68).

[155] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *22nd ACM SIGKDD*, 2016, pp. 1105–1114. DOI: 10.1145/2939672.2939751 (cited on pp. 70, 74, 75).

[156] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems 14*, 2002, pp. 585–591. [Online]. Available: http://papers.nips.cc/paper/1961-laplacian-eigenmaps-and-spectral-techniques-for-embedding-and-clustering.pdf (cited on pp. 70, 73, 74, 77).

[157] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *22nd International Conference on World Wide Web*, 2013, pp. 37–48. DOI: 10.1145/2488388.2488393 (cited on pp. 70, 74).

[158] S. Yuan, X. Wu, and Y. Xiang, "SNE: Signed network embedding," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, vol. 10235, 2017, pp. 183–195. DOI: 10.1007/978-3-319-57529-2_15 (cited on pp. 70, 79).

[159] J. Kim, H. Park, J.-E. Lee, and U. Kang, "SIDE: Representation learning in signed directed networks," in *World Wide Web Conference*, 2018, pp. 509–518. DOI: 10.1145/3178876.3186117 (cited on pp. 70, 79).

[160] T. Derr, Y. Ma, and J. Tang, "Signed graph convolutional network," in *18th ICDM*, 2018, pp. 929–934. DOI: 10.1109/ICDM.2018.00113 (cited on pp. 70, 80).

[161] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," in *MLG*, 2017 (cited on pp. 70, 77, 84, 102, 103, 105).

[162] F. Heider, "Attitudes and cognitive organization," *Journal of Psychology*, vol. 21, no. 1, pp. 107–112, 1946. DOI: 10.1080/00223980.1946.9917275 (cited on p. 71).

[163] F. Harary, "On the notion of balance of a signed graph," *Michigan Mathematical Journal*, vol. 2, no. 2, pp. 143–146, 1953. DOI: 10.1307/mmj/1028989917 (cited on p. 71).

[164] D. Cartwright and F. Harary, "Structural balance: A generalization of Heider's theory," *Psychological Review*, vol. 63, pp. 277–293, 1956. DOI: 10.1037/h0046049 (cited on p. 71).

[165] P. Doreian and A. Mrvar, "A partitioning approach to structural balance," *Social Networks*, vol. 18, no. 2, pp. 149–168, 1996. DOI: 10.1016/0378-8733(95)00259-6 (cited on p. 72).

[166] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," in *FOCS*, 2002, pp. 238–247. DOI: 10.1109/SFCS.2002.1181947 (cited on pp. 72, 81).

[167] J. A. Davis, "Clustering and structural balance in graphs," *Human Relations*, vol. 20, no. 2, pp. 181–187, 1967. DOI: 10.1177/001872676702000207 (cited on p. 72).

[168] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE TKDE*, vol. 30, no. 9, pp. 1616–1637, 2018. DOI: 10.1109/TKDE.2018.2807452 (cited on p. 72).

[169] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE PAMI*, vol. 29, pp. 40–51, 2007. DOI: 10.1109/tpami.2007.250598 (cited on pp. 72, 73).

[170] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018, ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2018.03.022 (cited on pp. 73, 74, 76, 96, 112).

[171] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. DOI: 10.1126/science.290.5500.2323 (cited on p. 73).

[172] X. Liang, D. Li, M. Song, A. Madden, Y. Ding, and Y. Bu, "Predicting biomedical relationships using the knowledge and graph embedding cascade model," *PLoS ONE*, vol. 14, 2019. DOI: https://doi.org/10.1371/journal.pone.0218264 (cited on pp. 74, 75).

[173] J. Li, L. Wu, R. Guo, C. Liu, and H. Liu, "Multi-level network embedding with boosted low-rank matrix approximation," in *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2019, pp. 49–56. DOI: 10.1145/3341161.3342864 (cited on pp. 74, 102, 103).

[174] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013. [Online]. Available: https://arxiv.org/abs/1301.3781 (cited on pp. 74, 75, 98).

[175] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *20th ACM SIGKDD*, 2014, pp. 701–710. DOI: 10.1145/2623330.2623732 (cited on pp. 75, 101, 103).

[176] B. Hou, Y. Wang, M. Zeng, S. Jiang, O. J. Mengshoel, Y. Tong, and J. Bai, "Customized graph embedding: Tailoring embedding vectors to different applications," *arXiv*, 2019. [Online]. Available: http://arxiv.org/abs/1911.09454 (cited on p. 75).

[177] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234. DOI: 10.1145/2939672.2939753 (cited on pp. 75, 76).

[178] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "HARP: Hierarchical representation learning for networks," in *32nd AAAI Conferenceon Artificial Intelligence*, 2018. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewFile/16273/15922 (cited on p. 75).

[179] B. Perozzi, V. Kulkarni, and S. Skiena, "Don't walk, skip! online learning of multi-scale network embeddings," in *IEEE/ACM ASONAM*, 2017, pp. 258–265. DOI: 10.1145/3110025.3110086 (cited on pp. 75, 101, 103).

[180] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: Graph representation learningwith generative adversarial nets," in *32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 2508–2515. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16611 (cited on p. 76).

