Approches multimodales pour la détection d'anomalies sur les réseaux sociaux numériques

Omar Jaafor

Thèse
de doctorat
de l'UTT

Omar JAAFOR

Approches multimodales pour la détection d'anomalies sur les réseaux sociaux numériques

Champ disciplinaire :
Sciences pour l'Ingénieur

2018TROY0046 **Année 2018**

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR
de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

EN SCIENCES POUR L'INGENIEUR

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Omar JAAFOR

le 5 décembre 2018

Approches multimodales pour la détection d'anomalies sur les réseaux sociaux numériques

JURY

M. H. SNOUSSI	PROFESSEUR DES UNIVERSITES	Président
M. G. BONFANTE	MAITRE DE CONFERENCES - HDR	Rapporteur
M. C. DU MOUZA	MAITRE DE CONFERENCES - HDR	Rapporteur
Mme A. BOYER	PROFESSEURE DES UNIVERSITES	Examinatrice
M. N. GROZAVU	MAITRE DE CONFERENCES - HDR	Examinateur
M. D. KOTZINOS	PROFESSEUR DES UNIVERSITES	Examinateur
Mme L. MERGHEM-BOULAHIA	PROFESSEURE DES UNIVERSITES	Examinatrice
M. B. BIRREGAH	MAITRE DE CONFERENCES	Directeur de thèse

Remerciements

Au terme de mes études doctorales, je voudrais dire un grand merci à tous ceux et celles qui m'ont soutenu d'une manière ou d'une autre, et sans qui je n'aurais pu réaliser mon parcours académique.

Je tiens tout d'abord à remercier mon directeur de recherche, Dr Babiga Birregah, qui m'a encadré durant ma thèse. Son aide précieuse m'a permis d'avancer dans mes travaux scientifiques.

Je souhaiterais également remercier ma famille et mes amis qui ont été à mes côtés.

Je tiens a exprimer ma gratitude à Mm. Guillaume Bonfante et Cédric Du Mouza, qui ont accepté d'être rapporteurs de ma thèse, pour l'attention qu'ils y ont portée et les observations pertinentes qu'ils ont faites sur ce travail. Je tiens spécialement à remercier Mme Boyer, M Grozavu, Mme Merghem-Boulahia, M Kotzinos, et M Snoussi pour avoir accepté d'évaluer cette thèse.

Je remercie Mmes Bernadette André et Véronique Banse, au secrétariat du laboratoire de Modélisation et Sûreté des Systèmes (LM2S), pour m'avoir aidé sur mes différentes tâches administratives.

Je voudrais également remercier Mme Ansaldi de m'avoir aidé à corriger ma thèse.

Cette thèse de doctorat est effectuée au sein du laboratoire LM2S - Pôle Recherche Opérationnelle, Statistiques Appliquées et Simulation (ROSAS) - ICD - Université de Technologie de Troyes.

Omar JAAFOR

Résumé

L'émergence des réseaux sociaux numériques tels que Twitter, Facebook ainsi que les blogs et forums ont permis pour la première fois d'effectuer des analyses à grande échelle sur le comportement humain. Alors que dans le passé, la majorité des études centrées sur l'humain était orientée terrain, ces réseaux ont permis le développement de méthodes statistiques qui analysent des traces de l'activité humaine. Les plateformes sociales ont ainsi permis d'étudier le comportement d'utilisateurs malveillant qui ont migré certaines de leurs activités sur ces plateformes. En effet, des groupes radicaux utilisent les réseaux sociaux pour recruter des jihadistes. Des revendeurs de cartes de crédits volées ont pu avoir accès à un marché important grâce aux réseaux sociaux. Aussi, des spammeurs utilisent des robots qui polluent le contenu de ces réseaux. Cette thèse concerne la détection de ces utilisateurs malveillant qui ont un comportement atypique sur ces réseaux sociaux numériques en prenant en compte différents modes d'interaction et de similarité (Retweets, Mentions, Similarité d'URL, similarité du texte, etc.). Nous avons développé des méthodes semi-supervisées, supervisées ainsi que des méthodes non-supervisées (basées sur la détection de communautés) afin de détecter ces utilisateurs.

Abstract

The emergence of on-line social networks such as Twitter, Facebook as well as blogs and forums has allowed for the first time to carry out large-scale analyzis of human interactions. Whereas in the past, the majority of human-centered studies were field-oriented, the emergence of these networks allowed the development of statistical methods which took as entry traces of human activity. On-line social networks have allowed for the first time to conduct large scale studies of malicious users that leverage these plateforms. For example, radical groups use social networks to recruit jihadists. Resellers of stolen credit cards are now able to access a large market through social networks. Also, spammers use robots that pollute the content of these networks.

This thesis concerns the detection of malicious users who behave in an anomalous manner in on-line platforms using different modes of interactions and similarity (Retweets, Mentions, URL similarity, text similarity). We have developed semi-supervised, supervised as well as unsupervised methods (based on community detection) to detect malicious users.

Table des matières

Introduction générale

Les réseaux sociaux sont devenus des plateformes incontournables qui s'inscrivent dans la vie quotidienne de plusieurs milliards d'utilisateurs. Ces plateformes sont aujourd'hui des lieux de socialisation, d'information, d'e-commerce et de distraction.

L'émergence des plateformes sociales a souvent conduit au rapprochement de communautés qui ne pourraient pas communiquer autrement. Cela a également donné la parole à des membres de communautés qui se sentent opprimés dans des pays où les médias traditionnels sont contrôlés. Cependant, ces réseaux ont bénéficié à des membres d'organisations malveillantes et/ou criminelles ayant des difficultés à propager leurs idéologies ou à vendre leurs produits. Plusieurs groupes d'utilisateurs malveillants comme des spammeurs, des revendeurs d'informations volées (carte bancaire) ou des apologistes au jihad retiennent l'attention de plusieurs études ces dernières décennies [1, 2, 3, 4, 5].

La majorité des méthodes de classification automatique ne sont pas nécessairement adaptées à l'analyse de données provenant de réseaux sociaux [6, 7, 8, 10, 11]. Ces dernières ont été conçues et validées sur des réseaux qui sont souvent caractérisés par un seul type d'interaction [12, 6]. Ces interactions peuvent représenter la communication entre nœuds, des relations familiales, professionnelles ou, simplement, la proximité géographique [13].

L'objectif de cette thèse est de développer des méthodes de détection de communautés virulentes qui propagent du malware ou qui effectuent des activités illicites, comme le phishing ou l'apologie au jihad. Ces communautés sont considérées comme des anomalies dans les réseaux sociaux étant donné qu'ils sont minoritaires et que les objectifs qui régissent leurs comportements diffèrent du reste des utilisateurs. Cette thèse propose des algorithmes de classification semi-supervisés, non supervisés ou supervisés qui séparent les anomalies du reste des utilisateurs.

La première approche proposée est semi-supervisée et consiste à détecter des utilisateurs malveillant en interconnectant différents blogs et réseaux sociaux en un réseau social multimodal. Nous avons découvert que plusieurs utilisateurs malveillant font peu attention à dissimuler leurs traces en publiant leurs contenus dans des réseaux ayant une faible sécurité et un faible auditoire. La classification sur ces réseaux est beaucoup plus efficace que sur les réseaux avec un grand nombre d'utilisateurs, qui généralement implémentent des mesures de sécurité. Aussi,

les utilisateurs malveillant redirigent les utilisateurs de réseaux sociaux vers leurs pages personnelles ou laissent des informations leur permettant d'être contactés. Ces traces laissées sur différents réseaux sociaux permettent d'interconnecter ces derniers, et de détecter ces utilisateurs sur les réseaux où la classification est plus difficile en utilisant le contenu qu'ils ont publiés sur les réseaux et blogs à faible auditoire [7].

La deuxième approche concerne des algorithmes de classification non supervisées. Ces méthodes ont pour objectif de détecter des petites communautés dans les réseaux sociaux numériques. Ces réseaux ont souvent une faible densité et un haut nombre d'interactions avec les leaders des communautés. Des méthodes de détection de communautés se basant sur des mesures de densité peuvent obtenir un faible résultat sur ces réseaux [14]. Nous avons ainsi développé deux méthodes de détection de communautés sur les réseaux sociaux. La première méthode vise à réduire la taille des réseaux en utilisant une mesure de densité locale ayant une faible complexité. Cette dernière minimise ensuite une mesure de distance basée sur les marches aléatoires, qui a une plus grande complexité sur un graphe de taille réduite. Ainsi, la méthode proposée permet d'effectuer une détection de communautés qui utilise la structure du réseau avec une très faible complexité. Nous avons effectué des expérimentations sur des réseaux synthétiques ainsi que des réseaux réels qui ont montré que la méthode proposée permet de détecter des communautés de petites tailles [8]. La seconde méthode de détection de communautés proposée est une méthode de la famille des méthodes de détection de communautés par les leaders. Cette méthode considère qu'un leader peut faire partie de plusieurs communautés. Elle utilise du pattern mining pour détecter des combinaisons de leaders qui définissent des communautés sur des graphes simples, attribués et disjoints. Les nœuds du réseau sont ensuite affectés aux groupes de patterns qui ont été découverts. Nous proposons une méthode permettant de sélectionner un nombre limité de patterns qui couvrent toutes les communautés d'un réseau. Ensuite, nous proposons un algorithme de détection de communautés basé sur ces patterns. Les expérimentations sur des réseaux synthétiques et réels montrent que la méthode proposée permet de détecter des communautés de tailles différentes. Les leaders de ces communautés peuvent ensuite être labellisés manuellement ou par des méthodes statistiques afin de détecter les communautés virulentes.

La troisième approche proposée durant cette thèse concerne des méthodes de classification collective sur les réseaux sociaux multimodaux. Ces méthodes supervisées permettent de construire des modèles à partir des données d'entraînement, et d'utiliser les interactions et attributs dans les données à classifier pour déterminer la classe d'utilisateurs de réseaux sociaux.

La première méthode développée est basée sur le K plus proche voisin qui a été modifié pour la problématique de classification avec classes non équilibrées [10].

La seconde méthode utilise les attributs des nœuds ainsi que les attributs des voisins pour construire un modèle de classification. Elle utilise des marches aléatoires sur un graphe multi-

modal pour identifier un nombre fixe de voisins proches, et fait varier ces derniers dans chaque étape de l'algorithme. Cette approche permet de limiter la propagation d'erreurs introduites par les labels des voisins. Le second objectif de cette méthode est de répondre à l'effet de la distribution des degrés qui suit une loi de puissance, et qui introduit du bruit dans la classification collective [15]. En effet, alors que l'utilisation des voisinages locaux implique que des nœuds populaires sont connectés à un nombre important de nœuds de classes différentes, l'utilisation de voisinages identifiés par des proximités structurelles permet une plus grande diversité de voisins avec un faible couplage entre les différentes classes. Nous avons validé cette méthode sur des réseaux réels, où elle a obtenu un meilleur résultat que les méthodes de classification collective qui utilisent les voisinages locaux des nœuds [11].

La troisième méthode de classification collective utilise un modèle de génération de données synthétiques afin d'améliorer la détection des classes minoritaires. Nous avons ainsi proposé une nouvelle méthode de génération de données synthétiques adaptée aux données sociales. Ensuite, nous avons introduit un algorithme basé sur le boosting qui sélectionne les nœuds à utiliser pour la génération de données synthétiques dans chacune de ses itérations.

Le quatrième algorithme de cette troisième approche utilise les données d'entraînement et de classification (semi-supervisée) pour inférer le thème des interactions sur les réseaux multimodaux. Cet algorithme utilise une méthode de Boosting pour orienter une détection non supervisée de thèmes qui utilise le texte dans les messages ainsi que la distribution des thèmes de l'auteur du message. Ces thèmes sont ensuite utilisés pour générer des attributs de voisinage (attributs structurels et textes des voisins). La méthode proposée a pu améliorer la détection de communautés virulentes sur des réseaux réels tout en maintenant un faible taux de faux positifs.

Cette thèse a ainsi permis le développement de plusieurs méthodes pour la détection d'anomalies sur les réseaux sociaux. La première perspective de ce travail est d'adapter les méthodes non supervisées pour la détection de communautés sur les réseaux multimodaux (reply, mention, retweet, etc.). Ces méthodes ne traitent pour le moment que deux modes d'interaction et de similarité (interaction et similarité du texte publié). La seconde perspective est d'utiliser la dimension temporelle dans la détection de communautés virulentes.

Organisation du manuscrit

Le premier chapitre permet de donner un cadre à notre étude et de préciser notre problématique. Ce chapitre se compose de quatre sections.

Dans les deux premières sections, une présentation des réseaux sociaux ainsi que des réseaux sociaux numériques est fournie. La troisième section concerne la cybercriminalité dans les réseaux sociaux numériques et la dernière section présente un outil de collecte de données sur Twitter développé durant cette thèse [16].

Le chapitre 2 offre un état de l'art sur les différentes méthodes de détection d'anomalies sur les réseaux sociaux. Cette thèse est centrée sur la détection d'utilisateurs malveillant par des méthode de classification. Ainsi, seules les grandes tendances sur les méthodes de détection d'anomalies seront présentées. Ce chapitre sera ainsi centré sur la détection de communautés ainsi que sur la classification avec classes non équilibrées et la classification collective. Ces trois familles de méthodes permettent de détecter des communautés ou des classes d'utilisateurs minoritaires, et ayant un comportement différents de l'ensemble des utilisateurs.

Le chapitre 3 présente une méthode de détection de revendeurs de cartes de crédit qui se base sur l'interconnexion de différents réseaux sociaux et blogs. Ce chapitre commence par proposer une typologie des réseaux sociaux [17], et présente ensuite la méthode de détection semi-supervisée.

Le chapitre 4 de ce mémoire est consacré aux méthodes de détection de communautés :

– Nous présentons dans la première section une méthode de détection de communautés pour les grands graphes dénommée SpongeWalker ;
– La seconde section est consacrée à la détection de communautés avec l'utilisation de pattern mining (WalkMiner).

Le cinquième chapitre concerne la classification collective sur les réseaux sociaux :

– La première partie présente une méthode de classification basée sur le K plus proches voisins pour réseaux avec classes non équilibrées. Cette méthode ne prend en compte que les attributs des nœuds ;
– La seconde section présente une méthode de classification collective basée sur la variation de voisinage ;
– La troisième section présente une méthode de classification basée sur le Boosting qui génère des données synthétiques sur les réseaux sociaux multimodaux ;
– La quatrième section introduit une méthode de classification collective pour les réseaux avec arêtes attribuées.

Une conclusion au mémoire sera ensuite présentée, ainsi que l'orientation des travaux futurs.

Chapitre 1

Réseaux sociaux numériques et cybercriminalité

Résumé du chapitre

Ce chapitre introduit la notion de réseau social et de réseau social numérique. Il poursuit par la présentation de la cybercriminalité sur les réseaux sociaux numériques. Enfin, il présente un outil de collecte et de visualisation de données développé dans le cadre de cette thèse [16].

1.1 Réseaux sociaux

Beaucoup de sociologues considèrent que tout individu est intégré dans des réseaux de relations sociales [18]. Un réseau de relations sociales ou un réseau social est un réseau de connexions entre des individus et des groupes [19]. Ces connexions peuvent être matérielles comme des échanges de biens, ou immatérielles comme la perception de l'autre.

Ainsi, on peut considérer tout groupe d'entités sociales connectées par des liens comme un réseau social. Contrairement aux autres réseaux, les réseaux sociaux exhibent plusieurs propriétés provenant de la nature des entités qui les composent (acteurs sociaux) [20]. Ces entités sont par exemple attirées par des acteurs centraux et ont une tendance à interagir avec ces derniers [13]. Aussi, elles ont tendance à s'organiser en communautés [13, 19]. Il existe des propriétés attribués à ces réseaux sociaux qui ne sont pas nécessairement valides dans d'autres réseaux, comme leur faible diamètre [21].

L'analyse des réseaux sociaux fait appel à des techniques et des méthodes incluses dans le domaine de l'analyse structurelle qui permettent de définir, de décrire et de mesurer les comportements interactifs des acteurs [13]. Ainsi, ces méthodes peuvent quantifier des notions abstraites comme la centralité d'un utilisateur, sa disponibilité ou son amabilité, etc. Dans cette

thèse, nous sommes concernés par l'analyse des réseaux sociaux pour la détection d'entités présentant certaines anomalies.

L'analyse des réseaux sociaux considère les postulats suivants [13] :

- Les acteurs ainsi que leurs actions dans un réseau social sont interdépendants ;
- Les relations entre les acteurs d'un réseau sont des canaux de transfert de "ressources" matérielles ou immatérielles ;
- Le réseau pose des limites sur le champ d'action de ces acteurs et leur offre également des opportunités ;
- La structure d'un réseau est considérée comme un schéma durable dans le temps.

Plusieurs réseaux tels que les réseaux de collaborations scientifiques ou les réseaux sociaux numériques (Facebook, Twitter, LinkedIn, etc.) peuvent être considérés comme des réseaux sociaux, étant donné qu'ils répondent aux critères ci-dessus. Des individus ou des organisations forgent des relations durables avec d'autres individus et organisations en utilisant ces réseaux. Ces relations permettent un transfert de connaissances, tendances, perceptions, etc. Ainsi, des méthodes permettant d'étudier les réseaux sociaux sont utilisées pour modéliser et analyser les réseaux sociaux numériques avec un objectif d'identifier des anomalies dans ces derniers.

1.2 Réseaux sociaux numériques

La forte hausse du nombre d'utilisateurs de réseaux sociaux numériques s'est également accompagnée par l'augmentation de travaux scientifiques centrés sur ces derniers. Cela rend nécessaire de fournir une définition de ces derniers qui puisse les séparer des autres plateformes Web permettant l'interaction entre utilisateurs (ex. blogs, forums, etc).

Boyd et Ellison [22] définissent les réseaux sociaux numériques (RSN) comme des services Web qui permettent à des individus de 1) construire un profil public ou semi-public 2) articuler une liste identifiant les utilisateurs avec lesquels ils partagent une connexion et 3) visualiser et traverser leurs listes de connexions ainsi que celles d'autres utilisateurs. Cette thèse va utiliser la définition ci-dessus étant donné qu'elle permet de séparer les réseaux sociaux d'autres services Web comme les blogs (qui ne permettent pas de construire une liste d'utilisateurs). Des exemples de réseaux sociaux populaires sont : Facebook, Twitter, Foursquare, LinkedIn, YouTube, etc.

Le nombre de consultations mensuelles de réseaux sociaux numériques a été estimé en 2016 à plus de 4 milliards [23]. Ainsi, la forte augmentation des utilisateurs de RSN a donné naissance à un marché très lucratif où les différents acteurs se livrent une compétition féroce pour augmenter leurs parts de marché. Cette compétition a conduit plusieurs RSN à offrir de nouveaux services et à s'éloigner du RSN traditionnel afin de se démarquer de leurs compétiteurs. LastFM, par exemple, qui est une radio, offre des services d'e-commerce multimédia. Foursquare s'est

positionné comme un réseau de recommandation basé sur la géolocalisation, et Flickr est centré sur le partage d'images.

Les différences entre ces réseaux ne se délimitent pas seulement aux services offerts. Les RSN ont adopté des modes de navigation ainsi que des politiques de confidentialité différentes afin de répondre aux besoins de leurs utilisateurs et de se démarquer d'autres RSN. Malgré des différences majeures entre les réseaux sociaux, ces derniers permettent la formation de communautés centrées autour de thèmes ou d'individus. Ce principe nous permettra de détecter des utilisateurs virulents, comme des spammeurs, apologistes au jihad et des revendeurs de cartes de crédit.

1.3 Cybercriminalité dans les réseaux sociaux numériques

La fraude peut être définie comme toute action permettant à un individu ou à une organisation d'obtenir un avantage injuste ou d'effectuer une activité illicite [24]. La cybercriminalité est un sous-ensemble d'actions frauduleuses utilisant des appareils de calculs connectés.

Les réseaux sociaux offrent plusieurs avantages à leurs utilisateurs, comme la possibilité de protéger leur identité, de diffuser des messages à un grand nombre d'utilisateurs, de promouvoir leurs talents et produits, etc. Il n'a pas fallu attendre longtemps pour que ces avantages n'attirent des utilisateurs malveillants pouvant bénéficier des plateformes sociales pour effectuer des activités illicites. Il a été estimé par exemple que 11 % des publications sur Twitter en 2010 étaient des SPAMs [3]. En effet, plusieurs activités illicites ont été observées sur les réseaux sociaux, comme l'envoi de SPAMs [25, 26, 27], les attaques d'ingénierie sociale [28], la diffusion de malwares [29] ou la propagation d'idéologies criminelles (ex. apologie au jihad) [30, 31, 5].

Concernant l'envoi de SPAMs, ces plateformes permettent aux spammeurs de diffuser leurs messages à un grand nombre d'utilisateurs avec un très faible coût. Cela est dû aux méthodes d'automatisation des interactions avec les plateformes sociales qui offrent souvent des interfaces de communication [32]. Aussi, étant donné que ces plateformes souhaitent souvent faciliter la création de comptes et permettre aux utilisateurs de rester anonymes, le coût induit par la détection des comptes de spammeurs est minime.

Les attaques d'ingénierie sociale suivent généralement quatre étapes : collecte d'informations, développement de relations avec les victimes, exécution et exploitation [33, 28]. Les trois premières phases sont susceptibles de se produire à l'extérieur d'une organisation, ce qui permet à l'attaquant de profiter de ressources multiples, comme le Web, ainsi que des outils basés sur les réseaux sociaux, comme les mécanismes de recherche et les forums et blogs. De plus, la disponibilité des outils de collecte de pages Web ainsi que les données provenant des réseaux sociaux a réduit les compétences et les efforts nécessaires pour exécuter une attaque automatisée d'ingénierie sociale [34, 35]. Ce développement sans précédent dans les outils de collecte

rend la protection des systèmes d'information contre ces attaques d'une importance vitale, surtout que les réseaux sociaux numériques contiennent aujourd'hui des informations précieuses sur le mode de vie, l'emploi du temps et les connaissances des victimes [36, 37]. Ces informations peuvent aider à exécuter des attaques qui tiennent compte du contexte de la victime, et qui sont particulièrement difficiles à détecter [38]. Dans le cas d'attaques de phishing par exemple, une expérience menée par [39] suggère qu'ils sont quatre fois plus susceptibles de réussir si la victime est sollicitée par quelqu'un qui semble être une connaissance.

Concernant les apologistes au jihad, les réseaux sociaux leur offrent également une plateforme qui leur permet de diffuser leur propagande vers une grande communauté. Ces réseaux peuvent être encore plus intéressants pour les apologistes que pour les spammeurs. La raison est que les spammeurs vont généralement perdre la réputation qu'ils ont créée après la détection de leurs comptes. Les apologistes en revanche peuvent réintégrer leurs communautés en s'identifiant à ces dernières. Leur réputation peut être restaurée après une courte période grâce à l'aide de leurs communautés.

L'Etat Islamique a lancé une campagne médiatique sur Twitter répondant aux objectifs suivants [40] :

- Recruter de nouveaux membres ;
- Terroriser leurs adversaires ;
- Informer leurs sympathisants des derniers développements dans leur région ;
- Projeter une image alternative du groupe afin d'augmenter leur influence.

Les plateformes sociales leur permettent de manipuler les images provenant des régions qu'ils contrôlent. Plusieurs experts ont noté que l'Etat Islamique met en avant des publications montrant des soldats qui jouent avec des animaux ou qui nagent dans les rivières, construisant ainsi l'image d'une vie idyllique afin de recruter de nouveaux membres [40].

Les méthodes utilisées par l'Etat Islamique pour convaincre de nouveaux membres sont diverses. Il existe par exemple des témoignages de femmes américaines ayant rejoint l'Etat Islamique après avoir développé une relation amoureuse en ligne avec des jihadistes au Moyen-Orient [41]. Plusieurs hommes célibataires partent également avec des promesses de mariage et d'emploi. D'autres sont radicalisés et partent spécifiquement pour combattre. L'Etat islamique utilise aussi Internet pour communiquer les trajets à suivre ainsi que les directives pour passer les douanes [41].

L'Etat Islamique publie également les fameuses vidéos de décapitation et d'immolation afin de terroriser leurs potentiels ennemis [40], les poussant à la fuite plutôt qu'au combat. Ainsi, leur campagne médiatique devrait être perçue comme faisant partie d'une stratégie militaire visant à renforcer leur régime.

Il est ainsi impératif de détecter les sympathisants de l'Etat Islamique et autres groupes terroristes afin de réduire leur influence. Néanmoins, il est également important, pour des raisons

de liberté d'expression et de profitabilité des plateformes sociales de réduire les faux positifs afin de ne pas censurer un utilisateur légitime.

Plusieurs méthodes permettant la détection de profils malveillants tirent profit du fait que ces derniers ont des difficultés à devenir centraux dans les réseaux. Ces méthodes nécessitent ainsi le développement de mesures de centralité robustes face aux diverses stratégies utilisées par les nœuds malveillants pour devenir centraux. L'une de ces stratégies est le Link Pharming, qui consiste à se faire suivre par des utilisateurs (légitimes ou malveillants) [26]. Ainsi, une des difficultés majeures liées à la détection de profils atypiques est que ces derniers ne sont pas forcement à la périphérie du réseau. Ghosh et al. [26] ont par exemple trouvé que ces profils se faisaient suivre par ce qu'ils appellent "les capitalistes sociaux". Ces derniers sont des profils fortement connectés qui réciproquent toute interaction. Des exemple de ces profils sont Barack Obama ou d'autres célébrités.

Dans cette thèse, nous supposons que les profils atypiques peuvent être fortement connectés aux réseaux, mais qu'ils exhibent des comportement atypiques liés à 1) leurs interactions avec des utilisateurs qui interagissent peu ensemble (spammers, phishing, etc.) ou 2) leurs interactions avec d'autres utilisateurs atypiques (carding, apologie au jihad, etc.). Les méthodes développées dans cette thèse visent à détecter des utilisateurs malveillant pouvant effectuer des attaques de phishing ou d'ingénierie sociale (revendeurs de cartes de crédit volées), des utilisateurs radicalisés (apologistes au jihad) et de spammeurs.

1.4 Collecte de données sociales

Les réseaux sociaux numériques offrent une variété de données sociales qui sont utilisées dans des domaines tels que le marketing, la cybersécurité, l'information ou le commerce électronique. L'obtention de ces données dépend néanmoins à la fois du réseau utilisé pour les collecter ainsi que de l'usage qui sera fait de ces dernières. Alors que certains réseaux de type Post-it (Twitter) ou d'auto-promotion (ex : YouTube, Google+) ont des politiques souples d'extraction de données, d'autres réseaux fermés ou semi-fermés comme Facebook posent des contraintes liées à l'extraction et l'utilisation de leurs données. En effet, les réseaux fermés ou semi-fermés veulent offrir à leurs utilisateurs un espace protégé qui protège la confidentialité de leurs utilisateurs, ou au moins, qui crée une image de respect de cette dernière [42].

Le réseau Twitter est très flexible concernant la collecte et l'exploitation des données d'utilisateurs. Une des raisons à cela peut être liée au caractère temporel de ces données, qui perdent de leur utilité dans plusieurs domaines d'application après un court intervalle de temps [42].

Twitter offre deux interfaces d'application (API) pour les développeurs souhaitant collecter des données sociales. La première interface se dénomme "l'Interface de Recherche Historique" [43]. Cette API permet d'effectuer des recherches de Tweets qui ont été publiés plus d'une

semaine avant le lancement de la requête. La seconde est "l'Interface Streaming" [44]. Cette dernière permet de collecter jusqu'à 2% des Tweets en utilisant des mots clés ou une géo-localisation.

Dans le cadre de cette thèse, nous avons développé une application nommée Tweetcapt [16] permettant de collecter et de visualiser des données sur Twitter. La figure 1.1 illustre le tableau de bord de cette dernière. Cette application permet de collecter des données sociales sur Twitter à partir de mots clés ou d'un rectangle de coordonnées en utilisant l'Interface Streaming ou l'Interface de Recherche Historique.

Tweetcapt permet également d'enrichir les données sociales collectées en récupérant les pages Web référencées dans les Tweets ainsi que des informations sur les serveurs de redirection utilisés. Ce processus nécessite l'extraction d'URL imbriquées dans des Tweets. Ensuite, Tweetcapt établit une connexion avec chaque serveur de redirection sollicité pour récupérer la page Web. Cela permet de récupérer les URLs des pages cachées par des services de raccourcissement de pages Web : bit.ly, tinyurl, urlshortener, etc. Ainsi, il est possible de créer des graphes à partir des URLs partagées même si ces dernières sont été cachées par des services différents.

L'application développée est illustrée dans la figure 1.1.

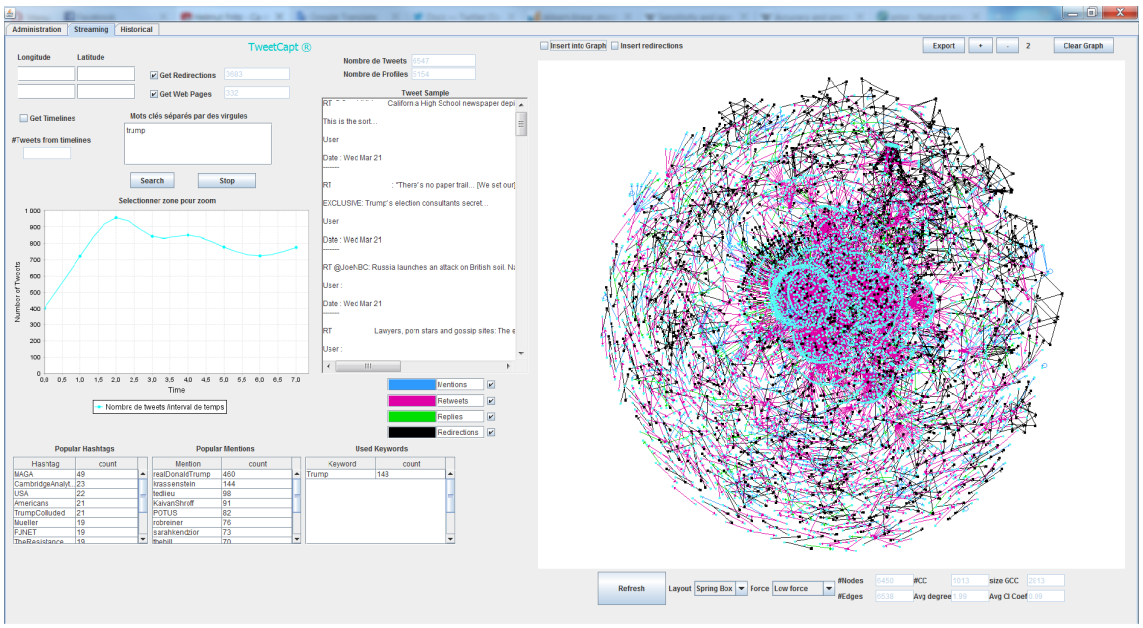

FIGURE 1.1 – Tweetcapt : application de collecte et d'enrichissement de données sociales

L'application permet également de visualiser le graphe multimodal construit à partir des interactions sur Twitter. Ce graphe est composé des quatre modes d'interaction suivants : Retweet, Reply, mention, utilisation de la même URL. L'utilisateur peut sélectionner les modes qu'il souhaite visualiser. Cette visualisation permet d'évaluer la qualité du graphe social collecté en temps réel, et d'orienter une collecte afin d'enrichir ce dernier.

Tweetcapt fournit plusieurs indicateurs sur le graphe social construit à partir d'une collecte. Un des indicateurs calculé est la taille de la plus grande composante connexe. Une grande composante connexe peut suggérer l'existence d'une thématique autour des mots clés utilisés pour la collecte. Un autre indicateur calculés par l'application est le coefficient de clustering moyen. Un haut score sur cet indicateur suggère l'existence de communautés. Enfin, l'application calcule des indicateurs qui décrivent le graphe comme le degré moyen, le nombre d'arêtes et le nombre de nœuds. La figure 1.2 illustre les indicateurs calculés par Tweetcapt.

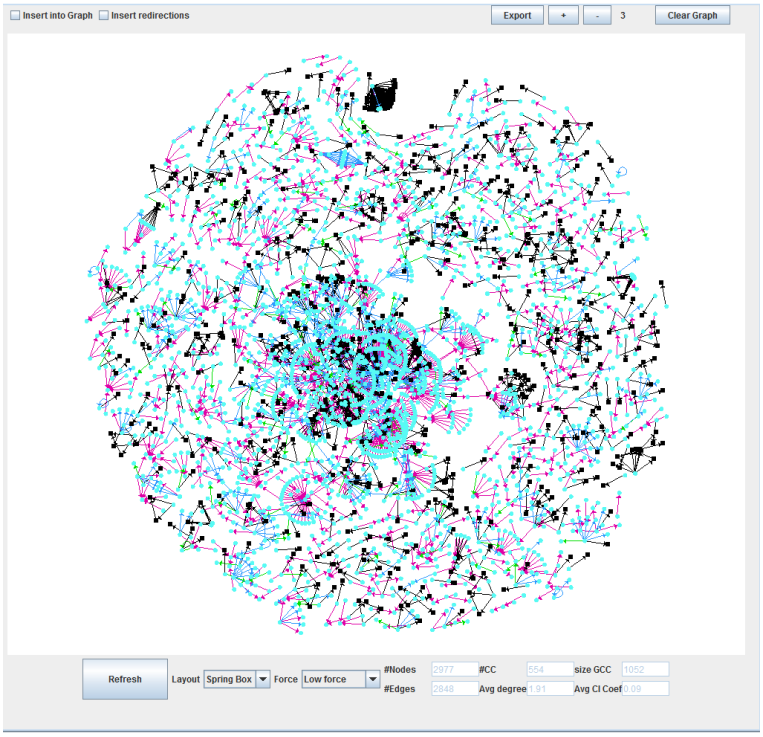

FIGURE 1.2 – Indicateurs pour un graphe

Tweetcapt permet également de suggérer des mots clés, hashtags et utilisateurs populaires à partir d'une requête. L'utilisateur peut ainsi affiner ou étendre une recherche en se basant sur ces informations. L'application met à jour ces indicateurs après un intervalle de temps k fourni comme paramètre. Aussi, la distribution des tweets par intervalle de temps est illustrée et mise à jour par l'application, et un échantillon de Tweets est affiché après chaque intervalle de temps K. La figure 1.3 illustre les indicateurs décrivant une collecte qui sont présentés par Tweetcapt.

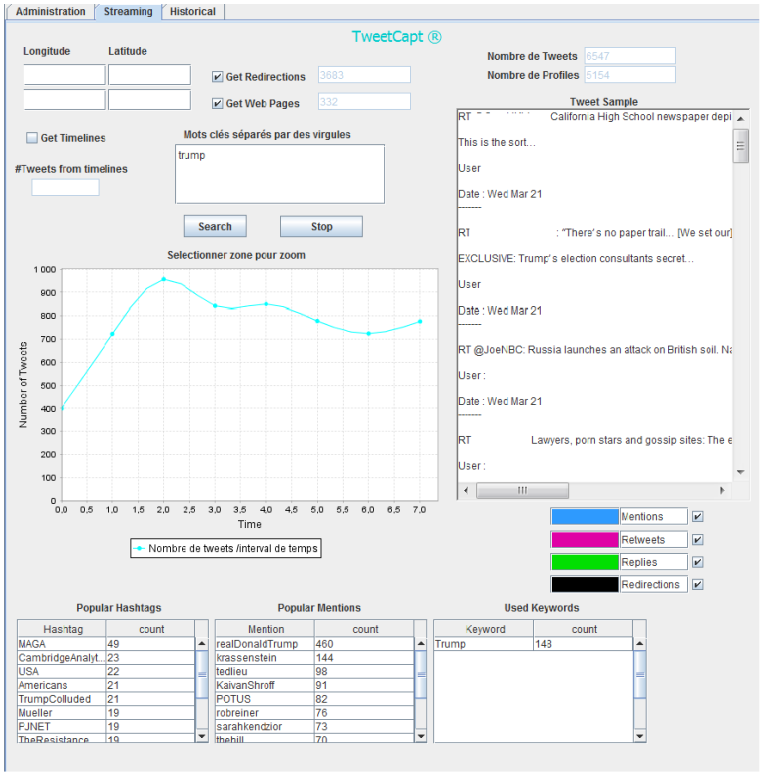

FIGURE 1.3 – Indicateurs pour une collecte

Chapitre 2

Étude bibliographique

2.1 Introduction

La première étape permettant de lutter contre la cybercriminalité consiste à développer du matériel et logiciel qui protège les utilisateurs [24] (Privacy/protection by design). La détection d'utilisateurs et d'actions malveillantes est une seconde couche de protection permettant de perturber les activités de ces derniers. Cette étude bibliographique se consacre aux méthodes de détection automatique d'utilisateurs malveillants sur les réseaux sociaux numériques.

Les réseaux sociaux numériques sont caractérisés par la haute dimensionnalité des données qu'ils contiennent. En effet, ces réseaux sont caractérisés par 1) des données textuelles des messages échangés, 2) des données textuelles et numériques des profils d'utilisateurs ainsi que 3) des données relationnelles (les interactions entre utilisateurs) [45]. Cette richesse permet d'augmenter la pertinence des différentes analyses effectuées dans ces réseaux, mais pose également plusieurs défis scientifiques liés au passage à l'échelle des méthodes d'analyse, la redondance d'attributs, et la présence de bruit dans les attributs et modes d'interaction, pour n'en citer que quelques uns. La grande majorité des modèles pour la représentation des réseaux sociaux considèrent seulement certains modes d'interaction. D'autres modèles, utilisés généralement par des méthodes de classification supervisées ou semi-supervisées (SVM, réseaux de neurones, etc.), se focalisent sur les données textuelles.

La première section de ce chapitre est dédiée à la représentation des données issues de réseaux sociaux. Elle décrit des modèles pour les réseaux sociaux uni-modaux et ensuite des modèles pour les réseaux sociaux multimodaux. La seconde section offre une étude bibliographique sur la détection d'anomalies. Elle commence par offrir une typologie des anomalies dans les graphes. Ensuite, elle présente des méthodes de détection de communautés dans les graphes souvent utilisées pour identifier des anomalies, et enfin, des méthodes de classification supervisées utilisées pour la détection d'anomalies.

2.2 Modélisation des réseaux sociaux

2.2.1 Modélisation des réseaux sociaux uni-modaux

Les réseaux sociaux uni-modaux sont des réseaux sociaux où seulement un type d'interaction est considéré. Ces réseaux sont généralement modélisés par des graphes finis.

Graphes finis

Les graphes finis sont des structures de données utilisées pour représenter les relations entre éléments. Un graphe fini $g = (V, E)$ est défini par les ensembles finis $V = \{n_1, n_2, ..., n_N\}$ et $E = \{e_1, e_2, ..., e_M\}$ tels que V est un ensemble fini d'éléments appelés nœuds (vertex, node) et E est un ensemble fini d'éléments appelés arêtes ou arcs (edges) [46] [47]. Dans le contexte des réseaux sociaux, les nœuds représentent généralement des utilisateurs ou entités de ces réseaux, et les arêtes représentent une relation ou interaction entre ces éléments. Une arête appartenant à l'ensemble E est définie par une paire de nœuds appelés les extrémités de e [46]. On dit également que les nœuds reliés par l'arête e sont adjacents à cette arête. Une arête e reliant les nœuds n_1 et n_2 peut être représentée par ces nœuds comme suit : $e = (n_1, n_2)$ [47]. Une arête reliant un nœud à lui-même est appelée une "boucle".

Un graphe fini peut être représenté par une matrice d'adjacence $A = [a_{i,j}]$, où l'entrée $a_{i,j}$ de la matrice A est égale au poids de l'arête entre le nœud n_i et le nœud n_j si cette dernière existe, ou 0 si l'arête (n_i, n_j) n'existe pas.

Graphes orientés

Un graphe orienté est un graphe dont les arêtes ont une orientation [48] (équation 2.2.1.0.0) :

$$E \subseteq V \times V$$

Dans ce type de graphes, on peut distinguer les origines et les destinations des arêtes. Dans l'arête $e = (n_i, n_j)$ par exemple, l'origine est v_i et la destination est v_j. L'arête e représente donc une relation allant de v_i vers v_j. Les réseaux sociaux sont souvent modélisés par des graphes orientés, parce que les relations ne sont pas toujours réciproques. Un exemple est Twitter, où une personne peut mentionner ou retweeter n'importe qui (président, personnalité, acteur, etc.) sans pour autant que ce dernier interagisse avec elle.

Graphes non orientés

Un graphe non orienté $g = (V, E)$ est un graphe dont les arêtes représentent des relations symétriques [47]. En d'autres termes, g est non orienté si :

$$\forall (n_i, n_j) \in E \implies (n_j, n_i) \in E$$

Une arête reliant deux nœuds dans un graphe non orienté crée simultanément une connexion dans les deux sens. On peut ainsi considérer un graphe non orienté comme un cas particulier du graphe orienté.

Graphes attribués

Un graphe fini peut contenir des attributs qui décrivent des nœuds ainsi que des arêtes. On défini la fonction $Atts_V : V \rightarrow \mathbf{R}^O$ avec O le nombre de dimensions dans les attributs des nœuds. $Atts_V(n_i)$ permet ainsi de retourner les attributs du nœud n_i. On défini un seconde fonction $Atts_E : E \rightarrow \mathbf{R}^Q$ avec Q le nombre de dimensions dans les attributs des arêtes. La fonction $Atts_E(e_i)$ permet de retourner les attributs de l'arête e_i. Un graphe attribué est donc un cas particulier du graphe fini où au moins une des fonctions $Atts_V$ ou $Atts_E$ sont définis.

2.2.2 Modélisation des réseaux sociaux multimodaux

Les réseaux sociaux multimodaux sont des réseaux sociaux permettant à leurs utilisateurs d'établir différents types d'interactions (mention, retweet, reply, like, etc.) avec d'autres utilisateurs sur le réseau.

La structure de données qui est généralement utilisée pour modéliser les réseaux sociaux multimodaux est le graphe multicouches $\mathcal{M} = \{G, \mathcal{C}\}$. Ce dernier est composé de L graphes $G = \{g_\alpha | \alpha \in \{1, ..., L\}\}$ et de $L \cdot (L-1)$ ensembles d'arêtes qui relient chaque paire de graphes $\mathcal{C} = \{E_{\alpha\beta} | \alpha \in \{1, ..., L\} \wedge \beta \in \{1, ..., L\} \wedge \alpha \neq \beta\}$.

Les modèles multicouches qui représentent des interactions sur les réseaux sociaux [49, 50, 51] émanent généralement de la conception sociologique de ces réseaux, qui est un ensemble d'utilisateurs ayant différents types de relations. Les seules connexions inter-couches dans ces modèles sont entre un utilisateur (ou entité) et ses occurrences dans différentes couches. Cette propriété rend la majorité des modèles multicouches non adaptés pour représenter la diversité des connexions inter-couches. Boccaletti et al. [52] décrivent différents modèles qui pourraient être adaptés pour représenter des interactions sur des réseaux multicouches (réseaux de réseaux, par exemple).

Un des modèles les plus courants est le graphe multiplex [52]. Ce dernier est composé de L graphes $\{g_\alpha | \alpha \in \{1, ..., L\}\}$ avec le même ensembles de nœuds $V_1 = V_2 = ... = V_L$. Les arêtes inter-graphes sont seulement autorisées à connecter les apparitions d'un nœud dans les différentes couches $(n_i^\alpha, n_j^\beta) \in \mathcal{C} \wedge \alpha \neq \beta \implies i = j$. Ainsi, un nœud appartenant à une couche α ne peut avoir qu'une seule arête le reliant à la couche β. Cette arête relie nécessairement deux apparitions du même nœud.

Un autre modèle qui peut être utilisé pour la représentation de réseaux multimodaux est le modèle des réseaux interconnectés [52]. Ce modèle est également composé de L graphes et $\{L \cdot (L-1)\}$ ensemble d'arêtes connectant les différents graphes. Un nœud dans la couche α

peut être connecté à plusieurs autres nœuds dans la couche β. Cependant, ce modèle ne distingue pas une arête inter-couches (représentant une interaction) d'une arête reliant l'apparition d'un nœud dans la couche α à son apparition dans la couche β (arête identité).

Un autre modèle qui mérite d'être mentionné est le graphe multidimensionnel [52] $g = \{V, E, D\}$. Ce dernier est composé d'un ensemble de nœuds V, d'un ensemble d'arêtes E et d'un ensemble de labels D décrivant chaque arête. Ainsi, l'entité $(n_i, n_j, d_\alpha) \in E \times D$ représente une arête (n_i, n_j) sur la dimension d_α. Ce modèle basé sur des triplets ne permet pas de modéliser des arêtes identités représentant les apparitions d'un même nœud sur différentes couches. En effet, un seul ensemble V représente les nœuds sur toutes les dimensions. Ainsi, le fait qu'un nœud ne soit pas présent (ex. l'utilisateur qu'il représente n'a pas de compte) sur une dimension ne peut être représenté par le modèle.

Kivala et al. [53] ont proposé un modèle basé sur des couches décrites par plusieurs dimensions. Ils considèrent que chaque dimension (qu'ils appellent un aspect) peut contenir une valeur discrète ou catégorique. Une couche serait décrite par des valeurs discrètes ou catégoriques sur un ensemble de dimensions. Un exemple est un réseau multicouches avec deux dimensions : 1) type de relation 2) temps. On considère que la dimension type de relation peut se voir assigner les valeurs suivantes : ami, collègue, parent et la dimension temps les valeurs suivantes : temps 1, temps 2. Une couche serait identifiée par une valeur sur chaque dimension : collègues, temps. Ce graphe permet de différencier les interactions des arêtes identités par un "aspect" contenant une valeurs binaire (identité, interaction). Ce modèle est utilisé pour représenter les réseaux multimodaux dans ce mémoire.

2.3 Détection d'anomalies dans les graphes

Une anomalie est un concept dont il n'existe pas de définitions unanimes. Ce dernier fait généralement référence à des éléments minoritaires qui n'obéissent pas à la distribution des données. Hawkins [54] définie une anomalie comme une observation qui diffère tellement des autres observations qu'on peut déduire qu'elle a été générée par un mécanisme différent. Comme spécifié par [55], cette définition est très générale et la catégorisation d'un signal comme anomalie dépend souvent du domaine.

Akoglu et al. [55] définissent une anomalie dans un graphe comme des nœuds, arêtes ou sous-structures qui sont 1) rares et 2) différents. Parmi les difficultés liées à l'identification d'anomalies, on peut noter la quantification de la rareté d'une observation et la construction de mesures qui peuvent déterminer si un élément est différent. Nous allons présenter dans cette section une typologie des méthodes de détection d'anomalies dans les graphes. Ensuite, nous allons décrire des méthodes de détection de communautés souvent utilisées pour identifier des anomalies. La troisième partie concerne la détection d'anomalies par des méthodes supervisées.

Enfin, nous allons présenter des mesures utilisés pour quantifier la qualité d'une classification.

2.3.1 Typologie des méthodes de détection d'anomalies dans les graphes

La figure 2.1 offre une typologie des méthodes de détection d'anomalies dans les graphes inspirée de [55]. Nous pouvons diviser les méthodes de détection d'anomalies en deux grands groupes, qui sont les méthodes utilisant des données labellisées (supervisées/ semi supervisées) et les méthodes n'utilisant que des données non labellisées (non supervisées). Dans chacun de ces groupes, il existe des méthodes qui détectent des anomalies dans des graphes simples ainsi que des méthodes qui ont été développées pour des graphes attribués. Concernant les méthodes supervisées, nous n'avons étudié dans cette thèse que les méthode pour graphes attribués. Les attributs dans les graphes peuvent être manifestes (messages publiés, attributs de profil, etc.) ou latents (labels inférés des voisins). Concernant les méthodes non supervisées (pour graphes simples et attribués), nous étudiant le sous ensemble de méthodes qui utilisent la détection de communautés pour inférer des anomalies. Nous considérons seulement les graphes non temporels représentant des captures d'interactions dans un intervalle de temps.

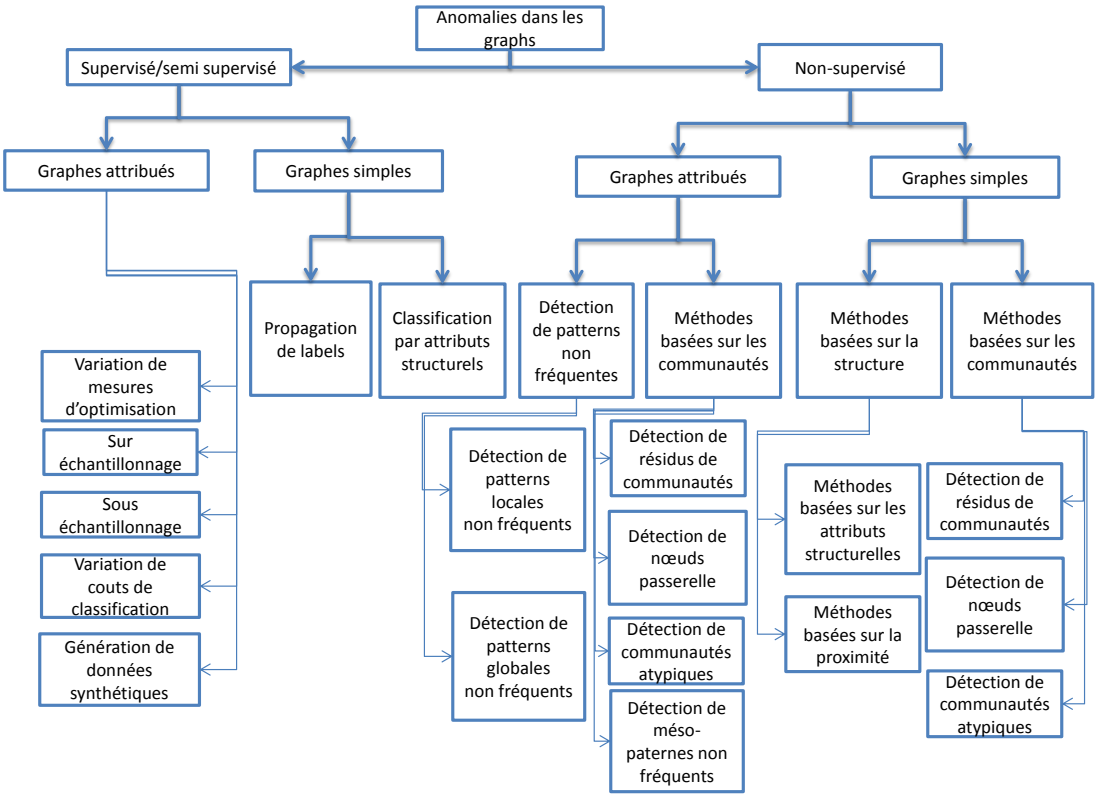

FIGURE 2.1 – Typologie des méthodes de détection d'anomalies dans les graphes

2.3.1.1 Méthodes non-supervisés

Nous pouvons diviser ces méthodes en deux grandes familles :

- Méthodes de détection d'anomalies dans les graphes simples ;
- Méthodes de détection d'anomalies dans les graphes attribués.

Méthodes pour graphes simples : Concernant ces méthodes, il est possible d'identifier deux orientations majeurs, qui sont : les méthodes basées sur la détection de communautés, et les méthodes basées sur la structure du graphe.

Les premières méthodes caractérisent une anomalies selon sa proximité des communautés présentes dans le graphes [56]. Les trois approches majeurs sont la détection de résidus des communautés [57, 58, 59, 60], la détection de nœuds passerelle [61] et la détection de communautés atypiques [62, 63]. La première approche considère que des nœuds présentant des anomalies peuvent difficilement être affectés à une communauté. Ces derniers ont des interactions qui ne suivent pas les distributions des communautés dans le graphe. La seconde approche considère que des nœuds peuvent être identifiés comme anomalie s'il sont des passerelles entre communautés. Enfin, la dernière approche tente d'identifier des communautés entière considérées atypiques.

Les méthodes basées sur la structure peuvent être catégorisées en méthodes basées sur les attributs structurels [64, 58] et les méthodes basées sur les proximités [65, 66, 67]. La première catégorie calcule des attributs structurelles comme le Pagerank, différentes mesure de centralité, le nombre de triades, etc. et utilise des méthodes de détection d'outliers qui prennent en entrée ces attributs structurels afin d'identifier des anomalies. Un exemple est la méthode OddBall [64] qui calcule des attributs structurels à partir des voisins directes d'un nœud afin d'identifier des anomalies. La seconde catégorie utilise l'auto-corrélation entre les différents nœuds d'un graphe pour identifier des anomalies. .

Méthodes pour graphes attribués : Il est possible de diviser les méthodes non supervisés de détection d'anomalies dans les graphes attribués en deux grandes familles qui sont : les méthodes basées sur la détection de communauté [68, 69], et les méthodes basées sur la détection de patterns fréquents [70]. Comme pour les graphes simples, les méthodes basées sur la détection de communautés définissent une anomalies en comparant chaque nœud avec les éléments des communautés qui lui sont proches. Comme pour les graphes simples, differents types d'anomalies peuvent être identifiés. Les méthodes basées sur les patterns fréquents œvrent à caractériser le comportement "normal" qui est suivi par la majorité des nœuds et considère que les nœuds qui n'obéissent pas à ce comportement sont des anomalies.

2.3.1.2 Méthodes supervisées/semi supervisées

Les méthodes de classification supervisées séparent généralement les nœuds d'un graphe en deux classes qui sont : la classe normale (négative) et la classe atypique (positive). Il est possible de diviser ces méthodes en deux grandes familles :

- Méthodes pour graphes simples ;
- Méthodes pour graphes attribués.

Méthodes pour graphes simples : Ces méthodes ont été développées pour partitionner des graphes simples en plusieurs classes (souvent deux). Ces méthodes n'utilise que la structure des graphes pour partitionner ces derniers. Nous pouvons identifier deux tendances concernant la classification supervisée de graphes simples, qui sont : la propagation de labels [71, 72] et la classification par attributs structurels [73]. Les premières méthodes propagent les labels des nœuds de départ en calculant des mesures de proximité qui utilisent la structure du graphe. Les secondes méthodes calculent des attributs structurels à partir de la structure du graphes, et génèrent ensuite des modèles de classification qui prennent en compte les attributs calculés.

Méthodes pour graphes attribués : Ces méthodes utilisent souvent les attributs manifestes et latents (ex. labels prédits) des nœuds en plus des attributs de leurs voisins. Nous pouvons identifier cinq grandes tendances parmi ces méthodes, qui sont :

- L'optimisation de mesures de qualité de partitions sensibles à la nature non équilibrée des classes : plusieurs méthodes tentent d'optimiser des critères qui prennent en compte la nature non équilibrée de la classification (ex. AUC, critères joints, ect.) [74, 75].
- Variation de couts d'erreurs de classification : Certaines méthode augmentent le cout lié aux erreurs de classification dans la classe positive afin d'augmenter la rappel de ces méthodes [76, 77, 78, 79]. Il faut noter que plusieurs méthode qui varient les couts d'erreurs de classification peuvent également être considérés comme des méthodes qui optimisent des mesures de qualité de partitions sensibles à la nature non équilibrée des classes.
- Méthodes basées sur le sur-échantillonnage de la classe minoritaire : Ces méthodes sur-échantillonnent les éléments de la classe minoritaire afin d'augmenter le rappel de cette dernière [80, 81, 82].
- Méthodes basées le sous-échantillonnage de la classe majoritaire : Ces méthodes sous échantillonnent la classe majoritaire afin de baisser la contribution de ces derniers dans la génération du modèle [77, 83, 84, 85].
- Génération de données synthétique : ces méthodes équilibrent les classes en combinant des éléments de la classe minoritaire afin de générer des données synthétiques [86, 87, 88].

– Méthodes mono-classe : Ces dernière n'utilisent qu'une seule classe pour générer un modèle de classification qui élimine les outliers [89, 90].

2.3.2 Détection de communautés dans les graphes

Cette dernière décennie a connu un grand développement dans les méthodes de détection de communautés dans les graphes. En effet, la qualité des communautés trouvées par ces algorithmes s'est largement améliorée, et surtout, leurs capacités de passage à l'échelle ont augmentées, permettant ainsi d'analyser des graphes à plusieurs millions de nœuds.

Malgré le développement rapide qu'a connu la détection de communautés dans les graphes, la majorité des méthodes concernent la détection de communautés dans les graphes uni-modaux, non orientés et non attribués. Le problème de détection de communautés dans les graphes orientés, multimodaux ou attribués est plus difficile à résoudre. Ces trois problèmes souffrent du manque de compromis sur ce que signifie une communauté dans ces graphes ainsi que sur la difficulté à adapter des concepts tels que la densité sur ces types de graphes.

Dans cette section, nous allons commencer par présenter certaines orientations prises par les méthodes de détection de communautés dans les graphes uni-modaux. La deuxième partie concerne la détection de communautés dans les graphes multimodaux. Ensuite, nous allons décrire des méthodes de détection de communautés dans les graphes attribués. Enfin, nous allons présenter une discussion sur les différentes méthode de détection de communautés dans les graphes.

2.3.2.1 Détection de communautés dans les graphes uni-modaux

Les premières définitions de communautés concernent des graphes uni-modaux non orientés. Chaque nœud représente un acteur et une arête représente une interaction, ou une proximité (ex. graphe de parenté). Les communautés de nœuds dans ces graphes sont définies en termes de densité, et on estime que la densité des interactions entre les nœuds d'une même communauté doit être plus élevée que la densité des interactions entre les nœuds de communautés différentes. Cette conception de communautés était déjà populaire dans le domaine du clustering (regroupement) de données spatiales et géographiques.

Une des premières mesures utilisée pour décrire la qualité d'une partition est la densité intra-communautés. Dans un graphe non orienté, cette densité est définie par l'équation 2.1, avec a_{ij} représentant l'entrée ij de la matrice d'adjacence, et c est la communauté des nœuds n_i et n_j dans une partition P ($P = \{c_1, c_2, ...c_r\}$).

$$D_{intra} = \sum_{c \in P} \frac{\sum_{i,j \in c^2} a_{ij}}{|c|} \tag{2.1}$$

L'optimisation de cette mesure permet de trouver des communautés proches des vérités terrain, mais elle nécessite de connaître le nombre de communautés à l'avance, car elle ne permet pas de comparer des partitions de tailles différentes.

Une mesure qui est devenue très populaire pour l'évaluation de partitions de graphes est la modularité [91]. Newman et Girvan [92] ont proposé un algorithme de détection de communautés qui supprime des arêtes en utilisant une mesure basée sur la centralité d'intermédiatité (betweeness). Ils ont ensuite proposé la modularité comme critère d'arrêt de leur algorithme. Étant donné que la modularité leur permet de trouver les meilleures partitions de graphes, Newman a ensuite proposé le premier algorithme de détection de communautés qui optimise cette mesure [91]. La modularité est définie par l'équation 2.2, où $deg : E \rightarrow \mathbb{N}$ retourne le degré d'un nœud et $\delta : c^2 \rightarrow \{0, 1\}$ est la fonction de Kronecker qui retourne 1 si ses entrées sont sont égales et 0 dans le cas contraire.

$$Q = \frac{1}{2M} \sum_i \sum_j a_{ij} - \frac{deg(i)deg(j)}{2M} \tag{2.2}$$

La modularité compare le nombre d'arêtes existantes entre deux nœuds (ou communautés) et une hypothèse nulle construite à partir des degrés des nœuds en question. Cette mesure permet de construire des partitions de graphes proches des vérités terrain sans connaître le nombre de communautés à l'avance.

Blondel et al. [93] ont proposé une heuristique qui optimise la modularité pour le partitionnement de grands graphes. Cet algorithme, appelé "algorithme de Louvain", se compose de deux étapes qui se répètent jusqu'à ce que la modularité soit maximisée :

1. Pour chaque nœud, placez-le dans la communauté voisine qui maximise la modularité. Répétez ce processus jusqu'à ce qu'il n'y ait plus de gain de modularité.
2. Fusionnez les nœuds appartenant à la même communauté où le poids des arêtes entre communautés est additionné, et les arêtes entre les nœuds d'une même communauté deviennent des boucles.

Cet algorithme est largement adopté, car il permet de construire des partitions de haute qualité, et a été utilisé sur des réseaux contenant des millions de nœuds. La complexité de cette algorithme ne peut être bornée qu'empiriquement, et elle a été évaluée à O(Nlog(N)) où N est le nombre de nœuds.

Fortunato et Berthelemy [94] ont analysé cette mesure de modularité et ont mis en évidence un problème de résolution où des communautés proches des vérités terrain, voire des cliques se font agréger dans des communautés plus grandes avec une qualité plus faible lorsque la taille du réseau augmente. Ainsi, la mesure de modularité favorise des communautés d'une certaine taille.

Pour résoudre ce problème, plusieurs mesures basées sur la densité ont été proposées où l'hypothèse nulle n'est pas influencée par la taille du graphe. Une des mesures les plus utilisées est la modularité densité (modularity density) proposée par Li et al. [95]. Ils définissent la mesure $L(V_1, V_2) = \sum_{i \in V_1, j \in V_2} a_{ij}$ tel que V_c est l'ensemble des nœuds appartenant à la communauté c. Ainsi, $L(V_1, V_1) = \sum_{i,j \in V_1^2} a_{ij}$ et $L(V_1, \bar{V_1}) = \sum_{i \in V_1, j \in \bar{V_1}} a_{ij}$. Si on considère qu'un graphe est partitionné en r communautés $\{V_1, V_2, ..., V_r\}$, la modularité densité est définie par l'équation 2.3, avec P la partition du graphe ($P = \{c_1, c_2, ..., c_r\}$).

$$D = \sum_{c \in P} \frac{L(V_c, V_c) - L(V_c, \bar{V_c})}{|V_c|} \tag{2.3}$$

Pons et Latapy [96] proposent une méthode de détection de communautés appelée Walktrap. Ils considèrent un cluster comme un ensemble de nœuds pouvant atteindre ou facilement être atteint par le même ensemble de nœuds dans des marches aléatoires de petites tailles. Par conséquent, les communautés détectées par cet algorithme ne sont pas nécessairement fortement connectées, bien que cette méthode puisse également détecter des communautés à forte densité (étant donné que des nœuds fortement connectés ont une grande probabilité d'atteindre et d'être atteints par le même ensemble de nœuds dans des marches aléatoires).

Leur méthode calcule des marches aléatoires (t = 4 par défaut) sur le réseau. Elle calcule ensuite une mesure de distance entre chaque paire de nœuds en utilisant la matrice stochastique utilisée pour effectuer des marches sur le réseau. Cette matrice est obtenu en normalisant la matrice d'adjacence par la somme des lignes de la matrice (équation 2.4).

$$m_{i,j} = \frac{a_{i,j}}{\sum_{k=0}^{k=|V|} a_{i,k}} \tag{2.4}$$

La distance proposée est décrite par l'équation 2.5 où deg est une fonction qui retourne le degré entrant d'un nœud.

$$D_{i,j} = \frac{\sum_{k=0}^{k=|V|} |m_{i,k} - m_{j,k}|}{deg(k)} \tag{2.5}$$

Il faut noter que Walktrap rend un dendrogramme qui est une hiérarchie de communautés. Pons et Latapy [96] utilisent ensuite la modularité pour retourner la meilleure communauté.

Subelj et al. [97] ont effectué une analyse quantitative et qualitative de plusieurs algorithmes de détection de communautés sur différents réseaux de citations. Dans une de leurs analyses, ils ont effectué un clustering en K-means (11 classes) d'algorithmes de détection de communautés en utilisant les partitions qu'ils ont générées comme attributs. Les résultats de ce clustering ont montré que Walktrap [96] et Louvain [98] n'étaient pas dans la même classe. Cela souligne le fait que l'utilisation de la modularité comme critère de sélection d'une partition dans un

dendrogramme (stratégie de Walktrap) peut offrir des résultats très différents de ceux obtenus en optimisant la modularité (stratégie de Louvain).

Yakoubi et al. [99] proposent une méthode nommée Licod qui appartient à la famille des algorithmes basés sur les représentants. L'objectif de cette méthode est d'utiliser des représentants afin de réduire la complexité des algorithmes afin d'analyser des réseaux de très grandes tailles. La première phase de Licod est d'identifier des nœuds qui sont des représentants convenables des différentes communautés qui peuvent exister sur le réseau analysé. Ils considèrent que des représentants convenables auraient des degrés plus élevés que leurs voisins, sans nécessairement avoir les degrés les plus élevés du réseau. Cela permet de trouver des représentants des petites communautés avec un degré moyen plus bas. Licod, ensuite, regroupe les représentants des différentes communautés si la proportion de leurs voisins en commun par rapport à l'union de leurs voisins est inférieur à un seuil (similarité de Jaccard). La prochaine étape est d'assigner chaque nœud à une communauté en calculant une mesure de proximité.

Bonchi et al. [100] proposent une méthode inspirée du clustering de corrélation. Cette méthode définit une fonction objectif qui croît avec la dissimilarité entre les nœuds appartenant aux mêmes communautés et la similarité des nœuds appartenant à des communautés différentes. Bonchi et al. étendent le modèle de clustering de corrélation pour prendre en compte la possibilité qu'un nœud appartienne à plusieurs communautés. Leur fonction objectif est décrite par l'équation 2.6.

$$C_{occ} = \sum_{u,v \in V \times V} |H(l(u), l(v)) - s(u, v)| \tag{2.6}$$

$s : V^2 \rightarrow [0, 1]$ est une fonction de similarité qui prend comme entrée deux nœuds, et rend une valeur [0,1] indiquant le poids du lien (s'il existe) entre deux nœuds. La fonction l retourne l'ensemble des communautés dont fait partie un nœud et la fonction H calcule la similarité entre deux ensembles de communautés (deux labellisation $[l_1^\alpha, l_2^\alpha, ..., l_m^\alpha]$ et $[l_1^\beta, l_2^\beta, ..., l_n^\beta]$). Les auteurs proposent d'utiliser une similarité de Jaccard pour la fonction H. Ils ont présenté des résultats satisfaisants (NMI > 0.9) sur un ensemble de données benchmark de UCI qu'ils ont modélisé en utilisant des graphes.

Une autre orientation pour la détection de communautés sur des graphes uni-modaux se base sur la propogation de labels. Ces méthodes s'inspire de l'algorithmes de propagation de labels introduit par [101]. La méthode commence par affecter chaque nœud à un label (communauté) différent, puis se base sur le consensus des voisins des nœuds afin de leur affecter itérativement de nouveaux labels. Cette méthode à une compléxité linéaire et permet ainsi de traiter des graphes de grande taille. Néanmoins, les partitions obtenus peuvent se dégrader quand les réseaux sont bruités.

La majorité des méthodes de détection de communautés non recouvrantes et même recou-

vrantes s'est focalisé sur la détection de communautés dans les graphes non orientés. Une des raisons de cette tendance est que la majorité des bases de données de graphes étaient non orientées. Elles représentent généralement des interactions symétriques ou des similarités (liens de parenté, proximité géographique, etc.). Cette tendance a changé avec l'évolution des réseaux sociaux numériques, qui ont mis à disposition des bases de données massives d'interactions asymétriques.

Dans les graphes orientés, la détection de communautés est un problème qui est intrinsèquement plus difficile à résoudre en comparaison avec la détection de communautés dans les graphes non orientés. On peut identifier les difficultés suivantes liées à la détection de communautés dans les graphes orientés :

- La représentation asymétrique des graphes (matrices d'adjacences asymétriques) rend l'analyse spectrale beaucoup plus coûteuse [102] ;
- Plusieurs concepts utilisés pour la détection de communautés dans les graphes non orientés ne sont pas facilement extensibles aux graphes orientés (ex. densité) [103] ;
- Il n'existe pas encore de définition acceptée par la communauté scientifique sur ce qu'est une communauté dans un graphe orienté [103] (bien que des tendances commencent à émerger).

Une des premières tendances de la détection de communautés dans les réseaux orientés est d'adapter des algorithmes de détection de communautés conçus pour des réseaux non orientés.

Ainsi, une des premières définitions de communautés sur un réseau orienté est une adaptation de la définition de communauté sur les réseaux non orientés. Elle considère qu'une communauté est un ensemble de nœuds ayant une plus forte densité d'interactions avec des nœuds de leurs communautés qu'avec des nœuds n'appartenant pas à leurs communautés.

Pour détecter ce type de communautés, il a fallu d'abord adapter la densité pour les réseaux orientés. La mesure de densité qui a le plus capturé l'intérêt des chercheurs est naturellement la modularité.

Leicht et al. [104] proposent une mesure de modularité pour les réseaux orientés où l'hypothèse nulle tient compte de l'orientation des arêtes. Leur méthode introduit un biais où l'orientation d'une arête a un effet important sur les valeurs de modularité. Cette mesure est décrite par l'équation 2.7, où M est le nombre d'arêtes et indeg et outdeg sont des fonctions qui retournent respectivement le degré entrant et le degré sortant d'un nœud.

$$Q = \frac{1}{2M} \sum_i \sum_j a_{ij} - \frac{indeg(i)outdeg(j)}{2M} \tag{2.7}$$

Yang et al. [105] proposent un algorithme qui optimise la modularité pour les réseaux orientés décrits par [104]. Ils commencent par une première optimisation de la modularité à l'aide d'un modèle de programmation linéaire en nombres entiers qui est rapide mais qui pourrait

être pris dans un optimum local. Pour éviter ces optimum locaux, ils exécutent plusieurs instances de leur algorithme avec des points de départ différents et choisissent le résultat avec la plus grande modularité (optimisation with restart). L'étape suivante consiste à d'utiliser un programme non linéaire en nombres entiers qui est plus difficile à résoudre, mais qui permet d'obtenir de meilleurs résultats. Ils utilisent la meilleure solution obtenue dans l'étape précédente comme point de départ à leur programme non linéaire.

Kim et al. [106] proposent une autre adaptation de la modularité dans les graphes orientés en utilisant des marches aléatoires. L'équation 2.8 définit la modularité dans des graphes orientés où π_i fait référence à la distribution stationnaire du nœud n_i. La distribution stationnaire est obtenu à partir de la matrice stochastique (équation 2.4). Cette dernière reflète les ratios de temps passé dans chaque nœud après un nombre infini de marches aléatoires. Ainsi, elle peut être utilisée comme une mesure de centralité d'un nœud. Le plus cette probabilité est élevée pour un nœud n_i, le plus il y a de chemins qui connectent ce nœud au reste du graphe.

$l_{ij} = \pi_i \cdot m_{ij}$ capture la probabilité d'atterrir sur le nœud n_i lors d'une marche aléatoire puis d'aller à n_j à partir de n_i. m_{ij} représente donc la valeur de la matrice stochastique construite à partir de la matrice d'adjacence normalisée par la somme des lignes. m_{ij} permet de capturer la probabilité d'aller au nœud n_j à partir du nœud n_i. La mesure proposée par les auteurs compare donc la probabilité d'atterrir dans un nœud puis d'aller aux nœuds de la même communauté avec l'hypothèse nulle $\pi_i\pi_j$ qui considère l'indépendance des probabilités d'atterrir sur chaque nœud.

$$Q = \sum_i \sum_j (l_{ij} - \pi_i\pi_j)\delta(c_i, c_j) \tag{2.8}$$

Une autre famille de méthodes de détection de communautés dans les réseaux orientés transforme un graphe orienté en graphe non orienté, puis utilise un algorithme de détection de communautés sur les graphes non orientés. L'objectif de cette transformation, souvent appelée symétrisation, est de capturer les patterns spécifiques à l'orientation des nœuds dans une mesure de similarité. Les méthodes proposées dans cette thèse utilisent la symétrisation de graphes non orientés.

Satuluri et al. [107] décrivent diverses méthodes de symétrisation de graphes orientés. L'une de ces méthodes consiste à de construire une matrice d'adjacence à partir des co-citations et des couplages bibliographiques (bibliographic coupling). Une des symétrisations qu'ils évaluent est décrite dans l'équation 2.9. Cette première méthode créé une arête entre deux nœuds de poids égal à la somme des co-citations et de leur couplage bibliographique. Un algorithme de détection de communautés dans les graphes non orientés peut ensuite être utilisé sur le graphe symétrique avec la matrice U. D'après les expérimentations effectuées dans le cadre de cette thèse, l'utilisation de la modularité comme critère de qualité dans un graphe symétrisé par le

couplage bibliographique regroupe des communautés qui sont faiblement connectées. Cela est dû au fait que la matrice U créée beaucoup plus de liens que ceux contenus dans le graphe de départ. En effet, des nœuds sont connectés dans le graphe symétrique si ces derniers co-citent ou sont co-cités par un même nœud.

$$U = A \cdot A^T + A^T \cdot A \tag{2.9}$$

Une autre procédure de symétrisation décrite par [107], qui est également basée sur les co-citations et le couplage biométrique, normalise la matrice U par 1) le degré des nœuds cités et des nœuds qui co-citent et 2) par le degré des nœuds citant et des nœuds co-cités. L'équation 2.10 décrit la mesure de similarité qu'ils proposent. Cette dernière augmente également la densité des interactions qui conduit à regrouper des communautés distinctes lorsque la modularité est utilisée comme critère de partitionnement. Les paramètres α et β sont initialisés à 0.5, ce qui signifie que la mesure est normalisée par la racine carrée des degrés.

$$B_d(i,j) = \frac{1}{outdeg(i)^{\alpha} outdeg(j)^{\alpha}} \cdot \sum_k \frac{a_{i,k} a_{j,k}}{indeg(k)^{\beta}} \tag{2.10}$$

Les auteurs ont expérimenté leur méthode de symétrisation sur plusieurs corpus, incluant un corpus de Wikipédia, de Cora, de Flickr et de Livejournal. Leurs meilleurs résultats ont été obtenus par la symétrisation décrite dans l'équation 2.10. Il faut noter qu'ils ont supprimé des liens dans leurs graphes en utilisant un seuil sans donner d'indication sur leur choix.

Klymko et al. [108] ont proposé une méthode de symétrisation de graphes orientés qui prend en compte le nombre de chemins de longueur 3 entre les nœuds ainsi que du nombre d'arêtes bi-directionnelles dans ces chemins. Leur pondération est décrite par l'équation 2.11

- r0 : indique s'il existe un chemin orienté de taille 3 entre nœud i et j ;
- r1 : indique s'il existe un chemin orienté de taille 3 entre nœud i et j avec une arête bi-directionnelle ;
- r2 : indique s'il existe un chemin orienté de taille 3 entre nœud i et j avec deux arêtes bi-directionnelles ;
- r3 : indique s'il existe un chemin orienté de taille 3 entre nœud i et j avec trois arêtes bi-directionnelles.

$$w_{i,j} = max(4r4, 3r2, 3r1, 2r0, 1) \tag{2.11}$$

Ils ont évalué leur méthode sur plusieurs corpus incluant des graphes construits à partir de recommandations d'Amazon, des graphes d'Epinion ainsi qu'un corpus Wikipédia. Ils ont trouvé que leur méthode de symétrisation améliorait la qualité des partitions dans certains corpus, mais leurs résultats étaient assez similaires à la symétrisation classique $w_{ij} = a_{ij} + a_{ji} + a_{ij} \cdot a_{ji}$.

Une autre tendance dans la détection de communautés dans les réseaux orientés est de considérer qu'une communauté est un ensemble de nœuds où un surfeur aléatoire peut être piégé.

Rosvall et al. [109] proposent un algorithme qui permet de trouver de telles communautés. Leur méthode commence par générer des marches aléatoires sur le réseau social. Ils optimisent ensuite la taille d'encodage minimal de ces marches. Cette dernière est obtenue en affectant à la même communauté les nœuds qui apparaissent souvent ensemble dans les marches aléatoires.

Zhou et al. [110] proposent une méthode de classification semi-supervisée dans des graphes orientés qui peut être utilisée dans un mode non-supervisé pour la détection de communautés. Ils proposent d'optimiser une fonction objectif (basée sur une mesure de densité intra-communauté) ayant une composante de lissage. Cette dernière capture la similarité entre les nœuds d'une communauté en considérant les nœuds autorité qu'ils citent ainsi que les nœuds hub qui les citent. Comme Walktrap [96], cette méthode génère un dendrogramme.

Agreste et al. [111] ont comparé plusieurs algorithmes de détection de communautés sur des réseaux orientés. Ils ont utilisé un générateur de réseaux synthétiques proposé par [112] ainsi que des réseaux réels. Leur plus haut score de NMI a été obtenu par Walktrap sur le réseau synthétique, même si ce dernier modélise un réseau orienté par un graphe non orienté. Les auteurs n'avaient pas de vérité terrain concernant les réseaux réels. Une étude de la taille des communautés générées a montré que Walktrap et Infomap généraient des communautés de tailles similaires. Les expérimentations effectuées par [111] montrent que, souvent, prendre en compte l'orientation des arêtes n'améliore pas les partitions générées. Ces améliorations ou dégradations dépendent surtout du type de réseau analysé. Dans des réseaux de citation, par exemple, prendre en compte l'orientation améliore souvent les partitions.

2.3.2.2 Détection de communautés dans les réseaux hétérogènes

Les réseaux sociaux sont souvent caractérisés par plusieurs types d'interaction (mention, retweet, reply, similarité d'URL, etc.). Aussi, les nœuds ainsi que les arêtes dans ces réseaux peuvent être décrites par des attributs numériques, catégoriques et textuels. Alors que ces réseaux sont souvent clairsemés et que la structure du graphe peut ne pas être suffisante pour partitionner un réseau, l'utilisation des attributs ainsi que des types d'interaction offre des informations utiles pour le partitionnement de ces derniers.

Cette section présente certains travaux sélectionnés sur la détection de communautés sur les réseaux multimodaux et/ou attribués. Pour un état de l'art complet, veuillez consulter [113].

Détection de communautés dans les réseaux multimodaux Berlingerio et al. [114] présentent une méthode basée sur la fouille de patterns qu'ils ont nommés Abacus. Ils commencent par utiliser une méthode de détection de communautés pour les réseaux uni-modaux sur chaque mode d'un réseau. L'étape suivante est d'utiliser un algorithme de fouille de patterns fréquentes

en considérant des nœuds comme des transactions. Les communautés multimodales sont obtenues en utilisant les communautés détectées par algorithmes de détection de communautés uni-modaux comme des items. Les communautés multimodales sont ainsi les ensembles de patterns fréquentes (frequent itemsets) retrouvées par l'algorithme de fouille de patterns.

De Domenico et al. [115] proposent une méthode basée sur la compression de flux permettant d'identifier les flux multimodaux. Cette méthode peut être considérée comme une extension d'Infomap [109] aux réseaux multimodaux où les connexions inter-couches sont pondérées par un paramètre r. Comme Infomap, ils génèrent des marches aléatoires, mais sur un graphe multimodal. Ensuite, ils compressent la taille de description minimale de ces marches en assignant les nœuds qui sont souvent inclus dans les mêmes marches à la même communauté. Cela permet de réduire le nombre de bits nécessaires pour représenter ces nœuds.

Tang et al. [116] proposent un algorithme basé sur l'optimisation de la modularité pour les réseaux multimodaux. Ils commencent par optimiser la modularité sur les différents modes du réseau, et suppriment les communautés qui ont un faible impact sur la modularité dans chaque mode. Ils utilisent ensuite une représentation en décomposition de valeurs singulières (SVD) afin de capturer les corrélations des communautés entre différents modes, puis K-means pour trouver les communautés multimodales. Ainsi, cette méthode nécessite de spécifier le nombre de communautés.

Kuncheva et Montana [12] introduisent une méthode de détection de communautés dans les réseaux multimodaux qui s'inspire de l'algorithme de Walktrap. Comme Walktrap, leur méthode commence par calculer des marches aléatoires, mais sur le réseau multimodal. Les liens inter-couches sont pondérés par la similarité des voisinages ainsi que par un paramètre utilisateur. Leur méthode calcule ensuite une distance similaire à celle de Walktrap pour regrouper les nœuds. Finalement, la meilleure partition est sélectionnée en utilisant une modularité pour les réseaux multimodaux.

Boden et al. [117] proposent un algorithme qui identifie les δ-quasi cliques sur les différents modes d'un réseau multimodal. Ainsi ils définissent une communauté comme un ensemble de nœuds dont chacun d'entre eux a au moins δ interactions avec des nœuds de sa communauté. Une communauté multimodale est donc composée de δ-quasi cliques sur chaque mode. Cette méthode nécessite de fournir le paramètre δ.

Brodka et al. [118] introduisent un coefficient de clustering multimodal qui permet de quantifier la proportion des voisins communs entre deux nœuds sur les différents modes du réseau. Ils identifient ensuite les communautés en supprimant itérativement les liens entre les nœuds avec le coefficient de clustering le plus bas. Leur méthode permet ainsi de générer un dendrogramme.

Hmimida et Kanawati [119] introduisent un algorithme qu'ils nomment mux-licod, qui est une extension de Licod [99] pour les réseaux multiplex. Ils utilisent un degré pour les réseaux multiplex afin d'identifier les représentants des communautés. Ce degré multiplex est basé sur

l'entropie des degrés sur chaque mode, augmentant ainsi les degrés des nœuds qui interagissent sur plusieurs modes. Cette méthode adapte également les mesures de similarités utilisées dans Licod pour regrouper les représentants et pour affecter les nœuds aux groupes de représentants afin de favoriser les nœuds qui interagissent sur plusieurs modes.

Détection de communautés dans les réseaux attribués Lin et al. [120] proposent une méthode de détection de communautés dans des réseaux attribués avec nœuds manquants. Ils utilisent les régions denses du graphe pour inférer une distance globale basée sur les attributs. Ils utilisent ensuite les liens ainsi que cette distance entre les nœuds pour détecter des communautés.

Ruggieri et al. [121] proposent une méthode appelé "SENC" qui détecte des communautés recouvrantes dans des graphes attribués. Ils commencent d'abord par définir des bornes supérieures et inférieures pour chaque communauté. Les bornes inférieures peuvent être des quasi-cliques par exemple. Ils utilisent ensuite un algorithme itératif qui estime la distribution des attributs d'une communauté en utilisant l'algorithme de maximisation d'espérance (EM) [122], puis en assignant des nœuds à cette dernière.

Li et al. [69] présentent une méthode qui utilise des représentants afin de détecter des communautés ayant des attributs à haute dimensionalité. Ils commencent par utiliser l'algorithme Apriori [123] pour trouver des représentants pour chaque communauté. La prochaine étape est d'utiliser l'Allocation de Dirichlet latente (LDA) [124] comme méthode de sélection d'attributs seulement sur les représentants. Cela permet de réduire la complexité du LDA. L'étape suivante est d'entrainer un SVM en utilisant les représentants ainsi que des éléments échantillonnés. Enfin, ils ajoutent des nœuds à une communauté si le SVM les accepte.

Horizonte et al. [125] présentent une méthode de pattern mining nommée "SCPM". Cette dernière détecte les $\delta-$quasi cliques ainsi que les itemsets fréquents. Ils définissent un critère nommé la "corrélation structurelle" qui calcule la probabilité qu'un élément appartienne à un $\delta-$quasi clique s'il a un itemset fréquent S. L'algorithme retourne les $\delta-$quasi cliques qui ont un score de corrélation structurelle qui dépasse un certain seuil.

Ruan et al. [126] proposent une méthode qui créé des arêtes basées sur la similarité d'attributs. Ils commencent par calculer la similarité cosinus entre chaque paire d'éléments. Ils utilisent ensuite l'union des arêtes topologiques et des arêtes créées par la similarité cosinus en liant les K-plus proches voisins. Le graphe obtenu est ensuite rendu moins dense en échantillonnant les arêtes qui ont le plus de poids. Enfin, un algorithme de détection de communautés sur les graphes est utilisé.

Wang et al. [127] modélisent les réseaux attribués par un réseau multimodal composé de liens entre utilisateurs, ainsi que de liens entre des utilisateurs et des attributs catégoriques. Ils utilisent ensuite une méthode de détection de communautés sur le graphe obtenu.

Zhou et al. [128] présentent une méthode de détection de communautés pour des réseaux composés de deux types de nœuds (auteurs, documents). Ils commencent par générer une partition des documents en utilisant une variation de K-means pour les attributs à haute dimensionalité [129]. L'étape suivante est de regrouper des auteurs aux clusters de documents en utilisant leurs liens. Enfin, ils utilisent l'optimisation de modularité afin de regrouper les clusters de documents et d'auteurs en modélisant les co-citations de documents comme des liens entre auteurs.

La méthode introduite par Steinhaeuser et Chawla [130] pondère les liens par un score qu'ils appellent "NAS" en utilisant les attributs des nœuds. Pour cela, les auteurs identifient deux types d'attributs :

- Les attributs catégoriques : le poids d'une arrête est incrémenté pour chaque attribut en commun entre les nœuds incidents.
- Les attributs continus : le poids est incrémenté par la distance euclidienne normalisée pour chaque attribut en commun entre les nœuds incidents.

Les auteurs ont conduit des expérimentations sur des réseaux téléphoniques, et ont trouvé que l'utilisation du NAS permet d'obtenir des partitions avec une modularité élevée. Ces résultats ont été obtenus malgré l'utilisation d'un algorithme très simple pour la détection de communautés : Les nœuds ayant une arête dont le poids excède un seuil sont regroupés dans la même communauté. Les auteurs ont néanmoins testé leur méthode sur un seul réseau, rendant leurs résultats difficiles à généraliser. Une de leurs hypothèses est que l'homophilie doit nécessairement permettre aux attributs d'améliorer les partitions. Cela n'est pas nécessairement vrai à cause du bruit dans certains attributs ainsi que de la présence d'hétérophilie dans certains réseaux (ex. réseaux de célibataires). Aussi, leur méthode est difficilement applicable quand la dimensionalité augmente (curse of dimensionalité).

Zhang et Al. [131] pondèrent les attributs des nœuds en fonction des communautés dont ils sont proches. Ils optimisent ainsi une fonction objectif qui détermine itérativement l'affectation de nœuds aux communautés, et la pondération des poids des attributs pour chaque communauté (méthode de co-clustering).

Dang et Viennet [132] proposent une méthode qui optimise un critère joint qui mesure la similarité topologique ainsi que la similarité d'attributs. Ils utilisent la modularité [92] pour évaluer la proximité topologique. Concernant la similarité d'attributs, ils proposent d'utiliser l'inverse de la distance euclidienne pour les attributs continus, une méthode basée sur la correspondance pour les attributs discrets et une méthode basée sur le tf-idf pour les attributs textuels. Ils introduisent un algorithme basé sur Louvain [93] pour optimiser leur critère joint.

Combe et Largeron [133] proposent une méthode basée sur l'algorithme de Louvain [93] qui optimise un critère joint composé de la modularité ainsi qu'une mesure pour la similarité des attributs de communautés d'une partition. La seconde mesure est nommée "la modularité

basée sur l'inertie". Elle mesure la qualité d'attributs numériques seulement. Leur second critère peut néanmoins conduire des nœuds ayant un modèle nul trop élevé (outliers) à rejoindre une communauté.

Cruz et Al. [134] proposent une méthode basée sur l'algorithme de Louvain ainsi que sur la minimisation de l'entropie par un algorithme de la classe Monte-Carlo. Ils ont modifié l'algorithme de Louvain en ajoutant une étape après l'optimisation de modularité (avant l'agrégation de nœuds). Dans cette nouvelle étape, ils changent la communauté de certains nœuds afin de minimiser l'entropie de sous-ensembles d'attributs (appelés "point de vus"). Le fait d'effectuer une optimisation d'entropie indépendamment de la topologie pourrait néanmoins dégrader les résultats obtenus par Louvain (dans l'étape précédente), contrairement aux méthodes qui utilisent un critère joint.

2.3.2.3 Discussion

La majorité des méthodes de détection de communautés ont été développé pour des graphes uni-modaux. La détection de communautés dans les graphes orientés ainsi que dans les graphes hétérogènes (attribués ou multi-modaux) est relativement sous-étudiée.

La densité d'interaction est l'un des critères les plus utilisés pour la segmentation de graphes uni-modaux. Plusieurs mesures s'inspirant de ce critère ont été développés. La mesure la plus utilisée est la modularité qui est soit optimisée pour segmenter un graphe ou utilisée comme critère d'arrêt par des algorithmes qui génèrent des dendrogrammes.

Alors que la modularité permet souvent de retrouver des partitions de qualité élevée, il existe plusieurs cas où cette dernière n'est adaptée au partitionnement de graphes. En effet, la modularité souffre d'un problème de résolution. Le modèle nul de cette dernière croît en fonction de la taille du graphe. Ainsi, de petites communautés peuvent être combinés dans des communautés voisines si la taille du graphe est importante. Plusieurs mesures ont été proposés pour répondre à ce problème [95, 135, 136, 137]. Néanmoins, la modularité reste la mesure la plus utilisée. En effet, cette dernière permet souvent d'obtenir des partitions de haute qualité pour une classe étendue de graphes. Aussi, elle a été largement étudiée par la communauté scientifique contrairement aux nouvelles mesures qui ont été proposés.

Une alternative à l'optimisation de mesures de densité est le calcul de distances entre nœuds basées sur la structure du graphe [96, 100, 138, 110]. Ces méthodes utilisent des calculs souvent couteux qui peuvent impliquer des marches aléatoires, analyse des co-citations ou autres calculs qui génèrent des patterns qui caractérisent les différentes communautés. Avec l'augmentation de la taille des graphes, l'utilisation de mesures couteuses peut néanmoins être prohibitive. Pour cela, il peut être intéressant de calculer ces mesures sur un graphe de taille réduite qui reflète le graphe initiale. Ce graphe peut être obtenu soit par l'échantillonnage [139, 140, 141] ou par la fusion de nœuds ayant une haute densité d'interaction [142, 8]. Il faut néanmoins noter que

l'échantillonage peut conduire à une perte d'information utile à la détection de communauté. La fusion, en contre partie, peut engendré des communautés qui sont le produit de la mesure de densité utilisée pour fusionner des nœuds, si le taille du graphe est fortement réduite. Ainsi, les mesures structurelles auront peu d'effet sur le partitionnement du graphe. Dans le cas contraire, la taille du graphe obtenu par la fusion se rapproche de la taille du graphe initiale (peu de nœuds sont fusionnés), engendrant une faible réduction de la complexité après la fusion. Ainsi, il est difficile de trouver un compromis sur le nombre de nœuds et d'arêtes fusionnés. Pour cela, nous proposons dans cette thèse d'utiliser à la fois, la fusion de nœuds ayant une forte densité d'interaction, ainsi que l'utilisation de représentant pour le calcul de distances structurelles. Cette méthode ne fusionne que les nœuds densément connectés, mais permet également de réduire la complexité de l'algorithme en utilisant des représentants de communautés.

Un deuxième problème dont souffre la modularité est lié à la distribution en loi de puissance des degrés dans certains graphes. Cela est notamment le cas des réseaux sociaux numériques de type Twitter (réseaux Post-it) où on peut identifier deux types de nœuds. Le premier type est composé de leaders qui sont souvent sollicités (mentionné, retweeté, etc.). Ces nœuds sont appelés les "représentants" [14]. Le second type est composé des membres de communautés ayant un faible degré qui sollicitent souvent les représentants. Ces derniers sont appelés les "auxiliaires" [14]. Dans le cas de réseaux à distribution en loi de puissance des degrés, la modularité n'agrège pas certaines communautés dans des communautés voisines (problème de résolution), mais divise des communautés en plusieurs petits segments [14]. Pour ce type de réseaux, des méthodes basés sur les représentants permettent d'obtenir de meilleurs partitions [99, 14]. Néanmoins, il faut noter que même si l'on souhaite obtenir une partition dur du graphe, les représentants peuvent faire partie de plusieurs communautés. La raison pour cela est que ces nœuds sont tellement actifs sur le réseau qu'ils peuvent avoir un rôle déterminant dans plus d'une communauté. Un exemple est Bill Gates qui peut être central dans une communauté qui discute les produits informatiques ainsi que dans une communauté qui discute les maladies en Afrique centrale. Pour cela, nous proposons dans cette thèse de détecter des patterns de leaders qui définissent des communautés. Ces patterns sont composés d'un ensemble de nœuds centraux dans la communauté, qui sont souvent sollicités par les membres de cette dernière. Ensuite, ces patterns sont utilisés pour obtenir une partition dure du réseau.

Concernant les réseau orientés, la détection de communauté est une tache qui est plus difficile. En effet, les méthodes spectrales sont plus couteuses [102], et plusieurs concepts comme la densité ne sont pas clairement définis [103]. Les méthodes de détection de communautés dans ces derniers ont ainsi initialement cherché à adapter une mesure de densité qui prend en compte l'orientation des arêtes [105, 106]. Un principe déterminant derrière ces mesures, qu'elles prennent en compte les structures locales ou globales, est qu'une arête bi-directionnelle a plus de poids dans l'affectation de nœuds a une communauté que deux arêtes ayant la même

orientation. Cette première tendance a pour objectif d'utiliser des algorithmes de détection de communautés pour réseaux uni-modaux qui sont largement étudiés pour détecter des communautés dans des réseaux orientés. Une seconde tendance est de symétriser des réseaux orientés [107, 108]. L'idée est d'utiliser des structures locales comme la co-citation, le nombre de cycles, d'arêtes bi-directionnelles et ainsi de suite, pour transformer un réseau orienté en réseau non-orienté dont les arêtes sont pondérés par les structures locales. Cette seconde tendance permet également d'utiliser des algorithmes développés pour des réseaux non orientés. Notez que les deux premières tendances n'améliorent pas forcement la qualité de partitions obtenus [111]. En effet, les expérimentations effectués dans cette thèse suggèrent que les apports obtenus par la symétrisation de graphes ou par l'utilisation d'une mesure de densité pour réseaux orientés varient fortement d'un graphe à l'autre, et que dans plusieurs cas, ils dégradent la qualité de partitions. La troisième tendance dans la détection de communautés dans les réseaux orientés est de capturer les flux d'information dans les réseau afin de partitionner ces derniers [109]. Ces mesures nécessitent des réseaux fortement connectés comme des réseaux de citations afin de produire des partitions de qualité.

Il est possible d'identifier deux grandes tendances parmi les méthodes de détection de communautés dans les réseaux multimodaux. La première tendance considère que ces réseaux sont composés d'un nombre limité de modes [115, 116, 12, 12, 119]. Ces méthodes sont généralement des adaptation d'algorithmes de détection de communautés dans des réseaux uni-modaux. La majorité de ces méthodes nécessitent que l'utilisateur fournisses un paramètre afin de contrôler le nombre de modes par communautés. En effet, dans certaines application, il est important que les communautés soient définis sur un nombre restreint de modes (ex. chaque mode représente une langue, un domaine scientifique, etc.). La seconde tendance considère que les réseaux peuvent contenir un nombre très élevée de modes [114, 117]. Ces méthodes utilisent du pattern mining afin de détecter des ensembles de modes qui sont souvent utilisés par les mêmes groupes de nœuds. Les méthodes de détection de communautés dans les réseaux multi-modaux souffrent du manque de données labellisés qui permettent de valider leurs approches, étant donné que cette problématique est relativement récente comparée à la détection de communautés dans les réseaux uni-modaux.

Concernant la détection de communautés dans les réseaux attribués, il est possible d'identifier quatre tendances.

1. La première est de pondérer les attributs en utilisant la structure du graphe [120, 69, 131]. Ces méthode utilisent les interactions afin de pondérer les attributs des nœuds, augmentant le poids des attributs qui confirment la dimension structurelle. Ces méthodes sont souvent utilisés dans des graphes ayant des attributs à haute dimension (ex. attributs textuels) et une dimension structurelle de bonne qualité (ex. graphes de co-citations, d'URL, etc.).
2. La seconde méthode pondère les arêtes par les attributs [130, 129]. Contrairement à la

méthode précédente, cette dernière considère que les attributs ont une grande importance dans le partitionnement d'un graphe. Dans le cadre des réseaux sociaux, une grande partie des messages publiés n'ont pas de dimension structurelle. Ces derniers peuvent être utiles dans la segmentation du réseau, surtout que les le nombre d'interactions dans ces réseaux est souvent très faible. Pour cela, nous proposons dans cette thèse une méthode qui pondère les arêtes par les attributs.

3. La troisième tendance consiste à de détecter des corrélations d'attributs et de nœuds qui définissent des communautés [125, 134]. Ces méthodes peuvent donner un avantage aux attributs ou aux liens en fonction de la difficulté lié à la détection de corrélations sur chacune de ces dimensions. Ces méthodes peuvent néanmoins avoir beaucoup de mal a détecter des corrélations en haute dimension avec un réseau clairsemé.

4. La dernière tendance utilise un critère joint qui prend en compte la structure du graphe ainsi que les attributs [132, 133]. Ces méthodes permettent de contrôler l'effet des attributs et de la structure du graphe sur la génération de partitions par un paramètre utilisateur. Ils nécessitent néanmoins une connaissance des réseaux étant donné que le paramétrage de ces méthodes n'est pas intuitif. Aussi, il faut noter que dans plusieurs cas, l'importance des attributs change d'une région du réseau à une autre. Plusieurs méthodes dans les trois tendances précédentes adaptent les poids des attributs en fonction de leur capacité à segmenter le réseau. Ainsi, l'importance des attributs dans chaque communauté n'est pas la même. Les méthode utilisant un critère joint pondère de manière uniforme les poids des attributs dans tout le réseau, ce qui peut détériorer les partitions obtenus.

2.3.3 Classification supervisée

La classification automatique peut être définie comme l'affectation d'objets appartenant à un ensemble de données à des classes [83]. Un objet peut être identifié par le couple (x,y), où $x \in \mathcal{X}$ est un vecteur d'attributs attaché à l'objet et $y \in Y = \{y_1, y_2, ..., y_N / y_i \in \mathbb{N}\}$ est la classe (ou label) de l'objet. L'espace $\mathcal{X}$ est obtenu à partir du produit des domaines d'attributs continus et catégoriques d'un objet x.

L'objectif de la classification est de construire un ensemble de règles $\mathcal{R}$ qui permettent de prédire la classe y d'un élément à partir de son vecteur d'attributs x ($\mathcal{R} : \mathcal{X} \rightarrow Y$). Dans le cadre de la classification supervisée, ces règles $\mathcal{R}$ sont construites à partir d'un ensemble labellisé $T = (x_1, y_1), (x_2, y_2), ..(x_m, y_m)$ (ensemble d'entrainement). Ils sont ensuite utilisés pour déterminer les classes d'autres jeux de données suivant la même distribution que le jeu ayant servi à construire les règles $\mathcal{R}$.

La conception d'algorithmes de classification est un domaine de recherche vibrant ayant connu un grand essor ces deux dernières décennies. Une des problématiques liées à la classifi-

cation supervisée est de négocier un compromis entre la réduction du biais et de la variance du modèle. Le biais est lié au bruit et aux particularités d'un jeu de données. En effet, si un algorithme de classification génère des règles complexes permettant de classifier tous les éléments d'un jeu de données, il est possible que ces règles ne se généralisent pas à d'autres jeux de données dont les classes sont inconnus (sur-apprentissage). La réduction de la variance permet à un jeu de données de classifier l'ensemble labellisé utilisé pour construire le modèle. Si les règles de classification sont trop simples pour obtenir des résultats satisfaisants sur le jeu labellisé, il est peu probable que ces dernières obtiennent des résultats satisfaisants sur des jeux de données à classifier.

Une autre problématique liée à la classification est la dérive de concept (concept drift) [143]. La dérive de concept est la modification de la relation entre les attributs et les classes dans le temps [143]. Cette problématique peut être observée dans des réseaux sociaux numériques, où de nouveaux termes apparaissent et d'autres perdent leur popularité avec le temps. Cette problématique est souvent résolue par l'apprentissage d'un nouveau modèle [143], nécessitant de labelliser un nouvel ensemble de données. L'utilisation d'ensembles de classifieurs entrainés sur des données échantillonnées et/ou des attributs échantillonnés permet de réduire l'effet de la dérive de concept, étant donné que plusieurs combinaisons d'attributs sont utilisés pour obtenir une prédiction finale.

Dans cette section, nous allons commencer par présenter des méthodes de classification pour classes non équilibrées. Nous allons ensuite présenter des méthodes de classification collective pour éléments connectées. Enfin, des métriques pour l'évaluation de la qualité de classifications seront détaillées.

2.3.3.1 Classification pour classes non équilibrées

Avant de présenter des travaux sur la classification collective dans les graphes, nous souhaitons tout d'abord introduire le sous-domaine de la classification avec classes non équilibrées. La classification avec classes non équilibrées a été considérée comme une des problématiques les plus difficiles dans le data mining [144]. La majorité des méthodes pour la classification collective ne traite pas séparément le cas de la classification avec classes non équilibrées. La tendance est d'effectuer un traitement en amont de la classification qui permet de résoudre le problème des classes non équilibrées (sous-échantillonnage, sur-échantillonnage, pondération, échantillons synthétiques, etc.), puis d'effectuer une classification collective.

Plusieurs méthodes utilisées pour la classification se dégradent quand les classes sont non équilibrées. La régression logistique par exemple sous-estime les probabilités conditionnelles des classes minoritaires [145, 83]. L'analyse de discriminante linéaire souffre aussi de plusieurs difficultés lorsque les classes sont non équilibrées. En effet, l'estimation de la matrice de covariance est dominée par la dispersion dans la classe majoritaire [83]. Lawrence et al. [146]

ont démontré que les garanties théoriques offertes par les réseaux de neurones ne tiennent pas dans un environnement avec les classes non équilibrées, et Akbani et al. [147] ont démontré que les SVM offrent de faibles résultats quand les classes sont non équilibrées. Concernant le K plus proche voisin, [76] a décrit leur biais vers la classe majoritaire pour la classification de documents textuels.

Il est possible d'identifier trois grandes difficultés liées à la classification non équilibrée [148]. La première est liée à l'utilisation de la majorité des algorithmes, de métriques tel le taux de bien classés qui donnent un avantage à la classe majoritaire. La seconde difficulté est due à la nature des règles de classification qui permettent de séparer les différentes classes. Dans une classification binaire par exemple, la classe minoritaire nécessite souvent des règles hautement spécialisées, alors que la classe majoritaire requiert souvent des règles assez générales pour éviter un sur-apprentissage. Ce phénomène est accentué par le fait que la classe minoritaire est souvent composée de petits groupes disjoints [149]. Cela nécessite de générer des règles de classification complexes pour chacun de ces groupes. La troisième difficulté est liée au bruit qui dégrade les résultats de la classe minoritaire. En effet, la génération de règles spécialisées peut conduire à l'apprentissage de bruit et affaiblir la capacité de généralisation du modèle. Il faut noter que la difficulté liée à la classification avec classes non équilibrées est surtout rencontrée quand la tâche de classification est non triviale [83]. L'approche la plus souvent utilisée pour les tâches de classification simples est d'augmenter le coût lié à la classification d'éléments de la classe minoritaire. Il faut également noter les difficultés héritées de la classification équilibrée comme le chevauchement de classes ou la négociation du compromis : réduction de variance et réduction de biais.

Les approches aux problèmes de classification avec classes non équilibrées peuvent être divisées en deux grandes catégories [79, 148, 83]. La première catégorie consiste à traiter le problème en amont de la tâche de classification. Ces approches consistent généralement à sous-échantillonner la classe majoritaire [77], sur-échantillonner la classe minoritaire [80], générer des données synthétiques [86] pour augmenter la classe minoritaire, ou une combinaison de ces méthodes. La seconde catégorie consiste à trouver une solution au niveau algorithmique. Ces méthodes essaient d'ajuster des algorithmes de classification afin de prendre en compte la nature non équilibrée des classes. Une de ces approches est d'utiliser des coûts différents aux bien classés dans la fonction objectif optimisée par l'algorithme de classification [150]. Un des problèmes liés à cette approche est la difficulté de choisir les coûts [83]. Une autre classe d'approches n'utilise que la classe positive pour l'apprentissage [90]. Ces méthodes classifient ensuite les données en fonction des règles apprises à partir de la classe minoritaire [74, 89, 90].

On peut identifier les problèmes suivants associés au sous-échantillonnage ou sur-échantillonnage [79] :

– La distribution optimale est inconnue à l'avance, rendant difficile de choisir la taille des

échantillons à utiliser ;
- Une mauvaise stratégie d'échantillonnage peut conduire 1) à une perte d'éléments importants pour la construction du modèle (sous-échantillonnage) ou 2) au sur-apprentissage de la classe minoritaire (sur-échantillonnage) ;
- Un coût additionnel pour le traitement et l'analyse des données est souvent associé aux méthodes de sur-échantillonnage.

Plusieurs méthodes de classification pour classes non équilibrées ont été introduites durant ces deux dernières décennies [80, 151, 152, 153, 154, 155, 156, 157].

Une des approches les plus courantes est dénommée SMOTE et a été introduite par Chawla et al. [86]. Cette méthode est basée sur la génération de données synthétiques pour agrandir la classe minoritaire. Ces données synthétiques sont générées en variant les attributs de la classe minoritaire. Pour générer un élément synthétique, cette méthode commence par sélectionner un élément de la classe minoritaire. Ensuite, un de ses K plus proches voisins est échantillonné (de la même classe). Enfin, le vecteur d'attributs de l'élément en question est rapproché de son voisin par un facteur entre 0 et 1 (également échantillonné). L'idée générale derrière cette méthode est d'éviter un sur-apprentissage en étendant la classe minoritaire dans l'espace de la classe majoritaire par la génération de données synthétiques. Une des difficultés rencontrée par cette méthode est liée à la possibilité de dégradation de l'espace de la classe majoritaire par des échantillons synthétiques.

Menardi et al. [83] proposent une méthode basée sur le sous-échantillonnage de la classe majoritaire ainsi que sur la génération synthétique de données. Leur méthode permet d'éviter un sur-apprentissage en étendant la classe minoritaire par des données synthétiques, mais également d'éviter la dégradation de la classe majoritaire en limitant la génération de données synthétiques. Une des difficultés liée à leur méthode est de fixer le taux de sous-échantillonnage.

Batista et al. [87] proposent une méthode nommée Smote-ENN, qui permet de répondre aux problèmes liés à SMOTE en effectuant un nettoyage des données synthétiques générées. Ils éliminent tout élément synthétique dont deux de ses trois plus proches voisins sont dans la classe opposée. Luengo et al. [148] ont effectué une comparaison de Smote-ENN [87], et du sous-échantillonnage par un algorithme évolutif [158] et ont obtenu de meilleurs résultats pour la seconde méthode sur un grand ensemble de jeux de données. Il faut aussi noter que, sur plusieurs jeux, les deux méthodes de pré-traitement de données ont dégradé la performance d'algorithmes de classification. Saez et al. [159] présentent une méthode similaire basée sur un filtre contre le bruit.

Certaines méthodes utilisent des méta-heuristiques d'optimisation afin de détecter des classes minoritaires.

Lee et Lee [81] proposent par exemple une méthode basée sur ANOVA, Fuzzy C-means (FCM) et l'optimisation par la recherche bactérienne (BFO). ANOVA est utilisé comme mé-

thode de sélection d'attributs et FCM identifie des classes recouvrantes initiale. Cette méthode utilise ainsi le FCM pour détecter un nombre élevé de clusters en supposant que certains de ces clusters définissent la classe minoritaire. Enfin, BFO utilise les clusters retournées par le FCM pour générer une partition des données.

Lee et al. [80] proposent une méthode basée sur le sur-échantillonnage, les arbres de décision et l'optimisation par essaims particulaires (PSO). La première étape de leur approche est de sur-échantillonner la classe minoritaire. Le PSO détermine ensuite le nombre d'arbres ainsi que le paramètre M (attributs par arbre).