[181] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021. DOI: 10.1109/tnnls.2020.2978386 (cited on p. 76).

[182] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: Algorithms, applications and open challenges," in *International Conference on Computational Social Networks*, ser. Lecture Notes in Computer Science, vol. 11280, 2018, pp. 79–91. DOI: 10.1007/978-3-030-04648-4_7 (cited on p. 76).

[183] N. de Lara and E. Pineau, "A simple baseline algorithm for graph classification," *arXiv*, 2018. eprint: 1810.09155 (cs.LG). [Online]. Available: https://arxiv.org/pdf/1810.09155.pdf (cited on pp. 77, 102, 103).

[184] S. F. Mousavi, M. Safayani, A. Mirzaei, and H. Bahonar, "Hierarchical graph embedding in vector space by graph pyramid," *Pattern Recognition*, vol. 61, pp. 245–254, 2017, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2016.07.043 (cited on p. 77).

[185] S. Verma and Z.-L. Zhang, "Hunt for the unique, stable, sparse and fast feature learning on graphs," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 88–98. [Online]. Available: http://papers.nips.cc/paper/6614-hunt-for-the-unique-stable-sparse-and-fast-feature-learning-on-graphs.pdf (cited on pp. 77, 102, 103).

[186] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller, "NetLSD: Hearing the shape of a graph," in *24th ACM SIGKDD*, 2018, pp. 2347–2356. DOI: 10.1145/3219819.3219991 (cited on p. 77).

[187] T. Huynh, T. T. Thi Ho, and B. Le, "Graph classification via graph structure learning," in *Asian Conference on Intelligent Information and Database Systems*, ser. Lecture Notes in Computer Science, vol. 13758, Springer, 2022, pp. 269–281. DOI: 10.1007/978-3-031-21967-2_22 (cited on p. 77).

[188] R. Winter, F. Noe, and D.-A. Clevert, "Permutation-invariant variational autoencoder for graph-level representation learning," in *Conference on Neural Information Processing Systems*, 2021 (cited on p. 78).

[189] L. Gutiérrez-Gómez and J.-C. Delvenne, "Unsupervised network embeddings with node identity awareness," *Applied Network Science*, vol. 4, p. 82, 2019. DOI: 10.1007/s41109-019-0197-1 (cited on p. 78).

[190] R. Al-Rfou, B. Perozzi, and D. Zelle, "Ddgk: Learning graph representations for deep divergence graph kernels," in *World Wide Web Conference*, 2019, pp. 37–48. DOI: 10.1145/3308558.3313668 (cited on p. 78).

[191] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *33rd International Conference on International Conference on Machine Learning*, 2016, pp. 2014–2023 (cited on p. 78).

[192] Z. Luo, L. Liu, J. Yin, Y. Li, and Z. Wu, "Deep learning of graphs with ngram convolutional neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 10, pp. 2125–2139, 2017. DOI: 10.1109/tkde.2017.2720734 (cited on p. 78).

[193] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th ICML*, 2017, pp. 1263–1272 (cited on pp. 78, 85).

[194] T. Pham, T. Tran, H. Dam, and S. Venkatesh, "Graph classification via deep learning with virtual nodes," *arXiv*, vol. cs.LG, p. 1708.04357, 2017 (cited on p. 78).

[195] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *17th SIAM International Conference on Data Mining*, 2017, pp. 327–335. DOI: 10.1137/1.9781611974973.37 (cited on p. 79).

[196] M. Cygan, M. Pilipczuk, M. Pilipczuk, and J. Wojtaszczyk, "Sitting closer to friends than enemies, revisited," in *Mathematical Foundations of Computer Science*, 2012, pp. 296–307. DOI: https://doi.org/10.1007/978-3-642-32589-2_28 (cited on p. 79).

[197] J. Huang, H. Shen, L. Hou, and X. Cheng, "Signed graph attention networks," in *International Conference on Artificial Neural Networks*, 2019 (cited on p. 80).

[198] Y. Li, Y. Tian, J. Zhang, and Y. Chang, "Learning signed network embedding via graph attention," in *34th AAAI Conference on Artificial Intelligence*, 2020 (cited on p. 80).

[199] N. Arnk, R. Figueiredo, and V. Labatut, "Multiplicity and diversity: Analysing the optimal solution space of the correlation clustering problem on complete signed graphs," *Journal of Complex Networks*, vol. 8, no. 6, 2021. DOI: 10.1093/comnet/cnaa025 (cited on p. 81).

[200] ——, "Multiple partitioning of multiplex signed networks: Application to european parliament votes," *Social Networks*, vol. 60, pp. 83–102, 2020. DOI: 10.1016/j.socnet.2019.02.001 (cited on p. 81).

[201] B. Rozemberczki and R. Sarkar, "Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models," in *29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1325–1334. DOI: 10.1145/3340531.3411866 (cited on pp. 83, 100).

[202] A. Galland, "Deep learning techniques for graph embedding at different scales," PhD thesis, Université Paris sciences et lettres, 2020. [Online]. Available: https://theses.hal.science/tel-03690033/ (cited on p. 83).