D'autres méthodes adaptent le KNN afin de réduire son biais vers la classe majoritaire. En effet, quand les classes sont hautement non équilibrées, les outliers de la classe majoritaire peuvent conduire à une mauvaise classification de plusieurs éléments.

Zhang et al. [151] introduisent une méthode basée sur les K plus proches voisins (KNN) pour la classification d'attributs textuels. Ils introduisent une mesure composée de la plus longue séquence en commun entre deux éléments. Cette mesure privilégie les documents ayant une forte similarité, et réduit ainsi l'effet du bruit dans la classe majoritaire.

Padmaja et al. [77] proposent une méthode basée sur le KNN inversé (RKNN). Ils utilisent SMOTE afin d'augmenter la classe minoritaire et un sous-échantillonnage réduisant la taille de la classe majoritaire. Ils utilisent ensuite RKNN pour éliminer les outliers.

Shang et al. [75] introduisent une version modifiée de l'index Gini prenant en compte les poids des classes. Il utilisent ensuite un KNN avec leur nouvelle mesure.

Tan [76] proposent un KNN pondéré par les poids des classes. La classe minoritaire obtient ainsi plus de poids. Néanmoins, ils ne tentent pas d'éliminer les outliers.

Gao et al. [160] introduisent une approche composée d'un KNN et d'un SVM pour classifier des documents textuels en chinois. En premier lieu, ils représentent les textes dans un nouvel espace dimensionnel en utilisant un kernel. Ils utilisent ensuite le KNN pour classifier les documents dans le nouvel espace obtenu par le SVM. Le kernel leur permet de mieux séparer les données améliorant ainsi les résultats du KNN.

Une troisième famille de méthode n'utilise que la classe minoritaire pour construire un modèle. Ces méthodes ont obtenu des résultats encourageants, souvent sur des jeux de données à faible dimensionalité.

Zhang et al. [90] proposent une méthode basée sur l'apprentissage de règles. Ce type de méthodes apprend des règles en utilisant la classe minoritaire seulement. Ils trouvent les règles fréquentes à partir de la classe minoritaire, et les utilisent pour classifier les nouveaux éléments.

Japkowicz et al. [89] entrainent un auto-encodeur sur la classe positive seulement. L'auto-encodeur apprend un modèle lui permettant de compresser et de restituer les données. Les éléments qui ne peuvent pas être correctement décompressés sont considérés comme faisant partie de la classe majoritaire.

Bhowan et al. [74] introduisent une méthode qui optimise la surface sous la courbe (AUC) en utilisant un algorithme génétique multi-objectif. Leur méthode permet d'optimiser à la fois les bien classés de la classe majoritaire ainsi que les bien classés de la classe minoritaire. Ils produisent un front Pareto permettant à l'utilisateur de sélectionner les meilleurs compromis.

Les méthodes d'ensembles sont devenues très populaires pour la classification non équilibrée, étant donné qu'elles permettent souvent de réduire le biais du modèle d'apprentissage généré [85]. Ces méthodes peuvent être catégorisées en deux grandes familles, qui sont le Bagging et le Boosting. La première famille de méthodes apprend des modèles différents sur plusieurs échantillons de données, et combine ensuite ces derniers. La seconde famille est basée sur un algorithme itératif qui sur-échantillonne les données mal classées. La seconde famille de méthodes est plus populaire pour la classification non équilibrée car elle sur-échantillonne également des éléments de la classe majoritaire qui sont difficiles à classifier (la frontière entre les classes). Les deux familles de méthodes permettent d'éviter le sur-apprentissage dû aux classes non équilibrées en combinant plusieurs modèles construits à partir de règles différentes.

Avant de présenter les méthodes basées sur le Boosting, il est important de décrire AdaBoost [161] (algorithme 1). Pour affecter les éléments $x_i \in X$ aux classes appartenant à l'ensemble Y, AdaBoost commence par initialiser une probabilité uniforme pour échantillonner chaque élément. Ensuite, l'algorithme effectue T itérations, avec T un paramètre fourni par l'utilisateur. Durant chaque itération, AdaBoost génère un modèle à partir des éléments échantillonnés. Ensuite, l'erreur ainsi que le paramètre α_t sont calculés. Ce dernier sert à la fois à pondérer l'apport du modèle généré durant l'itération t ainsi que pour générer un échantillon durant l'itération t+1. Si beaucoup d'éléments ont été mal classés par le modèle, l'erreur est grande et la valeur du paramètre α_t baisse. Cela implique une baisse dans le sur-échantillonnage de données mal classées pour corriger le modèle, ainsi qu'une baisse dans le poids du modèle généré. Ainsi, si un modèle génère peu d'erreurs, le poids de l'échantillonnage de ces dernières augmente, permettant à l'algorithme de focaliser sur ces données dans l'étape qui suit. Schapire et Singer ont prouvé que l'erreur d'entrainement d'AdaBoost est bornée par $exp(-2\sum_{t=1}^{T}(0.5-\epsilon_t)^2)$. L'algorithme AdaBoost n'est cependant pas adapté à la classification non équilibrée. En effet, ce dernier échantillonne les éléments bien ou mal classés des différentes classes avec les mêmes probabilités (il optimise le taux de bien classés).

Algorithm 1 ADABOOST

(x_i, y_i) : vecteur de données labellisées avec $y_i \in Y \wedge y_i \in \{-1, +1\}$

T : entier spécifiant le nombre d'itérations

WL : classifieur faible

I : fonction identité retourne 0 si les elements sont égaux et 1 dans le cas contraire.

1: Initialise $D^1(i) = \frac{1}{N}$

2: Pour t ∈ 1..T :

3: Générer le modèle h_t :WL(D^t,X,Y)

4: Calcul de l'erreur $\epsilon_t = \sum D^t \cdot I(h_t(x) \neq y)$

5: Calcul du paramètre $\alpha_t = 0.5.ln(\frac{1-\epsilon_t}{\epsilon_t})$

6: Mettre à jour les poids pour l'échantillonnage $D^{t+1}(i) = \frac{D_i^t exp(-\alpha h_t(x_i)y_i)}{Z_t}$ avec Z_t un facteur de normalisation $Z_t = \sum D_i^t exp(-\alpha h_t(x_i)y_i)$

7: Classification des données : y_i = signe($\sum_{t=1}^{T} \alpha_t h_t(x_i)$)

Sun et al. [79] proposent des adaptations d'AdaBoost pour les données non équilibrées. Les deux algorithmes ADAC1 et ADAC2 minimisent l'erreur pondérée par les coûts de la classe, alors que ADAC3 minimise l'erreur pondérée par le coût carré. Les algorithmes ADACi nécessitent un paramètre utilisateur C_i pour chaque classe i, qui est le cout de classification associé à tous les éléments de cette classe. ADAC2 repousse la frontière de la classe minoritaire même dans le cas où les éléments d'une classe sont toujours bien classés. Cela est dû au fait que le coût d'un élément est utilisé à l'extérieur de l'exponentiel dans l'équation définissant l'échantillonnage, conduisant un élément d'une classe minoritaire bien/mal classée à être échantillonné en proportion des coût associés à sa classe. Un des désavantages de ces trois algorithmes est qu'il est difficile de choisir un poids pour les différentes classes. La pondération par l'inverse du nombre d'éléments pour chaque classe peut conduire à des sur-échantillonnement excessifs pour la classe minoritaire (erreur >0.5).

1. ADAC1
   - $\alpha_t = \frac{1}{2} log(\frac{1+\sum_{i,y_i=h_t(x_i)} C_i D(i)^t - \sum_{i,y_i \neq h_t(x_i)} C_i D(i)^t}{1-\sum_{i,y_i=h_t(x_i)} C_i D(i)^t + \sum_{i,y_i \neq h_t(x_i)} C_i D(i)^t})$
   - $D_i^{t+1} = D_i^t \frac{exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t}$
2. ADAC2
   - $\alpha_t = \frac{1}{2} log(\frac{\sum_{i,y_i=ht(x_i)} C_i D(i)}{\sum_{i,y_i \neq ht(x_i)} C_i D(i)})$
   - $D_i^{t+1} = D_i^t C_i \frac{exp(-\alpha_t ht(x_i) y_i)}{Z_t}$
3. ADAC3
   - $\alpha_t = \frac{1}{2} log(\frac{\sum C_i D(i) + \sum_{i,y_i=ht(x_i)} C_i^2 D(i)^t - \sum_{i,y_i \neq ht(x_i)} C_i^2 D_i^t}{\sum C_i D(i) - \sum_{i,y_i=ht(x_i)} C_i^2 D(i)^t + \sum_{i,y_i \neq ht(x_i)} C_i^2 D(i)^t})$
   - $D_i^{t+1} = D_i^t C_i \frac{exp(-\alpha_t C_i ht(x_i) y_i)}{Z_t}$

Le paramètre α a les mêmes fonctions que celui utilisé dans Adaboost. Il est appris des erreurs de classification commises par l'algorithme dans chaque itération de Boosting. Z dans

le dénominateur permet de normaliser les valeurs de D^{t+1}. Ainsi, l'idée est de sur-échantillonner exponentiellement les éléments ayant été mal classés tout en considérant le coût d'un élément C_i. Le seul algorithme proposé garantissant la minimisation exponentielle de l'erreur pondérée est le AdaC2.

Batista et al. [87] introduisent une méthode de sous-échantillonnage dans un framework de bagging. Ils sous-échantillonnent la classe majoritaire dans chaque itération de bagging.

Wang et al. [82] utilisent une méthode similaire, mais basée sur le sur-échantillonnage. Leur méthode SMOTE-bagging génère des données synthétiques dans chaque itération de bagging. La première moitié de leurs échantillons est composée de la classe majoritaire, et la seconde moitié est composée de données synthétiques en plus de la classe minoritaire sur-échantillonnée (en utilisant un paramètre contrôlant le taux de données synthétiques).

Geng et al. [162] proposent une méthode d'ensemble basée sur la segmentation de l'espace de recherche. Ils commencent par apprendre des modèles sur plusieurs sous-espaces et pondèrent ensuite ces derniers par un ensemble de validation. Ils utilisent un KNN pour sélectionner le modèle à utiliser pour chaque élément à classifier.

RUS-Boost [84] sous-échantillonne la classe minoritaire dans chaque itération d'Adaboost [161]. Les poids de l'échantillon obtenus sont ensuite normalisés pour obtenir une distribution. EUS-Boost [85], qui est une extension de RUS-Boost, utilise un sous échantillonnage par un algorithme évolutif. Cette extension permet d'augmenter la diversité de l'ensemble obtenu par le Boosting.

Smote-Boost [163] génère M éléments synthétiques à partir de la classe minoritaire dans chaque itération de Boosting, en échantillonnant aléatoirement des éléments de cette classe. La méthode les combine avec les éléments échantillonnés par AdaBoost.

Krawczyk et al. [78] proposent une méthode basée sur un ensemble d'arbres de classification. Ils commencent par générer des sous-ensembles aléatoires d'attributs, et entrainent un classifieur sur chaque sous-ensemble. Ils utilisent ensuite un algorithme évolutif pour pondérer les différents classifieurs. Une des contributions de cette méthode est que l'algorithme évolutif permet de sélectionner le nombre optimal de classifieurs. Une pondération des classifieurs est effectuée seulement si ces derniers obtiennent le meilleur résultat pour un élément (sinon, leur poids est fixé à 0). Il nécessite néanmoins un nombre maximal de classifieurs pour l'algorithme évolutif.

2.3.3.2 Classification collective sur les réseaux

La classification collective est un sous-domaine de la classification semi-supervisée qui utilise les liens représentant les interactions ou similarités d'un élément, et peut également utiliser ses attributs. Sen et al. [15] définissent la classification collective comme l'utilisation des attributs d'un élément en plus des attributs observés et des labels non observés de son voisinage

afin de prédire son label (classe). Dans cette section, nous allons présenter certaines méthodes sélectionnées pour la classification collective sur les réseaux. Il existe très peu de méthodes qui ont abordé la classification collective sur les réseaux multimodaux ou pour les classes non équilibrées.

On peut considérer qu'il existe deux grandes familles d'algorithmes de classification collective sur les réseaux. La première va construire des modèles pour classifier des nœuds en utilisant leurs interactions et attributs ainsi que ceux de leurs voisins. La seconde famille de méthodes va propager les labels de nœuds sur les réseaux en utilisant seulement la structure du graphe. La second famille considère que les nœuds labellisés sont toujours présents sur le réseau (classification transductive). Cette supposition est souvent non valide, spécialement pour des domaines d'application comme la détection de spammeurs ou d'apologistes au jihad. Pour cette raison, nous allons nous focaliser sur les méthodes qui utilisent les attributs ainsi que la structure du graphe.

Nous pouvons identifier trois grandes orientations suivies par les algorithmes de classification collective sur des graphes attribués qui sont 1) d'utiliser un paramètre de régularisation qui pénalise l'affectation des nœuds proches structurellement à des classes différentes 2) de propager les labels une utilisant la structure et les attributs et 3) d'apprendre des imbrications de graphes (transformer l'interaction en vecteur d'attributs). Nous allons commencer par présenter quelques méthodes pour la classification transductive, puis le reste de la section présente des méthodes pour la classification inductive.

Subbian et al. [164] proposent une méthode de classification transductive qui utilise la matrice d'adjacence ainsi que la matrice d'attributs afin de créer une matrice laplacienne (méthode de régularisation). Cette dernière est générée à partir de données labellisées ainsi que des données à classifier. La première étape de leur méthode consiste à ajouter des liens entre les éléments de la classe minoritaire pour s'assurer que ces derniers auront une faible distance dans la matrice laplacienne. Ensuite, ils combinent la matrice d'adjacence avec une matrice de similarité d'attributs W en utilisant un paramètre de pondération. Enfin, ils utilisent un KNN pondéré pour classifier les données. Ils apprennent tous leurs paramètres à partir d'échantillons de contrôle.

Hu et al. [165] présentent une méthode pour la classification transductive qui identifie des patterns dans la structure du graphe pour propager les labels. Ils considèrent qu'un graphe est composé majoritairement de nœuds non labellisés, et que le peu de nœuds qu'il contient peuvent être utilisés pour propager les labels. Hu et al. [165] apprennent une métrique de distance pour chaque élément. L'idée est que cette métrique est inférieure à la distance euclidienne pour les éléments appartenant à la même classe, et supérieure à la distance euclidienne pour les éléments appartenant à des classes différentes. L'apport majeur de leur travail est d'utiliser des patterns de graphes pour propager les labels, alors que la majorité des méthodes n'utilisent que

les liens. Ils considèrent une clique de taille 3 ainsi que les co-citations générant une relation sémantique entre des nœuds, et apprennent une distance pour caractériser cette relation. Un des désavantages de leur méthode est la nécessité de paramétrer l'apport des cliques ainsi que des co-citations dans l'apprentissage de métriques.

Ahmed et al. [73] introduisent une méthode de classification collective qui s'inspire de Deepwalk [166] (imbrication de graphes). Ils commencent par effectuer un partitionnement des nœuds du graphe en utilisant leurs attributs seulement. Ils génèrent ensuite un graphe de plus petite taille où les communautés sont représentées par des groupes d'attributs fréquents. La prochaine étape consiste à exécuter des marches aléatoires sur ce graphe et à générer un espace à M dimensions à partir des marches fréquentes. Un algorithme de classification sur les attributs pourra être enfin utilisé pour détecter les classes. Cet algorithme dépend de l'efficacité du partitionnement initial des nœuds.

Yang et al. [167] introduisent une méthode de classification collective basée sur les réseaux de neurones. Ces réseaux de neurones utilisent les attributs des nœuds ainsi que des représentations de graphes. Ils proposent une méthode transductive qui n'utilise que les données d'apprentissage, et une méthode inductive qui met à jour le modèle en utilisant les données classifiées.

Concernant la classification inductive, les méthodes les plus populaires sont basés sur la propagation de labels. Ces méthode considèrent que la présence d'un voisin affectée à une classe précise est une information utile pour la classification d'un nœud. Ainsi, d'une itération à l'autre, ces méthode prennent en entré les labels affectés à leurs voisin afin d'améliorer leurs prédictions.

Un des algorithmes les plus utilisés dans la classification collective (inductive) est l'algorithme de classification itérative (ICA) [15]. Ce dernier entraine un modèle en utilisant les attributs des nœuds, ainsi que les attributs et labels de leurs voisins. L'algorithme 2 décrit le fonctionnement de l'ICA. Ce dernier effectue plusieurs prédictions en utilisant un classifieur comme les SVM, Naive Bayes ou autres. Dans chaque itération, il met à jour les labels des voisins qui affectent les résultats des prédictions. L'algorithme prend en paramètre le nombre maximal d'itérations.

Kong et al. [168] introduisent une méthode basée sur les méta-chemins de taille l et de l'algorithme ICA. Ils définissent un méta-chemin comme un ensemble de nœuds qui peuvent être atteints en suivant des types d'arêtes spécifiques. Un exemple est $(X_1PX_3CX_6PX_4)$, où P et C sont des types de relations et les X sont des nœuds. Des méta-chemins sont du même type s'ils suivent les mêmes types d'arêtes dans le même ordre. Leur approche est basée sur l'algorithme itératif de classification [15], où ils utilisent K ensemble de voisins par nœud où K est le nombre de types de méta-chemins.

Une autre méthode populaire pour la classification collective est l'échantillonnage de Gibbs

Algorithm 2 ICA

f :classifieur

T : nombre maximum d'itérations

1: *Pour chaque nœud* n
2: \\Calculer les labels en utilisant les attributs observés
3: $y_n \leftarrow f(n)$
4: Répéter \\Classification itérative
5: *Pour chaque nœud* n
6: \\Calculer les labels en utilisant les attributs observés et inférés
7: $y_n \leftarrow f(n)$
8: Arrêt quand les labels inférés se sont stabilisés ou quand T a été atteint

[169]. Comme l'ICA [15], cette méthode commence par générer un modèle en utilisant les attributs des nœuds, ainsi que les attributs et labels des voisins. Ensuite, T itérations sont effectuées où les labels sont affectés aux nœuds avec la probabilité retournée par l'algorithme de classification utilisé. La troisième étape est similaire à la seconde étape avec l'exception que les labels assignés à chaque nœud sont comptés. La dernière étape est d'assigner à chaque nœud le label le plus fréquent obtenu dans la troisième étape.

Sen et al. [15] ont évalué trois méthodes de propagation de labels qui sont souvent utilisées dans la classification collective. La première est l'ICA et la seconde est l'échantillonnage de Gibbs. La troisième méthode est basée sur les champs aléatoires de Markov, où des facteurs définissent des labels dans le graphe. Cette dernière méthode utilise un algorithme de propagation de croyance pour obtenir les labels des nœuds. Les expérimentations effectuées montrent que les trois algorithmes obtiennent des résultats similaires. Les champs aléatoires de Markov apportent une petite amélioration aux résultats mais nécessitent beaucoup de paramétrisation.

Kazienko et Kajdanowicz [170] introduisent des attributs obtenus à partir de données relationnelles qui dépendent des classes. Un exemple de ces attributs est la centralité d'intermédiarité qui dépend des classes. Cette dernière est obtenue en considérant seulement les nœuds étant assignés à un label précis dans le calcul de la centralité. Ils proposent deux algorithmes de propagation de labels qui utilisent les nouveaux attributs introduits.

- LDBootstrapping : Classifie chaque nœud une seule fois en utilisant les attributs qui dépendent des classes. Cet algorithme nécessite que les données d'entrainement soient dans le même graphe que les données qu'on souhaite classifier (apprentissage transductive).
- LDGibbs : comme son nom l'indique, cet algorithme est une version modifiée de l'échantillonnage de Gibbs. Les attributs qui dépendent des classes sont recalculés à chaque itération, étant donné que les labels assignés aux nœuds peuvent changer d'une itération à l'autre.

Zhou et al. [110] ont proposé une méthode pour la classification collective qui a été décrite

dans la section 2.3.2.1. Cette méthode optimise une fonction objectif qui calcule une similarité entre les nœuds. Cette fonction contient une composante de lissage qui assigne les mêmes labels aux nœuds pointant vers les mêmes nœuds autorités, et qui sont référencés par les même nœuds hubs. Ainsi, cette méthode utilise la structure du graphe comme une composante de régularisation.

Cao et al. [171] présentent une méthode de classification collective qui enrichit les attributs des nœuds par des "méta-chemins". Ces derniers sont constitués en effectuant une recherche en profondeur sur le graphe à partir d'un nœud et en agrégeant des attributs de nœuds traversées dans cette recherche. Cette méthode peut néanmoins avoir une complexité élevée dans des graphes denses (ex. graphes d'URLs).

Castillo et al. [62] proposent une méthode de classification collective qui permet de détecter des SPAMs sur le Web en utilisant les hyper liens ainsi que les données textuelles des pages Web. Leur méthode commence par générer un nombre d'attributs à partir des hyper liens qui sont soit locaux (basés sur le degré) soit structurels comme le score de pagerank, ou le score de trustrank (pagerank avec téléportation vers des pages de confiance). La première étape de cette méthode consiste à effectuer une classification initiale utilisant les attributs des nœuds. La seconde étape consiste à utiliser les voisins des nœuds pour affiner la classification. Ils considèrent trois méthodes pour effectuer cela. Ils ont ainsi comparé des représentants de trois familles de méthodes dans leur recherche. Les meilleurs résultats on été obtenus par la troisième méthode.

- La première méthode effectue une détection de communautés sur le graphe des hyper liens et labellise tous les nœuds d'une communauté par le même label. Ce dernier est déterminé par un vote de tous les membres d'une communauté ;
- La seconde méthode utilise une régularisation afin que les nœuds faisant souvent partie des mêmes marches aléatoires aient les mêmes labels ;
- La troisième méthode entraine un nouveau modèle en utilisant les attributs des nœuds et les labels de leurs voisins. Cette méthode se base sur l'ICA mais n'effectue qu'une seule itération.

Une méthode qui est utilisée pour répondre au problème de la classification non équilibrée est la prédiction de liens. En effet, il on peut souvent considérer que le réseau d'entrainement contient a une taille inférieur au réseaux utilisés dans la phase de classification (manque de données labellisées). Ainsi, plusieurs méthodes effectuent de la prédiction de liens sur le réseau d'entrainement afin de repousser la frontière de la classe minoritaire.

Gallagher et al. [172] proposent une méthode basée sur la prédiction de liens. Il utilisent des marches aléatoires pour ajouter des liens entre deux entités et proposent deux algorithmes pour classifier les nœuds. Le premier est basé sur la propagation de labels et utilise la moyenne des voisins directs. Le second algorithme utilise les données labellisées pour apprendre les poids des voisins ajoutés par la prédiction de liens ainsi que le poids des voisins présents dans

le graphe. Pour chaque nœud, les attributs des voisins ajoutés par la prédiction de liens sont agrégés dans six groupes différents. Le premier groupe contient les voisins les plus proches et le second contient des voisins plus distants, et ainsi de suite. Chaque groupe est assigné un poids différent.

Bilgic et al. [173] proposent une adaptation de l'algorithme de classification itérative [15] qui utilise la prédiction de liens. Dans chaque itération, leur méthode commence par prédire des liens en utilisant la régression logistique, puis classifie les nœuds en utilisant leur nouveaux voisinages. Ainsi, si la classification de nœuds s'améliore après chaque itération, la prédiction de liens s'améliore également étant donné qu'elle utilise les labels estimés. Cette méthode peut amplifier le problème de l'exagération des poids des labels estimés qui a été identifié pour l'algorithme de classification itérative.

Zhao et al. [174] introduisent une méthode de classification collective qui tire profit de la prédiction de liens. La première étape de leur méthode est d'extraire les attributs du graphe utilisés pour la classification. Ensuite, ils choisissent un algorithme de prédiction de liens utilisant la structure du graphe. La troisième étape est de prédire des liens en sur le graphe, et de les pondérer en utilisant la similarité d'attributs entre les voisins. Enfin, leur méthode utilise la moyenne pondérée des voisins directs pour prédire la classe d'un nœud.

Un autre groupe de méthodes de classification collective tire profit des méthodes d'ensembles. Ces méthodes sont adapté pour la classification sur des réseaux multimodaux. Cette tendance d'utiliser des méthodes d'ensembles sur des réseaux multimodaux est dû à la haute dimensionalité de ces derniers comparés aux réseaux uni-modaux. Les méthodes d'ensembles sont souvent mieux adaptés pour répondre a des problèmes en haute dimension. Il faut noter que les méthodes d'ensembles peuvent être intégrées dans un algorithme de propagation de labels comme l'ICA, ou utiliser une méthode de régularisation ou d'imbrication de graphes.

Vijayan et al. [175] proposent une méthode de classification sur les réseaux en utilisant différentes perspectives (combinaison d'attributs). Chaque perspective contient des attributs ou une structure de graphe (transformée en attributs structurels). Ainsi, les différents modes d'interaction sont modélisés par des perspectives différentes. Ils utilisent un classifieur pour chaque perspective et minimisent le désaccord entre les perspectives.

Shi et al. [176] proposent une méthode qui se base sur le boosting de gradient. Ils apprennent également un modèle sur un ensemble de perspectives où chacune représente une source de données. Pour chaque perspective, ils entrainent un classifieur. Ils minimisent ensuite une fonction qui calcul des poids pour chaque classifieur de sorte que leur désaccord soit réduit. Ces classifieurs pour chaque perspective sont ensuite utilisés dans un framework de boosting de gradient (l'itération n+1 s'entraine sur le résidu de l'itération n). Les auteurs utilisent le principe de l'homophilie où ils ajoutent une pénalité pour les entités connectées ayant des prédictions différentes.

Preisach et al. [177] proposent une méthode de classification collective pour graphes multimodaux. Ils utilisent un sous-graphe pour chaque mode et agrègent les attributs des voisins de leurs voisins dans chaque sous-graphe. Ensuite, ils apprennent un modèle pour chaque sous-graphe et combinent les estimations des modèles par un méta-classifieur. Comme pour l'ICA [15], ils utilisent des fonctions d'agrégation pour générer des attributs de voisins.

Eldardiry et Neville [178] offrent une justification analytique pour l'utilisation d'ensemble de classifieurs pour la classification collective. Ils ont démontré que ces ensembles permettent de réduire le biais et la variance des modèles. De plus, ils ont également démontré que l'utilisation de classifieur sur des sous-ensemble de graphes (ex : graphes obtenus par tirages aléatoires) permet de réduire le biais des modèles et d'obtenir de meilleures prédictions.

2.3.3.3 Discussion

Nous pouvons identifier deux grandes familles de méthodes pour la classification de données avec classes non équilibrées. La première effectue des transformations sur le jeu d'entrainement en amont de la génération du modèle [86, 77, 87, 158]. L'algorithme de classification est ensuite exécuté sur un jeu de données transformé dans le butte de répondre à la problématique de classes non équilibrées. La seconde résout le problème de classes non équilibrées au niveau algorithmique [84, 162, 163, 79, 74]. L'intuition derrière une grande partie de la seconde famille de méthodes est qu'il n'est pas nécessaire d'effectuer des transformations sur tout le jeu de données d'entrainement, mais seulement sur une région difficile à classifier. Différents algorithmes peuvent itérativement identifier les régions difficiles et effectuer des transformations sur ces dernières afin d'adapter des modèles de classification à la problématique de classes non équilibrées.

Les deux familles de méthodes font usage des mêmes mécanismes pour générer des modèles adaptés à la classification non équilibrée. Ces derniers sont

- Le sur-échantillonnage ;
- Le sous-échantillonnage ;
- La génération de données synthétiques ;
- L'apprentissage de règles à partir de la classe minoritaire ;
- L'optimisation de fonctions objectif qui privilégient la classe minoritaire ;
- L'augmentation de couts d'erreurs de classification d'éléments de la classe minoritaire.

Le sur-échantillonnage et l'augmentation du couts d'erreurs de classification ont un effet similaire avec l'exception que la seconde méthode n'augmente pas nécessairement la complexité des algorithmes. Ces deux méthodes forcent les algorithmes à repousser la frontière de la classe minoritaire. Ils nécessitent de sélectionner les couts d'erreurs de classification par classe qui ne sont souvent pas intuitifs à choisir.

Les méthodes de sous-échantillonage ont l'avantage de réduire la complexité des algo-

rithmes. Néanmoins, ils peuvent conduire à la perte d'éléments importants pour la classification. Une des méthodes les plus populaire pour répondre au problème de classes non équilibrée est le sous-échantillonage avec détection de représentants pour la classe minoritaire.

La génération de données synthétiques permet de repousser la frontière de la classe minoritaire. Contrairement au sur-échantillonage ou à l'augmentation de couts d'erreurs de classifications, cette méthode peut étendre la classe minoritaire dans la classe majoritaire en générant de nouveaux éléments. Elle permet également de connecter plusieurs régions déconnectés de la classe minoritaire. Un des désavantages de cette méthode est qu'elle peut réduire la précision de la classe majoritaire, augmentant ainsi les faux positifs. Plusieurs méthodes basées sur la détéction d'outliers ont été proposés pour répondre à ce problème.

L'apprentissage de règles à partir de la classe minoritaire peut offrir des résultats dépassant la classification binaire sur des jeux de données hautement non équilibrés. Cette méthode n'est cependant pas privilégiée quand les données sont en haute dimension.

Les méthodes qui optimisent des fonctions objectif privilégiant la classe minoritaire font également plusieurs choix compte tenu du cout d'un élément. Le pourcentage d'éléments bien classés pour chaque classe est souvent indirectement optimisé par ces mesures (la somme des couts pour chaque classe est égale à 0.5). Ce choix n'est souvent pas adapté quand le nombre d'éléments de la classe majoritaire est très élevé, étant donné qu'un faible pourcentage d'éléments mal classés peut conduire à un grand nombre de faux positifs.

Concernant les méthodes de classification collective, la majorité ne traitent pas le problème de classes non équilibrées. Il est ainsi nécessaire d'effectuer un traitement des données en amont de l'exécution des algorithmes afin de répondre a cette problématique. Les seuls méthodes qui peuvent répondre à ce problème au niveau algorithmique sont les méthodes basés sur la prédiction de liens. Ces derniers augmentent la taille des voisinages dans les données d'entrainement permettant ainsi d'étendre la frontière de la classe minoritaire. Il faut néanmoins noter que la prédiction de liens peut être un tache aussi difficile que la classification, et que d'après les expérimentations effectuées dans cette thèse, ces méthodes risquent de dégrader la classe majoritaire.

Nous pouvons identifier deux grandes familles de méthodes de classification collective. La première concerne la classification transductive ou les données d'entrainement et les données à classifier sont sur le même réseau. La seconde concerne la classification inductive où les données d'entrainement et les données à classifier sont dans des réseaux différents.

Pour les deux familles de méthodes de classification collective, il est possible d'identifier trois grandes orientations qui peuvent caractériser les algorithmes développées :

– Propagation de labels : cette première orientation considère que le label des voisins d'un élément est une information utile pour la classification. Ces méthodes offrent de meilleurs résultats quand l'information relationnel est déterminante pour une classification. Les méthodes de cette orientation peuvent également utiliser les attributs des nœuds ainsi que

les attributs de leurs voisins.

- Régularisation par la structure du graphe : Ces méthodes utilisent les attributs des nœuds pour classifier ces derniers et utilisent la structure du graphe afin "d'encourager" le regroupement de nœuds connectés dans les mêmes classes. Une des problématiques liés à ces méthodes est la sélection du facteur de régularisation. Cela nécessite une connaissance à priori des réseaux ou l'utilisation d'un jeu de validation d'un taille importante.
- Apprendre des imbrications de graphes : Cette troisième famille de méthode est souvent utilisé pour l'apprentissage transductive, même si des méthodes basés sur l'apprentissage de profils de nœuds sont utilisés dans l'apprentissage inductive. L'objectif est de transformer le graphe en vecteur, et d'apprendre un modèle sur ce dernier. Ces méthodes nécessitent ainsi d'avoir un graphe fortement connecté, ce qui n'est pas toujours le cas dans les réseaux sociaux.

2.3.4 Évaluation de la qualité de classifications

Dans la section ci-dessous, nous allons présenter plusieurs métriques qui permettent d'évaluer la qualité des résultats de classifications. Les premières mesures qui sont les plus souvent utilisées sont le nombre de vrais positifs, faux positifs, vrais négatifs et faux négatifs.

- Nombre de vrais positifs (TP) : Ce sont les éléments de la classe positive qui ont été bien classés. La classe positive est le plus souvent la classe minoritaire.
- Nombre de faux positifs (FP) : Le nombre d'éléments de la classe négative (souvent majoritaire) classés dans la classe positive.
- Nombre de vrais négatifs (TN) : Les éléments de la classe négative qui ont été bien classés.
- Nombre de faux négatif (FN) : Les éléments de la classe positives qui ont été classés comme négatifs.

D'autres indicateurs importants sont la sensibilité et la spécificité.

La sensibilité (rappel) ou le taux de vrais positifs (TPR) indique la probabilité qu'un élément soit classé positif par l'algorithme sachant que ce dernier appartient à la classe positive. La sensibilité est calculée par l'équation 2.12.

$$TPR = \frac{TP}{TP + FN} \tag{2.12}$$

Le taux de faux positifs (FPR) indique la probabilité qu'un élément soit classé négatif alors qu'il est positif. Ce taux est calculé par l'équation 2.13

$$FPR = \frac{FP}{FP + TN} \tag{2.13}$$

La spécificité (TNR) calcule la probabilité de rejeter un test pour des éléments négatifs, et dans le cadre de la classification, d'affecter des éléments à la classe négative sachant qu'ils sont négatifs. Cette dernière est calculée par l'équation 2.14.

$$TNR = \frac{TN}{FP + TN} \tag{2.14}$$

Le taux de faux négatifs (FNR) calcule la probabilité qu'un élément négatif soit classé positif. Ce taux est calculé par l'équation 2.15.

$$FNR = \frac{FN}{TP + FN} \tag{2.15}$$

Le tableau 2.1 résume les quatre mesures présentées ci-dessus.

TABLE 2.1 – Mesure de qualité d'une classification

	Prédits positifs	Prédits négatifs
Classe positive	Vrais positifs (TP)	Faux négatifs (FN)
Classe négative	Faux positifs (FP)	Vrais négatifs (TN)

Taux de bien classés : Ce taux calcule la probabilité qu'un élément soit affecté à sa classe. Ce taux est calculé par l'équation 2.16. Ce taux est optimisé par une majorité des algorithmes de classification.

$$Acc = \frac{TN + TP}{TN + TP + FN + FP} \tag{2.16}$$

Concernant le cas où les classes sont non équilibrées, l'optimisation du taux des bien classés conduit des algorithmes de classification à affecter tous les éléments (ou une majorité) à la classe majoritaire. Une adaptation du taux de bien classés quand les classes sont non équilibrées est le **taux de bien classés pondéré**. Ce dernier est calculé par l'équation 2.17.

$$WAcc = \frac{C_1 \cdot TN + C_2 \cdot TP}{C_1 \cdot TN + C_2 \cdot TP + C_2 \cdot FN + C_1 \cdot FP} \tag{2.17}$$

Une mesure qui nous vient du domaine de la recherche d'information qui est souvent utilisée pour quantifier la qualité d'une tâche de classification est la précision. Cette dernière est définie par le taux de vrais positifs sur le nombre d'éléments appartenant à la classe positive (eq 2.18)

$$Pr = \frac{TP}{TP + FP} \tag{2.18}$$

F-measure : permet de calculer la performance d'une classification relative à la classe minoritaire seulement. Cette mesure représente une moyenne harmonique entre la précision et le rappel. Puisque la F-measure est une moyenne harmonique, elle est maximisée lorsque la précision et le rappel sont élevés, et baisse lorsqu'une de ces deux mesures est faible. Ainsi, un algorithme qui obtient un résultat décent sur les deux mesures aura un meilleur F-measure qu'un algorithme qui maximise l'une des deux mesures et obtient de faibles résultats sur la seconde.

Cette mesure est calculée par l'équation 2.19.

$$F - measure = \frac{2TPR \cdot Pr}{TPR + Pr} \tag{2.19}$$

G-mean : Cette mesure est souvent utilisée quand la performance pour les deux classes est considérée [79]. Cette dernière a été introduite par Kubat et al. [179] et elle est définie par l'équation 2.20.

$$G - mean = \sqrt{TPR \cdot TNR} \tag{2.20}$$

Fonction d'efficacité du récepteur (courbe ROC) : Plusieurs algorithmes de classification assignent un score ou une probabilité à leurs prédictions [79]. En variant le seuil d'affectation à une classe, il est possible de créer des courbes de TPR et TNR. Il est ainsi possible de tracer une courbe de TPR et de FPR qui illustre les résultats d'un algorithme en fonction du seuil d'acceptation. Cette courbe rend difficile la tâche de comparaison entre algorithmes étant donné que ces résultats dépendent du seuil d'acceptation. Un algorithme peut être considéré comme ayant de meilleurs résultats si sa courbe domine celle des autres algorithmes.

Surface sous la courbe ROC (AUC) : Cette surface est calculée en utilisant la surface sous la courbe ROC. Elle permet de comparer des algorithmes en utilisant plusieurs seuils pour l'affectation d'un élément à une classe. Cette mesure est souvent utilisée pour quantifier la performance d'algorithmes de classification pour classes non balancées.

Discussion :

Le choix d'une mesure adéquate pour l'évaluation de la qualité d'une classification dépend largement du domaine d'application et du jeu de données utilisé. En effet, dans plusieurs cas, maintenir le nombre de faux positifs au plus bas est primordiale. Des exemples sont la détection d'apologistes au jihad, ou la détection de fraudeurs. Dans ces deux cas, labelliser un utilisateur bénin comme apologiste ou fraudeur peut conduire au bannissement de ce dernier de la plateforme sociale. Surtout, cela peut porter préjudice à l'utilisateur d'être enregistré comme jihadiste ou fraudeur dans une base de données. L'utilisation du G-mean, de la F-measure ou autre mesure pour la classification non équilibrée n'est pas adéquate dans ce cas. Il est ainsi

nécessaire de considérer le taux de faux positifs et de vrais positifs afin de déterminer si l'algorithme fourni des résultats acceptables.

Dans le cas ou le nombre d'éléments de la classe majoritaire est très élevé (pouvant atteindre des millions), les mesures pour la classification non équilibrée ne sont pas adéquates. Même un faible taux de faux positifs peut indiquer que des milliers, voir des millions d'utilisateurs sont mal classés. Ainsi, le taux de faux positifs et vrais positifs, ou le taux de biens classés et le taux de vrais positifs devraient être utilisés pour évaluer la qualité d'une classification.

Dans certains domaines où la détection d'éléments de la classe minoritaire est d'une grande importance comme pour la détection de clients potentiels ou la détection de nœuds vulnérables a des attaques, l'utilisation de mesure comme le G-mean ou la F-measure est justifiée, étant donné qu'il est très important pour la tache de classification d'augmenter son rappel de la classe minoritaire. Il est aussi intéressant d'utiliser ces mesures quand les erreurs de classification n'engendrent pas un couts supplémentaire important, comme pour la classification dr clients potentiels dans l'objectif de les rajouter à une liste de diffusion (mailing list).

La mesure AUC permet de comparer des algorithmes quand les critères de sélection ne sont pas connus à l'avance. Un algorithme obtenant un score élevé de AUC, permet d'obtenir un haut TPR par rapport à différents FPR. L'utilisateur pourra ensuite sélectionner un seuil de FPR acceptable en fonction de ses critères.

Chapitre 3

Détection de revendeurs de cartes de crédit par interconnexion de réseaux sociaux numériques

Les réseaux sociaux numériques sont souvent utilisés par des groupes malveillants afin de propager leurs messages. Plusieurs de ces utilisateurs publient leurs contenus sur des réseaux sociaux différents afin de maximiser leurs audiences. Ce comportement est surtout visible parmi les communautés des revendeurs de cartes de crédit qui publient leurs messages dans des blogs et réseaux sociaux avec un faible niveau de sécurité et une faible audience, ainsi que sur des réseaux populaires tels que Facebook et Twitter. Nous présentons dans ce chapitre une typologie des réseaux sociaux numériques [180] qui caractérise le niveau de sécurité, la visibilité et les politiques de recherche d'information sur ces réseaux. Nous présenterons ensuite un algorithme de détection d'anomalies [7] qui utilise les interconnexions entre réseaux sociaux afin de détecter des utilisateurs malveillants. Cet algorithme utilise les réseaux à faible mesure de sécurité permettant une navigation basée sur le contenu où les revendeurs de cartes de crédit sont souvent explicites en ce qui concerne leurs activités. Ces derniers sont généralement beaucoup plus discrets dans les réseaux populaires avec des navigations de type serendipity (contacts vers contacts de contacts) étant donné qu'être banni du réseau les obligerait à reconstruire une liste de contacts légitimes. En effet, nous avons découvert que plusieurs revendeurs de cartes de crédit peuvent avoir plus de 10k followers sur Twitter, qui ne se doutent pas pour la plupart des activités de l'utilisateur malveillant.

3.1 Typologie de réseaux sociaux numériques

La typologie proposée [17] catégorise les réseaux sociaux numériques selon 1) le type de contenu partagé sur ces réseaux 2) le type de contenu proposé aux utilisateurs par les plateformes de réseaux sociaux 3) la visibilité et les recherches d'utilisateurs et enfin, 4) les mécanismes de gestion des autorisations de consultation et de partage. Cette typologie a été inspirée de [181], qui a caractérisé les réseaux sociaux selon le type de navigation et les politiques de visibilité des utilisateurs. Ce travail ajoute deux nouvelles dimensions qui sont la gestion des autorisations et les services de géolocalisation, tout en simplifiant les premières dimensions.

1. **Le type de contenu partagé sur les RSN :** dans certains cas, le contenu partagé sur les RSN peut être la raison principale pour laquelle leurs utilisateurs rejoignent ces réseaux [182]. Des exemples sont les réseaux de partage de contenu tels YouTube et MySpace. Ainsi, le type d'objets que les utilisateurs partagent est un aspect important pour l'analyse de ces derniers. Nous proposons de diviser les RSN en trois catégories en se basant sur ce critère :
   - **RSN permettant des messages longs :** ces réseaux permettent aux utilisateurs de partager des messages de plus de 600 caractères. Ces derniers peuvent également contenir des ressources multimédias (vidéos, sons, images, etc.) ;
   - **RSN permettant d'echanger des messages courts :** Ces réseaux permettent à leurs utilisateurs d'échanger seulement des messages courts accompagnés de ressources multimédias. Un exemple est Twitter (280 caractères) et LinkedIn (600 caractères) ;
   - **RSN ne permettant pas l'échange de texte :** Ces réseaux permettent aux utilisateurs de publier des ressources multimédias. Ces derniers peuvent également commenter et partager ces ressources. Des exemples sont Flickr et YouTube.
2. **Contenu proposé par les RSN :** plusieurs RSN recommandent du contenu à leurs utilisateurs en se basant sur plusieurs critères. Parmi ces contenus, des RSN peuvent suggérer à leurs utilisateurs des événements avec une localisation géographique (cinémas, restaurants, galas, etc.). Les systèmes de recommandation de ces RSN prennent souvent en compte la localisation géographique de leurs utilisateurs afin de leur proposer ces contenus. Ainsi, les RSN peuvent faciliter la localisation de leurs utilisateurs, même si ces derniers ne partagent pas nécessairement cette information. Ce critère permet de catégoriser les RSN en réseaux qui recommandent du contenu en se basant sur la localisation de leurs utilisateurs et en réseaux qui ignorent cette information. La localisation d'un utilisateur doit être précise, une ville, par exemple. On ne considère pas que les RSN qui recommandent du contenu en se basant sur le pays des utilisateurs (ex : recommandation uniquement du contenu dans une langue spécifique).
3. **Mécanismes de recherche :** Les RSN ont créé plusieurs mécanismes permettant aux uti-

lisateurs de naviguer à travers leurs contenus. Ces mécanismes ont un impact important sur l'expérience des utilisateurs dans ces réseaux. Certains RSN enferment leurs utilisateurs dans des sandboxes où ils ne peuvent accéder qu'à des contenus publiés par des utilisateurs avec lesquels ils partagent une connexion. D'autres RSN, YouTube par exemple, permettent à leurs utilisateurs de naviguer à travers tout le contenu du réseau. Cardon [42] a catégorisé les RSN en cinq groupes en se basant sur la visibilité de leurs utilisateurs :

- **Réseaux fermés** : Dans ces réseaux, des individus partagent des éléments de leur vie privée avec des utilisateurs qu'ils ont sélectionnés. Les utilisateurs de ces réseaux ont tendance à donner accès à leur profil à une liste restreinte d'utilisateurs qu'ils connaissent également en dehors du réseau. Les utilisateurs avec lesquels ils ne partagent pas de connexion ont généralement un accès très limité. Le mécanisme de recherche permis sur ces réseaux est généralement basé sur le serendipity. Ce mécanisme permet de naviguer d'un utilisateur à un autre en partant des utilisateurs proches "amis" vers les "amis d'amis", et ainsi de suite. Des exemples de tels réseaux sont Cyworld et LiveJournal [42].
- **Réseaux d'auto-promotion :** Les utilisateurs partagent plusieurs aspects de leurs personnalités ainsi que leurs créations avec tout le réseau. Dans ces réseaux, les utilisateurs essaient de maintenir une liste étendue de connexions avec des utilisateurs souvent inconnus. Cela leur permet d'avoir une audience pour leurs créations. Les connexions avec d'autres utilisateurs peuvent également servir de marque-pages. Ces réseaux sont centrés sur le contenu où les interactions entre utilisateurs indiquent des affinités par rapport aux contenus partagés par ces derniers plutôt que l'existence d'une relation à l'extérieur du réseau.
- **Réseaux post-it :** Dans ces réseaux, les utilisateurs rendent leur présence connue en publiant de brèves mises à jour. Leurs publications sont diffusées à un cercle de contacts qui les suivent. La localisation géographique des utilisateurs de ces réseaux est très importante, étant donné qu'ils sont souvent utilisés par ces derniers pour explorer leur environnement ou pour organiser des événements. Des exemples de tels réseaux sont Twitter et Dodgeball.
- **Réseaux fermés permettant l'auto-promotion** : Ces réseaux partagent des caractéristiques avec les réseaux fermés et les réseaux d'auto-promotion. Les publications des utilisateurs sont visibles à une liste d'amis et d'amis d'amis. Les utilisateurs peuvent également rendre accessibles leurs publications à tout le réseau.
  Une des caractéristiques majeures de ces réseaux est qu'ils favorisent une recherche basée sur le serendipity (amis, puis amis d'amis) plutôt qu'une recherche basée sur les mots clés [42]. Un exemple de tels réseaux est Facebook et Instagram.
- **Réseaux fermés permettant l'auto-promotion et le post-it** : Ces réseaux partagent

des caractéristiques avec les trois premiers réseaux présentés ci-dessus. Ils permettent le post-it et l'auto-promotion, mais sont souvent utilisés comme des réseaux fermés. Ces derniers vont généralement interagir avec des utilisateurs qu'ils connaissement, mais vont également garder des liens avec des utilisateurs inconnus qu'ils utilisent comme marque-pages. Un exemple de ces réseaux est Google +.

4. **La gestion des autorisations :** plusieurs réseaux sociaux numériques offrent à leurs utilisateurs la possibilité de gérer la visibilité des contenus qu'ils diffusent. Certains de ces réseaux permettent une gestion fine de ces permissions, offrant à l'utilisateur le choix du public pouvant accéder à chaque publication. D'autres réseaux sociaux offrent une gestion de permissions binaires (public/privée) ou même, aucune gestion de permission. En utilisant ce critère, nous regroupons les réseaux sociaux numériques en deux catégories :

   – **Les RSN qui permettent la création de groupes de diffusion** : Ces réseaux permettent d'ajouter les contacts d'un utilisateur à différents groupes de diffusion. Ils peuvent ensuite assigner à chaque groupe des droits d'accès différents ou limiter les contenus qu'ils publient à certains groupes.
   – **Les RSN qui ne permettent pas la création des groupes de diffusion** : Ces réseaux offrent peu de mécanismes de gestion de permissions à leurs utilisateurs. Ces derniers peuvent soit partager leurs contenus avec le réseau ou le garder privé.

Le tableau 3.1 offre une classification de réseaux sociaux en utilisant les critères présentés ci-dessus. Nous pouvons remarquer que les réseaux qui sont orientés auto-promotion offrent généralement peu de mécanismes de gestion des permissions. Les réseaux Post-it permettent les messages courts, alors que les réseaux fermés ont tendance à permettre les messages longs. Les réseaux d'auto-promotion offrent des formats de messages variables. Cette thèse est focalisée sur l'étude de réseaux Post-it (ex : Twitter). Ces derniers sont caractérisés par des interactions hétérogènes (RT - Retweet, @ Mentions, Reply) où les utilisateurs interagissent souvent avec des utilisateurs distants dans le graphe. Ainsi, ces réseaux sont caractérisés par une structure étoilée où des leaders de groupes thématiques ont un degré très élevé (distribution des degrés en loi de puissance).

SNS	Crit. 1	Crit. 2	Crit. 3	Crit. 4
Facebook	Long mess.	Location content	Closed and social prom.	Perm. mngt
LinkedIn	Short mess.	Location content	Closed and social prom.	No perm. mngt
Foursquare	No mess.	Location content	Closed and social prom.	No perm. mngt
Google +	Long mess.	Location content	closed, post it & soc. prom.	Perm. mngt
Live Journal	Long mess.	No Loc. content	Soc. prom.	Perm. mngt
YouTube	No. mess.	No Loc. content	Soc. prom.	No perm. mngt
LastFM	No mess.	No Loc. content	Soc. prom.	No perm. mngt
MySpace	No mess.	No Loc. content	Soc. prom.	No perm. mngt
Flickr	No mess.	No Loc. content	Soc. prom.	No perm. mngt
Twitter	Short mess.	Location content	Post-it	Perm. mngt
Delicious	No mess.	No Loc. content	Soc. prom.	No perm. mngt
Stumble Upon	No mess.	No Loc. content	Soc. prom.	No perm. mngt
Tumblr	No mess	No Loc. content	Soc. prom.	No perm. mngt

TABLE 3.1 – Typologie de RSN populaires

3.2 Algorithme de détection de revendeurs de cartes de crédit

Les blogs, forums et réseaux sociaux numériques sont devenus populaires dans l'échange de biens volés. Parmi ces biens, la revente de cartes de crédit volées est de plus en plus commune. Il est à la portée de personnes ayant peu de connaissances informatiques de retrouver des cartes volées sur différents réseaux avec des prix variant de 1 à 40 dollars selon la balance dans la carte [183]. En plus de l'illégalité de cette activité, les revendeurs de cartes de crédit volées sont des menaces pour la sécurité d'un réseau numérique. En effet, ils peuvent rediri-

ger les utilisateurs vers des sites web qui enregistrent les informations des cartes lors d'achats (phishing). Ils peuvent également propager du malware dans le réseau et infecter un nombre important d'utilisateurs. Les utilisateurs non malveillants qui interagissent avec des revendeurs de cartes de crédit peuvent ainsi devenir des véhicules permettant l'infection du réseau. Les revendeurs de cartes de crédit volées publient souvent sur plusieurs réseaux sociaux et blogs. Ces revendeurs sont confrontés au dilemme de rendre leurs publications et profils explicites vis-à-vis de leurs activités afin de permettre aux clients potentiels de les retrouver sur ces réseaux, ou de cacher leurs activités afin d'éviter d'être détectés par les plateformes sociales. Souvent, ces revendeurs rendent leurs posts explicites sur les blogs et réseaux dont le nombres d'utilisateurs est limité, ainsi que dans les réseaux d'auto-promotion [80] où leurs listes de contacts ont peu d'effet sur la diffusion de leurs messages (navigation par contenus). Néanmoins, sur les réseaux de type Post-it ou sur les réseaux personnels (navigation par serendepity), ces derniers cachent leur comportement leur permettant de bâtir une réputation et de se doter d'une liste de diffusion. Pour cela, ils communiquent entre eux et avec des utilisateurs légitimes, et publient un nombre important de messages non malveillants. Certains utilisateurs adoptent même une activité licite comme l'agrégation d'information les rendant difficiles à détecter sur ces réseaux. Nous proposons, dans cette section, un algorithme semi-supervisé de détection de revendeurs de cartes de crédit. Cet algorithme effectue une détection sur les réseaux sociaux de promotion, ainsi que sur les blogs, forums. Il modélise ainsi dans sa première étape chaque réseau par un graphe unimodal. L'algorithme inter-connecte ensuite ces derniers avec les réseaux Post-it et réseaux personnels afin de détecter ces revendeurs dans ces plateformes. Il génère ainsi un graphe multimodal utilisé pour la détection de revendeurs sur des réseaux Post-it et personnels. Nous allons d'abord présenter la méthodologie, et ensuite détailler des expérimentations effectuées sur des milliers de blogs et sur Twitter.

3.2.1 Méthodologie

L'algorithme 3 décrit la méthode proposée. La première étape est d'entraîner un classifieur pour la détection de revendeurs de carte de crédit volées sur des blogs et réseaux d'auto-promotion. Un modèle de classification f est généré dans cette étape. Les publications sur ces réseaux sont explicites et la classification peut être précise. La figure 3.1 illustre des revendeurs de cartes de crédits sur des blogs et forums.

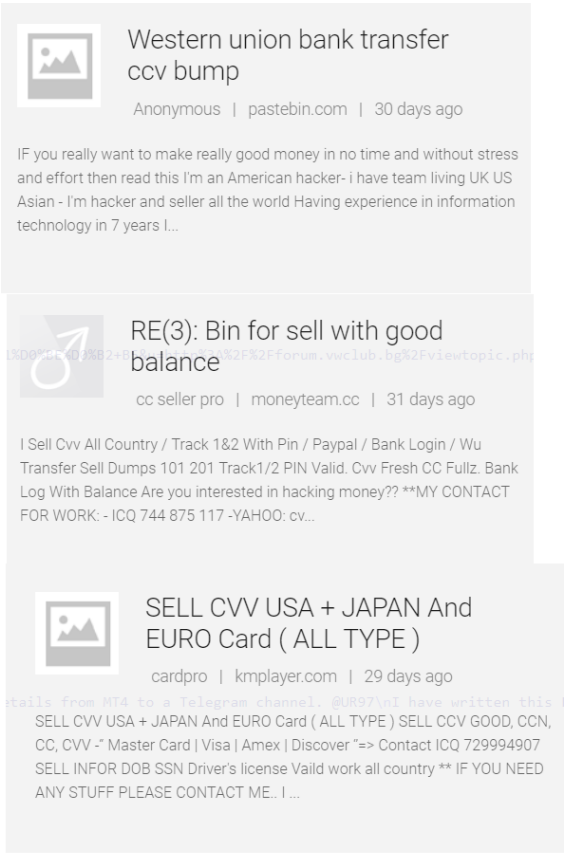

FIGURE 3.1 – Exemples de revendeurs de cartes de crédits sur des blogs et forums

La seconde étape est d'effectuer une détection de revendeurs de cartes sur ces réseaux de l'ensemble R_a. Ensuite, l'algorithme construit un graphe multimodal en inter-connectant les blogs et réseaux d'auto-promotion ainsi que les réseaux Post-it (ex : Twitter) et les réseaux personnels (ex : Facebook). Cela est fait par une méthode de résolution d'entités (em) sur les réseaux sociaux qui lie les comptes d'utilisateurs à travers plusieurs réseaux. La résolution d'entités permet de lier les différents comptes sur des plateformes différentes appartenant au même utilisateur. Alors qu'il existe plusieurs méthodes permettant d'effectuer la résolution d'entités sur les réseaux sociaux [184, 185, 186], nous avons adopté une méthode simple basée sur la détection d'identifiants uniques et sur les URL publiés. La raison derrière ce choix est que plusieurs blogs et réseaux de faible auditoire contiennent peu d'information pour pouvoir utiliser des méthodes avancées de résolution d'entités. Nous avons lié des utilisateurs ensemble s'ils ont publié la même adresse mail ou le même identifiant ICQ (I seek you). Ce dernier permet de se connecter à un programme de messagerie très populaire parmi les revendeurs de cartes de crédit. Nous avons également lié deux comptes si leur similarité de Jaccard calculée sur les URLs publiés est de plus de 0.5.

L'algorithme considère que si un nœud est classifié comme revendeur(classe 1) sur un réseau

Algorithm 3 Détection de revendeurs de cartes de crédit par interconnexion de réseaux

L :classifieur

em : méthode de résolution d'entités

R_a : réseaux de taille limitée (blogs et forums) et réseaux d'auto promotion (Youtube, Flickr, etc.)

R_p : réseaux Post it et réseaux personnels

1: f :[Générer un modèle de classification](R_a,L)
2: //Classifier les messages provenant de blogs et réseaux d'auto-promotion
3: Pour chaque g $\in R_a$:
4: Pour chaque nœud $n_i \in V_g$:
5: $y_i \leftarrow$ f(m)
6: // Modéliser les interactions sur les différentes plateformes par un graphe multimodal
7: $\mathcal{M} \leftarrow$[Effectuer une résolution d'entités entre les réseaux de R_a et R_p](R_a,R_p,em)
8: Pour chaque nœud n_i dans $V_{\mathcal{M}}$:
9: $y_i \leftarrow \max(\{y_j/(n_i, n_j) \in E_{identity} \vee j = i\})$
10: [Analyser les interactions sur les réseaux Post-it et personnels pour détecter les communautés de revendeurs]

ou blog, tous les comptes appartenant à ce nœud sont classé comme revendeurs (étape 9 de l'algorithme).

Enfin, la dernière étape est d'analyser les interactions sur les réseaux Post-it et personnels afin de détecter leurs voisins malveillant. Cela est effectué en entraînant un classifieur sur les comptes détectés et en classifiant les voisins de ces comptes. Ainsi, un utilisateur sur un réseau Post-it (ex : Twitter) est considéré comme malveillant s'il vérifie une des conditions suivantes :

1. La résolution d'entité le lie à un utilisateur classifié comme malveillant sur un blog réseau d'auto-promotion.

2. Il est classifié comme malveillant et il est connecté à un utilisateur détecté par la condition précédente.

3.2.2 Expérimentations

Nous présentons dans cette sous-section une expérimentation effectuée sur un jeu de données collecté sur Twitter et sur plusieurs milliers de blogs et de publications sur un réseau social d'auto-promotion qui est LiveJournal.

3.2.2.1 Collecte de données

Le processus de collecte de données est illustré dans la figure 3.2. La première étape de la collecte de données est de fournir des mots clés permettant de rechercher des publications sur l'agrégateur de blogs Webhose et sur Twitter. Ce dernier a été requêté en deux périodes à travers l'application Tweetcapt [16]. La première collecte a débuté le 29 juin 2015 à 12 :30 jusqu'au 1^er juillet à la même heure. La seconde collecte a débuté le 27 décembre à 16 :15 et s'est achevée le 7 janvier à 5 :00. Ces deux collectes ont permis de récupérer plus de 3.1 millions de Tweets. Les mots clés utilisés durant ces deux collectes sont mastercard, visa, amex, ccv, ccv2, paypal et dump.

La collecte sur Twitter introduit plusieurs biais. Le premier est lié à l'utilisation de mots clés dans l'API Streaming. En effet, tout revendeur qui n'utilise pas un des mots clés ci-dessus ne peut être détecté. Le second biais est lié à l'intervalle de temps de la collecte (environ 1 mois). Ainsi, des revendeurs ayant publié des message en dehors de cet intervalle de temps ne pourront être détectés. Cette expérimentation ne permet de détecter qu'un ensemble restreint de revendeurs de cartes de crédits sur Twitter ayant utilisé un mots clés ci-dessus dans l'intervalle de temps défini.

Webhose a été requêté deux fois avec les mêmes mots clés utilisés pour Twitter. La première collecte historique a été effectuée le 1^er juillet à 14 :30 et a permis de collecter 73k publications de 6k blogs et forums différents. La seconde collecte a été effectuée le 4 janvier à 10 :30 et a permis de récupérer 60k publications de 5k blogs et forums différents.

La collecte sur Webhose introduit également plusieurs biais. Le premier est lié à la plateforme qui ne récupérait pas, dans cette période, les données provenant du dark web. Ainsi, seulement les blogs et forums indexés par les moteurs de recherche ont été récupérés. Le second biais est lié à l'utilisation des mots clés, comme pour la recherche dans Twitter. Les revendeurs n'ayant pas utilisé ces mots clés ne pourront pas être détectés. Le troisième biais est lié à l'intervalle de temps. Webhose ne permet de récupérer que des messages publiés un ans plus tôt. Ce biais est moins important que pour Twitter où les messages sont récupérés en temps réel durant l'intervalle de temps de la collecte.

Les briques logiciel MySQL et MongoDB illustrées dans la figure 3.2 ont été utilisées pour stocker et requêter les données collectées. Les briques de résolution d'entités permettent de construire un réseau multimodal où chaque mode représente une plateforme sociale, et les interactions entre les modes représentent l'appartenance des comptes au même utilisateur.

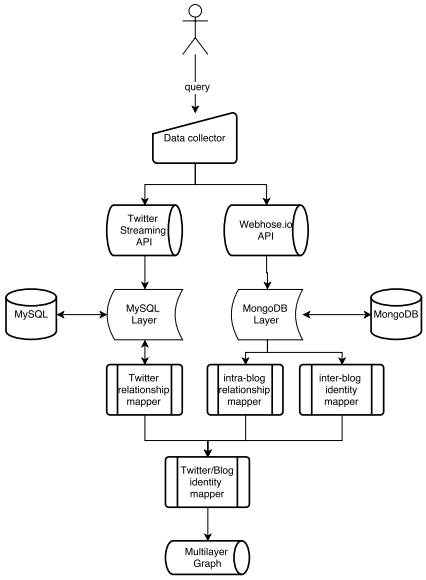

FIGURE 3.2 – Processus de collecte de données

3.2.2.2 Détection de revendeurs

La première étape de cette expérimentation est d'effectuer une classification binaire afin de détecter les revendeurs de cartes de crédit sur les blogs collectés à travers Webhose. Pour cela, un modèle sac de mots a été utilisé pour représenter le texte contenu dans ces publications.

Les forums et blogs sont ainsi représentés par des graphes unimodaux attribués. Un nœud représente un profil d'utilisateur dans un blogs ou forum et une arête indique que deux utilisateurs ont commenté sur la même discussion (thread) du forum ou blog. Chaque nœud est enrichi par un vecteur d'attributs textuels contenant les mots qu'il a utilisé dans ses messages. Un exemple de vecteur d'attribut pour le second utilisateur dans la figure 3.1 (le premier est anonyme) est comme suit : ((sell,2),(all,1),(country,1),(track,1),(pin,2),(paypal,1)...)

Le jeu d'entraînement a été labellisé manuellement, et contient 172 publications de revendeurs de cartes de crédit volées et 486 publications de non-revendeurs. Malgré la taille limitée du jeu d'entraînement, les publications sur les blogs sont explicites, permettant de séparer les différentes classes, contrairement aux publications sur Twitter. Nous avons testé plusieurs algorithmes de classification avec une méthode de validation croisée effectuée 10 fois. Les tests ont été effectués en utilisant la plateforme Weka 3.7.

TABLE 3.3 – Résumé du jeu d'entraînement

Indicateur	Value
# de positives	600
# de négatives	61,605
# de revendeurs	249
# d'utilisateurs	31,711
# de non revendeurs	31,710
# de blogs/forums	8,709
# de blogs/forums avec une publication de revendeurs	82
# de blogs/forums sans publication de revendeurs	8,627

TABLE 3.2 – Classification de revendeurs de cartes de crédit

Algorithm	Bien classés	Faux Pos.	Faux Neg.
Naive Bayes	95.28%	10.46%	2.67%
Logistic regression	98,32%	2.9%	1.23%
AdaBoost	96,50%	8,72%	1,64%
LibSVM	96.20%	13,37%	0.41%
J48	96.96%	5,81%	2.05%
RandomForest	98.17%	5.23%	0.2%

Nous avons sélectionné l'algorithme de la régression logistique pour effectuer cette expérimentation étant donné qu'il a permis d'obtenir les meilleurs résultats. Le tableau 3.3 fournit des informations sur le jeu de données ainsi que sur les prédictions obtenues par l'algorithme de classification. Nous pouvons remarquer que relativement peu de blogs sont utilisés par les revendeurs de cartes de crédit, permettant ainsi de surveiller ces derniers.

L'étape suivante est de représenter les données collectées par un réseau multimodal. On considère $\mathcal{M}$ un graphe multimodal composé des modes $L = \{l_1, l_2, ..., l_{8709}\}$. Chaque mode représente un blog ou un réseau social. $\mathcal{M}$ contient deux types d'arêtes. Les premiers sont des arêtes intra-modes qui indiquent que des utilisateurs ont commenté (like, retweet, reply, etc.) sur la même publication. Les seconds sont des interactions inter-modes qui indiquent que des comptes d'utilisateurs appartiennent à la même personne ou organisation. Ces interactions sont obtenues en effectuant une résolution d'entité sur ces blogs et réseaux sociaux.

La figure 3.3 illustre un sous-graphe G_{blogs} du graphe $\mathcal{M}$. Les nœuds n'ayant aucune interaction ont été omis de ce graphe afin d'en améliorer la visibilité. Les couleurs dans la figure 3.3 représentent les blogs et forums, et la taille des nœuds est proportionnelle à leurs degrés.

La figure 3.4 illustre les revendeurs de cartes de crédit dans G_{blogs} avec leurs degrés. Nous

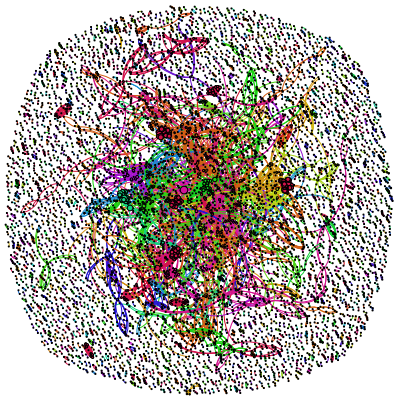

FIGURE 3.3 – Graphe G_1 illustrant les interactions sur les blogs et forums

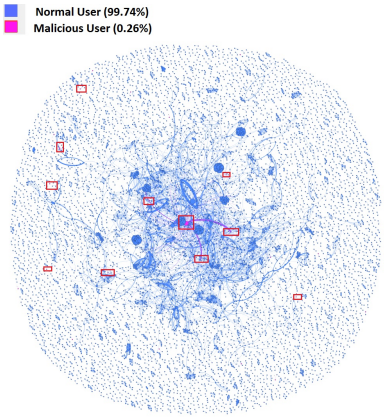

FIGURE 3.4 – Graphe G_1 illustrant les revendeurs de cartes de crédit volées

pouvons remarquer que peu de nœuds ont un haut degré et que les nœud à hauts degrés sont faiblement connectés entre eux. Cela indique la présence de plusieurs groupes différents parmi les revendeurs de cartes.

Le tableau 3.4 présente quelques indicateurs qui décrivent G_{blogs}. Nous pouvons voir que le degré moyen pondéré n'est pas très élevé par rapport au degré pondéré, ce qui indique que peu de nœuds interagissent ensemble régulièrement.

Après l'utilisation de la résolution d'entité à partir de l'ICQ, nous avons détecté 91 revendeurs de cartes de crédit sur des blogs qui étaient initialement classifiés comme non revendeurs par la régression logistique. Nous avons manuellement vérifié les 91 utilisateurs, et nous avons découvert qu'ils étaient tous revendeurs de cartes de crédit volées. Nous pouvons ainsi remarquer qu'une résolution d'entité basique a permis d'augmenter le rappel de 15%.

Nous avons également effectué une résolution d'entité par les identifiants email et les similarités d'URL. Cette deuxième résolution d'entités nous a permis de détecter seulement 10

TABLE 3.4 – Description of G_{blogs}

Indicateur	**Valeur**
Nombre de nœuds	8876
Nombre d'arêtes	15742
Degré moyen	3,08
Degré moyen pondéré	4,02
Nombre de composantes connexes	2847

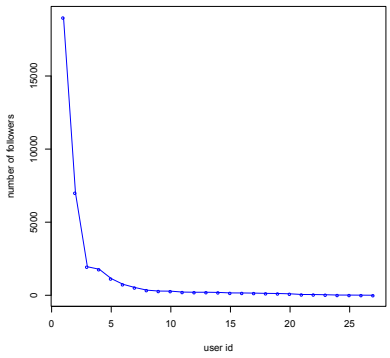

FIGURE 3.5 – Distribution des utilisateurs malveillants par nombre de followers

revendeurs de cartes de crédit volées, que nous avions déjà trouvés dans l'étape précédente.

Nous avons ensuite effectué une résolution d'entité entre les blogs et forum, et le jeu de données de 3.1 millions de Tweets collectés par TweetCapt. Nous avons détecté 31 utilisateurs malveillants qui ont publié sur les blogs et forums ainsi que sur Twitter. Tous ces utilisateurs ont été labellisés manuellement, et 30 sont des revendeurs de cartes de crédit volées et 1 utilisateur n'est plus présent sur Twitter (désabonnement ou suspension). Parmi les 30 utilisateurs détectés, 16 cachent la date de création de leur compte. Un utilisateur a créé son compte en 2008 et cinq autres ont créé leurs comptes en 2009. Les huit utilisateurs restants ont créé un compte avant 2013. Ainsi, nous pouvons voir que la majorité des utilisateurs ont été actifs durant plusieurs années sans être détectés par Twitter. Nous avons également remarqué que ces utilisateurs sont beaucoup moins explicites quant à leurs activités sur Twitter. Ils ont plusieurs publications qui ne concernent pas la revente de cartes de crédit volées.

La figure 3.5 illustre la distribution des utilisateurs malveillants détectés par le nombre de followers sur Twitter. Nous pouvons voir qu'un utilisateur a plus de 1.9k followers, qu'un second à 0.7k suiveurs et que 11 en ont plus d'une centaine. Ce nombre important de followers permet aux utilisateurs malveillants d'augmenter leur réputation et d'éviter la détection.

L'étape suivante a été d'analyser les interactions des utilisateurs malveillants sur Twitter afin de détecter s'ils sont connectés à des revendeurs de cartes de crédit. Nous avons utilisé une classification binaire sur un sous-ensemble des voisins des utilisateurs malveillants composé de

TABLE 3.5 – Nœuds malveillants par méthode de détection

Indicateur	**Valeur**
Logistic Regression	600
Entity mapping	111
Interactions	78

1k nœud. Le jeu d'entraînement est composé des revendeurs détectés à partir de la résolution d'entité. Nous avons détecté 78 revendeurs de cartes de crédit parmi leurs voisins.

Le tableau 3.5 résume les utilisateurs malveillants détectés dans chaque étape de l'algorithme proposé. Nous pouvons observer que la résolution d'entités et l'analyse des interactions ont permis d'augmenter largement le nombre de revendeurs détectés.

3.3 Conclusion

Les utilisateurs malveillants comme les revendeurs de cartes de crédit volées ont beaucoup de contraintes. L'une d'entre elles est le besoin de générer du profit poussant à diversifier les canaux qui leur permettent d'atteindre des clients. Encouragés par ce fait, ils publient souvent des messages explicites et répétitifs qui risquent d'être détectés par les plateformes sociales, si cette détection a peu d'impact sur leur auditoire. En effet, plusieurs plateformes sociales et blogs ont des mesures de sécurité souples permettant aux revendeurs de cartes de crédit d'être explicites. D'autres plateformes sont des plateformes d'auto-promotion où le contenu est souvent recherché à partir de mots clés. Dans ces plateformes, la détection n'a pas d'effets sur les revendeurs, étant donné que leurs listes de contacts ne servent pas à disséminer leurs contenus. Ainsi, les revendeurs de cartes volées peuvent être beaucoup plus explicites sur ces plateformes.

Il est souvent contraignant pour des revendeurs de cartes de crédit d'utiliser un email différent dans chaque blog, forum ou réseau social. Les utilisateurs malveillants ont ainsi tendance à réutiliser certains identifiants (même adresse email et/ou même identifiant ICQ,Skype, etc.). Ces identifiants sont nécessaires à leurs activités étant donné qu'ils permettent aux clients potentiels de les contacter. Ainsi, il est possible d'utiliser ces identifiants afin de connecter leurs différents comptes. La méthode proposée modélise les interactions d'utilisateurs par un graphe multimodal où chaque mode représente un réseau social différent ou blog.

La méthode proposée détecte les revendeurs de cartes de crédit sur les plateformes où ils sont explicites. Elle extrait ensuite des identifiants lui permettant de retrouver leurs comptes utilisateurs sur des plateformes tels Twitter où ces utilisateurs sont moins explicites. L'algorithme classifie ensuite la liste de contacts des utilisateurs, étant donné que plusieurs d'entre

eux se suivent pour augmenter leurs réputation.

La détection d'utilisateurs malveillant sur les réseaux ou forums à faible auditoire et faible sécurité ou sur des réseaux d'auto-promotion obligera ces derniers à être beaucoup moins explicites et répétitifs. Cela permettra à long terme de réduire leur visibilité, et ainsi ajouter des contraintes à leurs activités.

Chapitre 4

Classification non supervisée dans les réseaux sociaux

Nous proposons dans ce chapitre deux approches de classification non supervisée qui permettent de détecter des communautés dans les réseaux sociaux.

Une des motivations pour le développement de nouvelles méthodes de classification non supervisée dans les réseaux sociaux est l'amélioration de la scalibilité de la détection de communautés. Les méthodes utilisant la structure globale du graphe ont souvent des complexités qui dépassent N^2 (N : nombre de nœud) [96, 103, 187]. Cela encourage l'utilisation de méthodes se basant seulement sur les structures locales pour affecter des nœuds à des communautés. L'utilisation de la structure globale d'un graphe peut néanmoins améliorer la détection de communautés, comme il a été montré dans plusieurs études [96, 188, 189, 109]. Une seconde motivation est de permettre la détection de petites communautés dans le graphe. En effet, plusieurs méthodes se basant sur l'optimisation de la modularité ou l'analyse spectrale ignore les petites communautés qui peuvent être d'un grand intérêt (ex. communautés de revendeurs de cartes et communautés de spammeurs).

La première section de ce chapitre introduit un algorithme de détection de communautés se basant sur les marches aléatoires nommé "SpongeWalker". Cet algorithme utilise la structure locale du graphe dans une première étape afin de combiner des nœuds ayant un forte densité d'interactions. Il utilise ensuite la structure globale du graphe pour générer un dendrogramme. La première étape qui est peu coûteuse en temps de calcul (O(n)$\leq$ n.log(n)) réduit la taille du graphe en combinant les nœuds qui ont une forte probabilité d'appartenir à la même communauté. Cela permet ensuite d'effectuer des calculs plus coûteux qui font usage de la structure globale sur un graphe de taille réduite en utilisant des représentants sélectionnés. L'algorithme combine ainsi deux stratégies qui réduisent sa complexité (réduction de la taille du graphe et utilisation de représentants). Cela lui permet d'éviter de se baser uniquement sur la réduction de la taille du graphe pour réduire sa complexité. Cette stratégie peut en effet conduire à la gé-

nération de partitions similaires à celles obtenus par les algorithmes d'optimisation de mesures de densité. L'algorithme évite également d'utiliser un nombre trop faible de représentants qui peut dégrader la qualité de partitions. SpongeWalker retourne un denrogramme permettant de détecter des communautés de petite taille.

Une troisième problématique majeure, liée à la détection de communautés dans les réseaux sociaux, est la caractérisation de ces dernières. En effet, contrairement à la classification supervisée où les classes sont prédéfinies, la classification non supervisée nécessite la caractérisation des classes qui ont été générées. Plusieurs méthodes peuvent être considérées pour cette caractérisation, comme l'étiquetage manuel de représentants de classes, l'utilisation de statistiques liées aux attributs d'utilisateurs [59] ou l'utilisation de distances entre les nœuds d'une communauté [56, 190]. Dans cette thèse, nous nous focalisons sur la première méthode, qui nécessite d'identifier des représentants de classes qui peuvent la caractériser. Cette problématique est traitée dans la deuxième section de ce chapitre qui introduit une méthode de détection de communautés basée sur le pattern mining.

La section 2 introduit un algorithme de détection de communauté basé sur le pattern mining nommé "WalkMiner". Cet algorithme détecte des patterns qui sont considérées comme des représentants de communautés. Il permet d'obtenir des partitions de haute qualité sur des réseaux sociaux divers, incluant des réseaux de type Post-it (ex. Twitter). Ces derniers sont souvent difficiles à partitionner par des algorithmes classiques d'optimisation de densité à cause de leur distribution de degrés en loi de puissance et de leur faible densité [14]. WalkMiner permet de générer un ensemble réduit de patterns qui caractérisent les communautés d'un réseau social. Ainsi, l'algorithme présenté résout un des problèmes majeurs lié au pattern mining sur les graphes qui est la génération d'un nombre de patterns très élevé, pouvant dépasser le nombre de nœuds. Nous introduisons également une mesure basée sur le principe de description à taille minimale (minimum description length) pour affecter les nœud aux communautés. Cette dernière permet la détection de communautés de tailles différentes. Enfin, cette section introduit deux méthodes basées sur l'algorithme de pattern mining proposé qui détectent des communautés sur des graphes avec arêtes attribuées et sur des graphes disjoints.

4.1 SpongeWalker : détection de communautés dans les graphes

Il existe plusieurs méthodes qui réduisent la taille des réseaux avant de les analyser [119, 99, 14]. Certaines de ces méthodes sont basées sur l'échantillonnage [139, 140, 141], alors que d'autres sont basées sur la fusion de nœuds [142, 8]. Nous allons introduire dans cette section un algorithme de détection de communautés nommé "SpongeWalker", qui utilise les structures locales pour fusionner des nœuds et ensuite la structure globale du graphe. Cet algorithme permet ainsi d'éviter le problème de résolution dont souffre plusieurs mesures de détection de commu-

nautés par la densité, étant donné qu'il ne regroupe dans sa première étape (étape de détection de communautés basée sur la densité) que des nœuds fortement connectés. Il permet également de réduire la complexité, souvent élevée, de la détection de communauté utilisant la structure du graphe. Il effectue cela en calculant les mesures couteuses sur un graphe de taille réduite obtenu par la détection de communauté basée sur la densité, et en utilisant des représentants pour effectuer les calculs couteux. Cette approche nous permet de réduire la complexité de l'algorithme de détection de communauté .

4.1.1 Méthodologie

L'algorithme 4 explique le fonctionnement de SpongeWalker. Ce dernier commence par effectuer une détection de communautés basée sur la densité. Cette première étape est suivie par la fusion des nœuds affectés aux mêmes communautés, obtenant ainsi un graphe de taille réduite. L'algorithme sélectionne ensuite des représentants de communautés dans le graphe, et calcule des distances entre chaque paire de nœuds connectées en utilisant des marches aléatoires vers les représentants. Enfin, l'algorithme fusionne agglomérativement les communautés en minimisant la variance de distances structurelles (méthode de Ward [191]).

A notez que SpongeWalker commence par exécuter la première itération de l'algorithme de Louvain [98] en optimisant une mesure de densité qui regroupe des nœuds dans des communautés, puis combine les nœuds affectés aux mêmes communautés. L'algorithme utilise ensuite les distances entre les nœuds basées sur les marches aléatoires comme dans Walktrap [96] pour générer un dendrogramme. Il utilise cependant des représentants comme attributs des nœuds pour calculer ces distances afin de réduire la complexité de l'algorithme.

Les sous-sections 4.1.1.1 à 4.1.1.8 détaillent les differentes fonctions de l'algorithme 4.

4.1.1.1 Détection de communauté basée sur la densité :

La première étape de l'algorithme optimise une mesure de qualité de partitions qui se base sur la densité locale afin de fusionner des nœuds fortement connectés. Ainsi, cette partie consiste à réduire la taille de l'ensemble V des nœuds ainsi que de l'ensemble E des arêtes du graphe g.

Plusieurs mesures basées sur la densité peuvent être utilisées dans cette étape. Une des premières mesures envisagées est la modularité [91] (eq 2.2). Alors que cette dernière est souvent utilisée pour la détection de communautés sur les graphes, elle peut combiner des communautés de petite taille (problème de résolution de graphes)[94]. [95] propose une mesure inspirée de la modularité se dénommant "modularité densité" (équation 2.3) pour répondre à ce problème. Cependant, cette mesure ne permet pas de comparer des partitions ayant des nombres de communautés différents.

Algorithm 4 Sponge Walker

g : graphe simple

numRep : nombre de représentants (par défaut $\sqrt{N}$)

K : taille des voisinages utilisés pour sélectionner des représentants (par défaut 5)

t : nombre d'étapes pour le calcul de marches aléatoires

```
c← [Détection de communautés basée sur la densité](g)
g_s ← [Fusion des nœuds de la même communauté](g,c)
Rep←[Sélection des représentants](g_s,numRep,K)
M^t ←[Calcul de t marches aléatoires](g_s,t)
D ←[Calcul des distances structurelles entre communautés voisines](M^t,g_s,Rep)
Répéter
    c_i, c_j ←[Sélection des deux communautés voisines les plus proches](D)
    c, D ←[Fusion des communautés](C, D, c_i, c_j)
Tant que |c|>1
meilleure partition ←[Sélection de la meilleure partition]
retourner dendrogramme, meilleure partition
```

Une autre mesure qui utilise la structure locale d'un nœud est le test de Chi-carré (équation 4.1).

$$\mathcal{X}^2 = \sum_i \sum_j (a_{ij} - WE(a_{ij}))^2 / WE(a_{ij}) \tag{4.1}$$

La valeur de $WE(a_{ij})$ représente la somme des poids attendus entre le nœud i et j. Il existe plusieurs choix pour cette valeur comme par exemple le modèle nul utilisé pour le calcul de modularité $\frac{deg(n_i) \cdot deg(n_j)}{2M}$.

Cette mesure souffre du même problème de résolution identifié par [94] pour la modularité [91]. Cela est dû à la pénalité du modèle nul, dont la valeur baisse en fonction de la taille du graphe. Nos expérimentations ainsi que celles effectuées par [136] ont montré que cette mesure permet d'obtenir des communautés de tailles plus petites que celles obtenues par l'optimisation de la modularité [91].

Duan et al. [136] ont testé le rapport de vraisemblance pour la détection de communautés. Ce dernier est défini par l'équation 4.2. Les expérimentations effectuées par Duan et al. [136] ont montré que les tailles des communautés retournées par l'optimisation du rapport de vraisemblance sont souvent plus petites que celles retournées par la modularité, et plus grandes que celles retournées par le Chi-carré simplifié.

$$Likelihood = Pr(tp, o, n) / Pr(ep, o, n) \tag{4.2}$$

$$Pr(p, o, n) = C_n^o \cdot p^o (1-p)^{(n-o)} \tag{4.3}$$

Concernant les réseaux orientés, il n'existe pas encore de consensus dans la communauté scientifique sur une définition de la densité. Plusieurs variations de la modularité ont été proposées pour le partitionnement de ces réseaux.

Leicht et al. [104] ont proposé une mesure de modularité pour les réseaux orientés qui prend en compte l'orientation pour le calcul du modèle nul (eq 2.7). Cette mesure introduit un biais où l'orientation d'une arête a un effet trop important sur les valeurs de modularité.

Kim et al. [106] proposent une autre adaptation de la modularité dans des réseaux orientés utilisant les marches aléatoires (eq 2.8). Cette mesure utilise la distribution stationnaire des nœuds pour calculer le modèle nul. Elle est plus robuste que [104] étant donné qu'un changement de quelques arêtes ne modifie pas complètement les partitions obtenues.

Nous avons suivi les recommandations de [136] qui ont comparé le rapport de vraisemblance, la modularité, le Chi-carré et un rapport entre le nombre de liens et le modèle nul (rapport de probabilités). Ces derniers recommandent d'utiliser le rapport de vraisemblance si aucune information n'est disponible sur le jeu de données. Aussi, le rapport de vraisemblance permet d'obtenir des communautés de taille plus petites que celles obtenues par la modularité [136], mais plus grandes que celles obtenues par le Chi-carré ou le rapport de probabilité. Souvent, l'optimisation de la dernière mesure ne permet pas d'effectuer une réduction suffisamment importante du graphe pour impacter la complexité de SpongeWalker.

4.1.1.2 fusion des nœuds de la même communauté :

Dans cette étape, les nœuds affectés à la même communauté dans l'étape précédente sont fusionnés. Les nœuds résultants représentent ainsi les communautés détectés dans l'étape précédente. Ainsi, le poids d'une arête entre deux communautés est la somme des poids des arêtes entre chaque paire de nœuds qui les composent. Les arêtes entre les nœuds appartenant à la même communauté sont représentées par des boucles.

Cette étape permet de générer un graphe g_s dont le nombre de nœuds correspond au nombre de communautés (|C|) détectés dans l'étape précédente et le nombre d'arêtes est borné par $|C|^2$.

4.1.1.3 Sélection des représentants :

Cette étape concerne la sélection de représentants qui seront utilisés comme attributs pour le calcul de distances entre les nœuds du graphe. Les marches aléatoires permettent de caractériser chaque nœud par un vecteur de nœuds. Ce vecteur correspond à la probabilité d'atteindre chaque nœud après t marches. Ces vecteurs sont obtenus en multipliant t fois la matrice stochastique $\mathcal{M}$ générée à partir de la matrice d'adjacence A (équation 2.4).

Le calcul de similarités entre chaque paire de nœuds connectée, en utilisant tout le vecteur de nœuds (stratégie de Walktrap), peut être coûteuse et n'est souvent pas nécessaire pour dé-

terminer la similarité entre les nœuds. Par exemple, on considère deux nœuds n_1 et n_2. Ces deux nœuds sont représentés par deux vecteurs v_1 et v_2 indiquant la probabilité de transition de chaque nœud vers tous les nœuds du graphe. Les vecteurs ont une taille N qui est le nombre de nœuds dans le graphe. Ainsi, la distance entre n_1 et n_2 est calculée en utilisant les N éléments d'un graphe. Pour calculer la distance entre chaque paire de nœuds connectés, la complexité serait de $O(\frac{E \cdot N}{2}) > O(N^2)$. Il faut donc O(E) pour parcourir chaque paire de nœuds connectée (les nœuds non connectés ont une distance de 0), et pour chaque paire, effectuer un calcul sur un vecteur de N éléments (O(N)).

Il est possible d'identifier des leaders de communautés (représentants) et de les utiliser pour calculer les distances entre les nœuds. Les nœuds seront ainsi représentés par des vecteurs de taille |Rep|, où Rep est l'ensemble des représentants. Ces vecteurs indiqueront la probabilité de transition d'un nœud vers le représentant. Pour calculer la distance entre chaque paire de nœuds connectés, la complexité serait de $\frac{O(E)}{2} \cdot R$.

Wang et al. [14] présentent deux méthodes qui permettent de sélectionner des représentants de communautés sur les graphes. La première méthode, qui est une méthode d'optimisation gloutonne (Greedy optimisation), a une faible complexité. La compléxité dans le pire cas de l'algorithme est $O(numRep \cdot K \cdot E)$ quand le graphe est complet (tous les nœuds sont adjacents). Le nombre d'arêtes par nœuds est souvent considérée comme étant de l'ordre de log(N). On peut considérer que la complexité moyenne est $O(numRep \cdot K \cdot log(N))$. Les auteurs de l'algorithme [14] ont considéré qu'empiriquement, la complexité moyenne est dans l'ordre de $O(M + N)$. La seconde méthode obtient de meilleurs résultats d'après les expérimentations effectuées par [14], mais nécessite une condition d'arrêt fournie à l'algorithme en paramètre. Nous utilisons une variante de la première méthode, étant donné que les représentants ne sont utilisés que pour calculer des mesures de distance entre nœuds, et ne sont donc pas des racines de communautés comme dans [119, 99, 14].

La sélection des racines est décrite par l'algorithme 5. L'algorithme commence par sélectionner aléatoirement un nœud. Ensuite, K itération sont effectués, où un nœud voisin est ajouté à une liste S. Le nœud qui est ajouté est celui qui maximise la modularité avec l'ensemble S. L'ensemble S est ainsi composé des nœuds ayant une forte probabilité d'appartenir à la communauté du nœud échantillonné aléatoirement. La dernière étape de l'algorithme est de sélectionné le nœud à degré maximal dans S pour le rajouter à l'ensemble des représentants Rep.

La différence entre cet algorithme et celui présenté dans [14] est que nous maximisons la modularité pour sélectionner un représentant alors que [14] sélectionne le représentant avec le plus haut degré. Notre méthode permet d'éviter un biais trop élevé pour les communautés de grande taille, ce qui n'est pas toujours le cas dans la méthode présenté dans [14].

Algorithm 5 Sélection de représentants

numRep : nombre de représentants

K :Taille du voisinage pour la sélection d'un représentant

Rep :{}

Répéter :
    S← nœud aléatoire
    Répéter K fois :
        R←[Voisin de S maximisant le gain de modularité avec S]
        S ← S ∪ R
    Rep ← Rep ∪ nœud avec degré max(S)
Tant que |*Rep*|<*numRep* :
retourner Rep

4.1.1.4 Calcul de t marches aléatoires :

Cette étape consiste à calculer t marches aléatoires à partir de la matrice d'adjacence du graphe g_s. Notez que les marches aléatoires sont calculés à partir d'un graphe à taille réduite. Cela réduit significativement la complexité de SpongeWalker étant donné que le nombre de nœuds dans g_s est souvent inférieur au nombre de nœuds dans le graphe initial g.

Les marches aléatoires sont calculées à partir de la matrice d'adjacence A_s du graphe g_s. La première étape de ce calcul est de normaliser la matrice A_s par la somme des lignes de la matrice, transformant cette dernière en matrice stochastique (équation 4.4).

$$M_{i,j} = \frac{a_{i,j}}{\sum_{j \in V} a_{i,j}} \tag{4.4}$$

L'étape suivante est de multiplier t fois la matrice M. La matrice M^t qui est obtenu défini les probabilités de transition de chaque nœud après t sauts.

4.1.1.5 Calcul des distances structurelles entre communautés voisines :

Cette phase consiste à calculer des distances entre chaque paire de nœuds connectés sur le graphe g_s. La mesure de distance utilisée est similaire à celle présentée dans Walktrap [96], avec la différence que seuls les représentants sont utilisés comme attributs. L'utilisation de ces représentants rend ainsi l'algorithme beaucoup plus rapide que Walktrap, sans une grande perte dans la qualité des communautés, comme il sera montré dans les expérimentations.

L'équation 4.5 décrit la distance calculée entre les paires de nœuds connectés. Ainsi, la distance entre les nœuds i et j est la somme des distances entre les probabilités de transition des deux nœuds vers les représentants, normalisée par la racine carré du degré du représentant.

$$\mathcal{D}_{i,j} = \sum_{k \in Rep} \frac{m^t_{i,k} - m^t_{j,k}}{\sqrt{deg(k)}} \tag{4.5}$$

Notez que dans cette étape, les nœuds représentent une communauté générée par la détection de communautés basée sur la densité. Ainsi, la distance $\mathcal{D}_{i,j}$ entre les nœuds i et j du graphe g_s représente la distance entre les communautés i et j du graphe g.

4.1.1.6 Sélection des deux communautés voisines les plus proches :

SpongeWalker fusionne dans chacune de ces étape les deux communautés les plus proches. Cette étape consiste à sélectionner ces communautés. Nous avons suivis les recommandations de [96] et avons utilisé un arbre de recherche binaire pour enregistrer les distance entre les différentes communautés. Cette étape consiste donc à retrouver les communautés les plus proches en utilisant les distances calculés dans l'étape précédente. Notez que les communautés non voisines (ayant aucune arête incidente) ne sont pas envisagées dans cette étape.

4.1.1.7 Fusion des communautés :

L'algorithme fusionne les communautés les plus proches et met à jour les distances entre les communautés. Comme dans Walktrap [96], après chaque fusion, les entrées des nœuds fusionnés dans la matrice de distances sont combinées par une moyenne pondérée. Cette dernière est calculée par l'équation 4.6, avec N_i la taille de la communauté i, c_1 et c_2 les communautés fusionnées, et c_3 la communauté créée par la fusion.

$$\mathcal{D}_{c_3,c_j} = \frac{N_1 \cdot \mathcal{D}_{c_1,c_j} + N_2 \cdot \mathcal{D}_{c_2,c_j}}{N_1 + N_2} \tag{4.6}$$

4.1.1.8 Sélection de la meilleure partition

Les étapes précédentes produisent un dendrogramme qui peut être utilisé pour obtenir plusieurs communautés selon les besoins des utilisateurs. Il existe plusieurs méthodes qui permettent de sélectionner "la meilleure" partition dans un dendrogramme. Si le nombre de communautés n'est pas fourni, cet algorithme utilise la modularité [92] pour sélectionner cette partition. D'autres mesures peuvent également être utilisées ([136, 104])), mais nous utilisons la modularité étant donné que c'est la mesure qui a été la plus étudiée.

4.1.1.9 Complexité

Afin de pouvoir évaluer la complexité de SpongeWalker, il est nécessaire de calculer la complexité de chacune des étapes décrites dans la section 4.1.1.

La complexité exacte pour la détection de communautés basée sur la densité (étape 1) est la même que la complexité d'algorithme d'optimisation de Louvain [98] avant sa première fusion de nœuds. L'algorithme de Louvain dans le pire des cas n'est pas borné. Ainsi, sa complexité est la même que pour l'optimisation de modularité qui est un problème NP complet. Les expérimentations effectuées sur l'algorithme de Louvain suggèrent une complexité moyenne de $O(N.log(N))$ [98].

La complexité de la phase de sélection de représentants est de $O(M+N)$ d'après les auteurs de l'algorithme qui l'ont évalué empiriquement [14]. Le pire des cas se produit quand le graphe est complet et la complexité est $O(numRep \cdot K \cdot log(N))$, avec K un paramètre utilisateur indiquant la taille du voisinage utilisé pour sélectionner des représentants et $numRep$ le nombre de représentants.

Le calcul de distances dans l'étape 5 se fait sur le graphe retourné de l'étape 2. Le pire des cas se produit quand l'optimisation basée sur la densité ne permet de fusionner aucun nœud (cas rare se produisant quand la modularité du graphe est très faible). Ainsi, le calcul des t marches aléatoires peut être estimée à $O(MNt)$, avec N le nombre de nœuds, M le nombre d'arêtes et t le nombre d'étapes utilisés pour le calcul des marches. La complexité liée à la construction du dendrograme est quant à elle $O(M \cdot h \cdot numRep)$ selon les travaux de [96], avec h la hauteur du dendrogramme.

4.2 WalkMiner : Détection de communautés dans les graphes en utilisant les patterns de contraste

Alors que le pattern mining est très populaire pour le clustering de bases de données, cette approche n'a pas reçu l'attention méritée concernant la détection de communautés dans les graphes. Une des raisons est que la communauté scientifique lui a souvent reproché d'avoir une complexité trop élevée. D'autres raisons sont liées au fait que la réussite des méthodes spectrales ainsi que des méthodes d'optimisation de modularité a éclipsé plusieurs familles de méthodes [187]. Aussi, les méthodes basées sur le pattern mining nécessitent de spécifier des paramètres comme le support minimal qui ne sont souvent pas intuitifs à choisir.

Les méthodes de pattern mining peuvent néanmoins offrir plusieurs avantages pour la détection de communautés. Contrairement aux méthodes spectrales, le pattern mining facilite la caractérisation des communautés retournées. Cela est très important quand l'utilisateur cherche des communautés spécifiques (ex : spammeurs, apologistes au jihad, etc.). Aussi, les méthodes basées sur le pattern mining peuvent facilement justifier l'affectation d'un utilisateur à une communauté. Enfin, l'identification de patterns d'interaction peut faciliter la détection de communautés, spécialement dans les graphes clairsemés.

Nous proposons dans cette section un triptyque d'algorithmes de détection de communautés dans les réseaux sociaux basé sur les patterns mining (WalkMiner-simple, WalkMiner-edgeAtts, WalkMiner-DisGraphs). L'algorithme WalkMiner-simple utilise les patterns de contraste (contrast patterns) [192] permettant de détecter des petites communautés si leurs patterns d'interaction diffèrent des communautés voisines. Ces dernières sont souvent agrégées dans des communautés plus grandes par les algorithmes d'optimisation de modularité [102] ou les algorithmes de pattern mining. Les algorithmes de pattern mining ne sont souvent pas adaptés à la détection de petites communautés étant donné que les patterns caractérisant ces dernières peuvent avoir un support en dessous du support minimal spécifié (minSup). La détection de ces petites communautés nécessiterait de réduire le support minimal, ce qui peut augmenter significativement la complexité de l'algorithme. Dans plusieurs cas, la réduction du support minimal peut rendre un algorithme de pattern mining inutilisable, même pour des graphes de taille réduite.

L'algorithme proposé génère pour chaque nœud une liste de voisins (1) structurellement proches et (2) ayant une haute centralité locale. Ces voisins sont des candidats pour devenir des représentants de communautés. Ainsi, au lieu d'utiliser la liste d'adjacence pour détecter des patterns d'interaction, une liste de représentants structurellement proches des nœuds est utilisée. Cela facilite la détection de petite communautés en rapprochant les représentants de communautés des nœuds. L'utilisation de la liste d'adjacence nécessiterait de définir un support minimal très bas pour détecter des patterns représentant des petites communautés. Aussi, la liste de voisins structurellement proches élimine les patterns ayant un support très élevé. Ces dernières apparaissent dans le voisinage d'un nombre élevé de nœuds et ne conviennent pas à la détection de communautés. Ainsi, l'utilisation de la structure du graphe afin de détecter des voisins de nœuds encourage la génération de patterns discriminantes (patterns de contraste). Un troisième avantage lié à cette approche est que les nœuds dans des graphes clairsemés (ou régions clairsemées d'un graphe) peuvent être caractérisés par un nombre plus élevé de voisins que le nombre de voisins incidents (partageant une arête). Cela permet de détecter des patterns discriminantes même dans des graphes clairsemés avec peu d'interactions (ou régions clairsemées d'un graphe).

WalkMiner-simple adapte un algorithme d'élagage nommé DDPMine [193] introduit pour la classification supervisée. Dans chacune de ces itération, DDPMine identifie les meilleures patterns et supprime les transactions (nœuds dans le cas échéant) couvertes par K patterns. La suppression de transactions après chaque itération permet à l'algorithme de réduire les supports des patterns et de retourner un ensemble restreint de patterns discriminantes, parmi un nombre exponentiel de patterns qui vérifient la condition du support minimal [193]. Nous adaptons DDPMine pour le contexte non supervisé de la détection de communautés. Pour cela, nous proposons une mesure pour l'identification des meilleures patterns ainsi qu'une seconde mesure pour la sélection des transactions à supprimer. La version modifiée de DDPMine que nous

nommons DDPMine-utility permet à WalkMiner-simple d'effectuer une détection de communautés en utilisant un nombre restreint de patterns discriminantes. Cela permet ainsi d'utiliser l'algorithme sur des graphes de grande taille.

Nous proposons une extension de WalkMiner-simple pour les graphes avec arêtes caractérisées par des attributs textuels (WalkMiner-edgeAtts) ainsi que pour les graphe ayant plusieurs composantes connexes (WalkMiner-disGraphs). La première extension utilise Twitter-LDA afin de détecter des thèmes pour chaque arête. Ces thèmes sont ensuite utilisés afin de générer un graphe multimodal pondéré par la distribution de thèmes des nœuds. WalkMiner-simple est ensuite exécuté sur le graphe multimodal afin de détecter les communautés.

Concernant les graphes attribués avec plusieurs composantes connexes, WalkMiner-simple ou WalkMiner-edgeAtts est initialement exécuté sur chacune des composantes. Ensuite, nous proposons une régularisation qui rapproche les attributs des communautés partageant un grand nombre d'interactions. Le paramètre de régularisation baisse si le nombre d'arêtes intra-communautés est élevé. Ce paramètre de régularisation ne pénalise pas nécessairement des communautés sur plusieurs composantes connexes si leurs composantes interagissent avec des communautés ayant des attributs similaires, ou si leur nombre d'interactions intra-communauté dépasse largement leurs interactions inter-communautés.

Cette section est organisée comme suit : Nous commençons d'abord par présenter la notation utilisée. Ensuite, nous introduisons l'algorithme WalkMiner-simple pour les graphes simples. Les sections suivantes décrivent les algorithmes WalkMiner-edgeAtts et WalkMiner-DisGraphs. La section 4.3 présente des expérimentations en vue d'évaluer les performances des algorithmes proposés.

4.2.1 Notation

Un graphe g=(V,E) est composé d'un ensemble de nœuds V et d'un ensemble d'arêtes E. Un graphe peut être décrit par une matrice d'adjacence A. L'entrée $a_{i,j}$ de la matrice A fait référence à la ligne i et la colonne j de la matrice. Nous utilisons cette notation pour décrire toutes les matrices générées par l'algorithme.

Le pattern mining a été conçu pour la détection de tendances dans les tables de bases de données [123]. La taxonomie qui est suivie s'inspire donc de celle utilisée dans la gestion des bases de données. Les algorithmes de pattern mining prennent en entrée des tables qui peuvent être modélisées par des matrices. Une ligne de la matrice est appelée une "transaction" et une colonne non nulle est appelée un "item". Une pattern est un ensemble d'items, et la fonction supp calcul le support d'une pattern dans la base de données.

On considère que les patterns retournées par un algorithme de pattern mining appartiennent à l'ensemble P. Nous allons utiliser des lettres latines en majuscule pour faire référence à ces patterns. Concernant les items qui composent ces patterns, nous allons les indiquer par

des lettres latines en minuscule. Les transactions (ligne de la table) seront référencées par un indice (i, j, etc.). Il faut noter qu'une transaction est une ligne obtenue à partir d'une matrice décrivant la structure du graphe. Cette dernière est générée à partir de la matrice d'adjacence ou d'une transformation effectuée sur la matrice d'adjacence. Ainsi, une transaction représente les attributs structurels d'un nœud dans le graphe.

On dit qu'une pattern vérifie une transaction si tous les items dans la pattern sont contenus dans la transaction. En d'autres termes, toutes les colonnes représentées par les items sont non nulles dans la transaction.

L'utilité U d'une transaction est un concept qui provient de la fouille d'utilités (utility mining) [194, 195, 196]. La fouille d'utilités est un sous-domaîne du pattern mining effectué quand les transactions ou les items ont une valeur (utility) ou un coût. L'objectif est de détecter les patterns qui maximisent cette valeur. Dans cette section, nous effectuons du pattern mining sur une matrice ayant des valeurs continues. L'utilité d'une pattern fait référence à une valeur obtenue à partir des utilités de ces items (décrite dans WalkMiner-simple), et non à la somme des utilités des transactions qui vérifient cette pattern [194, 195].

4.2.2 WalkMiner-simple : Détection de communautés dans les graphes simples

L'algorithme 6 décrit WalkMiner-simple qui détecte des communautés sur des graphes simples. L'algorithme commence par effectuer des marches aléatoires. Les travaux de [96, 12] indiquent que des nœuds d'une même communauté tendent à converger vers les mêmes ensembles de nœuds dans des marches aléatoires courtes. L'étape suivante est de détecter des nœuds centraux dans leurs voisinages. Pour cela, seuls les nœuds ayant une forte centralité relative sont considérés, afin de rendre l'algorithme adapté à des graphes de grandes tailles. La troisième étape concerne la construction d'un vecteur d'attributs pour chaque nœud à partir des nœuds centraux avec lesquels ils ont une forte probabilité de transition. La méthode de génération de patterns utilise une variante de l'algorithme DDPMine [193] que nous introduisons dans l'étape 4. Cet algorithme permet de détecter un faible nombre de patterns peu redondantes qui représentent des communautés, contrairement aux méthodes qui utilisent FPGrowth [88] ou une de ces variantes. A partir de la cinquième étape, l'architecture de la méthode proposée s'inspire de la méthode de clustering de données non connectées à haute dimension de Fore et Dong [192]. Les mesures optimisées ont cependant été adaptées pour la détection de communautés dans les graphes. La méthode de génération de racines de communautés quant à elle est une adaptation de la méthode de sélection des K patterns les plus représentatives d'un jeu de données introduite par [197].

Algorithm 6 Pattern mining pour la détection de communautés dans les réseaux sociaux

g : graphe simple
minSup : support minimal
K : nombre de racines (par défaut $\frac{N}{minSup}$)
t : taille des marches aléatoires

1: $M^t \leftarrow$[Effectuer des marches aléatoires courtes](g,t)
2: $C \leftarrow$[Détecter les nœuds centraux](g)
3: $H \leftarrow$[Générer un vecteur d'attributs pour chaque nœud du graphe](C,M^t)
4: $P \leftarrow$[Effectuer une détection de patterns sur le vecteur d'attributs](H,minSup)
5: $\mathbb{R} \leftarrow$[Générer K racines des communautés dans le réseau](P,K)
6: $\mathbb{C} \leftarrow$[Affecter les patterns restantes aux racines](P,$\mathbb{R}$)
7: $\mathbb{C} \leftarrow$[Affecter les nœuds aux clusters de patterns](g,$\mathbb{C}$)
8: dendrogramme$\leftarrow$[Générer un dendrogramme en fusionnant itérativement les clusters les plus proches]
9: retourner dendrogramme, meilleure partition

4.2.2.1 Effectuer des marches aléatoires courtes :

Dans cette étape, nous effectuons des marches aléatoires courtes en utilisant la matrice d'adjacence normalisée par la somme de ces lignes (norme l1). Traditionnellement, les attributs sélectionnés pour faire du pattern mining sur les graphes proviennent de la liste d'adjacence d'un utilisateur (utilisateurs partageant une arête avec ce dernier). L'utilisation de cette dernière présente les trois problèmes suivants :

- La distribution en loi de puissance des degrés des nœuds génère des patterns fréquentes qui englobent plusieurs communautés. Cela peut réduire la pertinence des partitions obtenues ;
- La faible densité des graphes peut conduire à la génération de patterns qui ne caractérisent pas leurs régions peu denses ;
- De par la présence de petites communautés dans un réseau social, les patterns appartenant à ces dernières sont souvent ignorées à cause du critère de support minimal. La solution la plus intuitive est de réduire le support. Cela a néanmoins un effet sur la complexité des algorithmes, les rendant souvent inutilisables si la taille des communautés est petite.

L'utilisation de mesures basées sur les marches aléatoires permet de résoudre deux de ces trois problèmes (le premier et second) et de réduire l'effet du troisième problème. En effet, nous allons calculer un modèle nul qui caractérise la probabilité d'atterrir sur un nœud durant une marche aléatoire. Ce modèle nul permet d'éviter que les mêmes représentants ne soit fréquents dans plusieurs communautés (premier problème). L'utilisation des marches aléatoires permet

également de densifier la matrice d'adjacence et ainsi de rapprocher les nœuds des représentants de leurs communautés (second problème). Cette méthode rapproche les représentants de communautés de leurs nœuds, ce qui permet de maintenir un support minimal proche de la taille de la plus petite communauté. Si la liste d'adjacence est utilisée, le support minimal devrait être beaucoup plus bas que la taille de la plus petite communauté afin de générer des patterns qui caractérisent cette dernière. Ainsi, les voisinages proposés permettent également de faciliter la détection de petites communautés.

Le second problème lié à la détection de communautés de petite taille est également résolu par la méthode de pattern mining que nous proposons (DDPMine-utility). Cet algorithme génère un nombre réduit de patterns qui couvrent toutes les communautés, même en utilisant un support minimal très bas. Ainsi, il est possible de baisser le support minimal sans que cela augmente de manière considérable le nombre de patterns générées.