[203] B. Y. Weisfeiler and A. A. Leman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968 (cited on p. 84).

[204] C. Fang, Z. Liu, Y. Shi, J. Huang, and Q. Shi, "Functional code clone detection with syntax and semantics fusion learning," in *International Symposium on Software Testing and Analysis*, 2020, pp. 516–527. DOI: 10.1145/3395363.3397362 (cited on p. 84).

[205] X. Zhou, Y. Zhang, Z. Li, X. Wang, J. Zhao, and Z. Zhang, "Large-scale cellular traffic prediction based on graph convolutional networks with transfer learning," *Neural Computing and Applications*, vol. 34, pp. 5549–5559, 2022. DOI: 10.1007/s00521-021-06708-x (cited on p. 84).

[206] Q.-D. Ngo, H.-T. Nguyen, H.-A. Tran, and D.-H. Nguyen, "IoT botnet detection based on the integration of static and dynamic vector features," in *8th International Conference on Communications and Electronics*, 2021, pp. 540–545. DOI: 10.1109/ICCE48956.2021.9352145 (cited on p. 84).

[207] Z. Zhang, J. Liu, X. Zheng, Y. Wang, P. Han, Y. Wang, K. Zhao, and Z. Zhang, "RSGNN: A model-agnostic approach for enhancing the robustness of signed graph neural networks," in *ACM Web Conference*, 2023, pp. 60–70. DOI: 10.1145/3543507.3583221 (cited on p. 85).

[208] P. Doreian and A. Mrvar, "Partitioning signed social networks," *Social Networks*, vol. 31, no. 1, pp. 1–11, 2009. DOI: 10.1016/j.socnet.2008.08.001 (cited on p. 94).

[209] R. Figueiredo and G. Moura, "Mixed integer programming formulations for clustering problems related to structural balance," *Social Networks*, vol. 35, no. 4, pp. 639–651, 2013. DOI: 10.1016/j.socnet.2013.09.002 (cited on p. 94).

[210] J. Padilla Montani and P. Schüller, "TUWienKBS at GermEval 2018: German abusive tweet detection," in *GermEval 2018 Workshop*, 2018, pp. 45–50. [Online]. Available: https://www.oeaw.ac.at/fileadmin/subsites/academiaecorpora/PDF/GermEval2018_Proceedings.pdf (cited on p. 96).

[211] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the ACL*, vol. 5, pp. 135–146, 2017. DOI: 10.1162/tacl_a_00051 (cited on p. 99).

[212] L. Martin, B. Muller, P. J. Ortiz Suárez, Y. Dupont, L. Romary, E. de la Clergerie, D. Seddah, and B. Sagot, "CamemBERT: A tasty french language model," in *ACL*, 2020, pp. 7203–7219. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.645 (cited on p. 99).

[213] H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbé, L. Besacier, and D. Schwab, "FlauBERT: Unsupervised language model pre-training for french," in *12th Language Resources and Evaluation Conference*, 2020, pp. 2479–2490. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.302 (cited on pp. 99, 109, 113).

[214] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv*, 2019. [Online]. Available: https://arxiv.org/pdf/1907.11692.pdf (cited on p. 99).

[215] G. Wenzek, M.-A. Lachaux, A. Conneau, V. Chaudhary, F. Guzmán, A. Joulin, and E. Grave, "CCNet: Extracting high quality monolingual datasets from web crawl data," in *12th Language Resources and Evaluation Conference*, 2020, pp. 4003–4012. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.494.pdf (cited on p. 99).

[216] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *27th International Conference on Computational Linguistics*, 2018, pp. 1638–1649. [Online]. Available: https://www.aclweb.org/anthology/C18-1139 (cited on pp. 99, 117, 124, 127, 128).

[217] C. Donnat, M. Zitnik, D. Hallac, and J. Leskovec, "Learning structural node embeddings via diffusion wavelets," in *24th ACM SIGKDD*, 2018, pp. 1320–1329. DOI: 10.1145/3219819.3220025 (cited on pp. 102, 103).

[218] G. Nikolentzos, G. Dasoulas, and M. Vazirgiannis, "K-hop graph neural networks," *Neural Networks*, vol. 130, pp. 195–205, 2020. DOI: 10.1016/j.neunet.2020.07.008 (cited on pp. 102, 103).

[219] M. Zhang and P. Li, "Nested graph neural networks," in *Conference on Neural Information Processing Systems*, 2021, pp. 15 734–15 747. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/8462a7c229aea03dde69da754c3bbcc4-Paper.pdf (cited on pp. 102, 103).

[220] A. Rücklé, S. Eger, M. Peyrard, and I. Gurevych, "Concatenated power mean word embeddingsas universal cross-lingual sentence representations," *arXiv*, 2018. [Online]. Available: https://arxiv.org/abs/1803.01400 (cited on p. 107).

[221] U. Singer, I. Guy, and K. Radinsky, "Node embedding over temporal graphs," in *International Joint Conference on Artificial Intelligence*, 2019, pp. 4605–4612. DOI: 10.24963/ijcai.2019/640 (cited on p. 124).