Nous créons dans cette étape une matrice P qui sera utilisée dans les calculs de distances structurelles entre les nœuds du graphe. Cette matrice est générée par l'équation 4.7. L'entrée $p_{i,j}$ de la matrice P est obtenue en soustrayant l'entrée (i, j) de la matrice stochastique M^t obtenue en effectuant t marches aléatoires, d'un modèle nul. L'entrée $p_{i,j}$ est affectée 0 si la probabilité de transition $m_{i,j}$ est plus faible que le modèle nul $\frac{\sum_{i \in V} mi,j}{N}$ (N=|V|).

$$p_(i,j) = \begin{cases} \left(m_{i,j} - \frac{\sum_{k \in V} m_{k,j}}{N} \right)^2 & m_{i,j} > \frac{\sum_{j \in V} m_{i,j}}{N} \\ 0 & m_{i,j} \leq \frac{\sum_{j \in V} m_{i,j}}{N} \end{cases} \tag{4.7}$$

4.2.2.2 Détecter les nœuds centraux

La seconde étape consiste à calculer les centralités locales de chaque nœud. L'idée derrière le calcul de centralité est qu'il est inutile d'utiliser des nœuds qui ne sont pas centraux dans leur environnement local comme attributs pour le pattern mining étant donné qu'il ne serviront pas à trouver des patterns qui définissent des communautés. En effet, l'utilisation de nœuds qui ne sont pas centraux peut dégrader la détection de communautés étant donné qu'ils risquent d'apprendre du bruit. Cela a également comme effet de réduire le nombre de patterns considérés par l'algorithme DDPMine-utility accélérant ainsi le processus de pattern mining.

On calcule une mesure simple de centralité locale (équation 4.8) inspirée des travaux de [99]. On utilise la médiane des degrés (deg_{med}) comme facteur de régularisation afin d'éliminer les nœuds à très faible degré. La distribution des degrés dans un réseau social suit une loi de puissance [55] et on considère que même les communautés à faibles degrés ont des leaders qui devraient dépasser la médiane des degrés. La fonction $AvgDeg$ retourne la moyenne des degrés d'un ensemble de nœuds, et dans le cas échéant, les voisins du nœud n_i ($Voisins(n_i)$).

$$C(n_i) = log\left(1 + \frac{deg(n_i)}{deg_{med} + AvgDeg(Voisins(n_i))}\right) \tag{4.8}$$

4.2.2.3 Générer un vecteur d'attributs

Cette étape consiste à sélectionner les voisins de chaque nœud. Ces voisins seront utilisés pour effectuer du pattern mining. La figure 4.1 décrit l'étape de sélection des voisins par WalkMiner-simple, où la taille des nœuds est proportionnelle à leur centralité.

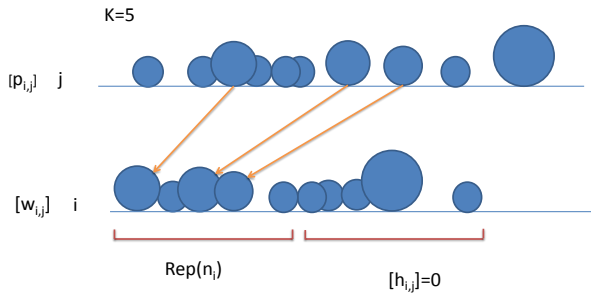

FIGURE 4.1 – Sélection des voisins par WalkMiner-simple

La sélection des voisins est à la fois basée sur la matrice P ainsi que sur la centralité relative des nœuds. Pour chaque champ non nul de la matrice P, on calcule une proximité basée sur la probabilité de transition et sur la centralité du nœud destination en utilisant l'équation 4.9.

$$w_{i,j} = \sqrt{p_{i,j}} \cdot C_j \tag{4.9}$$

Ensuite, pour chaque nœud du graphe, on sélectionne les ln(N) plus proches voisins en utilisant la proximité w, avec N le nombre de nœuds dans le graphe. Ces ln(N) plus proches voisins sont utilisés comme attributs des nœuds et sont fournis en entrée de l'algorithme de pattern mining. On dit aussi que les ln(N) plus proches voisins d'un nœud n_i (Voisins(n_i)) sont les attributs de ce dernier. Le choix de ln(N) a été fait parce que les représentants de communautés sont souvent proches des nœuds dans une matrice obtenue en effectuant des marches aléatoires [12]. Les vecteurs d'attributs de chaque nœud sont composés de ses ln(N) plus proches voisins et de leurs scores dans la matrice P (eq 4.10). Pour simplifier la notation, nous allons construire une matrice H avec $h_{i,j}$ le score de l'entrée (i,j) dans la matrice P si n_j est un voisin de n_i ($n_j \in Voisins(n_i)$).

$$h_{i,j} = \begin{cases} p_{i,j} & n_j \in Voisins(n_i) \\ 0 & n_j \notin Voisins(n_i) \end{cases} \tag{4.10}$$

4.2.2.4 Effectuer une détection de patterns sur le vecteur d'attributs

Le pattern mining peut générer un nombre très important de patterns, qui dans certains cas, grandit exponentiellement avec le nombre d'items (représentants dans le cas présent) [193].

Même l'utilisation de patterns fermées peut générer un nombre excessif de patterns [193]. L'algorithme DDPMine [193] permet de réduire de manière significative le nombre de patterns générées à partir d'un jeu de données. L'algorithme 8 décrit DDPMine et l'algorithme 7 décrit la procédure qui lui permet d'obtenir la meilleure pattern d'une FP-Tree [88] (structure représentant une table de transactions). Cet algorithme, conçu pour la classification supervisée, tente de couvrir un jeu de données en sélectionnant itérativement les patterns ayant obtenu le meilleur gain d'information (information gain), et en supprimant du jeu de données les transactions couvertes par ces patterns. L'efficacité de l'algorithme découle du fait qu'il calcule une borne supérieure (IG^{limite}) du gain d'information des patterns pouvant être générées à partir d'un préfixe, et évite celles dont la borne supérieure est inférieure au gain d'information de son préfixe (algorithme 7). Ainsi, seulement un sous-ensemble de patterns est considéré. Après avoir supprimé des éléments couverts du jeu de données, l'algorithme met à jour les gains d'information et effectue une nouvelle itération qui se concentre sur les éléments non couverts. Cette stratégie gourmande minimise le nombre de patterns nécessaires pour couvrir un jeu de données.

DDPMine n'est cependant pas adapté à la détection de patterns dans un contexte non supervisé étant donné que le gain d'information est une métrique pour la classification supervisée. Aussi, il faut noter que la matrice H contient des probabilités de transition qui sont des valeurs continues. Ces valeurs permettent de quantifier la proximité d'une pattern, et ainsi de sélectionner les meilleurs patterns. Ils peuvent également être utilisées pour sélectionner les transactions à supprimer dans chaque itération de l'algorithme (réduction de nombre du patterns). En effet, une transaction (ligne de la matrice H), peut être couverte plusieurs fois par des patterns dont les items ne sont pas proches. Ainsi, la transaction en question peut être supprimé par DDPMine sans que des patterns appartenant à sa communauté soient générées. La matrice H peut être utilisée pour sélectionner les transactions à supprimer qui sont couvertes par des patterns proches d'elles, ayant ainsi une forte probabilité d'appartenir à leurs communautés. Cela a comme effet de ne supprimer une transaction que si elle est couverte par des patterns proches, permettant ainsi de générer des patterns pour toutes les communautés du graphe.

Un second avantage lié à l'utilisation de la matrice H pour la génération de patterns est qu'elle permet de quantifier l'importance d'un item par rapport à une pattern dans un contexte

Algorithm 7 Branch-and-Bound

Input : FP-tree T, MinSup s, prefix A

Output : bestPat

Variables Globales : maxIG ← 0, bestPat : null

supp : fonction qui retourne le support d'une pattern

IG : fonction qui calcule le gain d'information d'une pattern

```
Pour chaque item i dans T :
    Générer pattern B=A ∪ i avec Supp(B) = Supp(i)
    Si Supp(B)>s :
        Calculer gain d'information IG(B)
        Si IG(B)> maxIG :
            bestPat ← B
            maxIG←IG(B)
        D_B ←Construire la base conditionnelle de B
        IG^limite(D_B) ← Calculer limite de IG(D_B)
        Si IG^limite(D_B) > maxIG :
            T_B ←Construire FP-tree de B
            Branch-and-Bound(T_B,s,B)
retourner bestPat
```

Algorithm 8 DDPMine

Input : FP-tree T, MinSup s, prefix B :null

Output : ensemble de patterns P

```
A = branch and bound(T, s, B)
t(A) ←Détecter les transactions contenant A
T← mettre à jour arbre (T,t(A))
P ← A∪ DDPMine(T,s) ;
retourner P
```

non supervisé. En effet, il est possible de créer une pattern Ae en concaténant la pattern A et l'item e. Cette nouvelle pattern Ae a un support inférieur ou égal à celui de A ($Supp(Ae) \leq Supp(A)$). Il est également possible de calculer une utilité liée à la génération de la pattern Ae (U_{Ae}) en utilisant la matrice H (équation 4.13). L'utilité de Ae baisse en comparaison avec celle de A en fonction de la faiblesse du support de Ae par rapport à celui de A. En contre parti, l'utilité de Ae augmente si les scores de e dans les transactions de la matrice H où Ae est vérifiée sont élevés. Ainsi, l'utilité de Ae est maximisée quand les deux conditions suivantes sont vérifiées :

– A et e sont vérifiés dans les mêmes transactions ;
– A et e ont des scores élevés dans les transactions où ils sont vérifiées.

L'utilisation de l'utilité pour générer des patterns permet ainsi de regrouper les items qui sont proches du même ensemble de nœuds, augmentant la qualité des patterns générées.

Afin d'expliquer les modifications qui ont été effectuées à DDPMine, il faut aborder quelques notions relatives à l'utilité d'une pattern. Nous allons d'abord présenter l'utilité d'un item i par rapport à un nœud n. Cette dernière est définie par l'équation 4.11.

$$u(n_i) = \sqrt{h_{n,i}} \tag{4.11}$$

L'utilité d'un nœud n par rapport à une pattern A est définie par l'équation 4.12. Cette dernière est égale à 0 si la pattern n'est pas vérifiée dans le vecteur d'attributs de l'élément. Dans le cas contraire, elle est égale à la racine carré de la somme des scores dans H de chaque item de la pattern.

$$u(n, A) = \begin{cases} \sqrt{\sum_{i \in A} h_{n,i}} & \forall (i \in A)/h_{n,i} \neq 0 \\ 0 & \exists (i \in A)/h_{n,i} = 0 \end{cases} \tag{4.12}$$

L'utilité totale d'une pattern est la somme des utilités de cette dernière pour chaque nœud dans le graphe.

$$U(A) = \sum_{n \in V} u(n, A) \tag{4.13}$$

L'algorithme 9 décrit DDPMine-utility que nous introduisons pour la génération de patterns dans un contexte non supervisé. La première modification apportée à DDPMine concerne le gain d'information. En effet la maximisation du gain d'information est remplacée par la maximisation de l'utilité des patterns. Ainsi, l'algorithme retourne itérativement les patterns avec la plus grande utilité. Plusieurs mesures ont été considérées dans le cadre de cette recherche pour remplacer le gain d'information. Le support par exemple offre un avantage aux patterns composées de peu d'items. Souvent, ces dernières ne permettent pas de caractériser une communauté. La somme des transactions où une pattern est vérifiée (somme des lignes dans H)

donne un avantage aux patterns très longues et augmente le nombre de patterns générées. L'utilité (équation 4.12) offre un compromis entre les deux stratégies, permettant d'identifier un petit ensemble de patterns de tailles moyennes qui couvrent H. Il est difficile néanmoins d'optimiser l'utilité afin de générer des patterns, étant donné qu'on ne peut pas obtenir l'utilité d'une pattern à partir des utilités des patterns qui la composent (il faut traverser les nœuds vérifiés par la pattern). Ainsi, nous utilisons une fonction de substitution décrite par l'équation 4.14 avec V(A) l'ensemble des nœuds vérifiant la pattern A, et Supp(A) est le support de la pattern A. Cette fonction calcule l'utilité en considérant que les poids dans la matrice H d'une pattern suivent une distribution uniforme. Les pattern retournées par cette fonction se rapprochent de celles retournées par l'utilité U. Nous utilisons la fonctions de substitution U^s au lieu de l'utilité réelle U seulement dans la phase de détection de patterns sur le vecteur d'attributs. WalkMiner-simple calcul ensuite les valeurs de U sur les patterns retournées par DDPMine-utility.

$$U^s(A) = Supp(A) \cdot \sqrt{\frac{\sum_{n \in V(A)} \sum_{i \in A} h_{n,i}}{Supp(A)}} \tag{4.14}$$

La seconde modification apportée à l'algorithme concerne la suppression des nœuds couverts par des patterns. Comme nous l'avons spécifié précédemment, certaines patterns apparaissent dans plusieurs communautés et leur détection peut couvrir plusieurs nœuds qui ne sont pas dans leurs communautés. Cela peut conduire à la non-représentation de certaines communautés dans les patterns générées. Nous supprimons une transaction (nœud dans H) seulement si les patterns qui la couvrent représentent plus de Q% de ces scores dans H. Q est un paramètre entré par l'utilisateur qui prend une valeur de 50% par défaut.

La dernière modification que nous effectuons à DDPMine est la fréquence des suppressions des éléments couverts. Après la suppression d'éléments, un parcours du FPTree [88] est nécessaire pour mettre à jour les utilités. Nous ajoutons un paramètre qui contrôle le nombre de ces parcours. Par défaut, nous parcourons l'arbre après la génération de ln(N) patterns (N le nombre de nœuds dans le graphe).

Algorithm 9 DDPMine-utility

Input : FP-tree T, MinSup s, matrice d'utilités H, inc, N : nombre de nœuds, prefix B :null, pourcentage de couverture d'une transaction avant la suppression Q :0.5

Output : ensemble de pattern P

1: incrémenter inc
2: A = branch and bound$_{util}$(T, s, B)
3: $t_C \leftarrow t_C \cup$ Calculer les transactions t dont $\sum_{i \in P \cup A} h_{t,i} > Q \cdot \sum_{i \in Rep(t)} h_{t,i}$
4: Si numIteration modulo $\lceil log(N) \rceil$ == 0
5: T$\leftarrow$mettre à jour arbre(T,t_C)
6: P $\leftarrow$ A$\cup$ DDPMine-utility(T,s,H,inc,N) ;
7: retourner P

4.2.2.5 Sélection des racines

Cette partie consiste à sélectionner les patterns de communautés qui seront utilisées comme racines. Ces racines $\mathbb{R}$ représentent les communautés de départ. La sélection de racines est une partie déterminante de l'algorithme, étant donné que, si des communautés ne sont pas représentées dans les racines, ces dernières ne pourront pas être retrouvées. La mesure optimisée pour la sélection de racines est inspirée d'une méthode de sélection des K patterns les plus représentatives d'une base de données [197]. Seules les patterns fermées sont éligibles pour devenir des racines, permettant ainsi de réduire le nombre de comparaisons effectuées dans cette étape.

La méthode d'optimisation qui identifie les K racines suit le même modèle que [192]. La sélection de racines commence d'abord par échantillonner T fois un nombre K de patterns fermées qui sont candidates pour être des racines. Le nombre K est soit obtenu comme paramètre utilisateur et correspondra ainsi au nombre de communautés soit est calculé par l'équation 4.15, avec MinSup le support minimal (par defaut, MinSup correspond à la taille minimale d'une communauté). Le nombre d'échantillons T est un paramètre qui prend la valeur par défaut $T = |patterns| \cdot log_2(|patterns|)$. Le paramètre T n'est pas déterminant dans cet algorithme. Il permet de contrôler le nombre de fois où la recherche de racines est effectuée (optimisation with restart).

Ensuite, les O partitions de racines avec le plus grand gain (équation 4.18) sont sélectionnées. Par défaut, le paramètre O prend la valeur $O = \log_2 |patterns|$. Une optimisation locale est effectuée pour chacune de ces O partitions en remplaçant itérativement la racine ayant la plus faible contribution au gain par la pattern qui maximise le gain (équation 4.18). L'algorithme s'arrête quand aucune amélioration au gain ne peut être apportée. Ensuite, parmi les O partitions, celle avec le plus haut gain est sélectionnée.

$$K = \frac{N}{MinSup} \tag{4.15}$$

Pour chacun des T échantillons, une variante de la mesure proposée par [197] est calculée. On commence d'abord par décrire la méthode de sélection des K patterns les plus représentatives d'une base de données proposée par [197]. Le gain obtenu en sélectionnant une pattern est décrit par l'équation 4.16.

$$G_{as}(A) = Supp(A) - \frac{1}{K-1} \sum_{A \in P^k} \sum_{B \in P^k \setminus A}^{K} R(A, B) \tag{4.16}$$

Supp(A) est une fonction qui retourne le support de la pattern A et $R(A, B)$ calcule la redondance entre deux patterns. La redondance [197] est définie par l'équation 4.17. Elle est calculée en multipliant l'inverse de la distance (D) et le minimum des supports de deux patterns. La distance utilisée est séléctionnée par l'utilisateur et doit être comprise entre 0 et 1 (ex. distance de Jaccard).

$$R(A, B) = (1 - D(A, B)) \cdot min(Supp(A), Supp(B)) \tag{4.17}$$

Le gain (G_{as}) est maximisée quand le support des patterns est élevé et quand leur intersection (ou redondance) est faible.

Nous effectuons une modification à la mesure proposée par [197] pour qu'elle soit adaptée à la détection de patterns dans les réseaux sociaux. La mesure que nous proposons est décrite par l'équation 4.18. Ainsi, au lieu d'utiliser le support, nous utilisons l'utilité des patterns. En effet, des patterns à support élevé ne sont pas nécessairement représentatives de communautés étant donné que le support privilégie les patterns composées de peu d'items. Les patterns à utilité élevée sont nécessairement composées d'items proches structurellement les uns des autres (ayant un haut score dans H dans des transactions similaires), et sont des meilleurs candidats pour caractériser des communautés.

La seconde modification apportée concerne la redondance entre les patterns qui n'est pas normalisée par $\frac{1}{K-1}$. Cette modification est effectuée afin de favoriser les patterns ayant très peu de redondances. Cela permet de détecter des représentants de petites communautés. Dans le cas où les redondances sont faibles, les patterns avec la plus grande utilité sont sélectionnées par la mesure.

La dernière modification concerne le calcul de la redondance qui utilise la matrice H. Cette redondance est maximisée quand des items ont des scores élevés dans les même transactions de H. Ainsi, cette modification permet de sélectionner des patterns qui ont une faible probabilité de représenter la même communauté.

$$G_u(P^k) = \sum_{A \in P^k} U(A) - \sum_{A \in P^k} \sum_{B \in P^k \setminus A} R(A, B) \tag{4.18}$$

Afin de calculer la redondance entre deux patterns, il est nécessaire de calculer l'intersection entre ces dernières et toutes les patterns voisines (patterns dont l'intersection est non nulle). Cette quantité est définie par l'équation 4.19. La fonction prox (équation 4.21) calcul la proximité entre deux patterns.

$$TI(A) = \sum_{B \in P} prox(A, B) \tag{4.19}$$

La redondance entre deux patterns R(A,B) est calculée par l'équation 4.20. Ainsi, cette dernière augmente si elles ont plusieurs patterns voisines en commun et si leur intersection est élevée avec les mêmes patterns voisines. Cette mesure permet de différencier entre des patterns qui définissent des communautés différentes, et des patterns d'une même communauté, étant donné que les dernières ont généralement plusieurs patterns voisines en commun (surtout des patterns à faible longueur et haut support).

$$R(A, B) = \frac{\sum_{C \in P} prox(A, C) \cdot prox(B, C)}{TI(A) \cdot TI(B)} \cdot min(U(A), U(B)) \tag{4.20}$$

La proximité entre un nœud et une pattern est obtenu en normalisant l'utilité d'un nœud par rapport à une pattern (eq 4.12) par l'utilité totale de la pattern (eq 4.13). Cette proximité est décrite par l'équation 4.21.

$$prox(n, A) = \frac{u(n, A)}{U(A)} \tag{4.21}$$

Notez que la somme des proximités entre une pattern et tous les nœuds dans V est égale à 1.
La proximité entre deux patterns A et B est définie par l'équation 4.22.

$$prox(A, B) = \sum_{i=1}^{N} prox(n, A) \cdot prox(n, B) \tag{4.22}$$

Le gain à maximiser permettant de trouver les racines initiales des communautés peut ainsi être reformulé dans l'équation 4.23 en remplaçant R(a,b) par ces valeurs.

$$G_u(P^k) = \sum_{A \in P^k} U(A) - \sum_{A \in P^k} \sum_{B \in P^k \setminus A} \sum_{D \in P^k} \frac{prox(A, C).prox(B, C)}{TI(A) \cdot TI(B)} \cdot min(U(A), U(B)) \tag{4.23}$$

4.2.2.6 Affecter les patterns restantes aux racines

Cette étape consiste à affecter toutes les patterns aux racines sélectionnées. Dans un premier temps, nous calculons la proximité $prox(A, E)$ (équation 4.22) entre chaque pattern fermée $A \in P$ et les racines $E \in \mathbb{R}$. Ensuite, une pattern fermée ainsi que les patterns sélctionnés par $DDPMine_{utility}$ qui la composent (ex. aef est composé par a,e,f,ae,af,fe) sont affectées à la racine ayant la plus grande proximité, si cette dernière est plus élevée que la médiane des proximités entre la racine et toutes les autres patterns fermées. Ainsi, des patterns proches des racines (plus proches que leurs médianes) sont affectées aux racines. On appelle les ensembles composés d'une racine et de ses plus proches patterns "cluster de patterns" ($\mathbb{C} = \{\mathbb{C}_1, \mathbb{C}_2, ..., \mathbb{C}_K\}$).

L'étape suivante est de calculer une proximité normalisée par l'utilité entre les patterns fermées non affectées et les clusters de patterns. La proximité entre une pattern et un cluster est calculée par l'équation 4.24. Cette dernière est la moyenne des proximité entre une pattern et toutes patterns dans un cluster pondéré par l'utilité des patterns.

$$prox(A, \mathbb{C}_i) = \frac{\sum_{B \in \mathbb{C}_i} prox(A, B) \cdot U(B)}{\sum_{B \in \mathbb{C}_i} U(B)} \tag{4.24}$$

L'algorithme 10 décrit l'affectation des patterns fermés aux clusters de patterns. Ce dernier affecte itérativement les patterns avec la proximité maximale à leurs clusters.

Algorithm 10 Affectation de patterns aux clusters de patterns

$\mathbb{C}$: clusters de patterns

P : Patterns

1: *Répéter tant qu'il y a un changement* :
2: *Pour chaque Cluster* $\mathbb{C}_i \in \mathbb{C}$
3: *Pour chaque pattern fermée A*$\in P$
4: Calculer $prox(A, \mathbb{C}_i)$
5: $\mathbb{C}(A) \leftarrow$ argmax($prox(A, :)$)
6: Attribuer les patterns non fermées qui composent la pattern fermées au même
7: retourner $\mathbb{C}$

4.2.2.7 Affecter les nœuds aux clusters de patterns

Pour chaque nœud, on calcule une taille de description (description length) vers chaque cluster de patterns. Cette mesure s'inspire de la mesure proposée par [198] qui utilise le principe de taille minimale de description (minimum description length). La méthode concerne la classification supervisée en utilisant les patterns minimales dans les bases de données. Une transaction dans une base de données qui ne satisfait aucune pattern dans une classe sera décrite par tous les

items de la transaction, augmentant ainsi la taille de description de cette transaction. En contrepartie, une transaction qui satisfait plusieurs patterns va être décrite par ces dernières, réduisant ainsi la taille nécessaire pour sa description. La taille nécessaire pour décrire une transaction est minimisée lorsque cette dernière satisfait les conditions suivantes :

- Elle satisfait plusieurs patterns dans une classe
- Elle satisfait des patterns de longues tailles (constituées de plusieurs items)
- Elle satisfait des patterns fréquentes (à haut support)

La mesure proposée par [198] tente de résoudre un des problèmes les plus courants en classification basée sur le pattern mining : les différentes classes ne sont pas équilibrées concernant le nombre et la qualité de patterns qu'elles contiennent. Cette mesure utilise des patterns disjointes afin de recouvrir les items d'une transaction (patterns qui partitionnent les items). Ainsi, même si une classe a un très haut nombre de patterns, seulement quelques-unes seront sélectionnées pour couvrir une transaction. Cette méthode n'offre pas une solution parfaite au problème de classes non équilibrées, étant donné que les classes riches en patterns auront une plus forte probabilité de recouvrir une transaction. La raison pour cela est que les transactions contiennent souvent des items qui n'appartiennent pas à leurs classes. Pour résoudre ce problème, il faut utiliser des mesures qui quantifient l'importance d'un item dans une transaction.

La mesure proposée par [198] ne peut pas être utilisée sur la matrice H, étant donné qu'elle ne prend pas en compte le poids (ou l'utilité) d'un item dans une transaction. En effet, la matrice H donne plus de poids aux items (voisins) structurellement proches d'un nœud. La mesure que nous proposons tente de réduire l'effet de la nature non équilibrée des classes en suivant l'intuition derrière la méthode de Dong et al. [198] et en l'adaptant au modèle utilisé dans cette section. Nous allons d'abord définir des quantités essentielles au calcul de la mesure que nous souhaitons minimiser.

La première quantité que nous souhaitons définir capture l'importance d'une pattern dans un cluster de patterns (eq : 4.25). Le PatternStrengthRatio est minimisé lorsque l'utilité d'une pattern $U(A)$ est plus élevée que l'utilité médiane des patterns dans le cluster $\mathbb{C}_i$. La pondération par la médiane des utilités répond au problème de classes non équilibrées, réduisant l'effet de patterns à faible utilité dans les classes minoritaires. Le PatternStrengthRatio est ainsi compris entre 0 et 1 (PatternStrengthRatio $\in]0, 1]$).

$$PatternStrengthRatio(A, \mathbb{C}_i) = \frac{1}{1 + \frac{U(A)}{mediane(U(K \in \mathbb{C}_i))}} \tag{4.25}$$

La seconde quantité que nous définissons est la proximité d'un item à un cluster de patterns (équation 4.26). Cette quantité est la moyenne des proximités d'un item à toutes les patterns d'un cluster normalisée par l'utilité de chaque pattern. Nous rappelons qu'un item est un élément d'une pattern et que ces items représentent les nœuds centraux (représentants).

$$prox(a, \mathbb{C}_i) = \frac{\sum_{A \in \mathbb{C}_i} U(A) \cdot prox(a, A)}{\sum_{A \in \mathbb{C}_i} U(A)} \tag{4.26}$$

Nous définissons ensuite la quantité qui permet d'attribuer un nœud à une communauté (eq : 4.27). Cette quantité nécessite de générer une partition $\mathcal{P}^k$ à partir d'un cluster de patterns $\mathbb{C}_i$. Dans cette partition, les patterns ont une intersection nulle. En d'autres termes, les représentants (items) dans chaque pattern de la partition $\mathcal{P}^k$ sont différents.

$$\mathcal{L}(n, \mathcal{P}^k, \mathbb{C}_i) = \sum_{A \in \mathcal{P}^k} \begin{cases} PatternStrengthRatio(A, \mathbb{C}_i) \cdot \sqrt{\sum_{a \in A} h_{n,a}^2 \cdot (1 - prox(a, \mathbb{C}_i))^2} & \forall a \in A / a \in Rep(n) \\ \sum_{a \in A} \sqrt{h_{n,a}^2 \cdot (1 - prox(n, \mathbb{C}_i))^2} & \exists a \in A / a \notin Rep(n) \end{cases} \tag{4.27}$$

Quand un élément vérifie une pattern, cette mesure est composée de deux parties. La première est le PatternStrengthRatio, qui capture l'importance d'une pattern dans une communauté de patterns. La seconde capture la proximité du nœud au cluster de patterns $\mathbb{C}$. Ainsi, $\mathcal{L}$ est minimisé quand les conditions suivantes sont remplies :

- Le vecteur $h_{i,:}$ de la matrice H est couvert par des patterns à hautes utilités relatives dans le cluster $\mathbb{C}_i$ (PatternStrengthRatio faible).
- Le vecteur $h_{i,:}$ est couvert par des patterns longues ($\sqrt{v_1 + v_2} \leq \sqrt{v_1} + \sqrt{v_2}$). Le fait qu'un vecteur soit couvert par des patterns longues est un indicateur qu'il appartient à la communauté [198].
- Les patterns dans $\mathbb{C}_i$ couvrent les items ayant les plus hauts scores dans $H_{i,:}$. En effet, les items d'un vecteur sont obtenus en soustrayant la probabilité de transition du modèle nul (équation 4.7). Une pattern qui vérifie plusieurs items à hauts scores a une plus forte contribution à l'affectation d'un nœud qu'une pattern vérifiant des items à faibles scores, ou que plusieurs patterns vérifiant ces items individuellement.
- La proximité des voisins d'un nœud d'un cluster de patterns ($prox(a, \mathbb{C}_i)$) peut l'affecter à celui-ci. Cette proximité (équation 4.26) permet d'affecter des nœuds peu connectés au graphe ayant une faible appartenance aux communautés. Les patterns quant à elles permettent de mieux séparer les communautés et d'affecter les nœuds aux frontières de celles ci.

L'algorithme 11 optimise la mesure $\mathcal{L}$ et affecte les nœuds aux clusters de patterns. Une communauté est ainsi définie par son cluster de patterns et par les nœuds qui lui sont affectées.

4.2.2.8 Générer un dendrogramme en fusionnant itérativement les clusters

Cette dernière étape consiste à agréger des clusters de patterns ainsi que les nœuds qui leur sont affectés afin de former un dendrogramme. La première étape est de détruire les clusters de

Algorithm 11 Affectation des nœuds aux clusters de patterns

Input

V : ensemble de nœuds d'un graphe ;

$\mathbb{C}$: clusters de patterns.

Output

com : vecteur représentant les communautés des nœuds.

Pour chaque nœud n $\in V$:
    Pour chaque Cluster $\mathbb{C}_i$
        $\mathbb{P}^k_{\mathbb{C}_i} \leftarrow$ [Générer K partitions de $\mathbb{C}_i$]
            pour chaque partition $\mathbb{P}^i \in \mathbb{P}^k_{\mathbb{C}_i}$
                Calculer $\mathcal{L}(\mathbb{P}^i, n)$
        $\mathcal{L}(C, n) \leftarrow min(\mathcal{L}(\mathbb{P}^:, n))$
    com(n) $\leftarrow$ argmin($\mathcal{L}(:, n)$)
retourner com

patterns qui sont attribués un nombre de nœuds inférieur à MinSup. Leurs patterns et ensuite leurs nœuds sont réaffectés aux clusters les plus proches en répétant les deux procédure des sous-sections ci-dessus.

Ensuite, nous regroupons itérativement les deux clusters de patterns qui minimisent une mesure de proximité qui se rapproche du couplage des moyennes (average linkage) [199]. La mesure introduite est inspirée de la modularité basée sur le couplage introduite dans [135]. La modularité basée sur le couplage répond au problème de résolution qui affecte la modularité [94]. Nous allons tout d'abord présenter la dispersion d'une communauté qui permet de calculer la mesure proposée.

La dispersion d'une communauté est définie par l'équation 4.28. Cette mesure est la somme des proximités (eq 4.22) entre les patterns d'une même communauté divisée par la somme des proximités entre ces patterns et toutes les autres patterns du graphe. Cette mesure est minimisée quand les patterns d'un cluster ont une forte proximité entre elles et une faible proximité avec les patterns appartenant à des clusters différents. Elle mesure ainsi la dispersion des patterns dans d'autres clusters. Il faut noter que, généralement, un nombre restreint de patterns ont des éléments en commun, ce qui permet de calculer cette mesure efficacement (P représente l'ensemble des patterns).

$$Disp(\mathbb{C}_i) = \frac{\sum_{A \in \mathbb{C}_i} \sum_{B \in \mathbb{C}_i} prox(A, B)}{\sum_{A \in \mathbb{C}_i} \sum_{D \in P} prox(A, D)} \tag{4.28}$$

La proximité entre deux clusters est calculée par l'équation 4.29. Cette mesure est maximisée quand les patterns des deux clusters ont une forte similarité et quand la dispersion des deux clusters est faible. Ainsi, si un cluster $\mathbb{C}_i$ à une proximité similaire avec deux clusters $\mathbb{C}_j$ et $\mathbb{C}_k$.

il sera affecté à celui ayant la plus faible dispersion. La raison est qu'un cluster à dispersion élevée serait proche de plusieurs autres clusters, et sa proximité avec le cluster $\mathbb{C}_i$ pourrait être non significative.

$$prox(\mathbb{C}_i, \mathbb{C}_j) = \frac{\sum_{A \in \mathbb{C}_i} \sum_{B \in \mathbb{C}_j} prox(A, B)}{1 + \sqrt{(Disp(\mathbb{C}_i))} + \sqrt{(Disp(\mathbb{C}_j))}} \tag{4.29}$$

4.2.3 WalkMiner-edgeAtts : Détection de communautés dans des graphes ayant des arêtes avec attributs textuels

L'objectif de cet algorithme est de détecter des communautés sur des graphes ayant des arêtes avec attributs textuels. L'algorithme 12 décrit WalkMiner-edgeAtts (illustré par la figure 4.2). La première étape de l'algorithme est d'effectuer un LDA [124] afin de détecter les thématiques échangées sur le réseau social. Le LDA est effectué sur une agrégation des messages de chaque nœud en utilisant la méthode Twitter-LDA [200]. Ensuite, chaque message est assigné une thématique parmi les thématiques détectées dans l'étape précédente. La quatrième étape consiste à construire un graphe multiplex où le poids des arêtes dans chaque mode est pondérés par la distribution des thèmes des utilisateurs. Enfin, la dernière étape est d'utiliser WalkMiner-simple sur le graphe généré. Ainsi, cet algorithme pondère les arêtes entre les nœuds par la proximité de leurs attributs. Le LDA est utilisé comme une méthode de réduction de dimensionalité.

Algorithm 12 WalkMiner-edgeAtts

g : graphe attribué

1: $c_1 \leftarrow$[Effectuer une détection de thèmes en utilisant Twitter-LDA](g) ;
2: $\mathcal{M} \leftarrow$[Construire un graphe pondéré par la distribution des thèmes](g,c_1) ;
3: $c_2 \leftarrow$[Utiliser WalkMiner-simple sur le graphe généré]($\mathcal{M}$). retourner c_2

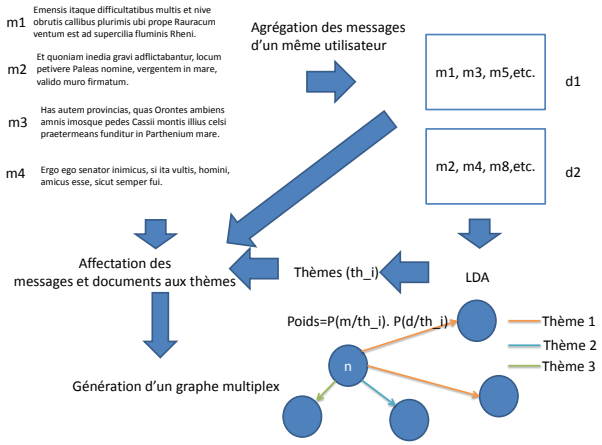

FIGURE 4.2 – WalkMiner-edgeAtts

4.2.3.1 Effectuer une détection de thèmes en utilisant Twitter-LDA

Twitter-LDA [200] est utilisé afin de détecter les thèmes des arêtes avec attributs textuels. Ces arêtes représentent souvent des messages publiés sur des réseaux. Dans les cas où l'algorithme analyse des graphes non orientés comme des graphes de similarité où une arête est crée à partir de deux messages (ex : graphe d'URL oû une arête est créée pour chaque deux messages partageant la même URL), il est considéré que chaque paire de nœuds adjacents est connectée par deux arêtes orientées. Chaque arête hérite ainsi les attributs du message de son nœud source.

Les messages échangés dans plusieurs réseaux comme des réseaux sociaux sont souvent très courts. L'utilisation d'un LDA sur ces messages détecte des thèmes composés de mots fréquents et peu significatifs qui s'éloignent des thèmes réellement abordés sur les réseaux sociaux [200]. Twitter-LDA (algorithme 13) détecte des thèmes en utilisant des documents composés de l'agrégation des messages d'un utilisateur. Cela permet au LDA utilisé dans l'algorithme de générer des thèmes qui se rapprochent des thématiques réelles abordées dans le réseau. Ensuite, cette méthode utilise le LDA généré afin de prédire les thèmes des messages courts. Cette prédiction prend en compte la distribution des thèmes de l'utilisateur (agrégation de messages) ainsi que la distribution des thèmes d'un message afin de sélectionner le thème le plus probable pour chaque message. Notez que Twitter-LDA agrège des messages qui ne sont pas nécessairement utilisés pour générer des arêtes (messages sans contenu relationnel). Ces messages, souvent majoritaires, permettent d'améliorer la qualité des thèmes retournés.

L'équation 4.31 décrit l'affectation d'un thème (Th_i) à une arête (n_1, n_2). La première étape consiste à calculer pour chaque thème un score composé de la probabilité que l'utilisateur source de l'arête ait utilisé ce thème, et de la probabilité que le message encapsulé dans l'arête aborde ce thème. La première probabilité est obtenu par le LDA sur les documents ((Doc) composés de l'agrégation de messages d'utilisateurs. La seconde probabilité est obtenue par le même modèle

de LDA, mais sur un message.

$$W(Th_i, (n_1, n_2)) = Pr(Th_i/(n_1, n_2)) \cdot Pr(Th_i/D(n_1)) \tag{4.30}$$

$$Theme((n_1, n_2)) = argmax(W(:, (n_1, n_2))) \tag{4.31}$$

Algorithm 13 Twitter-LDA

E : ensemble d'arêtes ayant des attributs textuels

Doc : documents (agrégation des arêtes attribuées de chaque utilisateur (nœud source de l'arête))

L : nombre de thèmes

```
D← [agréger les messages d'utilisateurs (E)]//Étape 1
Thèmes ← LDA(D,L)
Pour d dans Doc :
    Pr(Th_i/d) ← [affecter un mixture de thème]
    Pour e dans d :
        W(Th_i, e) [Affecter 1 thème par arête (e=(n_i, n_j))]
```

Cette section n'étudie pas le nombre de thèmes que prend en entrée le LDA. Étant donné que ce dernier est utilisé comme une méthode de réduction de dimensionalité, il n'est pas nécessaire de trouver le nombre de thèmes optimal. Nous utilisons l'algorithme HDP [201] pour sélectionner un nombre adéquat de thèmes.

4.2.3.2 Construire un graphe multiplex pondéré par la distribution des thèmes

Cette étape consiste à utiliser les thèmes détectés par Twitter-LDA afin de générer un graphe multiplex composé de graphes thématiques. La figure 4.3 illustre un segment d'un graphe multiplex $\mathcal{M}$ généré dans cette étape. Le graphe multiplex $\mathcal{M}$ est composé des graphes thématiques $\mathcal{M} = \{G_1 \times G_2 \times ... \times G_L\}$ avec L le nombre de thèmes. Chaque graphe G_i définit un thème obtenu dans l'étape précédente. Un graphe G_i est donc composé des arêtes E_i ayant été affectées au thème i ainsi qu'à leur nœuds incidents V_i (nœuds source et nœuds destination des arêtes). Chaque arête est pondérée par le poids $W(Th_i, e)$ calculé par Twitter-LDA. Ainsi, les arêtes ayant un thème qui est central pour l'utilisateur source voient leur poids augmenter comparé aux arêtes dont le thème est moins important.

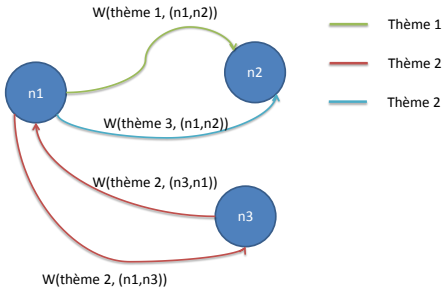

FIGURE 4.3 – Segment d'un graphe multiplex pondéré par la distribution des thèmes

4.2.3.3 Utiliser WalkMiner-simple sur le graphe multiplex

La dernière étape de l'algorithme est d'utiliser WalkMiner-simple sur le graphe multiplex généré dans l'étape précédente. Kuncheva et al. [12] utilisent également des marches aléatoires afin de détecter des communautés sur des graphes multicouches. Concernant la pondération des arêtes identités (connectant les différentes thématiques composant le graphe multimodal), Kuncheva et al. [12] calculent pour chaque nœud un ratio de l'intersection des arêtes qui existent sur deux couches et l'union de ces arêtes. Ensuite, ce ratio est pondéré par un paramètre utilisateur qui contrôle la capacité de l'algorithme à créer des communautés inter-couches. Notre approche est différente sur deux volets. Le premier est que nous utilisons un graphe multiplex qui ne contient pas d'arêtes identité. La seconde est que nous utilisons Twitter-LDA pour pondérer les différents graphes thématiques, comme il est décrit dans la sous-section précédente.

4.2.4 WalkMiner-DisGraphs

Cet algorithme permet de détecter des communautés sur des graphes ayant plusieurs composantes connexes. L'algorithme privilégie les communautés connectées, mais permet également d'utiliser les attributs des nœuds afin de regrouper des segments de graphes disjoints. Des segments non connectés sont regroupés dans la même communauté dans les cas suivants :

– Les nœuds dans ces segments ont des attributs similaires ;

– Les nœuds dans ces segments ont des liens avec des nœuds aux attributs similaires.

L'algorithme 14 décrit WalkMiner-DisGraphs. La première étape de l'algorithme est de détecter des communautés sur chaque composante connexe d'un graphe g en utilisant un des deux algorithmes décrits ci-dessus (WalkMiner-simple ou WalkMiner-edgeAtts). Cette étape utilise les interaction dans g afin de retourner des communautés c_1. La seconde étape consiste

à effectuer une régularisation qui rapproche les attributs des communautés qui interagissent avec des communautés similaires. Dans cette seconde étape, la similarité fait référence à la proximité des attributs, et non pas à la proximité structurelle. Ainsi, des communautés dans des composantes connexes différentes sont rapprochées si ces leurs nœuds interagissent avec des nœuds aux attributs similaires.

Le taux de rapprochement entre une communauté c_i et une communauté c_j est calculé par l'équation 4.32 à partir de la matrice d'adjacence A. Cette valeur asymétrique entre c_i et c_j représente le nombre d'interactions entre les nœuds des deux communautés sur le nombre total d'interaction des nœuds dans c_i. Ainsi, le taux de rapprochement baisse si la communauté c_i à un haut nombre d'interactions totales ou un faible nombre d'interactions avec c_j. Cela signifie que les communautés avec un faible nombre d'interactions inter-communautés comparé à leurs interactions intra-communauté ont une faible régularisation. Le rapprochement de deux communautés augmente si leur taux d'interaction sur leur nombre d'interactions total est élevé.

Algorithm 14 WalkMiner-DisGraphs

g : graphe attribué disjoint

numCom : nombre de communautés

1: $c_1 \leftarrow$ [WalkMiner-edgeAtts ou WalkMiner-simple sur les composantes connexes de g](g) ;
2: $Att^r \leftarrow$ [Régulariser les attributs des communautés c_1](g,c_1) ;
3: $c_2 \leftarrow$[Effectuer un K-means (ou K-means sphérique) sur c_1](Att^r,numCom).
4: retourner c_2

$$TR(c_i \rightarrow c_j) = \frac{\sum_{x \in c_i, y \in c_j} a_{x,y}}{\sum_{x \in c_i, y \in V} a_{x,y}} \tag{4.32}$$

On considère que la fonction $Att(c)c \rightarrow R^O$ retourne les attributs d'une communauté c_i. Le calcul des attributs régularisés par la dimension structurel d'un graphe g est décrit par l'équation 4.33.

$$Att^r(c_i) = TR(c_i \rightarrow c_i) \cdot Att(c_i) + \sum_{c_j \in c \setminus \{c_i\}} TR(c_i \rightarrow c_j) \cdot Att(c_j) \tag{4.33}$$

4.2.4.1 Effectuer un K-means sur C_1

La dernière étape de l'algorithme est d'effectuer un clustering qui considère les communautés avec attributs régularisés dans l'étape précédente comme des éléments individuels d'un jeu de données. Nous avons proposons dans cette étude le K-means [202] ou le K-means sphérique [203], étant donné que leurs comportements a été amplement étudié par la communauté scientifique, mais n'importe quel autre algorithme de clustering peut être utilisé.

4.3 Expérimentations

Cette section présente des expérimentations effectuées sur des benchmarks synthétiques et réels bien connus dans la littérature. Le générateur présenté dans [112] a été utilisé pour générer des réseaux synthétiques. Concernant les données réelles, nous avons utilisés les réseaux suivants :

- Réseau de Karate de zachary (Zachary Karate Club) [204] ;
- Réseau d'amitié d'une université en Angleterre [205] ;
- Réseau d'échange de courriel [**?**] ;
- Réseau de similarités basé sur les URLs sur Twitter collecté à partir de TweetCapt [16] ;
- Réseau de similarités basé sur les URLs sur Twitter contenant des faux amis ainsi que des utilisateurs bénins [206] ;
- Réseau de similarités basé sur les URLs sur Twitter contenant des spammeurs ainsi que des utilisateurs bénins [206].

Nous comparons les méthodes proposées avec les méthodes suivantes :

- Walktrap [96] : Un algorithme agglomératif qui utilise des marches aléatoires courtes pour calculer une mesure de distance entre nœuds. Il fusionne ensuite ces nœuds en communautés en utilisant la méthode de Ward ;
- Infomap [109] : Un algorithme qui identifie des communautés en calculant le nombre de bits nécessaires pour la représentation de marches aléatoires courtes ;
- Louvain [98] : Un algorithme itératif qui optimise la modularité ;
- Markov Clustering Process (MCL) [207] : Cet algorithme utilise des séquences de matrices stochastiques obtenues par la multiplication matricielle, ainsi qu'un processus d'inflation permettant de mieux séparer les régions d'un graphe. Cette dernière opération est effectuée en élevant chaque colonne par un opérateur exponentiel r, puis en normalisant les lignes ;
- SpinGlass [208] : Cet algorithme extrait des communautés en calculant l'état stationnaire d'un processus mécanique ;
- EigenDec [209] : Méthode qui maximise la modularité en utilisant les vecteur propres d'une matrice générée par l'algorithme, qui dénommant "matrice de modularité" ;
- Propagation de labels [101] : méthode qui propage des labels sur le graphe. Elle commence par affecter chaque nœud à un label (communauté) différent, puis se base sur le consensus des voisins des nœuds afin de leur affecter itérativement de nouveaux labels ;

De plus, afin d'évaluer l'utilisation de la modularité comme critère d'arrêt, nous comparons les algorithmes proposés avec une version du WalkMiner-simple que nous appelons "WalkMiner-noModOpt". Cette dernière approche consiste à utiliser, WalkMiner-simple sans la génération du dendrogramme (dernière étape de l'algorithme). Ainsi, les communautés obtenus par l'affec-

tation des nœuds aux racines sont retournées par l'algorithme, et la modularité n'est pas utilisée pour sélectionner la meilleur partition.

4.3.1 Réseaux synthétiques

Les trois premiers jeux de données sont des réseaux synthétiques avec 1 000 nœuds construits par un générateur de graphes sociaux [112]. Les trois jeux suivants (4 à 6 inclus) sont des réseaux de 10 000 nœuds et le jeu 7 est composé de réseaux à 100 000 générés également par [112]. Le générateur de jeux synthétiques [112] a pour objectif de modéliser des réseaux sociaux numériques (distribution de degrés en loi de puissance, diamètre faible, regroupement des nœuds en communautés).

Les jeux de données synthétiques sont décrits dans le tableau 4.1. Les jeux 1,4 et 7 ont été générés en utilisant les paramètres par défaut du générateur [112] : un degré entrant moyen (15), un degré maximal (50), taille de communauté minimale (20) et taille de communauté maximale (50). Les jeux 2 et 5 ont un degré entrant moyen plus bas (10), une taille de communauté maximale plus élevée (500) et un degré maximal plus élevé (100). La détection de communautés est plus difficile dans ces jeux à cause de la plus grande variance de la taille de leurs communautés ainsi que de la plus grande variance dans les degrés de leurs nœuds.

Les jeux 3 et 6 ont une densité plus faible que les autres jeux (degré moyen de 7), une taille de communauté maximale de 500 (similaire aux jeux 1,4 et 7) et un degré maximale de 100 (similaire aux jeux 1,4 et 7). La détection de communauté est plus difficile dans ces jeux à cause de leur faible densité. En effet, une majorité d'algorithmes obtiennent de faibles résultats sur ces jeux qui modélisent des graphes similaires au graphe social de Twitter (Réseaux Post it) [14].

Pour chaque jeu de données, on fait varier le paramètre de mélange μ qui affecte la qualité des partitions

($\mu = max\frac{\#liens\,inter\,communauts}{\#\,liens\,intra\,communauts}$). Plus ce paramètre est élevé, plus il est difficile de détecter des communautés, étant donné que le ratio du nombre de liens inter-communautés sur le nombre de liens intra-communauté augmente. Pour chaque jeu de données et pour chaque paramètre de μ, 20 graphes ont été générés en utilisant des racines différentes.

TABLE 4.1 – Description des graphes synthétiques

Nom du jeu	moyenne des degrés	degré max.	taille minimale de com.	taille max de com	Nombre de nœuds
Jeu 1	15	50	20	50	1 000
Jeu 2	10	100	20	500	1 000
Jeu 3	7	100	20	500	1 000
Jeu 4	15	50	20	50	10 000
Jeu 5	10	100	20	500	10 000
Jeu 6	7	100	20	500	10 000
Jeu 7	15	50	20	50	100 000

L'annexe A présente la moyenne des NMIs, modularités et mesures F1 obtenus par le générateur de graphe LFR [112] pour les jeux de données synthétiques. Chaque moyenne est calculée à partir de 20 graphes générés en utilisant une racine aléatoire.

Les scores de NMI sont illustrés par la figure 4.4. La qualité des partitions se dégrade avec l'augmentation du paramètre μ.

On peut voir que SpongeWalker et WalkMiner obtiennent des NMIs satisfaisants sur les données synthétiques.

Pour le jeu 1, plusieurs algorithmes obtiennent des résultats similaires. Cela peut être dû au fait que ce jeu soit relativement facile à partitionner comparé aux autres jeux. La propagation de labels, le MCL et la décomposition par vecteurs propres obtiennent de faibles résultats sur ce jeu. Les algorithmes proposés obtiennent des résultats proches des NMI des algorithmes aux plus hauts scores. Ils sont dépassés seulement par Louvain et la propagation de labels et dépassent les autres algorithmes sur la majorité des paramètres μ. Concernant le quatrième jeu de données, WalkMiner obtient les meilleurs scores de NMI.

WalkMiner obtient les meilleurs résultats sur le jeu 4. SpongeWalker dépasse Louvain et Walktrap sur ce quatrième jeu sur la majorité des variations de μ. WalkMiner et SpongeWalker obtiennent de meilleurs résultats que Walktrap sur le cinquième jeu, mais leur courbe sont dominées par celle de Louvain. WalkMiner obtient les meilleurs résultats sur le jeu 6 et SpongeWalker est dépassé seulement par la propagation de labels.

WalkMiner obtient un des meilleurs scores sur le jeu 7, proche de Louvain et la propagation de labels.

SpongeWalker obtient des NMI qui sont souvent bornés entre les NMI de Louvain et de Walktrap. Cela est du au fait que SpongeWalker combine les deux algorithmes. Sur les jeux à 1 000 nœuds (jeu 1,2), les NMI de SpongeWalker sont plus proches de Louvain, et sur les jeux à 10 000 nœuds, ses NMI sont plus proches de ceux de WalkTrap. Cela peut être expliqué par le fait que l'optimisation de densité effectuée par SpongeWalker génère des partitions proches des vérités terrain quand les réseaux ont un faible nombre de nœuds, contrairement au cas où les réseaux ont 10 000 nœuds. Dans ces seconds réseaux, l'algorithme se repose sur sa seconde phase qui combine des communautés en utilisant des distances structurelles.

WalkMiner et SpongeWalker obtiennent des NMI satisfaisant sur les tous les jeux. Le NMI de SpongeWalker se dégrade sur les jeux à faible densité (jeu 3 et 6). En effet, l'algorithme SpongeWalker n'est pas efficace sur ces graphes à faible densité à cause de son utilisation de l'optimisation de densité dans sa premier phase. L'optimisation de mesures de densité (ex : optimisation de modularité) peut obtenir de faibles résultats dans des réseaux à faible densité et hauts degrés [14]. Cela est dû à l'effet des nœuds à hauts degrés sur le calcul de mesures de densité, pouvant conduire à la génération de partitions très différentes et aux densités similaires (instabilité des résultats). WalkMiner obtient les meilleurs résultats sur les jeux à faible densité.

Cela est dû à son utilisation de marches aléatoires et de centralités relatives pour définir des voisinages de nœuds à partir des représentants de communautés. En effet, WalkMiner sélectionne des représentants de communautés qui sont proches structurellement sans qu'ils soient nécessairement connectés aux nœuds, permettant à un algorithme de pattern mining de trouver des patterns pour des nœuds peu connectés.

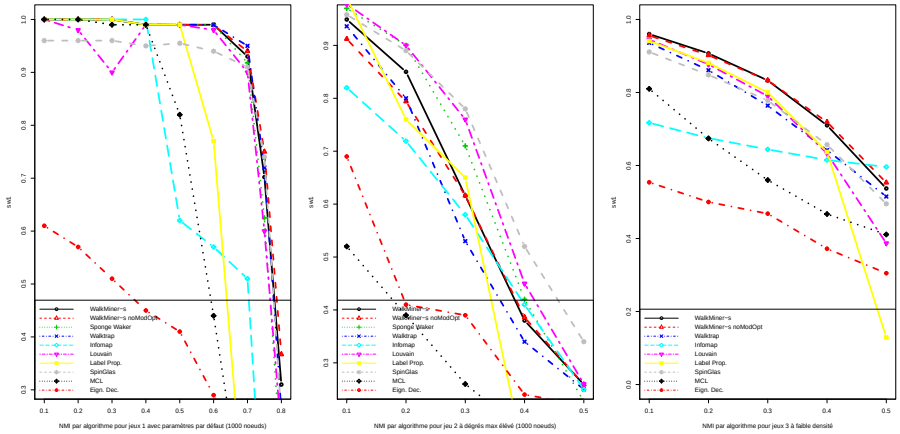

FIGURE 4.4 – Distribution des NMIs par μ pour jeux à 1 000 nœuds

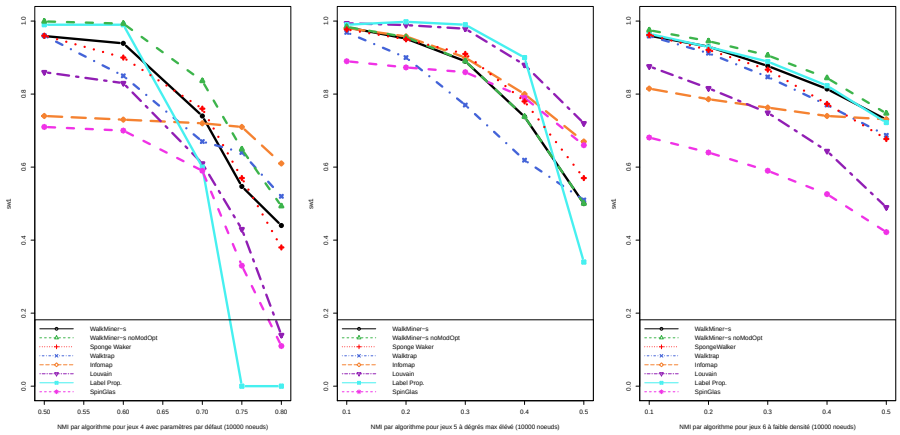

FIGURE 4.5 – Distribution des NMIs par μ pour jeux à 10 000 nœuds

La figure 4.7 illustre la distribution des scores de modularité par μ. Nous pouvons remarquer que Louvain et SpinGlass obtiennent les meilleurs scores de modularité pour les différents types de graphes. Concernant le graphe LFR avec paramètres par défaut (jeu 1), les scores de modularité des différents algorithmes sont très similaires, excepté le MCL, Infomap et EigenDec. L'algorithme de décomposition par vecteurs propres obtient des scores plus bas. Les variations dans les scores de NMI sont plus élevées que les variations dans la modularité, et de petites

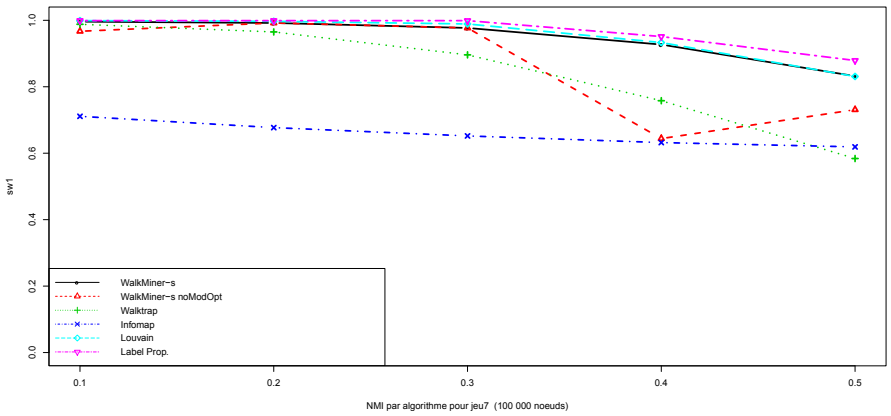

FIGURE 4.6 – Distribution des NMIs par μ pour jeux à 100 000 nœuds

variations dans la modularité peuvent conduire à de grands changements de NMI, comme pour l'algorithme de propagation de label pour $\mu = 0.6$ (jeu 1).

Concernant les graphes avec degré maximum élevé (jeux 2 et 4), la modularité de SpongeWalker est supérieure à celle de Walktrap et inférieure à celle de Louvain. On peut également remarquer que le score de modularité de SpongeWalker se rapproche plus de celui de Louvain que de celui de Walktrap pour les deux jeux. Les scores de modularité de SpongeWalker et de WalkMiner sont très proches les uns des autres. Aussi, les scores de WalkMiner et WalkMiner sans optimiser la modularité (WalkMiner-noModOpt) sont proches. En effet, WalkMiner détruit les clusters de patterns si peu de nœuds (moins de 15 nœuds par défaut) leur sont affectés. Cela conduit l'algorithme à démarrer la phase de génération de dendrogramme avec un nombre de communautés proche du nombre sélectionné par la modularité.

SpongeWalker obtient un des scores de modularité les plus élevés, dépassé seulement par Walktrap, Louvain et SpinGlass. Il obtient néanmoins de meilleurs NMIs que SpinGlass sur tous les graphes, et dépasse Louvain et la propagation de labels sur plusieurs graphes. La modularité est ainsi corrélée avec de meilleurs scores de NMI, sans pour autant avoir un effet de causalité.

La figure 4.8 illustre les variations de scores F1 pour la plus petite communauté dans chaque graphe. Nous avons d'abord identifié la plus petite communauté pour chaque graphe, et ensuite calculé les moyennes de scores F1 qui prennent en compte la précision et le rappel de cette communauté.

Concernant les graphes LFR avec paramètres par défaut (jeux 1 et 4), SpongeWalker obtient des scores très élevés, dépassé seulement par Walktrap et WalkMiner sans optimisation de modularité dans les graphes à 1000 nœds (jeu 1) et WalkMiner et la propagation de labels dans le jeu à 10 000 nœuds (jeu 4). SpongeWalker obtient des scores f1 bornés entre les scores de Louvain et de Walktrap pour les jeux à hauts degrés (jeu 2 et 5). Ses scores sont plus proches

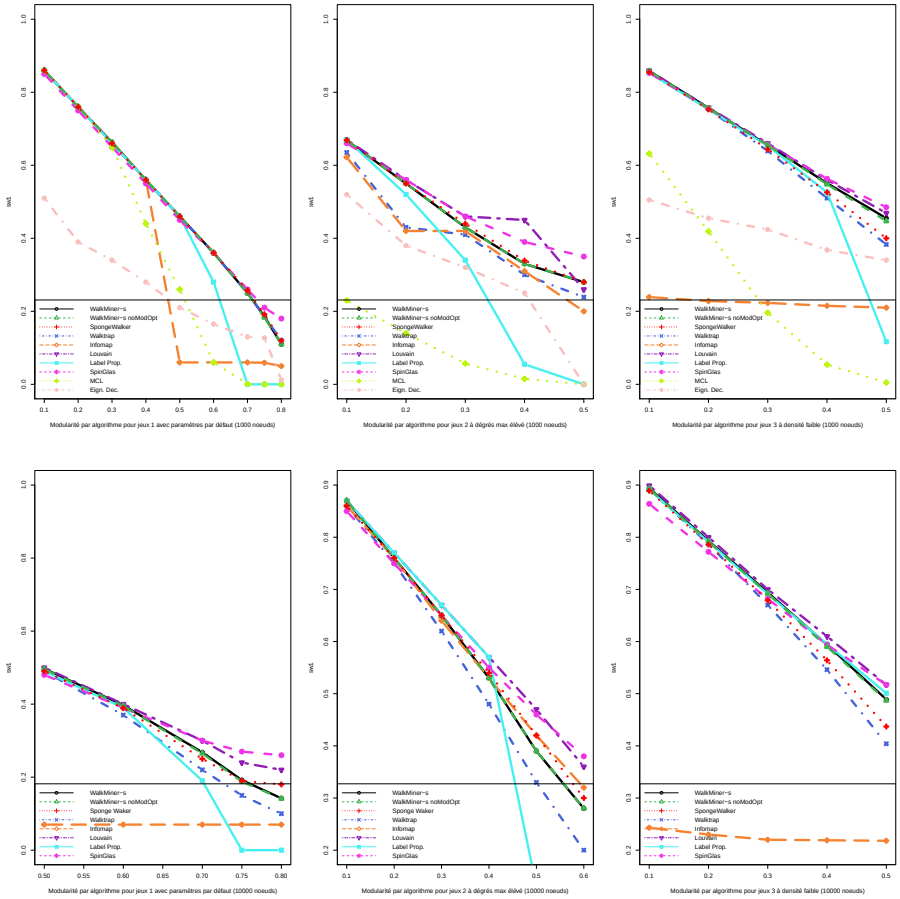

FIGURE 4.7 – Distribution des scores de modularité par μ

de ceux de Louvain dans les jeux à 1 000 nœuds et plus proches de ceux de Walktrap dans les jeux à 10 000 nœuds, comme pour son NMI. Comme indiqué dans la discussion sur les scores de NMI, cela est du à la plus grande importance de sa phase d'optimisation de modularité dans les graphes de petites taille. Dans les graphes de grande taille, sa seconde phase qui est inspirée de Walktrap a plus d'importance. SpongeWalker obtient des scores moins importants dans les graphes à faible densité (jeu 3 et 6), mais dépasse néanmoins la majorité des algorithmes. WalkMiner sans optimisation de modularité obtient les meilleur résultats pour le quatrième jeu de données. Cet algorithme est dépassé de peu par Walktrap dans le premier jeu de données. WalkMiner obtient de très hauts résultats dans le premier jeu, mais voit ses résultats se dégrader dans le jeu à 10 000 nœuds (jeu 4). Ainsi, afin de détecter des petites communautés dans des grand graphes, il est parfois nécessaire de récupérer des partitions plus bas dans le dendrogramme de WalkMiner que la partition sélectionnée par la modularité.

Concernant les graphes à hauts degrés maximum (jeux 2 et 4), le classement des différents algorithmes varie selon le paramètre μ. Pour le jeu 2, nous pouvons remarquer que les scores de Louvain baissent, comparé au score de Walktrap. Cela peut être dû au problème de résolution dont souffre la mesure de modularité [94]. En effet, l'algorithme qui obtient le meilleur score sur ce type de graphe n'optimise pas la modularité (Infomap). Pour ces deux jeux de données, le classement de WalkMiner peut augmenter avec la baisse du paramètre μ. En effet, cet algorithme peut détecter des petites communautés dans les jeux bruités, et la baisse de μ rapproche les partitions détecté par WalkMiner de celles détectées par WalkMiner sans optimisation de modularité, étant donné que la modularité deviens difficile à optimiser dans des réseaux bruités. WalkMiner obtient les meilleurs résultats sur les jeux à faible densité (jeu 3 et 6).

Nous présentons ci-dessous une analyse des patterns retournées par WalkMiner-simple sur le jeu 4 à 10 000 nœuds. Nous avons sélectionné ce jeu pour la variance dans la taille de ses communautés plus élevée que celle des autres jeux. Nous comparons DDPMine-utility avec les algorithmes suivants :

- FPGrowth(adj) : Utilise FPGrowth [88] sur la matrice d'adjacence d'un graphe.
- FPGrowth(H) : Utilise FPGrowth sur la matrice H générée dans la seconde étape de WalkMiner-simple.
- DDPMine(H) : Utilise DDPMine [193] sur la matrice H.
- DDPMine.utility2(H) : Utilise DDPMine sur la matrice H en considérant que l'utilité est la somme des poids des arêtes. Cette définition de l'utilité est similaire aux méthodes de pattern mining pour la maximisation de l'utilité (utility mining) [194, 195, 196].
- DDPMine(adj) : Consiste à utiliser DDPMine sur la matrice d'adjacence.

La figure 4.9 illustre les variations dans les gains d'information des patterns générées par algorithme. Il est possible de voir que l'algorithme proposé ($DDPMine\_utility(H)$) retourne les patterns avec le plus grand gain d'information en moyenne. Cela indique que les patterns re-

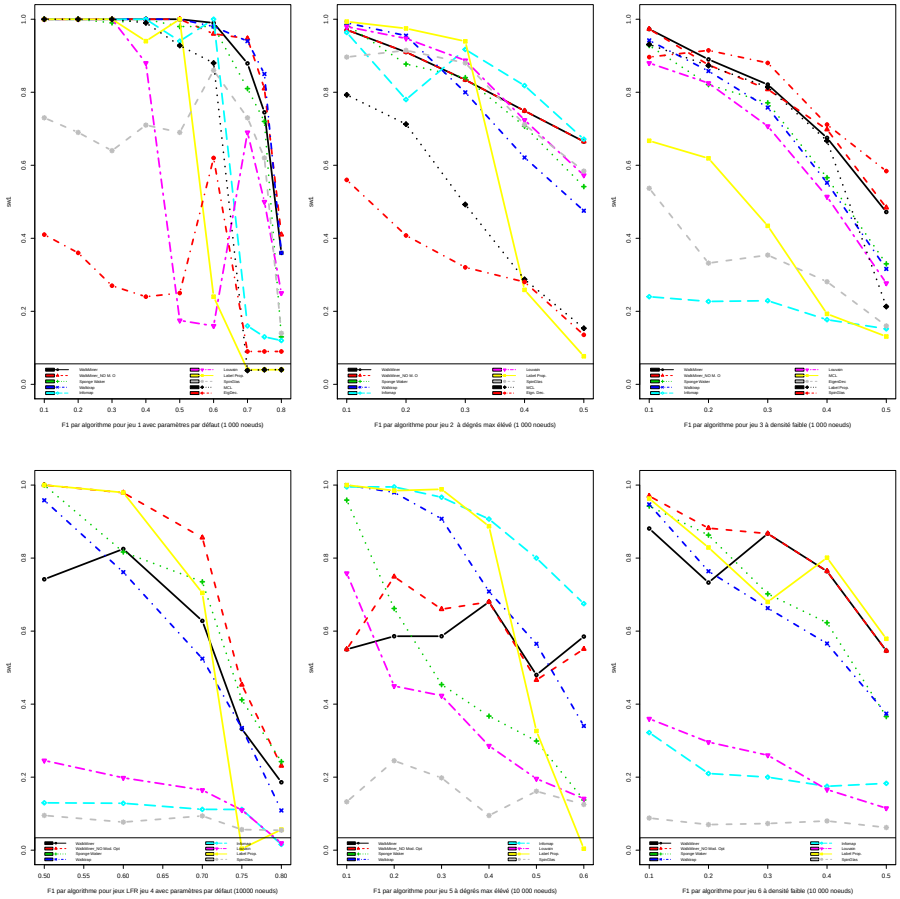

FIGURE 4.8 – Distribution des scores de mesure F1 par μ

tournées par $DDPMine\_utility(H)$ sont plus discriminantes en moyenne que celles retournés par les autres algorithmes.

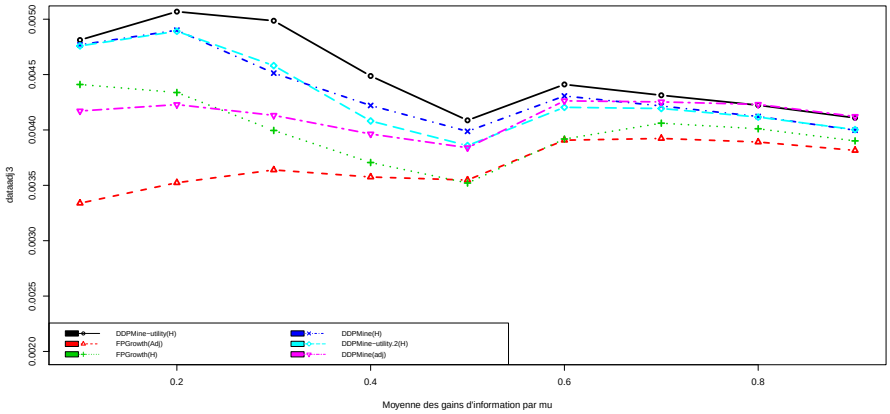

FIGURE 4.9 – Moyenne des gains d'information par μ

La figure 4.10 illustre le nombre de patterns retournés par chaque algorithme. On peut voir que l'algorithme proposé retourne très peu de patterns comparé aux autres algorithmes. Il est dépassé seulement par DDPMiner(adj) qui a un faible gain d'information par pattern.

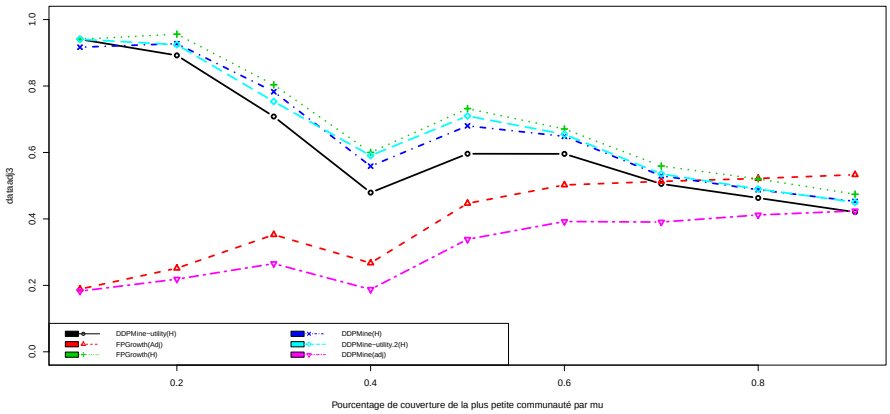

FIGURE 4.10 – Moyenne du nombre de patterns fermées par μ

La figure 4.11 illustre le pourcentage de couverture des voisins des nœuds dans la plus petite communautés de chaque graphe. On peut voir que DDPMine-utility obtient des résultats proches d'algorithmes qui génèrent beaucoup plus de patterns que ce dernier. Même si ce dernier couvre un plus faible pourcentage de voisins que d'autres algorithmes, il faut noter que ses patterns sont en moyenne plus discriminantes (gain d'information élevé) et permettent des séparer les communautés.

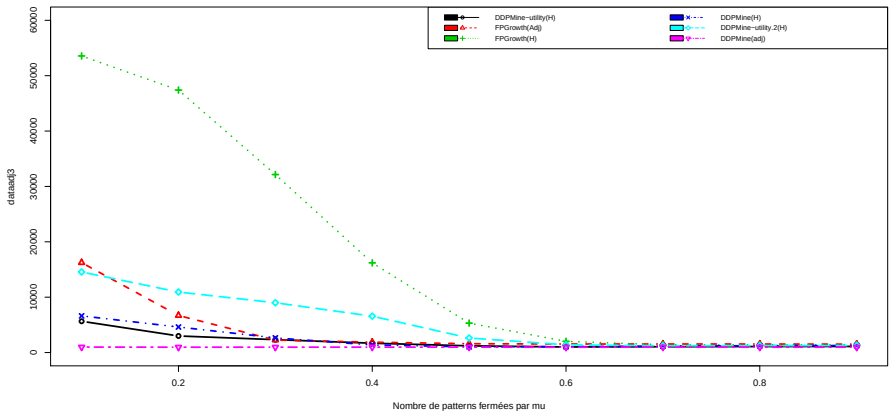

FIGURE 4.11 – Moyenne du pourcentage de voisins couverts par μ pour la plus petite communauté

4.3.2 Réseaux réels

La première expérimentation sur les réseaux réels a été effectuée sur le benchmark du club de karaté de Zachary [204]. Ce club de karaté universitaire avait deux leaders qui ont divisé le club en deux communautés. Les interactions dans ce graphe font référence aux activités que les membres ont effectuées en commun. Ce graphe est composé de 34 nœuds et 78 arêtes.

Les NMIs et les modularités obtenus par les algorithmes de détection de communautés sont présentés dans le tableau 4.2. La première remarque est que les algorithmes ayant obtenu les meilleurs NMI ne sont pas nécessairement ceux ayant obtenu les scores de modularité les plus élevés. Par exemple, Walktrap et Louvain ont obtenu une modularité plus élevée que Infomap et MCL, mais ces derniers ont obtenu un meilleur NMI. SpongeWalker a obtenu des scores de modularité et de NMI plus bas que ceux obtenus par Walktrap et Louvain dans ce graphe. La taille réduite de ce graphe peut expliquer que des algorithmes utilisant seulement les structures locales du réseau sont suffisants pour partitionner ce dernier. Il faut néanmoins noter que SpongeWalker a obtenu le même score de NMI que l'algorithme de décomposition de vecteurs propres (EigenDec), malgré la complexité plus élevée de ce dernier. Aussi, la différence de NMI entre SpongeWalker et la majorité des algorithmes est relativement faible. WalkMiner-simple a obtenu le meilleur score de NMI pour ce graphe.

TABLE 4.2 – Scores de NMI et modularité pour le club de karaté de Zackary [204]

SW		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s	
NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.
0.65	0.35	0.69	0.411	0.825	0.39	0.68	0.41	0.72	0.18	0.65	0.41	0.82	0.39	0.70	0.41	0.83	0.35

La seconde expérimentation sur les réseaux réels a été effectuée sur un réseau de colla-

borations universitaire [205]. Le poids des arêtes a été déterminé à partir d'un questionnaire évaluant l'importance des collaborations entre chercheurs, et les communautés font référence aux différents départements de l'université. Ce graphe est composé de 81 et 817 arêtes.

Les scores de NMI et de modularité sont présentés dans le tableau 4.5. SpongeWalker obtient un des plus hauts scores de NMI dans ce réseau, dépassant Walktrap et Infomap ainsi que la majorité des algorithmes. Comme pour les résultats des données synthétiques, le NMI et la modularité de SpongeWalker sont compris entre celles de Walktrap et de Louvain. Le score de NMI de WalkMiner-simple dépasse celui de Walktrap et de de SpongeWalker, mais ce dernier est inférieur au score de Louvain et de EigenDec. En effet, WalkMiner-simple ne peut pas retrouver des communautés composées de 2 ou 3 nœuds, étant donné que toute communauté ayant une cardinalité inférieure au support minimal est considérée comme du bruit (minSup = 10).

TABLE 4.3 – Scores de NMI et modularité pour le réseau universitaire en Angleterre [205]

SW		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s	
NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.
0.77	0.40	0.69	0.39	0.66	0.39	0.79	0.41	0.55	.21	0.79	0.41	0.67	0.34	0.75	0.40	0.764	0.392

La troisième expérimentation sur des données réelles concerne un réseau d'échange de mails entre les membre d'une institution de recherche européenne. Les 42 communautés de ce réseau représentent des départements et les nœuds sont des chercheurs. Ce graphe est composé de 1005 nœuds et de 25k arêtes. Le tableau 4.4 décrit les NMI et modularités de différents algorithmes obtenus sur ce graphe. Nous pouvons voir que Infomap obtiens les meilleurs résultats. Le reste des algorithmes obtiennent des NMIs beaucoup plus bas que Infomap. SpongeWalker obtiens un score de NMI plus proche de celui de Walktrap (relativement élevé) que de celui de Louvain (relativement faible). Aussi, il est possible de voir que la modularité est n'est pas un indicateur adéquat pour la qualité des partitions dans ce réseau.

TABLE 4.4 – Scores de NMI pour le réseau d'échange de courriels [**?**]

SW		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		WalkMiner-s	
NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.	NMI	Mod.
0.243	0.372	0.263	0.381	**0.50**	0.0613	0.132	0.438	0.160	0.208	0.106	0.341	0.160	0.002	0.134	0.438

Nous avons également effectué des expérimentations sur un jeu de données réel collecté à partir de Twitter. Les données ont été collectées en utilisant TweetCapt [16] entre le 16/04/2018 et le 21/04/2018. Si nous considérons les interactions sur Twitter (Retweet, Reply, Mentions), le graphe généré est souvent peu dense et la majorité des méthodes de détection de communautés obtiennent des partitions peu représentatives de la vérité terrain. Le graphe qui a été utilisé

pour ce test est un graphe de similarité généré à partir des URLs imbriqués dans les Tweets. Ainsi, si deux utilisateurs ont publié un message avec une même URL, une arête entre les deux utilisateurs est créée. Le poids des arêtes est le nombre d'URLs similaires que deux utilisateurs ont publiés. Ce graphe est beaucoup plus dense qu'un graphe d'interaction obtenu sur Twitter (ex. graphe des mentions). Il contient 45K nœuds et 15 millions d'arêtes. Nous n'avons pas la vérité terrain de ce réseau et nous ne pourrons donc pas calculer les scores de NMI. Ce réseau servira seulement à étudier les similarités des partitions retournées par différents algorithmes. Le tableau 4.5 présente ces similarités de partitions calculées à partir des NMI des partitions entre chaque paire d'algorithmes. Nous pouvons voir que la partition retournée par Louvain est celle qui est la plus similaire à la partition retournée par SpongeWalker. Cependant, la similarité entre SpongeWalker et Louvain dépasse à peine 0.5, ce qui signifie que SpongeWalker a fait grandement usage de sa phase de fusion de communautés basée sur Walktrap. Cela explique le fait que la partition de SpongeWalker a un NMI de 0.36, avec celle de Walktrap.

TABLE 4.5 – NMI entre les partitions d'algorithmes de clustering pour le graphe d'URL de Twitter

μ	**S. W.**	**Walktrap**	**Louv.**	**Label Prop.**
S. W.	1	0.36	0.514	0.46
Walktrap	0.36	1	0.189	0.647
Louv.	0.514	0.189	1	0.231
Label Prop.	0.46	0.647	0.231	1

Les cinquième et sixième jeux réels concernent des graphes sur Twitter. Dans ces jeux, nous allons comparer les algorithmes suivants :

- WalkMiner-simple ;
- WalkMiner-edgeAtts ;
- Louvain [98] ;
- Louv.-atts : On utilise l'algorithme de Louvain [98] sur le graphe multiplex généré dans l'étape 2 de WalkMiner-edgeAtts. Nous rappelons que ce graphe multiplex rapproche les nœuds ayant publié des messages dans des thématiques similaires ;
- Label prop : algorithme de propagation de labels [101] ;
- Label Prop-atts : algorithme de propagation de labels [101] sur le graphe multiplex généré dans l'étape 2 de WalkMiner-edgeAtts ;
- Walktrap [96] ;
- Walktrap-atts : Walktrap sur le graphe multiplex généré dans l'étape 2 de WalkMiner-edgeAtts ;
- InfoMap [109] ;
- InfoMap-atts : InfoMap sur le graphe multiplex généré dans l'étape 2 de WalkMiner-

edgeAtts ;
- EigenDec [209] ;
- EigenDec-atts : EigenDec sur le graphe multiplex généré dans l'étape 2 de WalkMiner-edgeAtts ;
- WalkMiner-DisGraphs-X : WalkMiner-DisGraphs où le K-means sphérique prend X classe comme paramètre.
- WalkMiner-DisGraphs-NOREG-X : WalkMiner-DisGraphs sans régularisation (étape 2 de WalkMiner-DisGraphs) où le K-means sphérique prend X classes comme paramètre ;
- Sph.Kmeans-X : K-means sphérique [203] avec X classes ;
- LDA-X : LDA [124] avec X thèmes. Notez que le LDA retourne une distribution de thèmes pour chaque élément et effectue ainsi un partitionnement souple des données. Nous affectons le thème ayant la plus grande probabilité à chaque élément pour obtenir une partition dure.

Les expérimentations effectuées dans le cinquième jeu de données concernent la détection de faux amis (fake followers) sur les réseaux sociaux. Ces faux amis sont des profils qui peuvent être achetés pour augmenter la visibilité d'un utilisateur. Ce comportement peut être considéré comme du SPAM [206] étant donné qu'il consiste à tromper le réseau social pour faire apparaître des publications en tête de liste. Aussi, il permet à des spammeurs et autres utilisateurs malveillants de se faire suivre par des faux amis afin d'éviter la détection. Ce jeu de données est composé de des profils bénins ainsi que de faux amis collectés sur Twitter et mis à disposition par [206]. Le jeu de données est non équilibré avec 2565 comptes normaux et 851 faux amis et contient 275K arêtes. Les auteurs ont achetés les faux comptes des trois plateformes suivantes : fastfollowerz.com, intertwitter.com et twittertechnology.com. Ainsi, le jeu de données contient des biais introduits par les plateformes qui revendent ou créent des faux comptes. Les comptes normaux ont été collectés en sélectionnant aléatoirement des utilisateurs sur Twitter. La méthode d'échantillonnage n'a pas été décrite par les auteurs, mais nous pouvons remarquer qu'un échantillon important de comptes publient des messages en langue italienne. Cela indique que l'échantillonnage utilisé contient plusieurs biais.

Pour chaque algorithme, nous calculons le NMI, la modularité et l'AUC. Concernant l'AUC, on considère que l'algorithme affecte les éléments d'une même communauté à la même classe. Cela est typiquement effectué en labellisant manuellement un échantillon de la communauté et en affectant au membres de cette dernière le label majoritaire de l'échantillon. Il faut noter que le score AUC privilégie les méthodes qui retournent un nombre élevé de communautés étant donné que les membres de ces dernières sont plus densément connectés les uns aux autres et ont une plus forte probabilité d'appartenir à la même classe (homophilie). Aussi, cette mesure pénalise les méthodes ayant une grande variance dans la taille des communautés alors que ces méthodes peuvent être préférées si l'utilisateur dispose d'informations a priori sur les commu-

nautés recherchées. La raison pour cela est que les communautés de grande taille ont une plus forte probabilité d'avoir des éléments de classes différentes.

Nous nous basons sur le NMI pour comparer les différents algorithmes, étant donné que ce dernier permet de comparer des partitions de tailles différentes. Il est possible de voir que WalkMiner-DisGraphs obtient les plus hauts NMI, en comparaison avec les autres algorithmes. Aussi, WalkMiner-simple et WalkMiner-edgeAtts obtiennent de meilleurs NMI que les autres algorithmes pour graphes connectés. On peut également voir que la régularisation proposée augmente le NMI et l'AUC pour les différentes instances de WalkMiner-DisGraphs. Aussi, les algorithmes pour graphes simples retournent un nombre important de communautés, rendant la tâche de caractérisation de ces dernières très difficile. Ainsi, on peut se rendre compte de l'intérêt d'utiliser un algorithme pour graphes disjoints qui permet de retourner un nombre réduit de communautés afin de faciliter la tâche de caractérisation de ces dernières.

TABLE 4.6 – Évaluation des différents algorithmes pour la détection de faux amis

Algo.	Mod	NMI	AUC	num. com.
WalkMiner-simple	0.69	0.28	0.97	307
WalkMiner-edgeAtts	0.68	0.311	0.99	310
Louv.	0.71	0.268	0.96	349
Louv.-atts	0.59	0.051	0.70	424
Label Prop	0.70	0.28	0.986	384
Label Prop-atts	0.57	0.065	0.83	1141
Walktrap	0.70	0.24	0.968	524
Walktrap-atts	0.45	0.13	0.986	3663
InfoMap	0.70	0.26	0.99	492
InfoMap-atts	0.56	0.062	0.83	1049
EigenDec	0.70	0.27	0.94	297
EigenDec-atts	0.029	0.062	0.584	368
WalkMiner-DisGraphs-10	0.50	0.227	0.816	10
WalkMiner-DisGraphs-20	0.68	0.315	0.917	20
WalkMiner-DisGraphs-30	0.67	0.271	0.888	30
WalkMiner-DisGraphs-NOREG-10	0.50	0.231	0.815	10
WalkMiner-DisGraphs-NOREG-20	0.67	0.239	0.84	20
WalkMiner-DisGraphs-NOREG-30	0.67	0.264	0.882	30
Sph.Kmeans-10	0.654	0.276	0.913	10
Sph.Kmeans-20	0.60	0.23	0.935	20
Sph.Kmeans-30	0.60	0.235	0.95	30
LDA-10	0.53	0.223	0.87	10
LDA-20	0.60	0.226	0.866	20
LDA-30	0.487	0.171	0.868	30

La sixième expérimentation sur des réseaux réels concerne des graphes avec des comptes de spammeurs sur Twitter. Ces utilisateurs interagissent avec différents comptes afin de les in-

citer à les suivre. Cela leur permet d'augmenter leur visibilité. Le jeu de données a été collecté par [206]. Il est composé d'utilisateurs bénins ainsi que de spammeurs traditionnels (8% des Spammeurs) et de Spammeurs sociaux (91% des Spammeurs). La première classe de Spammeurs concernent ceux qui publient des messages répétitifs et explicites, se souciant peu de la détection. Ces derniers créent de nouveaux comptes s'ils sont détectés par la plateforme. La deuxième classe de Spammeurs construit une réputation sur le réseau social afin d'éviter la détection. La deuxième classe de Spammeurs peut avoir plusieurs milliers de followers et publie souvent du contenu légitime. Après avoir enlevé les utilisateurs et Spammeurs n'ayant effectué aucune interaction, le jeu contient 5719 comptes et 4.5M d'arêtes (principalement similarité d'URL). Ce jeu de données est équilibré avec 2568 utilisateurs bénins et 2368 Spammeurs. La sélection d'utilisateurs bénins suit la même méthode d'échantillonnage effectué pour le jeu précédant.

Les spammeurs ont été détectés sur Twitter en utilisant des faux comptes pots de miel (honey pots). Ces comptes ont été crées par les auteurs afin de détecter des spammeurs. Les auteurs ont ensuite labellisé manuellement les comptes ayant interagit avec les pots de miels, et n'ont gardé que les comptes de spammeurs. Les faux comptes crées par les auteurs mentionnent/partagent les mêmes comptes (politiciens et autres), et peuvent donc être visibles par les mêmes groupes de spammeurs. Cette homogénéité dans le comportement des pots de miel introduit un biais concernant les comptes de spammeurs détectés. Ces derniers sont nécessairement proches structurellement des comptes retweetés/mentionnés par les pots de miel.

Nous pouvons voir que WalkMiner-simple obtient un plus haut NMI que le reste des algorithmes pour graphes simples. Aussi, l'utilisation du graphe multiplex améliore les NMI pour tous les algorithmes. Enfin, l'utilisation de WalkMiner-DisGraphs permet d'améliorer le NMI et l'AUC, en comparaison avec la version non régularisée (sauf pour 10 classes), ainsi qu'avec les K-means sphérique et LDA.

4.4 Conclusion

La détection de communauté est une tâche de data mining qui a des objectifs divers comme l'exploration des données ou la description de ces derniers. Nous avons proposé dans cette section deux algorithmes de détection de communautés dont la complexité est réduite. Ces derniers permettent de détecter des petites communautés sur les réseaux sociaux comme il a été montré dans les expérimentations.

Nous avons ensuite utilisé la détection de communautés afin de regrouper des utilisateurs malveillants dans les réseaux sociaux. Cette tâche a été effectuée sur le graphe de similarité d'URL qui permettent de connecter les utilisateurs dont les patterns d'interaction sont similaires. Cette section introduit également deux méthodes de détection de communautés dans des

TABLE 4.7 – Évaluation des différents algorithmes pour la détection de SPAMs

Algo.	Mod	NMI	AUC	num. com.
WalkMiner-simple	0.609	0.172	0.952	1231
WalkMiner-atts	0.59	0.173	0.952	1226
Louv.	0.628	0.197	0.959	1224
Louv. atts	0.595	0.178	0.956	1301
Label Prop	0.617	0.163	0.984	1744
Label Prop-atts.	0.568	0.159	0.990	2429
Walktrap	0.610	0.159	0.992	3367
Walktrap-atts.	0.452	0.1501	0.999	8656
InfoMap	0.013	0.132	0.996	6315
InfoMap-atts	0.560	0.156	0.990	2269
EigenDec	0.611	0.175	0.848	1186
WalkMiner-DisGraphs-10	0.56	0.184	0.792	10
WalkMiner-DisGraphs-20	0.582	0.25	0.868	20
WalkMiner-DisGraphs-30	0.598	0.245	0.88	30
WalkMiner-DisGraphs-NOREG-10	0.57	0.245	0.828	10
WalkMiner-DisGraphs-NOREG-20	0.52	0.228	0.848	20
WalkMiner-DisGraphs-NOREG-30	0.59	0.235	0.885	30
Sph.Kmeans-10	0.16	0.137	0.81	10
Sph.Kmeans-20	0.52	0.14	0.80	20
Sph.Kmeans-30	0.59	0.220	0.918	30
LDA-10	0.24	0.12	0.80	10
LDA-20	0.11	0.11	0.83	20
LDA-30	0.09	0.10	0.835	30

graphes attribués. La première utilise les attributs des arêtes pour rapprocher des nœuds partageant les mêmes intérêts, et la seconde utilise les attributs des nœuds afin de détecter des communautés dans des graphes à plusieurs composantes connexes. Les expérimentations effectuées sur des données réelles mettent en évidence l'intérêt de combiner la structure des graphes en plus des attributs afin de détecter des communautés.

Chapitre 5

Classification collective dans les réseaux sociaux multimodaux

La classification collective est un sous-domaîne de la classification qui vise à utiliser les données relationnelles afin de regrouper des éléments dans des classes prédéfinies. Les méthodes de classification collective peuvent également utiliser les attributs des nœuds d'un graphe, en plus de leurs interactions [15, 110, 167, 62, 164]. Les interactions d'un nœud peuvent fortement améliorer les résultats de classification, étant donné que ces derniers suivent souvent des patterns qui peuvent être apprises par un algorithme de classification.

Un des principes utilisé pour la classification collective est le principe d'homophilie [169]. Ce dernier postule que des nœuds similaires ont une plus forte probabilité d'interagir ensemble. Dans certains réseaux néanmoins, il est possible d'identifier l'hétorophilie où des nœuds interagissent avec d'autres nœuds différents (ex : réseaux de célibataires hétérosexuels). Le principe d'homophilie est néanmoins le plus utilisé pour la classification collective.

La classification collective sur les réseaux sociaux peut être dégradée à cause de plusieurs propriétés de ces derniers. La première propriété est la distribution de degrés qui suit une loi de puissance [169]. Ainsi, des nœuds avec un degré élevé peuvent être connectés à des nœuds de classes différentes, alors que des nœuds à faible degré ont très peu de voisins. Ainsi, pour plusieurs nœuds, l'information relationnelle n'est pas suffisante pour effectuer une classification (peu de voisins, connexions à des nœuds à haut degré). Aussi, plusieurs réseaux permettent différents types d'interactions (retweet, reply, like, etc.) qui ont un effet différent sur la classification. Peu de méthodes de classification prennent en compte cette hétérogeneité des interactions, considérant ainsi un seul type d'arêtes.

Cette section présente des méthodes de classification collective sur les réseaux sociaux. La première méthode adapte le KNN pour effectuer une classification sur les attributs des nœuds d'un réseau. La seconde méthode réduit l'effet de la distribution en loi de puissance des degrés sur la classification en générant un voisinage qui utilise la structure du graphe. La troisième

méthode introduit un modèle pour la génération de données synthétiques sur un réseau social. Ensuite, elle propose un algorithme basé sur le boosting qui sélectionne les nœuds à combiner afin de générer des données synthétiques. Enfin, nous proposons un algorithme qui génère des modes d'interaction à partir des modes existants (retweet, reply, like, etc.) et des attributs des arêtes. Cet algorithme utilise ensuite les modes générés afin d'identifier des relations sémantiques entre utilisateurs.

5.1 KNN-LC : Classification pour datasets non équilibrés avec un algorithme basé sur le KNN et des centralités locales

Cette section présente un algorithme de classification supervisée pour les classes non équilibrées. Cette méthode concerne les données non relationnelles, mais peut être facilement étendue aux graphes attribués (relations et attributs). L'objectif de la méthode proposée est d'adapter le KNN pour la classification avec classes non équilibrées en réduisant l'effet du bruit dans les données.

Plusieurs méthodes de classification, se basant sur la pondération de classes [76, 77, 151] permettent d'améliorer la précision et le rappel de la classe minoritaire, mais dégradent la précision de la classe majoritaire. KNN-LC utilise des distances relatives aux voisins, lui permettant d'améliorer la classification de la classe minoritaire. Aussi, cet algorithme utilise les centralités des données d'entraînement durant la phase de classification lui permettant de réduire l'effet du bruit et des éléments non représentatifs. Ces scores de centralité permettent de maintenir une haute précision pour la classe majoritaire. Cette section commence par la méthodologie de KNN-LC. Elle présentera ensuite des expérimentations sur des benchmarks pour les données non équilibrées.

5.1.1 Méthodologie

Cette section commence par présenter le processus global de la méthode proposée, décrit ensuite le calcul du facteur de centralité, et présente enfin la phase de classification de la méthode.

5.1.1.1 Processus global

L'algorithme 15 illustre le processus global de la méthode. KNN-LC commence par construire un arbre couvrant [210] si la dimensionnalité du jeu de données le permet. Ensuite, l'algorithme cherche les K plus proches voisins pour chaque élément dans les données d'entraînement en utilisant l'arbre couvrant. Il faut noter que ses plus proches voisins sont recherchés parmi les

éléments de la même classe. Cela est effectué pour calculer la centralité des éléments d'entraînement dans leurs classes respectives.

Concernant la sélection du paramètre K pour le calcul des plus proches voisins, il est conseillé de choisir $K \geq 10$ dans cette première phase, qui sert au calcul de centralités. Un paramètre K trop bas permettrait à des outliers proches de quelques voisins de ne pas être détectés. Le K choisi dans cette première phase peut être différent du paramètre K choisi durant la phase de classification.

La deuxième phase de cette méthode consiste à construire un graphe g_i pour chaque classe à partir des K plus proches voisins. Ainsi, un nœud dans ces graphes représente un élément d'entraînement, et une arête orientée (n_i, n_j) indique que l'élément n_j est un des plus proches voisins de l'élément n_i.

Les graphes construits sont ensuite utilisés pour calculer un facteur de centralité pour chaque nœud. Ce facteur de centralité est calculé en utilisant le réseau des voisins d'un nœud du second ordre (voisins des voisins).

La dernière phase, qui est la classification des nouveaux éléments, utilise les K plus proches voisins des éléments à classifier, les graphes construits à partir des données d'entraînement, ainsi que les facteurs de centralité des données d'entraînement.

Algorithm 15 KNN-LC

Input

K : nombre de voisins à considérer

X_E : jeu d'entraînement

X : jeu non labellisé

Output

Y : vecteur de labels du jeu X

1: [Calculer les KNN pour les données d'entraînement](X_E)
2: $[g_c]$ ←[Construire le graphe des KNN pour chaque classe]
3: CF ←[Calculer la centralité (CF) pour chaque élément des données d'entraînement](X_E,$[g_c]$)
4: Y ←Classification(X,CF,X_E,$[g_c]$)
5: retourner Y

5.1.1.2 Facteur de centralité (CF)

Il existe une littérature abondante sur la quantification de la centralité d'un nœud [211, 212, 213, 214, 215], et il est possible de sélectionner différentes mesures qui pourraient être incorporées dans la méthode proposée. Ces mesures doivent néanmoins répondre aux critères suivants :

– Les nœuds à faible centralité peuvent être identifiés comme des outliers ;

– La proximité aux nœuds représentatifs de la classe doit avoir un plus haut effet sur l'augmentation de la centralité que la proximité aux nœuds peu représentatifs.

Le score de pagerank répond aux deux critères ci-dessus, et permet ainsi de détecter les outliers en utilisant le graphe des voisins de chaque nœud. Ce score est également rapide à calculer, étant donné que seuls les graphes des voisins du second ordre sont utilisés.

La majorité des méthodes utilisant le pagerank calculent cette mesure sur l'intégralité du graphe, étant donné qu'elle permet de trouver les zones marginales. Nous avons préféré calculer le pagerank seulement sur un graphe des voisins du second ordre pour que des zones peu représentées d'une classe ne soient pas fortement pénalisées. Ainsi, les outliers que nous souhaitons identifier sont des outliers locaux, définis par la proximité de leurs voisins. Cela permet de ne pas privilégier les nœuds proches du centre de masse d'une classe dans le calcul du facteur de centralité.

5.1.1.3 Phase de classification

Le KNN est considéré comme un algorithme paresseux étant donné que tous les calculs effectués par l'algorithme sont faits dans la phase de classification. Pour cette raison, il est important que la phase de classification de l'algorithme proposé soit simple pour ne pas augmenter la complexité de ce dernier.

La phase de classification est décrite par l'algorithme 16.

Algorithm 16 KNN-LC : Classification

Input :

X,CF,X_E,$[g_i]$

Output :

Y : labels de X

1: Pour chaque élément n dans X :
2: Pour chaque classe c :
3: $KNN_c(n) \leftarrow$ [calculer les K plus proches voisins de n dans la classe c](n,X)
4: $TP_{n \to c} \leftarrow$ [calculer la probabilité de transition de n vers la classe c](n,X)
5: $TP_{KNN_c(n) \to n} \leftarrow$ [calculer les probabilités de transition des voisins de n dans la classe c vers n](n,$KNN_c(n)$,g_c)
6: $CF_c(KNN_c(n)) \leftarrow$ [calculer les facteurs de centralité des voisins de n]($KNN_c(n)$,g_c)
7: $c_n \leftarrow$ [Sélection de la classe de n]($TP_{n \to c}$,$TP_{KNN_c(n) \to n}$,$CF_c(KNN_c(n))$)

A la réception de nouveaux éléments à classifier, la première étape, similaire à celle du KNN classique, consiste à identifier les K plus proches voisins. La différence entre les deux algorithmes dans cette étape est que le KNN-LC va identifier les K plus proches voisins dans chaque classe c. Ainsi, concernant la classification binaire, le KNN-LC va identifier les K plus

proches voisins de chaque élément dans la classe positive ainsi que ses K plus proches voisins dans la classe négative.

L'étape suivante est de calculer des probabilités de transitions de l'élément n vers toutes les classes. Cette probabilité permettra le calcul d'un score final entre un élément et les classes auxquelles il peut appartenir.

La probabilité de transition est une mesure entre un élément à classifier n et ses voisins dans une classe c. Ces derniers font partie des données labellisées (données d'entraînement). La probabilité de transition se base sur une mesure de proximité présentée dans l'équation 5.1. Cette proximité entre un élément à classifier n et un de ses K plus proches voisins v est obtenue en soustrayant la distance entre n et v de la distance entre n et le voisin le plus distant parmi les K plus proches voisins de n. En d'autres termes, cette proximité est obtenue en inversant les distances entre n et ses voisins (distances transformés en proximités). Nous utilisons la distance euclidienne comme distance entre les vecteurs d'attributs de deux nœuds, mais toute autre distance peut être utilisée. La probabilité de transition de n vers un voisin v est présentée dans l'équation 5.2. Elle est calculée à partir de la proximité dans l'équation 5.1 normalisée par la somme des proximités vers les K plus proches voisins de n.

$$P^c_{n\to v} = max(dist[n, KNN_c(n)]) - dist[n, v] \tag{5.1}$$

$$TP^c_{n\to v} = \frac{P^c_{n\to v}}{\sum_{i\in KNN_c(n)} P^c_{n\to i}} \tag{5.2}$$

La probabilité de transition de n vers une classe c est calculée par l'équation 5.3. Cette dernière est la somme de probabilité de transition entre n et les voisins de n dans la classe c normalisée par un facteur Z, pour que la somme des probabilités de transition pour toutes les classes soit égale à 1.

$$TP_{n\to c} = \frac{\sum_{v\in KNN_c(n)} TP^c_{n\to v}}{Z} \tag{5.3}$$

$$Z = \sum_c \sum_{v\in KNN_c(n)} TP^c_{n\to v} \tag{5.4}$$

Après avoir calculé la probabilité de transition de chaque élément n pour toutes les classes, un score est calculé qui prend en compte 1) cette dernière, 2) les probabilités de transition des voisins de n vers ce dernier, 3) la centralité des voisins de n. La probabilité de transition des voisins d'un élément n vers ce dernier est calculée par la même manière que la probabilité de transition de n vers ses voisins (équation 5.3).

La centralité d'un voisin v est calculée par l'équation 5.5. Cette centralité est composée du pagerank pondéré par un paramètre α et d'un modèle nul pondéré par (1-α). Le module nul quantifie la valeur attendue du score pagerank du voisin. Le score est normalisé par $N_{voisinage}$,

qui est le nombre de nœuds dans le graphe de voisinage du second ordre. Le paramètre de contrôle $\alpha \in [0, 1]$ permet d'augmenter ou de réduire l'importance de la centralité pour une classification. S'il est estimé que les données d'entraînement sont bruités, il est conseillé d'augmenter la valeur de α afin de réduire la contribution des voisins qui peuvent être des anomalies. Par défaut, on initialise α à 0.5 dans toutes les expérimentations qui seront présentées.

$$CF_c(v) = (1 - \alpha) \cdot \frac{1}{N^2_{voisinage}} + \alpha \cdot \frac{TP_{v \to c}}{N_{voisinage}} \tag{5.5}$$

Le score final entre un élément à classifier n et une classe c est calculé par l'équation 5.6. Ce score est composé de la probabilité de transition de n vers la classe c. Cette dernière est multipliée par la racine carrée de la moyenne des probabilités de transition des K plus proches voisins de n dans la classe c vers n. Les probabilités de transition sont pondérées par les centralités des voisins. Ainsi, les voisins qui peuvent être des anomalies ou qui sont peu représentatifs de leurs voisinages ont une faible contribution au score final. La seconde composante de ce score ($TP_{v \to n}$) permet de répondre au problème de classes non équilibrées, étant donné que les éléments de la classe majoritaire ont une plus grande probabilité d'être proches les uns des autres, et ainsi, d'avoir une probabilité de transition plus faible avec l'élément à classer. En effet, la seconde composante permet de normaliser les proximités entre l'élément a classer n et le K plus proche voisins dans une classe v, par les voisins de v, permettant à la classe minoritaire d'augmenter ses probabilités de transition et ainsi son rappel.

$$score_c(n) = TP_{n \to c} \cdot \sqrt{\sum_{v \in KNN_c(n)} \frac{TP_{v \to n} \cdot CF_c(v)}{K}} \tag{5.6}$$

Enfin, l'élément à classifier est affecté à la classe dont le score est le plus élevé (équation 5.7).

$$c_n = argmax(score_{:}(n)) \tag{5.7}$$

5.2 Expérimentations

On présente dans cette section des expérimentations effectuées sur deux benchmarks avec des données hautement non équilibrées. La méthode proposée est comparée au KNN, KNN avec sous-échantillonnage de la classe majoritaire uniquement, KNN avec sous-échantillonage de la classe majoritaire et sur-échantillonnage de la classe minoritaire, et KNN avec augmentation par données synthétiques (SMOTE [86]). Nous ne présentons pas les résultats avec sur-échantillonage seulement, étant donné que les résultats sont peu concluants. On utilise 66% des

données pour l'entraînement et 33% pour le test en respectant la distribution des données par classe.

Le jeu de données utilisé [216] dans cette première expérimentation concerne des transactions par cartes de crédit. La classe positive concerne les transaction frauduleuses et la classe négative qui est majoritaire concerne des transactions non frauduleuses. Ce premier jeu de données contient 284,315 négatifs et 492 positifs.

Il est possible de voir que les algorithmes avec le plus haut taux de bien classés sont le KNN et KNN-LC pour K=10. Concernant le KNN, ce haut taux de biens classés s'explique par le fait que si tous les éléments positifs sont classés comme négatifs, il est possible d'obtenir un haut taux de biens classés. En effet, ce taux n'est pas adapté à l'évaluation de la classification pour classes non équilibrées, mais nous le présentons étant donné sa popularité. L'algorithme proposé a le deuxième score de biens classés pour K=5 après celui du KNN classique. Ainsi, nous pouvons voir que l'amélioration des vrais positifs par le KNN-LC ne conduit pas à une réduction forte des vrais négatifs, contrairement aux autres méthodes présentées. Aussi, la méthode proposée obtient le plus haut TPR pour les deux valeurs de K. Ce taux est suivi de près par le KNN avec SMOTE, qui néanmoins obtient des valeurs plus basses pour le TNR.

TABLE 5.1 – Données de transactions de cartes de crédit K=10

	KNN-LC	**KNN**	**KNN-Sous échant.**	**KNN-Sous et Sur Echant.**	**KNN-SMOTE**
Acc	99,93%	99,94%	97,48%	97,15%	97,98%
TNR	99,96%	99,98%	97,5	97,16%	97,99
FNR	0.04%	0.02%	2.5	2.83%	2%
TPR	99,41%	76,75%	89,53	87,8%	92
FPR	0.58%	23.25%	10.47	12.2%	8%

TABLE 5.2 – Données de transactions de cartes de crédit K=6

	KNN-LC	**KNN**	**KNN-Sous échant.**	**KNN-Sous et Sur Echant.**	**KNN-SMOTE**
Acc	99,95%	99,99%	95,31%	96,42%	96,48%
TNR	99,95%	99,98%	95,32%	96,52%	96,48
FNR	0.05%	0.02%	4.68%	3.48%	3.52%
TPR	98,83%	79,53%	90,64%	90,65%	95,51
FPR	1.17%	20.47%	9.36	9.35%	4.49%

Les tables 5.3 et 5.4 présentent des expérimentations effectuées sur les données de classification de mesures du Ionosphère [217]. Ce jeu de données est constitué de mesures radar dans la région du Labrador au Canada. Les mesures correspondant à certaines structures dans le Ionosphère sont considérées positives alors que les mesures ne correspondant à aucune structure

sont considérées négatives. Le jeu de données est composé de 351 éléments décrits par 32 attributs. Il est plus équilibré que le jeu de transactions par cartes de crédit, étant donné qu'un tiers des éléments sont positifs.

On présente les résultats du KNN-LC, du KNN avec sous-échantillonage et du KNN avec SMOTE. Nous avons écarté les résultats du KNN classique, étant donné que ses performances sur la classe minoritaire étaient très faibles. Il est possible de voir que la méthode proposée a le plus haut taux de biens classés ainsi que le plus haut TPR pour les deux valeurs de K.

TABLE 5.3 – Mesures du Ionosphère K=10

	KNN-LC	**KNN-Sous Echant.**	**KNN-SMOTE**
Acc	98.36%	81.95%	91.8%
TNR	97.43%	96.15%	98.7
FNR	2.57%	3.85%	1.3
TPR	100%	56.81%	79.54
FPR	0%	43.19%	20.26

TABLE 5.4 – Mesures du Ionosphère K=6

	KNN-LC	**KNN-Sous Echant.**	**KNN-SMOTE**
Acc	97.54%	82.78%	96.61
TNR	96.15%	98.71%	97,29
FNR	3.85%	1.28%	2.71
TPR	100%	54.54%	95.45
FPR	0%	45.46%	4.55

5.2.1 Conclusion

La classification des données non équilibrées est devenue un des sujets les plus importants dans la classification automatique. Cette thématique a reçu beaucoup d'attention de la part de la communauté des chercheurs ainsi que des professionnels de la classification automatique, après qu'ils aient observé que plusieurs algorithmes obtenaient de faibles performances sur les jeux non équilibrés.

Cette section a introduit un algorithme nommé KNN-LC qui calcule des centralités locales des données d'entraînement avant d'effectuer une classification basée sur le KNN. Ces centralités locales donnent plus de poids au K plus proches voisins qui sont centraux dans leurs voisinages. Cette méthode a l'avantage de réduire l'effet des outliers dans les données. Cela permet d'améliorer la classification de la classe minoritaire tout en maintenant le taux de biens classés.

Plusieurs expérimentations ont été effectuées sur des benchmarks qui comparent la méthode proposée avec des variations de KNN. Ces expérimentations montrent que la méthode proposée permet d'obtenir un haut taux de biens classés ainsi qu'un haut TPR.

5.3 Classification collective sur les réseaux sociaux multimodaux avec variation de voisinage

Cette section présente une méthode de classification collective qui utilise la structure du réseau pour définir et faire varier les voisinages des nœuds. Le premier objectif de cette méthode est de réduire la propagation d'erreurs de classification dans le graphe introduites par l'agrégation de labels de voisins [218]. Le second objectif de cette méthode est de répondre à l'effet de la distribution en loi de puissance des degrés qui introduit beaucoup de bruit dans une classification collective [219]. En effet, alors que l'utilisation des voisinages locaux implique que des nœuds populaires sont connectés à un nombre important de nœuds de classes différentes, l'utilisation de voisinages identifiés par des proximités structurelles retourne une plus grande diversité de voisins avec un faible couplage entre les différentes classes.

On modélise un réseau social ayant différents types d'interactions par un graphe multimodal. La méthode de classification introduite est basée sur l'algorithme itératif de classification (ICA) [15] qui prend en compte les attributs des nœuds ainsi que les attributs et les labels prédits de leurs voisins (figure 5.1). L'ICA apprend les poids correspondants aux labels des voisins sur des données d'entraînement dont la labellisation est souvent de haute qualité (labellisées manuellement). Cela induit que l'algorithme donne un grand poids aux labels des voisins, qui sont dans la phase d'utilisation du modèle, prédits. L'algorithme présenté introduit une méthode de sélection de voisinages pour les réseaux multimodaux afin de réduire l'effet de la distribution en loi de puissance des degrés, et varie les voisins dans la phase de classification pour limiter la propagation d'erreurs induites par les labels des voisins.

5.3.1 Méthodologie

L'algorithme proposé que nous nommons "ICA-SN" est basé sur l'algorithme ICA [15] présenté dans le chapitre 2.3.3.2. Il effectue ainsi une propagation itérative de labels en utilisant les labels des voisins, les attributs des nœuds et les attributs des voisins. Il diffère de l'ICA dans les aspects suivants :

– ICA-SN considère qu'un réseau social est composé de différents types d'arêtes. Ainsi, il effectue une pondération des modes pour identifier des voisinages pour chaque nœud. La modélisation multimodale permet d'améliorer la classification, étant donné que les modes

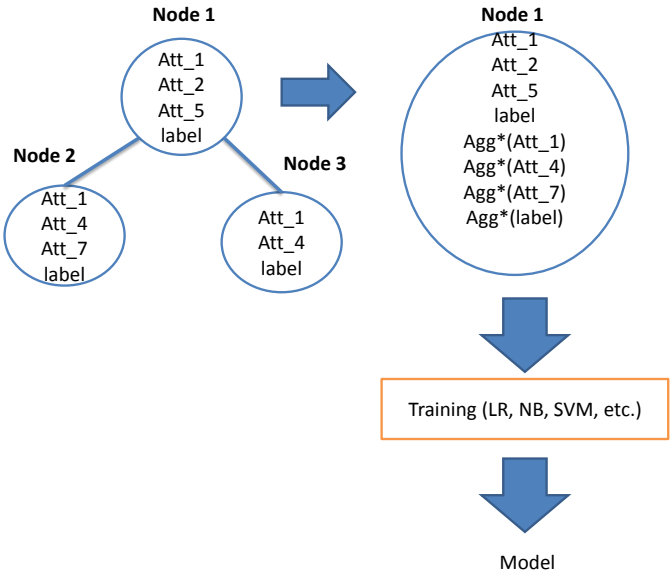

FIGURE 5.1 – Génération de données synthétiques par SMOTE-SN

qui introduisent beaucoup de bruit reçoivent un poids inférieur aux modes moins bruités, et ont ainsi peu d'effet sur la génération de voisinages.

– Un nouveau voisinage pour chaque nœud est échantillonné à chaque itération de l'ICA-SN, contrairement à l'ICA. Comme noté par [218], la phase d'entraînement de l'ICA est effectuée avec un voisinage dont les labels sont corrects. Cependant, dans la phase de classification, les labels des voisins sont ceux obtenus par l'algorithme dans chaque itération, et contiennent donc plusieurs instances mal prédites. Étant donné qu'un modèle appris avec des labels de voisins corrects va donner plus de poids à ces derniers, des erreurs de classification risquent de se propager sur le réseau.
  L'algorithme proposé réduit l'effet de cascade d'erreurs de classification en faisant varier les voisins de nœuds à chaque itération.
– L'ICA-SN affecte un nœud à la classe qui lui été attribué le plus fréquemment. Notez qu'il est possible d'ignorer les premières itérations si on considère que les labels des nœuds changent fréquemment, et nécessitent plusieurs itérations afin de se stabiliser. Ce processus ressemble à l'algorithme de Gibbs [169], avec la différence que les labels ne sont pas échantillonnés.

ICA-SN est utilisé pour classifier des profils d'utilisateurs ayant plusieurs modes d'interaction. On considère que chaque mode d'interaction $l \in L$ définit un graphe $g_l = V_l \times E_l$, où V_l est l'ensemble des nœuds de g_l et E_l est l'ensemble des arêtes de g_l. Un nœud qui n'a aucune interaction appartient à l'ensemble des nœuds V, sans appartenir à aucun des graphes g_l.

L'algorithme 17 décrit l'ICA-SN. La première étape de l'algorithme est de pondérer les modes d'interaction en fonction de leurs capacités à discriminer les différentes classes. La seconde étape est de construire un graphe multimodal $\mathcal{M}$ à partir des modes et de leurs pondérations. Ensuite, l'algorithme construit les voisinages des différents nœuds ($Voisinage(n)$). Enfin, l'algorithme effectue plusieurs itérations où il échantillonne des voisins à partir des voisinages des nœuds. Les voisins échantillonnés ainsi que les nœuds sont utilisés pour générer des vecteurs d'attributs fournis en entrée pour le modèle de classification f.

Algorithm 17 ICA-SN

T : nombre d'itérations
L : modes dans le graphe
E : ensembles d'arêtes pour chaque mode ($E_1, E_2, ..E_l$)
V : ensembles de nœuds pour chaque mode ($V_1, V_2, ..V_l$)
f : modèle de classification (SVM, Naive Bayes, Arbres, etc.)

W(L)←pondération des modes (L,E,V)
$\mathcal{M}$ ← Construction du graphe multimodal (L,W,V,E)
Pour chaque $n \in V$
    Voisinage(n)←Génération des voisinages (n,$\mathcal{M}$)
Pour t dans 1..T :
    Pour chaque $n \in V$
        Voisins(n) ← échantillonnage pondéré (Voisinage(n))
        $y_t(n)$ ←f(n,,Voisins(n))//Classification
Agrégation des résultats(y)

Pondération des modes : dans cette étape, les données d'entraînement sont utilisées afin d'assigner un poids à chaque mode d'interaction. Plusieurs méthodes peuvent être utilisées pour cela, incluant la génération d'un ensemble de validation où chaque mode est utilisé à tour de rôle pour la génération de voisins, puis est évalué sur l'ensemble de validation. Cette méthode qui semble intuitive nécessite néanmoins un jeu d'entraînement assez grand pour que le jeu de validation soit représentatif.

Nous proposons de détecter les voisins des nœuds dans chaque mode, et de calculer un ratio de voisins intra-classe, sur l'ensemble des voisins. On utilise un paramètre de lissage afin que les modes avec peu d'observations soient pénalisés par rapport aux modes ayant un grand nombre d'observations. L'algorithme 18 introduit la pondération des modes utilisées dans ICA-SN.

Afin d'identifier les voisinages pour chaque nœud, l'algorithme 18 commence d'abord par calculer la transition t de la chaîne de Markov construite à partir de la matrice d'adjacence des différents modes $l \in L$. Cette chaîne est construite en normalisant la matrice d'adjacence d'un graphe G_l par la somme de ces lignes (eq 5.8).

Algorithm 18 pondération des modes

L : modes dans le graphe

E : ensembles d'arêtes ($E_1, E_2, ..E_l$)

V : ensembles de nœuds ($V_1, V_2, ..V_l$)

1: Pour chaque mode $l \in L$
2: $M_l^t \leftarrow$ [Calcul de la transition t de la chaîne de Markov du mode] (E_l)
3: Pour chaque paire de nœuds connectés $(n_i, n_j) \in V_l^2$
4: $D^l(n_i, n_j) \leftarrow$ [Calcul de distance structurelle] (n_i, n_j, M_l^t)
5: Pour chaque nœud $n \in V_l$
6: Voisinage(n) $\leftarrow$ KNN(D^l,n)
7: W_l = [Calcul du poids] (Voisinage(V_l))

$$M_{i,x}^l = \frac{a_{i,x}^l}{\sum_{j \in V_l} a_{i,j}^l} \tag{5.8}$$

On pourrait être tenté d'utiliser la chaîne de Markov afin de déterminer un voisinage des nœuds en considérant que si un nœud n_1 a une forte probabilité de transition vers un autre nœud n_2, le second devrait être considéré comme un voisin. Néanmoins, ce voisinage ne prend pas en compte la distribution en loi de puissance des degrés d'un réseau social. Ainsi, les nœuds à hauts degrés vont être voisins de nœuds appartenant à des classes différentes, réduisant la capacité du modèle à séparer les classes.

Après avoir calculé la transition t de la chaîne de Markov, on utilise une mesure de distance entre les nœuds, qui prend en compte la structure du réseau. On considère les deux critères suivants pour évaluer une mesure de distance utilisée pour la sélection d'un voisinage pour les nœuds d'un réseau social.

- Les nœuds ayant une faible distance doivent avoir une forte probabilité d'appartenance à la même classe ;
- La majorité des nœuds d'un réseau peuvent être assignés suffisamment de voisins pour effectuer la tâche de classification.

La communauté de recherche travaillant sur la classification non supervisée a également œuvré pour proposer des mesures de distances permettant de séparer les nœuds appartenant à des communautés différentes [103]. Plusieurs méthodes de classification non supervisée utilisent des mesures de distances qui se basent sur les distances bibliométriques (voisins en commun) [103, 14, 96].

La mesure que nous utilisons est celle également utilisée par Walktrap [96] pour la détection de communautés. L'équation 5.9 décrit cette mesure. Elle est calculée à partir de la chaîne de

Markov à l'itération t construite à partir de la matrice d'adjacence. Cette mesure est la somme des distances entre deux nœuds. La distance entre deux nœuds est calculée à partir de la chaîne de Markov normalisée par la racine carrée d'une fonction de centralité d'un nœud, deg est une fonction qui retourne le degré d'un nœud.

$$D^l(i,j) = \frac{\sum_{k \in V_l} |m^l_{i,k} - m^l_{j,k}|}{deg^l(k)} \tag{5.9}$$

Ensuite, on identifie les K plus proches voisins (KNN) pour chaque nœud (calcul de voisinages) dans un mode. Ces derniers sont sélectionnés parmi les voisins d'un nœud dans chaque graphe g_l pour éviter le calcul coûteux du KNN. Si un nœud a peu de voisins, il est conseillé de calculer son voisinage parmi les voisins de ses voisins. Ensuite, on calcule le nombre de voisins appartenant à la même classe pour chaque nœud présents dans le mode ($nb_{intra}(V_l)$), et le nombre total de voisins pour chaque nœud, également présents dans le mode ($nb(V_l)$).

La pondération qui est effectuée est introduite par l'équation 5.10. Le poids d'un mode est le nombre d'arêtes intra-classe sur la somme des arêtes dans un mode. Cette mesure est lissée par un paramètre α pour pénaliser les modes avec peu d'observations.

$$W_l = \frac{nb_{intra}(V_l)}{\alpha + nb(V_l)} \tag{5.10}$$

Construction du graphe multimodal : Si le réseau social n'est pas multimodal, l'algorithme saute directement à l'étape suivante. Dans le cas contraire, un graphe multimodal utilisant le modèle introduit dans [180] est construit à partir des différents modes d'interaction. Les poids des arêtes de chaque mode sont pondérés par les poids identifiés dans l'étape précédente (W_l).

Génération de voisinages :

Cette étape comprends le calcul de l'étape t de la chaîne de Markov du graphe multimodal $\mathcal{M}$. Ensuite, les K plus proches voisins de chaque nœud sont identifiés, comme ce qui a été effectué dans l'étape de pondération des modes. La différence entre ces deux étapes est que le calcul des voisinages s'effectue sur le graphe multimodal $\mathcal{M}$ et non sur les graphes des différents modes G_l. Pour cela, une distance D_M est calculée pour chaque pair de nœud connectés dans $\mathcal{M}$ (équation 5.9). Ce calcul est effectué sur la chaîne de Markov à l'étape t du graphe multimodale $\mathcal{M}$. Ensuite, les K plus proches voisins de chaque nœud sont retournés en utilisant D_M.

Échantillonnage pondéré :

Après le calcul des voisinages, un échantillonnage pondéré est effectué afin de déterminer les voisins de chaque nœud à chaque itération (fonction Voisins). Le poids assigné à chaque voisin est calculé par l'équation 5.11. Ce dernier est l'exponentiel de la négation de la distance entre deux voisins, sur la moyenne des distances entre un nœud et ses voisins. Cette valeur est normalisée par $\mathcal{Z}$ pour la transformer en probabilité (eq 5.12). Ainsi, les voisins échantillonnés

sont structurellement proches des nœuds. Cet échantillonnage de voisins permet de limiter la propagation d'erreurs, étant donné que les voisins des nœuds changent dans chaque itération de l'algorithme permettant de corriger certaines erreurs induites par l'agrégation de leurs attributs.

$$p_{(i,j)} = \frac{e^{(-\frac{D_M[i,j]}{moyenne(D_M(i,KNN(i)))})}}{\mathcal{Z}} \tag{5.11}$$

$$\mathcal{Z} = \sum_{(i,j)\in V^2} e^{(-\frac{D_M(i,j)}{moyenne(D_M(i,KNN(i)))})} \tag{5.12}$$

Classification : Des fonctions d'agrégation (min, max, sum, mean, etc.) sont utilisées ensuite pour agréger les attributs des voisins de chaque nœud. Ces derniers sont ajoutés aux attributs du nœud pour générer un vecteur d'attributs. Le nœud est ensuite classifié en utilisant un modèle généré par un algorithme de classification (Naïve Bayes, Régression Logistique, Arbres de Décision, etc.).

Agrégation des résultats : Dans cette étape, la classe qui a été la plus fréquemment affectée à chaque nœud est sélectionnée (eq 5.13).

$$y_n = argmax(y_{\cdot}(n)) \tag{5.13}$$

5.3.1.1 Conclusion

Plusieurs méthodes de classification collective sur les graphes utilisent le principe d'homophilie qui considère que des nœuds de la même classe ont une plus forte probabilité d'être connectés. Ce principe n'est pas toujours vérifié à cause de la distribution en loi de puissance des degrés des nœuds. En effet, les nœud à hauts degrés peuvent être connectés à d'autres nœud de classes différentes. Aussi, plusieurs méthodes de classification collective utilisent les labels des voisins pour effectuer leurs prédictions. Ces modèles sont générés sur des données d'apprentissage où les nœuds sont généralement labellisés manuellement. Ainsi, les algorithmes de classification peuvent donner beaucoup de poids à ces attributs. Lorsque ces algorithmes sont mis en production, les labels des voisins sont ceux qui ont été prédits par l'algorithme, et contiennent donc beaucoup de fausses prédictions.

L'algorithme proposé dans cette section modélise les interactions sur les réseaux sociaux par un graphe multimodal. Ce dernier donne plus de poids aux modes d'interaction discriminants. L'algorithme effectue ensuite une sélection des voisins potentiels de chaque nœud en utilisant des distances qui prennent en compte la structure du réseau multimodal. Ces voisinages ne sont pas affectés par la distribution en loi de puissance des degrés, et ont donc plus de diversité. L'algorithme échantillonne ensuite des voisins dans chacune de ces itérations afin de limiter la propagation d'erreurs sur le réseau.

5.4 SN-Boost : Méthode de classification non équilibrée sur les RSN basée sur la génération de données synthétiques

Plusieurs tâches de classification sur les réseaux sociaux numériques sont caractérisées par des classes non équilibrées. La détection de spammeurs, d'apologistes au jihad ou de faux profils sont des exemples. Alors que la classification dans les réseaux sociaux a reçu beaucoup d'attention de la part de la communauté scientifique, la classification non équilibrée sur ces réseaux a été peu étudiée. En effet, la majorité des méthodes de classification collective répondent au problème de classes non équilibrées en amont de l'utilisation de leurs algorithmes. Nous proposons dans cette section un algorithme basé sur le Boosting qui génère itérativement des données synthétiques sur les réseaux sociaux multimodaux.

La génération de données synthétiques [86, 83, 87] est une méthode populaire pour répondre au problème des classes non équilibrées. Le principe de cette méthode est de générer des données en utilisant une interpolation des éléments et de leurs plus proches voisins. Cela permet de d'étendre la frontière de la classe minoritaire, et souvent, d'améliorer la classification. La méthode la plus utilisée pour la génération de données synthétiques est la méthode SMOTE [86] ainsi que ses variantes. La méthode SMOTE a été conçue pour les données continues et n'est pas adaptée aux données textuelles et structurelles caractérisant les réseaux sociaux. Dans plusieurs tâches de classification, la présence ou l'absence d'un attribut textuel a plus d'importance que le poids qui lui est attaché. L'utilisation du SMOTE classique sur les attributs textuels augmentera le nombre d'attributs non nuls de l'élément synthétique par rapport aux éléments source, créant de nouveaux éléments qui diffèrent de la distribution du jeu d'entraînement.

Certaines variantes de SMOTE [220, 86] permettent de générer des données synthétiques pour les variables catégoriques ou binaires, mais ne sont pas adaptées aux données textuelles à haute dimension. Une des stratégies est de créer un élément synthétique en utilisant un vote par majorité des voisins [86]. Ainsi, la valeur d'un attribut binaire ou catégorique prend la valeur la plus fréquente parmi les voisins de l'élément. Dans le cas des données textuelles, les voisins ont tellement peu d'éléments en commun qu'il est difficile de générer des éléments en suivant cette stratégie.

Une seconde stratégie souvent adoptée pour la génération de données synthétiques avec attributs textuels est d'effectuer une réduction de dimensionalité sur les données textuelles et ensuite d'effectuer un SMOTE classique [221]. Ces méthodes sont limitées par l'efficacité de la réduction de dimensionalité. La réduction de dimensionalité bien qu'elle soit justifiable dans certains cas, peut conduire à la perte d'attributs déterminants pour la classification [222]. Certaines méthodes de réduction de dimensionalité nécessitent de connaitre la distribution à priori des classes [221] et d'autres peuvent dégrader les performances de la classe minoritaire. Aussi, ils ne prennent pas en compte les attributs structurels qui ont des relations non linéaires entre

eux. En effet, plusieurs de ces attributs sont obtenus en effectuant des calculs impliquant la structure du graphe. Nous proposons dans cette section une méthode qui permet de générer des données synthétiques pour nœuds dans des graphes avec arêtes ayant des attributs textuels.

Une seconde problématique liée à la génération de données synthétiques est l'identification des plus proches voisins. Les méthodes basées sur SMOTE utilisent la distance euclidienne pour identifier ces éléments. Dans le cadre de la classification sur les réseaux sociaux, la distance euclidienne est affectée par la haute dimensionalité des éléments. Une alternative est d'utiliser une distance (ou proximité) comme le cosinus qui est plus adaptée aux données à hautes dimensions. Les réseaux sociaux sont néanmoins caractérisés par des interactions en plus des attributs textuels. Cela permet de calculer des distances basées sur la structure des graphes sociaux en plus des distances basées sur les attributs textuels. Nous proposons une méthode de sélection des voisins qui utilise le graphe social ainsi que les attributs textuels. La méthode proposée permet d'éviter le chevauchement des classes et ajoute de la diversité dans l'algorithme de Boosting.

Cette section commence par détailler la méthode de sélection des voisins utilisée pour la génération de données synthétiques. Ensuite, elle va présenter une méthode permettant la génération de ces données. La troisième sous-section introduit l'algorithme de Boosting proposé.

5.4.1 Sélection de voisinages pour la génération de données synthétiques

Les réseaux sociaux sont caractérisés par plusieurs types d'interactions (Retweet, Mention, Similarité d'URL, etc.) en plus d'attributs textuels et numériques. Un des critères pour la sélection de voisins pour la génération de données synthétiques est que les attributs des voisins doivent être proches [86]. Une des raisons derrière ce critère est que la combinaison de deux éléments distants pour la génération d'un élément synthétique peut conduire au chevauchement des classes. En effet, plusieurs expérimentations ont montré que SMOTE peut conduire à ce chevauchement et réduire la qualité de classifications [148]. Une seconde raison derrière ce critère est que la combinaison de deux éléments distants peut générer un élément synthétique qui est dissimilaire aux données. Ce type d'élément peut conduire à la génération de modèles complexes.

Ainsi, il est important de sélectionner des voisinages de nœuds qui sont proches vis-à-vis des attributs qui sont déterminants pour la classification. Si des voisins partagent des attributs en commun mais ont des valeurs différentes concernant les attributs déterminants (déterminants pour une classification), la génération de données synthétiques peut conduire au chevauchement des classes.

SMOTE considère que les éléments proches les uns des autres en utilisant la distance euclidienne auront également tendance à être proches vis-à-vis des attributs déterminants [223]. Cette supposition peut être souvent valide quand la dimensionalité est faible, ce qui n'est pas le cas pour les réseaux sociaux. Si on considère que chaque mode d'interaction génère des attributs

structurels et textuels, il est possible que les voisins proches vis-à-vis des attributs déterminants soient des voisins dans un des graphes sociaux (retweet, mention, URLs, etc.). Nous présentons une méthode de sélection de voisins qui permet d'éviter le chevauchement des classes en évaluant la qualité d'un voisin sur les différents graphes qui représentent les modes d'interaction et similarités. En plus des différents modes d'interaction dans un réseau social, nous utilisons la similarité des attributs (similarité cosinus) pour générer un graphe à partir des K plus proches voisins [126].

On considère $Voisins_l : V \rightarrow V^k$ une fonction de voisinage retournant pour chaque nœud des voisins dans un mode d'interaction $l \in L$. Cette fonction peut être simple comme l'identification des nœuds ayant interagi sur le réseau, ou peut comporter l'utilisation de la structure du graphe (ex : eq 5.9). Les voisins d'un nœud n_i dans un mode d'interaction l sont $Voisins_l(n_i)$.

On considère que chaque nœud n_j peut séparer un jeu de données en deux groupes. Le premier groupe est composé des nœuds qui contiennent n_j dans leurs voisinages ($\{n_i \in V_l / n_j \in Voisins_l(n_i)\}$). Le second groupe est composé des nœuds qui ne contiennent pas n_j dans leurs voisinages ($n_i \in V_l / n_j \ \forall \ oisins_l(n_i)$). Il est donc possible de calculer le gain d'information (information gain) relatif à la sélection du nœud n_j dans un mode l. L'équation 5.14 décrit le calcul du gain d'information. Ce dernier est calculé en soustrayant l'entropie (équation 5.16) sachant que n_j est un voisin, de l'entropie (équation 5.15). Dans les équations 5.15 et 5.16, c fait référence aux différentes classes dans une classification. Dans le cadre de la classification binaire, c prend deux valeurs $\{0, 1\}$. Le gain d'information permet ainsi de calculer la capacité d'un voisin à séparer un jeu de données en groupes de telle sorte que les éléments du même groupe aient une forte probabilité d'être dans la même classe. Ainsi, l'utilisation du gain d'information permet d'affecter une mesure de qualité (IG) à chaque voisin.

$$IG(c, n_i) = H(c) - H(c/n_i) \tag{5.14}$$

$$H(c) = -\sum_{c_k \in c} P(c_k) log(P(c_k)) \tag{5.15}$$

$$H(c/n_i) = -\sum_{c_k \in c} P(c_k, n_i) \frac{log(P(c_k, n_i))}{P(n_i)} \tag{5.16}$$

L'utilisation du gain d'information pour sélectionner les voisins privilégie les nœud qui sont souvent dans la liste des voisinages de nœud de la même classe. Aussi, cette mesure privilégie les nœud fréquemment retrouvés dans les listes de voisinages (nœuds centraux). Cette seconde propriété est utile dans le cadre de la classification non équilibrée où il est légitime de privilégier les nœud ayant un nombre élevé d'occurrences pour éviter d'apprendre du bruit.

La méthode de sélection de voisinage est décrite par l'algorithme 19. La première étape est de calculer le gain d'information pour chaque voisin potentiel dans chaque mode. On considère

que le voisinage d'un nœud est composé de ses voisins dans tous les modes d'interaction. La dernière étape est de sélectionner dans le voisinage d'un nœud les K voisins avec les plus hauts gains d'information. Ainsi, un nœud qui est voisin dans plusieurs modes peut être sélectionné plusieurs fois si son gain d'information est élevé dans ces modes. Si ce dernier a un gain d'information élevé dans seulement un mode d'interaction, il sera sélectionné une seule fois. Cela permet d'éviter de dégrader la frontière entre les classes si peu de voisins sont de bonne qualités (en sélectionnant des voisins plusieurs fois). Notez qu'un voisin aura differents scores de gain d'information (IG) dans les differents modes d'interaction (et de similarité comme la similarité du texte publié).

Algorithm 19 Sélection de voisins

k : nombre de voisins
$\mathcal{M} = \{V, E, L\}$: graphe (simple ou multimodale)
c : les différentes classes

Pour chaque mode $l \in L$
    Pour chaque nœud $n_i \in V$:
        Calculer IG_l(c,n_i)
Pour chaque nœud $n_i \in V$
    $Voisinage(n_i) = \bigcup_{l \in L} Voisins(n_i, l)$
Pour chaque nœud $n_i \in V$:
    S_voisins(n_i) = Sélectionner k voisins avec max IG ($Voisinage(n_i)$)
retourner S_voisins

L'algorithme proposé permet ainsi de sélectionner des voisins sur plusieurs modes différents tout en minimisant le chevauchement des classes. Un des inconvénients de cet algorithme est qu'il réduit la diversité des données synthétiques générées, comparé à une méthode qui échantillonne des voisins dans le voisinage des nœuds. Cet inconvénient est fortement réduit dans les réseaux sociaux multimodaux, où les différents modes d'interaction permettent d'identifier un nombre important de voisins. De plus, l'intégration de cette méthode dans un algorithme de Boosting permettra d'ajouter beaucoup de diversité dans les voisinages, étant donné que le gain d'information sera calculé sur l'échantillon de données générées par le Boosting.

5.4.2 SMOTE-SN : Génération de données synthétiques sur les réseaux sociaux

Cette section décrit la génération de données synthétiques (SMOTE-SN) sur des réseaux sociaux avec plusieurs modes d'interaction ainsi que des attributs textuels et numériques. On utilise le modèle de Boyd et Ellison [22] pour représenter un réseau social. Ce dernier décrit des

nœud par des attributs de profil, des messages générés par l'utilisateur et une liste de contacts. Les attributs de profils peuvent être une description écrite par l'utilisateur (attributs textuels), des informations sur sa géolocalisation, les études qu'il a suivies, ses orientations politiques, son âge ou son sexe (attributs catégoriques et numériques). Les messages générés par les utilisateurs contiennent des attributs textuels. Ces derniers permettent également de générer plusieurs graphes sociaux selon le mode d'interaction (Retweet, Reply, Mention, similarité d'URL). Les messages permettent aussi de générer des attributs numériques (nombre de hashtags, nombre de mentions, etc.). Enfin, la liste de contacts d'un utilisateur permet de générer un graphe social. Dans le cadre de la classification, les messages publiés par l'utilisateur contiennent souvent une plus grande richesse que les attributs de profils, surtout dans des réseaux Post-it [180, 181] comme Twitter.

La figure 5.2 décrit la génération de données synthétiques par SMOTE-SN. Afin de générer un nœud synthétique, la première étape est de sélectionner un nœud de départ n_i. Ensuite, un voisin n_j est sélectionné par la méthode introduite dans l'étape précédente (algorithme 19). Nous suivons l'approche de SMOTE [86] où un paramètre d'écart e$\in [0, 1]$ est généré aléatoirement. Concernant les attributs numériques et catégoriques des profils, l'approche utilisée est SMOTE-NC [86]. En ce qui concerne les attributs textuels des profils, nous sélectionnons la description de l'utilisateur ou bien celle de son voisin en fonction de la valeur c générée aléatoirement. Si cette valeur est inférieure à 0.5, nous sélectionnons la description du profil n_i, sinon, c'est la description de son voisin n_j qui est choisie. La liste de contacts du nœud synthétique est échantillonnée aléatoirement des listes des deux nœuds source n_i et n_j en respectant le paramètre d'écart e. Si des mêmes contacts de n_i et n_j sont échantillonnés, ces contacts dupliqués sont remplacés par un nouvel échantillon en utilisant le paramètre d'écart e. Dans le cas où il n'existe pas de remplaçants potentiels (n_i et n_j ont un haut chevauchement de voisins), les dupliqués sont écartés.

On considère que le nombre de messages publiés par n_i est $|me_i|$ et le nombre de messages publiés par n_j est $|me_j|$. Afin de générer la liste de messages me_z pour le nœud synthétique n_z, nous échantillonnons $\lceil e \cdot |me_i| \rceil$ messages de n_i et $\lfloor e \cdot |me_j| \rfloor$ messages de n_j. Le nœud n_z pourra ainsi être ajouté aux différents graphes sociaux (retweet, reply, mention, etc.) à partir des messages me_z. Cela permettra de calculer ensuite les attributs numériques (nombre de retweets, replys, URLs etc.) et structurels (nombre de triades, pagerank, degré, centralité d'intérmediarité, etc.) ainsi que d'agréger les attributs textuels, numériques et catégoriques des voisins du nœud synthétique n_z. Cette méthode permet ainsi de générer des nœuds synthétiques tout en préservant les corrélations entre attributs structurels, textuels et numériques, contrairement à l'utilisation du SMOTE classique qui peut générer des nœuds synthétiques qui s'éloignent de la distribution de leurs classes.

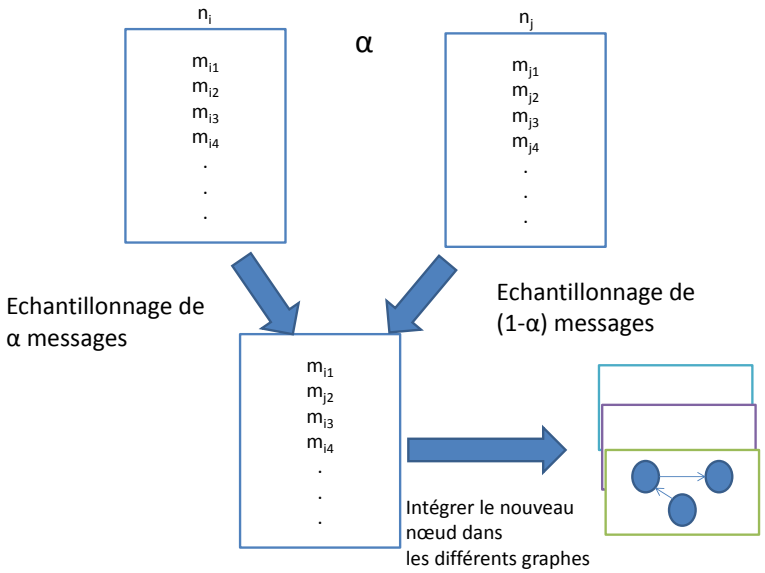

FIGURE 5.2 – Génération de données synthétiques par SMOTE-SN

5.4.3 Algorithme de classification pour classes non équilibrées

Cette section présente l'algorithme de classification proposé qui se base sur AdaBoost [161] (section 2.3.3.1). Malgré le sur-échantillonnage effectué par AdaBoost sur les données mal classées, la classe minoritaire est souvent peu échantillonnée en comparaison de la classe majoritaire [79]. Sun et al. [79] ont proposé des méthodes de sur-échantillonnage qui se basent également sur AdaBoost. Leurs méthodes permettent d'augmenter le nombre d'échantillons de la classe minoritaire en utilisant les poids des classes. D'autres méthodes demandent à l'utilisateur de sélectionner un nombre d'échantillons de la classe minoritaire qui est ajouté au jeu de données généré par AdaBoost [223]. Une autre stratégie consiste à pondérer les poids utilisés par l'algorithme de Boosting de manière à ce que les sommes des poids dans chaque classe soient les mêmes [224]. Cette section n'est pas concernée par la stratégie utilisée pour le sur-échantillonnage de la classe minoritaire. Une des trois stratégies présentées ci-dessus peut être sélectionnée par l'utilisateur. Par défaut, nous utilisons la troisième stratégie présentée où les poids des deux classes sont égaux et le nombre d'échantillons est égale à |V|, le nombre de nœuds dans le graphe.

Après avoir déterminé le nombre d'éléments de la classe minoritaire à utiliser dans une itération de Boosting, les méthodes de Boosting basées sur la génération de données synthétiques doivent sélectionner la proportion d'éléments qui seront modifiés par SMOTE (ou une variante), parmi les éléments de la classe minoritaire. Il est possible de citer deux stratégies différentes pour déterminer cette proportion, qui sont, l'utilisation d'une proportion fixe définie par un

paramètre [163], ou la modification de tous les éléments de la classe minoritaire [223]. Nous proposons de varier le nombre d'éléments modifiés par la génération de données synthétiques d'une itération de Boosting à l'autre en fonction de la difficulté de l'échantillon dans l'itération de Boosting. Ainsi, si l'algorithme peine à séparer les éléments sur-échantillonnés à cause du chevauchement des classes ou de l'éparpillement des données, nous proposons de limiter le nombre de données synthétiques générées. Cela permet à l'algorithme de classer ces éléments difficiles et d'augmenter les chances que le classifieur obtienne un taux d'erreurs inférieur à 50% (une condition pour AdaBoost). Quand l'algorithme de classification arrive à séparer les éléments échantillonnés par le Boosting, la méthode augmente le nombre d'éléments synthétiques générés en se focalisant sur la frontière entre les classes.

La dernière question concerne la sélection des nœuds de départ à utiliser pour générer les nœuds synthétiques. Nous suivons le principe utilisé par RAMOBoost [223] qui est de focaliser la génération de données synthétiques sur la frontière entre les classes. L'algorithme 20 décrit le fonctionnement de RAMOBoost. Ce dernier utilise la distribution W retournée par AdaBoost pour échantillonner des données. Ensuite, une seconde distribution est générée pour échantillonner les éléments positifs à partir de la distribution W. Cette seconde distribution r favorise les éléments ayant beaucoup de voisins négatifs. Le paramètre "a" contrôle le biais de l'algorithme vers le sur-échantillonnage d'éléments de la frontière (les auteurs appellent ce paramètre α, qui peut être confondu avec le paramètre α Boosting). Ensuite, l'algorithme utilise SMOTE sur tous les éléments positifs et exécute AdaBoost pour générer une nouvelle distribution W.

Algorithm 20 RAMOBoost

T : nombre d'itérations

a : paramètre qui contrôle le biais vers la frontière entre les classes.

WL : classifieur faible

W : Distribution initiale pour l'échantillonnage.

1: Pour t $\in$ 1..T
2: Échantillonner D en suivant la distribution W :
3: Diviser D en un jeu avec éléments positifs D_p et un jeu avec éléments négatifs D_n
4: Pour chaque élément n dans D_p :
5: En utilisant les K plus proches voisins de n, calculer $r_n \leftarrow \frac{1}{1+exp(-a \cdot \delta_n)}$ avec δ_n le nombre de voisins de la classe négative de n et a un paramètre utilisateur.
6: Normaliser le vecteur r
7: $D_p \leftarrow$Échantillonner S éléments de la classe positive en utilisant r comme distribution.
8: $D_p \leftarrow$Pour chaque élément positif dans D_p, utiliser SMOTE pour générer un élément synthétique.
9: W,f_t,$\alpha_t \leftarrow$ AdaBoost (W,D,WL)
10: retourner f,α

Dans le cadre des réseaux sociaux multimodaux, un nœud a différents voisins dans plusieurs modes. Un nœud dans la classe positive peut donc avoir beaucoup de voisins négatifs dans un mode d'interaction, justifiant ainsi son sur-échantillonnage, et beaucoup de voisins positifs dans un autre mode justifiant un sous-échantillonnage. Ainsi, il est possible d'obtenir des signaux contradictoires en fonction du mode utilisé. Nous proposons l'algorithme 21 pour la génération d'un modèle de Boosting sur les réseaux sociaux multimodaux qui permet de résoudre ce problème en calculant l'importance d'un mode d'interaction dans une classification. La figure 5.3 décrit la génération de données dans chaque itération de SN-Boost. Dans chaque itération, l'algorithme sépare les nœuds échantillonnés par W (la distribution dans l'étape précédente) en échantillon positif D_p et échantillon négatif D_n. lI sépare ensuite l'échantillon positif en deux échantillons D_p^0 et D_p^{SM} dont la taille est déterminée en fonction de la difficulté du dataset. L'échantillon D_p^0 ne sera pas transformé par SMOTE-SN. L'échantillon D_p^{SM} sera réechantillonné en utilisant une distribution r qui le rapproche de la frontière de la classe négative (en utilisant les différents modes d'interaction) et sera ensuite transformé par SMOTE-SN.

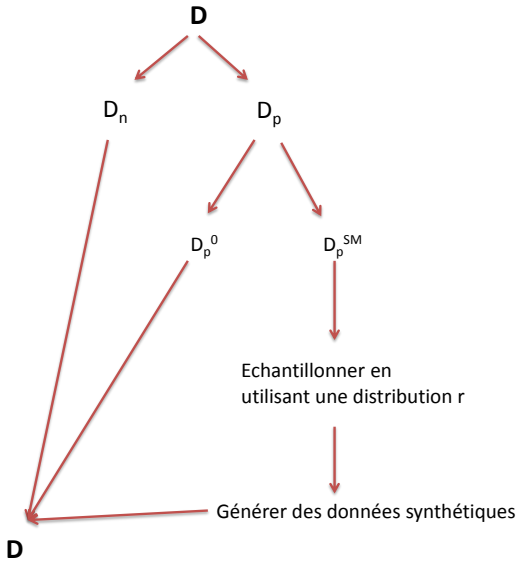

FIGURE 5.3 – Génération de données dans une itération de SN-Boost

Comme RAMOBoost, SN-Boost utilise la distribution W retournée par l'algorithme de Boosting. Pour chaque nœud de la classe positive dans D, un vecteur rm est généré. Ce dernier donne plus de poids aux nœuds qui n'ont aucune interaction, ou qui ont des interactions avec des nœuds de classes différentes. On considère que les nœuds ayant peu d'interactions sont souvent plus difficiles à classifier que les nœuds ayant des interactions avec une classe différente des leurs. Cela est dû à la tendance à interagir avec certains types de nœuds dans la classe opposée, ce qui peut être appris par un algorithme de classification. Un vecteur r est gé-

Algorithm 21 SN-Boost

T : nombre d'itérations.

a : paramètre qui contrôle le biais vers la frontière entre les classes.

WL : classifieur faible.

W : Distribution initiale pour l'échantillonnage.

1: Pour t $\in$ 1..T

2: Échantillonner D en suivant la distribution W

3: Diviser D en un jeu avec éléments positifs D_p et un jeu avec éléments négatifs D_n

4: Pour chaque élément n dans D_p :

5: Pour chaque mode $l \in L$:

6: En utilisant les voisins de n dans le mode l, calculer $rm_n^l \leftarrow \frac{1}{1+exp(-\beta \cdot \delta_n)}$ avec $\beta = a \cdot |Voisins_l(e)|$ et

$$\delta_n = \begin{cases} \frac{|Voisins_l(n) \cap D_p|}{|Voisins_l(n)|} & |Voisins_l(n)| \geq 0 \\ 1 & |Voisins_l(n)| = 0 \end{cases} \tag{5.17}$$

7: Poids(L) $\leftarrow$ [Calcul du poids de chaque mode](rm,L)

8: Générer le vecteur $r_n = \sum_{l \in L} rm_n^l \cdot Poids(l)$

9: Normaliser le vecteur r pour obtenir une distribution

10: Évaluer la difficulté de l'ensemble W pour obtenir le nombre S d'éléments à transformer par SMOTE-SN.

11: séparer D_p en deux sous-ensembles D_p^0 et D_p^{SM} avec $|D_p^{SM}| = S$ et $|D_p^0| = |D_p| - S$.

12: D_p^{SM}=Échantillonner S éléments de la classe positive en utilisant r comme distribution.

13: D_p^{SM} $\leftarrow$Pour chaque élément positif de D_p^{SM}, utiliser SMOTE-SN pour générer un élément synthétique.

14: W,f_t,$\alpha_t \leftarrow$ = AdaBoost (W,D,WL)

15: retourner f,α

néré dans l'étape 8 en agrégeant pour chaque nœud les valeurs de rm pour les différents modes. Cette agrégation est effectuée en calculant un poids pour chaque mode ($Poids(l)$).

Le poids d'un mode quantifie son importance dans une classification. L'objectif de cette quantification est de favoriser le sur-échantillonnage de nœuds ayant peu d'interactions ou beaucoup d'interactions inter-classes dans des modes déterminants dans une classification (modes non bruités). Le calcul du poids d'un mode est effectué par une adaptation du rapport de vraisemblance [136]. L'équation 5.18 décrit le calcul de ce poids, où E_l est le nombre d'interactions dans le mode l, E_l^{intra} est le nombre d'interactions entre nœuds positifs dans le mode l, et E représente le nombre totale d'interactions. Ce poids donne de l'importance aux modes ayant un plus haut nombre d'interactions intra-classe. Aussi, le nombre total d'interactions dans l'exposant permet d'augmenter le poids des modes ayant beaucoup d'observations, ce qui est légitime dans le cadre de classifications non équilibrées, où l'apprentissage de bruit peut fortement dégrader une classification.

$$Poids(l) = \big(\frac{|E_l^{intra}|}{E_l}\big)^{(1-\frac{|E_l}{E})} \tag{5.18}$$

L'objectif de l'étape 10 est de déterminer un nombre S d'éléments positifs à varier par SMOTE-SN. Le nombre total d'éléments positifs à varier peut suivre une des stratégies présentées dans [163, 79, 224, 223]. Nous corrélons la proportion d'éléments positifs variés par SMOTE-SN à la capacité de SN-Boost à prédire la classe des éléments dans D. Pour cela, on génère un modèle en utilisant l'ensemble D, et on prédit les labels de tous les éléments dans V. L'étape suivante est de calculer le taux d'erreurs (équation) 5.19. Dans cette équation, I est la fonction identité qui retourne 0 si des éléments sont similaires, et 1 s'ils sont dissimilaires. La fonction f retourne la classe prédite par le modèle et c_i est la classe d'un nœud.

Ce taux est utilisé pour déterminer le nombre d'échantillons à varier par SMOTE-SN (équation 5.20). Ainsi, quand l'erreur s'approche de 1 indiquant que l'algorithme obtient un taux de biens classés (pondéré) proche de 50%, on rapproche le nombre d'éléments à échantillonner en suivant la nouvelle distribution r de 0 afin de permettre à l'algorithme de séparer les éléments de D. Au fur et à mesure que les performances de l'algorithme augmentent, le taux d'éléments positifs échantillonnés en suivant r augmente également afin de se focaliser sur la frontière entre les classes. Les étapes suivantes de l'algorithme sont similaires à celles de RAMOBoost [223] (algorithme 20).

$$err = \frac{\sum_{n_i \in V} I(f(n_i), c_i) \cdot W_i}{\sum W} \tag{5.19}$$

$$S = max(\sqrt{err} - 1, 0) \tag{5.20}$$

Les figures 5.4, 5.5 et 5.6 illustrent les différentes stratégies de génération de données synthétiques pour SMOTEBoost [163], RAMOBoost [223] et SN-Boost quand seulement la similarité des attributs est considérée. SMOTEBoost échantillonne les éléments positifs aléatoirement,

et génère les éléments synthétiques à partir des K plus proches voisins. La figure 5.4 montre que plusieurs éléments synthétiques ont peu de contributions à la génération du modèle qui sépare les classes et augmente la taille du jeu de données. La stratégie de RAMOBoost est de générer des données synthétiques près de la frontière entre les classes. Cela permet de générer des éléments synthétiques qui repoussent la frontière de la classe positive, et affecte le modèle de classification. Il est possible de voir que plusieurs éléments synthétiques peuvent dégrader une classification. Le premier groupe d'éléments synthétiques encerclés dans la figure est généré par des voisins qui peuvent être considérés comme du bruit. Le deuxième groupe d'éléments synthétiques également encerclés est généré à partir de voisins qui sont dans la frontière entre les deux classes. Ces derniers sont entourés de plusieurs nœuds de la classe négative et conduisent au chevauchement des classes. La figure 5.6 illustre la stratégie utilisée par SN-Boost. Ce dernier génère des éléments synthétiques proches de la frontière des classes en utilisant une stratégie similaire à celle de RAMOBoost. La différence entre les deux algorithmes est que SN-Boost sélectionne des voisins proches des nœuds sur-échantillonnés par l'algorithme de Boosting (proches de la frontière), mais privilégie ceux qui ont moins de voisins de la classe négative (en utilisant le gain d'information). Ainsi, SN-Boost évite de sélectionner des nœuds pouvant être considérés comme du bruit en tant que voisins pour la génération de données synthétiques. Il effectue cela en utilisant le gain d'information comme critère pour la sélection des voisins. Ce gain d'information est calculé en utilisant la distribution W de données échantillonnés, et permet ainsi de maintenir de la diversité dans la génération de données synthétiques d'une itération à l'autre.

5.5 CollectiveBoost : Méthode basée sur le Boosting pour la classification collective sur les RSN

Les réseaux sociaux numériques sont souvent caractérisés par des attributs de profils (textuels et numériques) ainsi que des interactions entre utilisateurs générées à partir des messages échangés. Ces interactions sont souvent elles-mêmes décrites par des attributs textuels et numériques (nombre d'URL dans l'interaction, nombre de mentions, etc.). Alors qu'il existe plusieurs travaux sur la classification dans les graphes avec attributs des nœuds, la problématique de classification collective avec arêtes attribuées est sous-étudiée. Le développement de méthodes qui traitent ce problème est cependant très important, étant donné que la majorité des interactions sur les réseaux sociaux numériques peuvent être décrites par des attributs textuels ou numériques.

Les messages échangés sur les réseaux sociaux numériques peuvent être caractérisés par des thèmes. Ces derniers définissent des sujets de conversation abordés par les utilisateurs sur

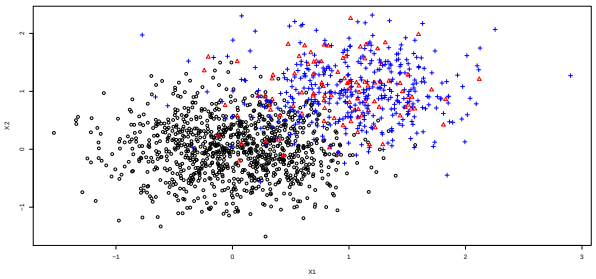

FIGURE 5.4 – Génération de données synthétiques pour SMOTEBoost

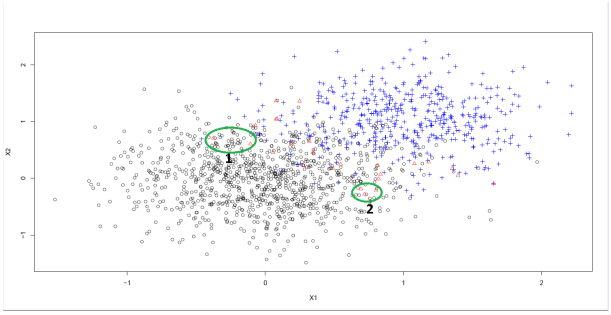

FIGURE 5.5 – Génération de données synthétiques pour RAMOBoost

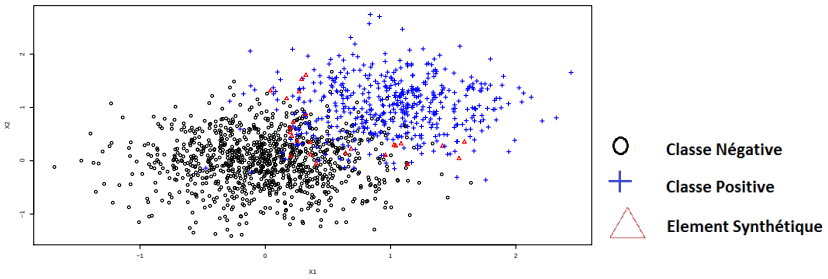

FIGURE 5.6 – Génération de données synthétiques pour SN-Boost

ces réseaux. En fonction de la combinaison d'attributs (existence de mots, combinaisons de mots), il est possible d'identifier un des thèmes abordé sur ces réseaux. L'identification des thèmes sur les réseaux sociaux peut augmenter la qualité d'une classification, surtout lorsque des algorithmes de classification utilisent la structure du graphe social. L'algorithme que nous proposons identifie ces derniers afin de construire plusieurs graphes sociaux thématiques. Cela permet de calculer ensuite des attributs structurels sur ces graphes, qui peuvent être déterminants dans plusieurs tâches de classification.

La classification de profils d'utilisateurs sur les réseaux sociaux est affectée à la fois par la capacité à catégoriser le texte échangé et par les graphes sociaux. Ces derniers sont souvent utilisés pour agréger les attributs textuels dans les voisinages des profils. Ainsi, la catégorisation d'attributs textuels est centrale, surtout dans des réseaux clairsemés comme Twitter (réseau Post-it). La classification de texte est affectée par plusieurs problématiques comme la polysémie, la synonymie et autres [**?**]. La détection des thèmes des messages peut répondre au problème de polysémie étant donné qu'un attribut textuel est défini par le binôme (mot, thème). Le problème de la synonymie peut être résolu pour chaque thème par des méthodes classiques de réduction de dimensionalité [222]. L'avantage d'effectuer une réduction de dimensionalité après la détection de thèmes est que cette dernière est effectuée après la résolution du problème de polysémie, éliminant les synonymes d'un mot après avoir projeté les différents sens que ce mot peut porter dans les différents thèmes.

La classification sur les réseaux sociaux peut également être affectée par le manque de données labélisées dans la classe minoritaire. C'est le cas dans plusieurs domaines comme le marketing, la détection d'apologistes au jihad ou de spammeurs. Pour répondre à ce problème, nous proposons une version semi-supervisée de l'algorithme CollectiveBoost qui effectue une détection thématique sur les données labellisées ainsi que les données à classifier. L'utilisation des données à classifier permet de compenser le manque de données labellisées. En effet, les attributs textuels dans les données à classifier permettent d'améliorer la qualité de la détection de thèmes effectuée sur le corpus. Cet algorithme génère des données synthétiques en utilisant les messages d'utilisateurs difficiles à classifier dans chacune de ces itérations afin d'orienter la détection de thématiques. Cela permet de fragmenter certaines thématiques qui sont larges, en plusieurs sous-thèmes et d'affiner le modèle de classification en prenant en compte les éléments difficiles. Aussi, l'algorithme proposé utilise une méthode basée sur le Boosting qui prend en compte le poids des classes. Cela permet aux éléments appartenant à la classe minoritaire d'être sur-échantillonnés, et ainsi représentés par des thèmes.

Cette section présente l'algorithme CollectiveBoost qui oriente le modèle de classification (Boosting) ainsi que la modélisation des données (détection thématique) par le sur-échantillonage et la génération de donnés synthétiques. Cet algorithme de classification collective utilise l'architecture de l'algorithme de classification itérative (ICA) [15]. La sous-section suivante intro-

duit la méthodologie. Enfin, des attributs pouvant être calculés sur des graphes thématiques sont introduits dans la dernière section.

5.5.1 Méthodologie

La méthode proposée se base sur une architecture de Boosting avec une détection de thématiques dans chacune de ses itérations. Cette détection de thématiques permet de générer un ensemble de graphes ayant différents effets sur la classification. Dans chaque itération de Boosting, l'algorithme effectue F échantillonnages de graphes qui ont été générés à partir de la détection thématique. CollectiveBoost construit ensuite un modèle à partir de chaque échantillon de graphes. Les F modèles générés effectuent un vote par majorité pour déterminer la classe d'un élément. Ce processus est ensuite répété dans chaque itération de Boosting. Ainsi, chaque vote par majorité effectué par des échantillons de graphes est pondéré par un paramètre $alpha_i$ obtenu à partir de la méthode de Boosting utilisée.

Avant d'expliquer en détail la méthode proposée, il est important de décrire certaines des caractéristiques souhaitées :

- La méthode doit pouvoir utiliser des attributs calculés à partir de la structure du graphe. Alors que plusieurs méthodes se contentent d'utiliser des attributs des voisins (attributs locaux), d'autres méthodes ont pu améliorer leurs résultats en considérant des attributs calculés à partir de voisinages ayant un sens sémantique différent [73, 171, 168, 177, 11]. Ces voisinages sont souvent identifiés à partir de la structure du graphe (voisins des voisins, marches aléatoires, co-citations, quasi-cliques, etc.). L'identification de ces voisinages se complique lorsque les interactions sont décrites par des attributs qui caractérisent un thème. La majorité des méthodes de classification ignorent ces attributs lors du calcul des voisinages, considérant ainsi que toutes les arêtes ont un même sens sémantique. Cela peut conduire à la perte d'informations utiles à la classification.
- La méthode doit pouvoir propager les labels des voisins. En effet, plusieurs méthodes de classification collective considèrent que les labels affectés aux voisins peuvent être utiles pour identifier le label d'un nœud [15, 170, 165, 164, 171]. Une des méthodes les plus utilisées pour propager les labels est l'algorithme de classification itératif [15]. Cette méthode agrège les labels des voisins ainsi que les attributs des voisins en utilisant des fonctions d'agrégation telles que : moyenne, somme, min, max, existe. Cette méthode, largement étudiée, permet d'obtenir de bons résultats sur plusieurs types de graphes. Nous nous baserons sur une de ces variantes (ICA-SN) présentée dans la section 5.3.
- La méthode doit échantillonner plusieurs sous-graphes afin de réduire le biais des données d'apprentissage. Les travaux de [178] ont prouvé par induction qu'une méthode basée sur le bagging et avec échantillonnage d'arêtes permet d'améliorer les résultats de classification collective. L'échantillonnage d'arêtes permet de réduire l'effet de certaines

interactions qui sont le résultat de l'aléa ou sont caractéristiques du jeu d'entraînement. Nous introduisons dans cette section une méthode réduisant le biais basée sur l'échantillonnage de sous-graphes. Cela permet de réduire le biais en apprenant des modèles de sous-ensembles des graphes sociaux, tout en préservant la connectivité de graphes souvent peu denses.

– La méthode proposée doit permettre de calculer des attributs structurels sur des graphes avec arêtes attribuées. Il existe certaines méthodes dans la littérature qui traitent ce problème [225, 226, 227], mais ces dernières ne sont souvent pas adaptées aux graphes avec attributs textuels qui ont une haute dimension. CollectiveBoost crée plusieurs graphes thématiques permettant de calculer des attributs structurels sur chacun d'entre eux. Un graphe thématique est un sous-graphe du graphe social composé uniquement d'interaction avec un contenu appartenant à un thème. Les thèmes sur les réseaux sociaux sont générés à partir d'algorithmes de classification non supervisées, mais ces derniers doivent prendre en compte certaines spécificités des réseaux, comme le faible contenu textuel des messages.

5.5.1.1 Architecture globale

CollectiveBoost utilise l'algorithme de classification itérative pour les réseaux sociaux (ICA-SN) qui est présenté dans la section 5.3. Il fait appel à ICA-SN dans l'étape de prédiction des labels.

Le modèle f utilisé dans le framework de ICA-SN est généré par un algorithme basé sur le Boosting. L'architecture du modèle f est illustrée par la figure 5.7. Ce dernier est composé de plusieurs modèles empilés verticalement et horizontalement. Les modèles empilés horizontalement $f_{x,}$ sont générés lors d'une même itération de Boosting. Ces modèles sont organisés dans une architecture de Bagging où chacun est entraîné sur des sous-graphes différents. Un vote majoritaire est effectué par chaque pile $f_{x,}$. Un deuxième vote, cette fois pondéré par un paramètre α, est ensuite effectué pour obtenir la classe finale de chaque élément. Le paramètre α est obtenu par un algorithme de Boosting présenté dans [161], ou un algorithme de Boosting pour données non équilibrées présenté dans la section 5.4 (SN-Boost). L'échantillonnage de graphes permet d'éviter à l'algorithme d'apprendre du bruit, en créant un comité de modèles sur des sous-ensembles d'arêtes. Le Boosting permet à la fois de sur-échantillonner les éléments difficiles à classer pour réduire la variance du modèle finale, et d'orienter la détection thématique pour faciliter la classification d'éléments difficiles. L'utilisation de graphes thématiques permet de maintenir une haute connectivité dans les graphes échantillonnés (contrairement à l'échantillonnage d'arêtes [178]), de calculer des attributs thématiques qui facilitent la classification, et de résoudre le problème de la polysémie (un terme avec plusieurs sens).

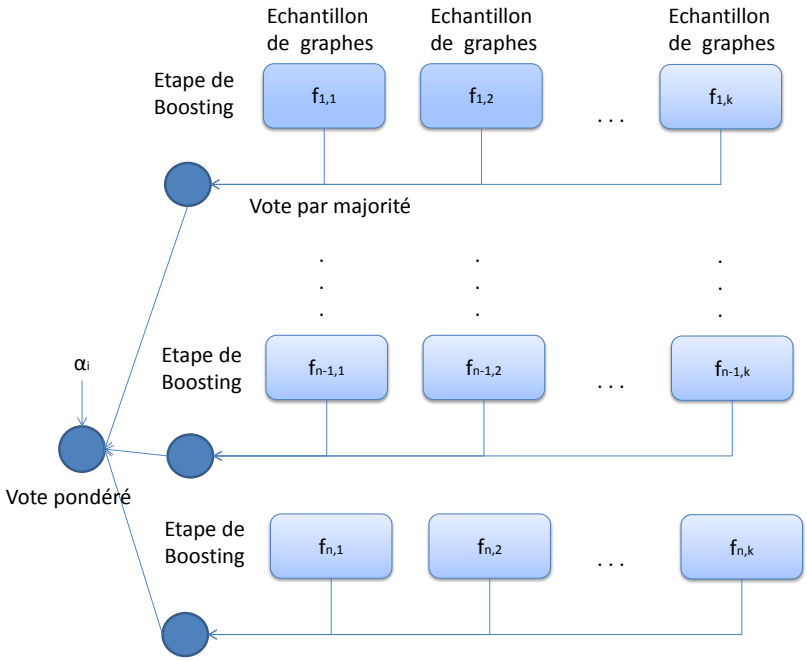

FIGURE 5.7 – Architecture de CollectiveBoost

5.5.1.2 Génération des modèles de classification

La génération des modèles de classification est détaillée par l'algorithme 22. L'algorithme proposé se différencie du Boosting classique par un clustering de Tweets qui permet de définir des classes sémantiques différentes (thématiques), ainsi que par l'échantillonnage de graphes pour les modèles empilés horizontalement. L'empilement horizontal des modèles permet à l'algorithme d'apprendre sur plusieurs sous-graphes.

Le générateur du modèle CollectiveBoost prend en entrée plusieurs paramètres incluant les paramètres K, T et C. Le premier paramètre fait référence au nombre d'échantillons de graphes à effectuer durant chaque étape de Boosting. Le second paramètre définit le nombre d'itérations dans l'algorithme de Boosting. Le troisième paramètre fait référence au cout d'erreurs de classification par classe.

Les premières étapes de l'algorithme consistent à d'initialiser les vecteurs W et W_{theme}. Ces deux vecteurs sont utilisés respectivement pour déterminer la probabilité d'échantillonner un élément comme donnée d'entraînement et la probabilité d'échantillonner un élément pour le clustering thématique. L'initialisation de les probabilités d'échantillonnage W est similaire à celle effectuée dans AdaBoost (1/nombre d'éléments). CollectiveBoost effectue ensuite T itérations générant des modèles à partir des éléments échantillonnés dans chacune. **Échantillonnage de nœuds** : Dans cette étape, CollectiveBoost utilise la distribution W afin de générer les données d'entrainement (D). Il effectue cela dans chaque itération.

Algorithm 22 Générateur du modèle

K : nombre de graphes empilés horizontalement

T : nombre d'itération

$C_i \in (0 + 1]$: Couts d'erreurs de classification par classe

Params : paramètres pour l'algorithme de Boosting sélectionné

ABoost : algorithme de Boosting séléctionné (ex. Adaboost ou SN-Boost)

$\mathcal{M}$: graphe simple ou multimodal

Messages : messages publiés

NT : nombre de thèmes pour le LDA

p : nombre de graphes par échantillon

Y :labels des nœuds

Initialise W(i)$\leftarrow \frac{1}{|N|}$

Initialise $W_{theme}(i) \leftarrow \frac{C_i}{\sum C_i}$

Pour t $\in$ 1 .. T

    D$\leftarrow$ [Échantillonnage de nœuds] (W,$\mathcal{M}$)

    D_{theme} $\leftarrow$[Échantillonnage et génération de données synthétiques par SMOTE-SN] (W,$\mathcal{M}$) \\section 5.4.2

    $Clust_t \leftarrow$ Clustering_thématique(D_{theme},Messages,NT)

    $g_t \leftarrow$ Génération_graphes (D,$Clust_t$)

    Pour k $\in$ 1..K

        $G_t^k \leftarrow$ [Échantillonnage pondéré de graphes] (G_t,P)

        $f_{t,k}, err_k \leftarrow$ [Génération du modèle de classification] (g_t^k,D)

    $W_{theme}^{t+1}(i) = \frac{C_i \cdot \sqrt{1-f_{t,}(x_i) y_i \cdot moyenne(err)}}{\mathcal{Z}_t}$// Mise à jour des poids, $\mathcal{Z}_t$ permet de normaliser W_{theme} pour obtenir une distribution ($\sum_i W_{theme}^{t+1}(i) = 1$)

    W,α_t=[itération d'ABoost](Params,D,$f_{t,}$)//Échantillonnage des données d'entraînement

retourner f,α

Échantillonnage et génération de données synthétiques : L'algorithme commence d'abord par échantillonner N éléments en suivant la distribution W_{theme} (N est le nombre d'éléments dans le jeu de données). La génération de données synthétiques par SMOTE-SN est effectuée sur chaque élément sur-échantillonné de la classe minoritaire. Ces éléments synthétiques seront utilisés dans le clustering thématique. Ils améliorent l'affectation des données de test aux clusters thématiques détectés.

Clustering_thématique : Cette étape permet de générer des classes sémantiques (thèmes) à partir des messages publiés par les utilisateurs. Les interactions construites à partir de ces messages peuvent ensuite être affectées à des graphes définis par un des thèmes détectés. L'algorithme peut prendre en compte différents modes d'interaction (@mention, RT retweet, partage de la même URL, #partage du même Hashtag, etc.). Dans le cas où plusieurs modes d'interaction sont présents, un graphe est défini par le binôme (mode d'interaction, thème).

Les messages sur les réseaux sociaux sont souvent peu riches en texte et ne permettent pas de détecter des thèmes qui reflètent les sujets de discussion sur ces réseaux. En effet, la détection de clusters thématiques à partir des messages échangés génère souvent des thèmes construits à partir de mots fréquents qui ont peu de sens [200].

Pour cela, il est important d'utiliser les informations de la source du message (l'utilisateur qui publie) pour générer des classes sémantiques valides. Nous utilisons une méthode nommée Twitter-LDA [200] expliquée par l'algorithme 23. L'idée étant qu'un utilisateur publie des messages en suivant une distribution de thématiques. En effet, un utilisateur va souvent discuter d'un sous-ensemble restreint de thématiques sur un réseau social. Twitter-LDA détecte les probabilités qu'un utilisateur discute une thématique à partir d'une agrégation de tous les messages publiés par ce dernier. Ainsi, l'ensemble des messages publiés par un seul utilisateur sont agrégés dans un document, et ces derniers sont utilisés pour 1) détecter les thématiques par le LDA [124], puis pour 2) affecter des probabilités qu'un utilisateur discute des thématiques détectées. Cela permet de détecter des thématiques à partir des agrégations des messages d'utilisateurs qui sont beaucoup plus riches en texte que les messages. La distribution des thèmes d'un utilisateur est ensuite utilisée pour déterminer les thèmes des messages publiés par ce dernier. Cette dernière étape est effectuée en multipliant la probabilité qu'un message appartienne à un thème par la probabilité que l'utilisateur ayant publié ce message discute de ce thème (eq 5.21). Ensuite, le message est affecté au thème avec la plus grande probabilité (eq 5.22).

$$We(Th_i, M) = Pr(Th_i/M) \cdot Pr(Th_i/Parent) \tag{5.21}$$

$$Theme(M) = argmaxWe(, M) \tag{5.22}$$

Étape Générations_graphes : Cette étape utilise les thématiques identifiées dans la phase précédente pour générer une série de sous-graphes. Chaque sous-graphe est généré à partir des

Algorithm 23 Twitter-LDA

NT : nombre de thèmes pour le LDA (peut être obtenu automatiquement)
D_{themes} : utilisateurs échantillonnés pour le clustering thématique
Messages : messages publiés

Docs← agréger les messages d'utilisateurs (messages,D_{themes})
Th ← LDA (D,NT)//génération de thèmes
Pour d dans Docs :
    $Pr(Th_i/d)$ ← [affecter un mixture de thème](Th,d)
Pour m dans Messages :
    $Clust(M)$ ← [Affecter 1 thème par message] (m,Th,Pr)
retourner $Clust$

messages assignés à une thématique.

Échantillonnage pondéré de graphes : Cette étape concerne l'échantillonnage de graphes construits à partir des thématiques (algorithme 24). Pour chaque modèle construit dans une itération de l'algorithme de Boosting, un sous-ensemble g_1 composé de P graphes est échantillonné aléatoirement (50% du nombre de thèmes par défaut). Ensuite, on échantillonne de nouveau un sous-ensemble g^2 composé de P graphes en utilisant cette fois un échantillonnage pondéré. Le premier ensemble de graphes échantillonnés g^1 permettra de calculer des attributs intra-thème et les échantillons de graphes g^1 et g^2 permettront de calculer des attributs inter-thèmes. La pondération dans le deuxième échantillonnage est obtenue en calculant la proximité de Jaccard entre l'ensemble des arêtes de g_1 et ceux des graphes non échantillonnés $Sim_{g_a,g_b} = \frac{|E_{g_a} \cap E_{g_b}|}{|E_{g_a} \cup E_{g_b}|}$ (E_{g_i} = les arêtes du nœud n_i).

Algorithm 24 Échantillonnage pondéré de graphes

P : nombre de graphes échantillonnés G : graphes thématiques

$g^r \leftarrow \{\}$//Graphes à retourner
$g^1 \leftarrow$ Échantillonnage aléatoire de P graphes (G)
$g^2 \leftarrow$ Échantillonnage aléatoire de P graphes (G)
pour $g_a \in g^1$
    $g^r \leftarrow g^r \cup g_a$
    pour $g_b \in g^2 \setminus \{g_a\}$
        \\Calcul de la proximité de Jaccard entre les ensembles d'arêtes
        $Prox_{g_a,g_b} = \frac{|E_{g_a} \cap E_{g_b}|}{|E_{g_a} \cup E_{g_b}|}$.
        $g^r \leftarrow g^r \cup$ [Échantillonnage de 1 graphe pondéré par la proximité de Jaccard](g^2,$Prox$)
retourner g^r

Génération du modèle de classification : Cette étape concerne la génération des modèles

pour chaque échantillon de graphes, pour chaque itération de l'algorithme de Boosting. Cette étape génère le modèle et calcul l'erreur (err), qui est le nombre d'éléments bien classés (pondéré par le coût C) sur le nombre total d'éléments (pondéré par le coût). L'équation 5.23 décrit le calcul de l'erreur, avec I, la fonction identité qui retourne soit 1 si deux éléments sont identiques, ou 0 dans le cas contraire. L'erreur err est utilisée dans l'étape suivante pour pondérer l'échantillonnage des éléments servant à générer de nouveaux thèmes.

$$err = \frac{\sum C_i \cdot I(h(i), y_i)}{\sum C_i} \tag{5.23}$$

La première étape de la génération du modèle de classification est de construire un vecteur à partir des attributs de profil d'utilisateurs. Ensuite, on enrichit ce vecteur par les attributs de voisins. Une méthode de sélection de voisinage est utilisée pour déterminer les voisins de chaque nœud. Plusieurs stratégies peuvent être utilisées dans cette phase. Nous en citons quelques-unes :

1. Utilisation des arêtes pour sélectionner les voisins. Cette stratégie considère que deux nœud sont voisins s'ils sont connectés par une arête. Elle est la plus utilisée des méthodes de classification collective.
2. Utilisation de la structure du graphe pour déterminer les voisins. Ces méthodes considèrent qu'il existe des relations sémantiques autres que l'interaction qui peut être utilisée pour la classification [177, 168, 11].
3. Méthodes basées sur la prédiction de liens : Ces méthodes enrichissent les voisinages de graphes en prédisant des liens entre les nœud [174, 173, 172].

Nous utilisons la méthode introduite dans la section 5.3, qui appartient à la deuxième famille de méthodes. Cette méthode effectue des marches aléatoires et calcule une mesure de proximité entre les utilisateurs à partir de ces marches. Nous enrichissons les modèles par des attributs calculés à partir de la structure des graphes ainsi que par des attributs structurels calculés à partir de l'intersection de plusieurs graphes (attributs inter-thèmes).

Mise à jour des poids pour le clustering thématique : Souvent, la classe avec classes non équilibrées a tellement peu d'observations qu'il est difficile de détecter les thèmes de cette dernière. Il est ainsi important de sur-échantillonner la classe minoritaire afin de faire ressortir les thèmes discutés dans cette dernière. Notez que les éléments sur-échantillonnés de la classe minoritaire seront variés par SMOTE-SN afin que chaque élément soit unique. Nous utilisons une équation qui sur-échantillonne les éléments mal classés en suivant un facteur pondéré par la racine carré au lieu de l'exponentielle utilisée par le Boosting. Sur-échantillonner les éléments mal classés pour le clustering thématique permet de séparer certains clusters en deux et d'en agréger d'autres dans chaque étape de Boosting. Les messages des éléments mal classés sont sur-échantillonnés et auront une plus grande influence sur la génération des thèmes. Ainsi,

CollectiveBoost oriente le clustering vers une meilleure représentation des éléments difficiles à classer, facilitant ainsi la génération d'un modèle par l'algorithme de Boosting. Le second avantage de la variation des thèmes d'une étape de Boosting à l'autre est l'augmentation de la diversité des modèles générés par le Boosting.

Nous utilisons l'erreur err, calculée dans l'étape précédente pour déterminer le taux de sur-échantillonnage. Quand l'erreur est élevée, le taux de sur-échantillonnage des éléments mal classés baisse, et quand l'erreur est faible, ce taux augmente. Nous suivons ainsi la même logique que le sur-échantillonnage effectué par le Boosting, mais en utilisant un fonction beaucoup plus lisse. Cela permet de détecter les thèmes discutés dans le réseau (grâce à l'utilisation d'une fonction lisse), et de ne pas doublement pénaliser les éléments mal-classés (par le sous-échantillonnage effectué par le Boosting et par le sous-échantillonnage dans la détection thématique). Le sur-échantillonnage effectué permet de donner un avantage à la détection de thèmes des éléments mal-classés, en poussant leurs frontières par l'utilisation du SMOTE-SN. Cependant, quand le taux d'erreur est élevé, nous réduisons cet avantage afin de classifier correctement tous les éléments mal-classés, comme dans le Boosting.

5.5.2 Attributs structurels sur des réseaux multimodaux

Plusieurs méthodes de classification calculent des attributs à partir de la structure des graphes [227, 63, 64]. Certains de ces attributs sont le nombre de cycles de taille K passant par un nœud, le degré, les composantes principales, le degré pondéré par le poids des arêtes, le nombre de voisins, etc. Ces attributs sont utiles pour l'apprentissage de modèles de classification. Des attributs structurels composites proposés par [64] permettent des détecter des anomalies dans les réseaux sociaux. Ces derniers sont :

- Densité Egonet : Cet attribut est composé du nombre d'arêtes incidentes à un nœud et du nombre de voisins de ce dernier. Akoglu et al. ont découvert que ces deux attributs obéissent à une loi de puissance dans plusieurs graphes sociaux. Cet attribut permet de détecter des quasi-cliques et quasi-étoiles.
- Poids Egonet : Cet attribut est composé du poids des arêtes incidentes et du nombre d'interactions. Ce dernier détecte des interactions récurrentes entre des paires de nœuds.
- Paire Dominante : La valeur propre de la première composante principale de la matrice d'adjacence pondérée et le poids des arêtes incidentes à un nœud. Cet attribut permet de détecter une haute affinité entre un nombre restreint de nœuds.

Dans le cadre d'un graphe avec arêtes contenant des attributs à hautes dimensions, il n'existe pas de méthodes simples pour calculer les attributs structurels des graphes. L'approche la plus souvent utilisée consiste à ignorer les attributs textuels des graphes causant une perte d'information utile pour la classification. CollectiveBoost détecte les thèmes des messages échangés sur un réseau social. Cela permet de créer un graphe thématique où toutes les arêtes du graphe sont

générées à partir de messages partageant les mêmes thèmes. Des attributs structurels peuvent ensuite être calculés sur ces graphes thématiques.

Les attributs structurels sur les graphes thématiques peuvent être plus déterminants pour une classification que ces mêmes attributs sur un graphe où les attributs des arêtes sont ignorés. En effet, les nœuds de différentes classes ont tendance à discuter de thèmes avec des fréquences différentes ou à avoir un comportement relationnel différent en fonction des thèmes discutés. Il est également possible de calculer des attributs inter-thèmes. Ces attributs peuvent refléter le comportement d'utilisateurs sur plusieurs thèmes. Contrairement à des attributs calculés sur un seul graphe thématique, les attributs inter-thèmes peuvent indiquer une relation plus forte entre utilisateurs. On considère les graphes thématiques G=$\{g_1, g_2, ..., g_l\}$. Nous présentons ci-dessous des attributs inter-thèmes construits à partir de ces graphes.

- Nombre de voisins inter-thèmes : Le nombre d'arêtes incidentes à un nœud présentes dans un sous-ensemble de graphes de l'ensemble G. Ce degré reflète le nombre de voisins avec lesquels un utilisateur a discuté tous les thèmes du sous-ensemble de G. Il reflète le nombre de voisins discutant un grand nombre de thèmes avec le nœud, et indique ainsi qu'un nœud a des relations fortes sur le graphe.
- Degré inter-thèmes : nombre d'interactions entre un nœud et ses voisins distincts inter-thèmes.
- L'inverse de la valeur propre de la composante principale de la matrice d'adjacence : Cet attribut est obtenu en multipliant, un à un, l'inverse de la valeur propre de la composante principale obtenue dans un sous-ensemble de graphes dans G. Un haut score signifie que le nœud reflète un comportement peu fréquent sur un sous-ensemble de graphes thématiques.
- Le poids des interactions d'un nœud : Cet attribut est obtenu en multipliant les poids des interactions d'un nœud sur un sous-ensemble de graphes dans G. Cela permet de détecter les nœuds ayant un comportement fréquent sur tous les thèmes reflétés dans le sous-ensemble de graphes.
- Densité Egonet : est calculé à partir du degré et du nombre de voisins distincts inter-thèmes. Permet de détecter des δ-cliques et des étoiles construites à partir d'interactions entre utilisateurs incluant un sous-ensemble de thèmes dans G.
- Poids Egonet : permet de quantifier la relation entre le poids des interactions et le degré inter-thème dans G. Cet attribut permet de détecter des interactions récurrentes qui discutent toutes les thématiques dans G entre une paire de nœuds.
- Paire Dominante : Similaire à la version uni-modale, mais permet de détecter s'il existe des paires avec une haute affinité sur tous les thèmes de G.

Notons qu'il est préférable de calculer des attributs inter-thèmes sur un nombre restreint de thèmes (2 ou 3) étant donné la faible densité qui caractérise souvent les réseaux sociaux.

5.6 Expérimentations

Dans cette section, nous allons présenter des expérimentations effectuées sur quatre benchmarks de Twitter.

Les attributs utilisés pour la classification sont construits à partir du texte de tous les messages qu'un utilisateur a publié, ainsi de son voisinage (texte publié par les voisins, attributs structurels (section 5.5.2)). Les algorithmes testés sont les suivants :

- CollectiveBoost (3,6 et 9 classes) : L'algorithme proposé avec différents nombres de classes thématiques.
- SN-Boost : L'algorithme proposé dans la section 3.
- ADAC2 [79] : Un algorithme de Boosting pour classes non équilibrées. Ce dernier utilise le poids des classes pour adapter les poids des échantillonnements retournés par le Boosting.
- Échantillonnement d'arêtes (EdgeSample) [178] : Cet algorithme de Bagging crée plusieurs modèles en échantillonnant un pourcentage des arêtes du graphe social.
- Algorithme de classification itérative (ICA) [15] : Cet algorithme souvent utilisé pour la classification collective utilise les attributs ainsi que les labels des voisins d'un nœud pour identifier son label. Il propage itérativement les labels en utilisant les voisinages locaux des nœuds.
- Algorithme de classification itérative pour les réseaux sociaux (ICA-SN) [11] : Cet algorithme basé sur l'ICA varie les voisinages des nœuds en utilisant la structure du graphe pour éviter la propagation d'erreurs.
- Sur/Sous-échantillonnement : Cette méthode sous-échantillonne la classe majoritaire et sur-échantillonne la classe minoritaire. Cette méthode échantillonne les nœud utilisés pour l'apprentissage avec des poids permettant d'équilibrer les deux classes. Le nombre de nœuds échantillonnés est le même que l'ensemble disponible pour l'entraînement. Ainsi, les nœuds de la classe majoritaire sont sous-échantillonnés et les nœuds de la classe minoritaire sont sur-échantillonnés.
- SMOTE-ENN [87] : Génère des données synthétiques en utilisant SMOTE, puis supprime les données qui ont 2 voisins sur 3 voisins qui n'appartiennent pas à leurs classes.
- SMOTE-NE [87] : Génère des données synthétiques en utilisant SMOTE puis remplace les données qui ont 2 voisins sur 3 qui n'appartiennent pas à leurs classes par des éléments de la classe opposée.
- SMOTE-IPF [88] : Génère des données synthétiques en utilisant SMOTE. Cette méthode divise les données générées en K paniers, où chaque panier est ensuite utilisé pour générer un modèle. Les éléments synthétiques qui ne sont pas correctement classifiés par la majorité des modèles sont supprimés.

Nous utilisons une classification qui pondère les coûts des erreurs de classification en fonction de la taille d'une classe pour les algorithmes qui n'utilisent pas d'échantillonnement. Aussi, tous les modèles de classification sont des modèles de régression logistique. Les méthode de Boosting utilise une haute régularisation (C=0.0001) afin que l'algorithme soit un apprenant faible (weak learner).

5.6.1 Détection d'apologistes au jihad

Le premier jeu de données est obtenu de l'association Fifth Tribe. Ce jeu concerne des utilisateurs identifiés comme apologistes au jihad ainsi que des utilisateurs identifiés comme étant des chevaliers blancs (qui luttent contre les apologistes). Les deux classes utilisent souvent les mêmes termes et interagissent également avec les mêmes utilisateurs. Les classes sont non équilibrées, avec 117k chevaliers blancs, et seulement 109 apologistes au jihad. Fifth Tribe a utilisé des comptes d'apologistes connus comme racines pour récupérer les comptes restants. Ainsi, les apologistes présents ne représentent que des utilisateurs dédiés à l'État Islamique, et non les comptes sympathisants qui peuvent être plus difficiles à détecter.

Le tableau 5.5 présente des expérimentations en utilisant la méthode SN-Boost ainsi que CollectiveBoost avec 3, 6 et 9 classes et ICA-SN. Nous avons utilisé un algorithme de régression logistique avec un haut facteur de régularisation (0.0001) comme classifieur dans chaque itération de boosting pour tous les algorithmes. Il faut noter que la régression logistique, qui n'est pas recommandée pour traiter les problèmes de classification avec classes non équilibrées [83], obtient de bons résultats grâce au Boosting. En effet, la régression logistique a tendance à sous-estimer les probabilités conditionnelles de la classe minoritaire [83], mais les variations aléatoires obtenues par le boosting permettent de répondre à ce problème.

Nous pouvons voir que CollectiveBoost obtient les meilleurs scores d'AUC, qu'elle soit utilisée avec 3 ou 6 ou 9 classes sémantiques. La méthode à 9 classes sémantiques obtient de meilleurs résultats, comparé à CollectiveBoost à 6 ou 9 classes sémantiques. Il est également possible de remarquer que SmoteBoost, avec le modèle Smote proposé (SMOTE-SN), obtient de meilleurs résultats que SmoteBoost-RegSmote. Cela est également valable pour SN-Boost, dont le score AUC se dégrade avec l'utilisation du SMOTE proposé par [86]. Ainsi, la méthode de génération de données synthétiques (Smote-SN) améliore le AUC de plusieurs algorithmes. Aussi, il est possible de remarquer que les méthodes basées sur le Boosting obtiennent de meilleurs résultats que les méthodes qui utilisent la régression logistique avec une faible régularisation, et que la méthode ICA-SN obtient un meilleur AUC que l'ICA.

TABLE 5.5 – Classification d'apologistes au jihad

Method	TP	FP	TN	FN	AUC
CollectiveBoost (3 classes)	85.18	14.82	99.39	0.61	0.9876
CollectiveBoost (6 classes)	88.88	11.12	99.61	0.39	0.9910
CollectiveBoost (9 classes)	87.03	12.97	99.73	0.27	**0.9936**
SN-Boost	92.59	7.41	98.68	0.32	0.9837
SN-Boost-RegSmote	85.18	14.82	99.68	0.32	0.9837
RamoBoost	74.07	25.93	99.76	0.24	0.9610
SmoteBoost	90.74	9.26	97.99	2.01	0.9674
SmoteBoost-RegSmote	83.33	16.67	99.68	0.32	0.9652
RUSBoost	79.62	20.38	98.70	1.3	0.9522
ADAC2	100	0	0	100	0.9896
EdgeSample	40.74	59.26	99.94	0.06	0.9206
ICA-SN	62.96	37.04	99.83	0.17	0.9299
ICA	62.96	37.04	99.83	0.17	0.9233
Sur/Sous échantillonnement	64.81	35.19	99.85	0.15	0.9245
SMOTE-ENN	20.37	79.63	100	0	0.9581
SMOTE-NE	66.66	33.34	99.84	0.16	0.9392
SMOTE-IPF	50	50	99.93	0.07	0.9374

5.6.2 Détection de pollueurs de contenu

Ce jeu de donnés concerne les pollueurs de contenu sur Twitter. Ces utilisateurs interagissent avec différents comptes afin de les inciter à les suivre. Cela leur permet d'augmenter leur visibilité. Le jeu de données a été collecté par [228] en 2011. Après la suppression des comptes qui n'ont effectué aucune interaction, le jeu est composé de 7K nœuds et 21M d'arêtes (principalement des similarités d'URL). Le jeu est équilibré, mais nous n'avons utilisé que 10% des pollueurs et 50% d'utilisateurs légitimes pour l'entraînement.

Le tableau 5.6 présente les résultats obtenus par les différents algorithmes. Il est possible de voir que les méthodes n'utilisant pas le boosting obtiennent les meilleurs résultats pour ce jeu de données (EdgeSample,SMOTE-NE et SMOTE-IPF). En effet, le Boosting a tendance à éviter le sur-apprentissage [161] et évite ainsi d'apprendre des règles trop complexes. Cela peut réduire son efficacité sur des jeux de données où le jeu de test a une haute similarité avec le jeu d'entraînement.

Parmi les algorithmes de Boosting, AdaC2 obtient le meilleur résultat, suivi par CollectiveBoost à 9 classes. Les méthodes CollectiveBoost et SN-Boost obtiennent de meilleurs résultats que les autres méthodes de Boosting (sauf ADAC2). Aussi, l'utilisation du ICA-SN réduit le score AUC, comparé au ICA. Cela peut être dû au fait que le graphe est clairsemé et composé de plusieurs segments non connectés, qui réduisent la qualité des voisins structurels.

5.6.3 Détection de "faux amis"

Ces expérimentations concernent la détection de faux amis (fake followers) sur les réseaux sociaux. Ces faux amis sont des profils qui peuvent être achetés pour augmenter la visibilité d'un utilisateur. Ce comportement peut être considéré comme du SPAM [206] étant donné qu'il consiste à tromper le réseau social pour faire apvis-à-visre les publications en tête de liste. Aussi, il permet à des spammeurs et autres utilisateurs malveillants de se faire suivre par des faux amis afin d'éviter la détection. Ce jeu de données est composé de profils bénins ainsi que de faux amis collectés sur Twitter et mis à disposition par [206]. Afin de mettre en évidence les avantages et les inconvénients des différents algorithmes de classification collective, nous n'avons gardé que les profils ayant effectué une interaction (mention, retweet ou similarité d'URL). Le jeu de données est non équilibré avec 2565 comptes normaux et 851 faux amis. Le jeu est composé de 4067 nœuds et 275K arêtes (principalement similarité d'URL). Nous avons utilisé 50% des données pour l'entraînement et 50% pour le test.

Le tableau 5.7 présente les résultats obtenus par les différents algorithmes. Les meilleurs scores sur ce jeu de données sont obtenus par des algorithmes qui n'utilisent pas le Boosting (EdgeSample et SMOTE-NE). Concernant les algorithmes de Boosting, RUSBoost obtient les meilleurs résultats, suivi de CollectiveBoost à 6 classes. SN-Boost obtient un meilleur score

TABLE 5.6 – Classification de pollueurs de contenu

Method	TP	FP	TN	FN	AUC
CollectiveBoost (3 classes)	82.78	17.22	99.84	0.16	0.9313
CollectiveBoost (6 classes)	84.43	15.57	99.84	0.16	0.9356
CollectiveBoost (9 classes)	83.52	17.22	99.92	0.08	0.9451
SN-Boost	83.36	16.64	99.29	0.71	0.9234
RamoBoost	75.91	24.09	100	0	0.9052
SN-Boost-RegSmote	83.44	16.56	99.53	0.47	0.9245
SmoteBoost	83.77	16.23	99.29	0.71	0.9409
SmoteBoost-RegSmote	83.44	16.56	99.53	0.47	0.9245
RUSBoost	95.03	4.97	67.28	32.72	0.9144
ADAC2	100	0	1.71	98.29	0.9558
EdgeSample	93.62	6.38	99.61	0.39	0.9721
ICA-SN	94.53	5.47	76.79	23.21	0.8392
ICA	95.36	4.64	77.25	22.75	0.877
Sur/Sous échantillonnement	94.45	5.55	80.91	19.09	0.8819
SMOTE-ENN	47.51	52.49	100	0	0.9292
SMOTE-NE	89.23	10.77	99.45	0.55	0.9701
SMOTE-IPF	86.67	13.33	99.92	0.08	0.9664

AUC que SmoteBoost et RamoBoost. Aussi, les expérimentations montrent que l'utilisation de la méthode proposée pour la génération de données synthétiques (Smote-SN) améliore les résultats de SmoteBoost ainsi que de SN-Boost. ICA-SN obtient également un meilleur score que ICA.

TABLE 5.7 – Classification de faux amis

Method	TP	FP	TN	FN	AUC
CollectiveBoost (3 classes)	76.78	23.22	69.00	31	0.8909
CollectiveBoost (6 classes)	74.57	25.43	100	0	0.8910
CollectiveBoost (9 classes)	74.01	25.99	100	0	0.8866
SN-Boost	75.04	24.96	99.84	0.16	0.8827
SN-Boost-RegSmote	78.42	21.58	69.54	30.46	0.8540
RamoBoost	67.96	32.04	99.92	0.08	0.8460
SmoteBoost	81.33	18.67	62.69	37.31	0.8525
SmoteBoost-RegSmote	80.20	19.8	67.91	32.09	0.8662
RUSBoost	89.68	10.32	64.64	35.36	0.9179
ADAC2	100	0	3.42	96.58	0.8862
EdgeSample	84.94	15.06	100	0	**0.9437**
ICA-SN	84.52	15.48	94.15	5.85	0.8806
ICA	84.09	15.91	93.06	6.94	0.8773
Sur/Sous échantillonnement	85.55	14.45	95.71	4.29	0.9015
SMOTE-ENN	48.07	51.93	100	0	0.8734
SMOTE-NE	80.53	19.47	99.53	0.47	0.930
SMOTE-IPF	76.07	23.93	99.84	0.16	0.914

5.6.4 Détection de SPAM

Le dataset suivant concerne la détection de SPAM sur Twitter [206]. Il est composé d'utilisateurs bénins ainsi que de spammeurs traditionnels (8% des Spammeurs) et de Spammeurs sociaux (91% des Spammeurs). La première classe de Spammeurs concernent ceux qui publient des messages répétitifs et explicites, se souciant peu de la détection. Ces derniers créent de nouveaux comptes s'ils sont détectés par la plateforme. La deuxième classe de Spammeurs construit une réputation sur le réseau social afin d'éviter la détection. La deuxième classe de Spammeurs peut avoir plusieurs milliers de followers et publie souvent du contenu légitime. Après avoir enlevé les utilisateurs et Spammeurs n'ayant effectué aucune interaction, le jeu contient 5719 comptes et 4.5M d'arêtes (principalement similarité d'URL). Ce jeu de données est équilibré

avec 2568 utilisateurs bénins et 2368 Spammeurs. Nous avons utilisé dans ce jeu 50% des utilisateurs bénins et seulement 10% des Spammeurs pour l'entraînement et le reste pour le test des modèles.

Le tableau 5.8 présente les résultats obtenus par les différents algorithmes.

Le plus haut score AUC est obtenu par CollectiveBoost. CollectiveBoost avec 3 classes obtient le meilleur score AUC, et CollectiveBoost avec 6 classes obtient le quatrième score après ADAC2. Nous n'avons pas testé CollectiveBoost avec 9 classes, étant donné que plusieurs de ces classes contenaient peu d'arêtes. En effet, le nombre de classes sémantiques (thèmes) pour ce jeu de données est limité. Nous pouvons également voir que, dans ce jeu de données, ICA-SN obtient un meilleur score AUC que ICA.

TABLE 5.8 – Classification de SPAMs

Method	TP	FP	TN	FN	AUC
CollectiveBoost (3 classes)	85.83	14.17	70.95	29.05	**0.9072**
CollectiveBoost (6 classes)	77.90	22.1	74.61	25.39	0.8938
ICA	87.19	12.81	91.58	8.42	0.8849
ICA-SN	86.63	13.37	94.7	5.3	0.8931
ADAC2	80.48	19.52	79.20	20.8	0.9026
EdgeSample	74.48	25.52	99.84	0.16	0.9006
Sur/Sous échantillonnement	89.02	10.98	88.55	11.45	0.888
RUSBoost	87.05	12.95	45.24	54.76	0.7337
SmoteBoost	52.25	47.75	99.84	0.16	0.894

5.7 Conclusion

Les réseaux sociaux numériques sont caractérisés par des données textuelles et des données relationnelles. Les interactions sur les réseaux sociaux suivent souvent une loi de puissance où certains nœuds sont sollicités par un grand nombre d'utilisateurs. Ces nœuds de degrés élevés peuvent réduire l'efficacité de classifications à cause de la redondance de voisins [229]. Ainsi, il est souvent souhaitable d'utiliser d'autres relations sémantiques que l'interaction pour définir le voisinage des nœud.

En effet, certaines relations sémantiques comme l'existence d'une triade (3-clique), la similarité de co-citations ou les similarités basées sur les marches aléatoires peuvent être plus discriminantes que les interactions. Alors qu'il existe plusieurs travaux concernant l'identification de voisinages basés sur ces relations sémantiques pour graphes simples, il y a peu de travaux qui concernent les graphes multimodaux où les liens sont attribués. Ces derniers peuvent plus naturellement représenter les activités sur les réseaux sociaux numériques étant donné que des

messages contenant du texte sont souvent utilisés pour déterminer les interactions : Retweet, Reply, Like, etc.

Ce chapitre a présenté quatre méthodes de classification collectives sur les réseaux sociaux numériques. La première méthode permet de réduire l'effet du bruit et du chevauchement de la classe majoritaire en introduisant un algorithme basé sur le KNN adapté à la classification avec classes non équilibrées.

La seconde méthode permet de réduire l'effet de la distribution en loi de puissance des degrés sur ces réseaux. L'algorithme pondère les différents modes d'interaction en fonction de leur capacité à distinguer les différentes classes. En utilisant des marches aléatoires, il détermine un voisinage pour chaque nœud. L'algorithme échantillonne ensuite les voisins pour éviter de propager des erreurs d'un voisin à l'autre. Cet algorithme agrège les différents modes d'interaction, perdant ainsi des informations permettant de distinguer les différentes classes. Cette méthode est comparé au ICA classique [15], étant donné que les deux méthodes utilisent la régression logistique et la pondération des erreurs de classification. ICA-SN a obtenu de meilleurs scores que le ICA sur 3 jeux de données. Notez que ICA-SN peut prendre en entré un des algorithmes proposés dans les sections 3 et 4.

La troisième méthode introduit un modèle pour la génération de données synthétiques ainsi qu'un algorithme qui permet de sélectionner les nœuds à combiner dans chacune de ses étapes. La méthode de génération de données synthétiques introduite a été évalué sur différents algorithmes avec plusieurs jeux de données. Cette dernière a augmenté les scores de la majorité des algorithmes en comparaison avec l'utilisation du Smote classique [163]. Aussi, l'algorithme introduit (Smote-SN) obtient de meilleurs scores que la majorité des algorithmes de Boosting. Il dépasse RAMOBoost dont il s'inspire sur tous les jeux de données.

Le quatrième algorithme permet de générer de nouveaux modes à partir des attributs des arêtes. Il s'inspire du second algorithme proposé dans sa méthode d'échantillonnage des voisins, mais il effectue cela sur chaque mode. Cet algorithme utilise une méthode de Boosting afin de déterminer les classes sémantiques et de générer le modèle de classification. Il échantillonne des relations sémantiques dans chacune de ses itérations. Cet algorithme obtient de hauts scores AUC comparé à la majorité des méthodes qui ont été évaluées.

Chapitre 6

Conclusion générale

L'émergence des réseaux sociaux a permis pour la première fois de connecter des utilisateurs vivant sur des continents différents. En effet, ces réseaux se sont rapidement inscrits dans la vie quotidienne de millions d'utilisateurs, devenant des plateformes de socialisation, d'information, de commerce et de divertissement. Cette diversité des usages des réseaux sociaux numériques a également été accompagnée par des individus et des communauté malveillantes. En effet, les réseaux sociaux numériques offrent plusieurs avantages à ces derniers, comme l'anonymat, la capacité d'atteindre une large audience ainsi que la capacité d'automatiser leurs activités.

L'analyse des réseaux sociaux numériques a permis de vérifier un grand nombre de comportements sociaux, comme la distribution des degrés qui suit une loi de puissance, le court diamètre des réseaux humains ou l'organisation des acteurs d'un réseau en communautés. Cette dernière propriété est particulièrement utile pour la détection d'utilisateurs malveillants. Il est souvent possible de détecter des communautés malveillantes comme des apologistes au jihad en étudiant les interactions sur les réseaux.

La majorité des méthodes permettant le partitionnement d'un réseau ignore l'hétérogénéité des interactions sur les réseaux sociaux. En effet, les réseaux sociaux numériques sont caractérisés par différents modes d'interaction entre utilisateurs (Reply, Retweet, Like, Mentions, etc.). Ces interactions affectent différemment la segmentation d'un réseau étant donné que certains groupes ne peuvent être identifiés qu'en utilisant un sous-ensemble des modes disponibles. De plus, la majorité des réseaux sociaux sont plus naturellement modélisés par des modèles avec arêtes attribuées. Ces attributs modélisent le plus souvent le texte qui est retrouvé sur les messages. Ainsi, les attributs des arêtes définissent des sens sémantiques aux interactions qui affectent la segmentation.

Nous avons introduit dans cette thèse des méthodes de segmentation de réseaux non supervisées et semi supervisées. Les deux méthodes non supervisées qui ont été introduites se basent sur le principe de réduction de la taille du réseau par la sélection de représentants et par le regroupement de représentants de communautés. La première méthode échantillonne aléatoi-

rement des représentants, alors que la seconde méthode utilise du pattern mining pour détecter une combinaison d'utilisateurs centraux (patterns) considérés comme représentants. La seconde méthode a nécessité l'introduction d'un modèle utilisé pour le pattern mining sur les réseaux sociaux. Les deux méthodes regroupent ensuite les représentants en communautés, et affecter des utilisateurs à leurs représentants.

Trois méthodes de classification collective introduite dans cette thèse se basent sur l'algorithme itératif de classification (ICA) [15]. Ces algorithmes utilisent les attributs des nœuds ainsi que les attributs de leurs voisins et les labels prédits des voisins pour inférer itérativement les labels des nœuds. L'algorithme (ICA-SN) proposé répond au problème de distribution de puissance des degrés des nœud dans un réseau, ainsi qu'au problème de cascade d'erreurs induites par l'ICA. Il répond à cela en définissant de nouveaux voisinages, en utilisant la structure du graphe, ainsi qu'en échantillonnant les voisins des nœud pour réduire la cascade d'erreurs. Le second algorithme, qui se base sur ICA-SN, est utilisé pour générer des données synthétiques sur le réseau afin de répondre au problèmes de classes non équilibrées. L'algorithme SN-Boost permet de générer des données synthétiques sur les réseaux sociaux multimodaux. Ce dernier obtient des résultats satisfaisant comparé à plusieurs algorithmes de Boosting. Le dernier algorithme proposé utilise une architecture de Boosting afin de 1) générer des modes à partir des attributs textuels sur les arêtes et 2) générer des modèles de classification permettant de segmenter un réseau attribué. L'algorithme échantillonne les modes identifiés dans l'étape 1 afin de générer plusieurs graphes. Ainsi, chaque étape de Boosting génère plusieurs modèles (1 modèle par échantillon de graphes).

Cette thèse a permis le développement de modèles et d'algorithmes permettant la segmentation d'un réseau. Nous avons seulement considéré un partitionnement dur de ces derniers alors qu'il est souvent possible qu'un utilisateur appartienne à plusieurs communautés. Les travaux futurs devront ainsi considérer le cas où un utilisateur peut appartenir à plusieurs réseaux. Aussi, la méthode non supervisée qui a été développée ne considère pas l'hétérogénéité des modes (reply, retweet, similarité d'URL). L'adaptation de ces méthodes aux problématiques ci-dessus fera partie de travaux futurs.

Annexes

Annexe A

Tableaux pour la détection de communautés sur des réseaux synthétiques

Cet annexe présente les scores de NMI, de modularité et de F-mesure obtenus sur les réseaux synthétiques présentés dans la section 4.3.1. Nous rappelons que les scores de F-mesure sont calculés pour chaque graphe sur sa plus petite communauté. Si un graphe a plusieurs communautés de taille minimale, une est sélectionnée aléatoirement. Les tableau A.1,A.2 et A.3 présentent respectivement les scores de NMI, de modularité et de F-mesure pour le jeu 1 présenté dans la section 4.3.1.

TABLE A.1 – Scores de NMI pour graphes LFR avec paramètres par défaut (jeu 1)

μ	CPM		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s.		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.
0.1	1	0	1	0	1	0	1	0	1	0	0.61	0.12	1	0	0.96	0.001	1	0	1	0
0.2	1	0	1	0	1	0	0.99	0.001	1	0	0.57	0.06	1	0	0.96	0.007	1	0	1	0
0.3	0.99	0.002	1	0	1	0	0.98	0.003	0.99	0.004	0.51	0.09	0.99	0.02	0.96	0.008	1	0	1	0
0.4	0.99	0.002	0.99	0.005	1	0	0.99	0.005	0.92	0.025	0.45	0.08	0.99	0.06	0.95	0.007	0.99	0	0.99	0
0.5	0.99	0.004	0.99	0.001	0.62	0.01	0.99	0.005	0.82	0.033	0.41	0.067	0.99	0.003	0.955	0.008	0.99	0.001	0.99	0.001
0.6	0.99	0.013	0.99	0.003	0.57	0.016	0.98	0.01	0.44	0.06	0.29	0.05	0.77	0.39	0.94	0.01	0.99	0.01	0.989	0.003
0.7	0.918	0.04	0.95	0.02	0.51	0.017	0.90	0.03	0.03	0.01	0.16	0.03	0	0	0.90	0.017	0.93	0.02	0.94	0.01
0.75	0.62	0.083	0.728	0.04	0.45	0.02	0.60	0.07	0.003	0.002	0.14	0.02	0	0	0.67	0.06	0.702	0.03	0.75	0.03
0.8	0.157	0.03	0.26	0.038	0.41	0.014	0.21	0.02	0	0	0.09	0.01	0	0	0.20	0.02	0.31	0.03	0.367	0.03

TABLE A.2 – Scores de Modularity pour graphes LFR avec paramètres par défaut (jeu 1)

μ	CPM		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s.		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C..	M.	E.C..	M.	E.C.
0.1	0.86	0.001	0.86	0.001	0.86	0.001	0.86	0.001	0.86	0.001	0.51	0.05	0.86	0.001	0.85	0.002	0.86	0.0015	0.86	0.0015
0.2	0.76	0.001	0.76	0.001	0.76	0.001	0.76	0.001	0.76	0.001	0.39	0.03	0.76	0.001	0.75	0.002	0.76	0.001	0.76	0.001
0.3	0.66	0.001	0.66	0.001	0.66	0.001	0.66	0.001	0.65	0.009	0.34	0.04	0.66	0.001	0.65	0.001	0.663	0.002	0.663	0.002
0.4	0.56	0.002	0.56	0.002	0.56	0.002	0.56	0.002	0.44	0.004	0.28	0.025	0.56	0.003	0.55	0.002	0.56	0.001	0.56	0.001
0.5	0.46	0.001	0.46	0.001	0.06	0.001	0.46	0.002	0.26	0.032	0.21	0.024	0.46	0.002	0.45	0.002	0.46	0.0019	0.46	0.0019
0.6	0.36	0.002	0.36	0.001	0.06	0.001	0.36	0.002	0.06	0.01	0.165	0.016	0.28	0.147	0.36	0.002	0.36	0.002	0.36	0.002
0.7	0.257	0.008	0.25	0.003	0.06	0.001	0.26	0.004	0.0009	0.0008	0.13	0.008	0	0	0.26	0.002	0.25	0.004	0.25	0.004
0.75	0.192	0.008	0.186	0.005	0.059	0.001	0.19	0.005	0	0	0.127	0.005	0	0	0.21	0.002	0.186	0.005	0.183	0.005
0.8	0.127	0.003	0.12	0.003	0.05	0.001	0.16	0.003	0	0	0.12	0.005	0	0	0.187	0.001	0.11	0.005	0.11	0.004

TABLE A.3 – Scores de F-mesure pour graphes LFR avec paramètres par défaut (jeu 1)

μ	SW	Walk.	Info.	Louv.	MCL	EigenDec	L. Prop	SpinG.	WalkMiner-s.	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.	M.	M.	M.	M.
0.1	1	1	1	1	1	0.41	1	0.73	1	1
0.2	1	1	1	1	1	0.36	1	0.69	1	1
0.3	0.99	1	1	1	1	0.27	1	0.64	1	1
0.4	1	1	1	0.88	0.99	0.24	0.94	0.71	1	1
0.5	0.98	1	0.94	0.175	0.928	0.25	1	0.69	1	1
0.6	0.988	0.986	1	0.168	0.88	0.62	0.24	0.86	0.99	0.96
0.7	0.81	0.94	0.16	0.69	0.038	0.09	0.04	0.73	0.879	0.947
0.75	0.72	0.85	0.13	0.50	0.04	0.09	0.04	0.62	0.745	0.81
0.8	0.13	0.36	0.12	0.25	0.04	0.09	0.04	0.14	0.36	0.41

Les tableaux A.4, A.5 et A.6 présentent respectivement les scores de NMI, de modularité et de F-mesure pour les graphes du jeu 2 introduit dans la section 4.3.1.

TABLE A.4 – Scores de NMI pour graphes LFR à haut degrés et 1 000 nœuds (jeu 2)

Mode	SW		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
μ	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.
1	0.97	0.02	0.936	0.04	0.82	0.09	0.97	0.029	0.52	0.10	0.69	0.18	0.99	0.006	0.958	0.21	0.949	0.029	0.949	0.029
2	0.90	0.05	0.80	0.07	0.719	0.089	0.90	0.05	0.39	0.07	0.50	0.15	0.93	0.15	0.90	0.06	0.85	0.062	0.85	0.062
3	0.71	0.20	0.53	0.16	0.58	0.17	0.76	0.20	0.26	0.08	0.39	0.105	0.65	0.279	0.78	0.19	0.61	0.20	0.61	0.20
4	0.42	0.14	0.34	0.10	0.41	0.149	0.45	0.17	0.18	0.05	0.24	0.05	0.12	0.18	0.52	0.19	0.386	0.15	0.386	0.15
5	0.227	0.08	0.25	0.06	0.25	0.12	0.26	0.10	0.15	0.03	0.22	0.067	0	0	0.34	0.129	0.259	0.087	0.259	0.087

TABLE A.5 – Scores de modularité pour graphes LFR à haut degré et 1 000 nœuds (jeu 2)

μ	SW		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.
1	0.67	0.05	0.635	0.16	0.622	0.158	0.67	0.05	0.23	0.11	0.52	0.135	0.67	0.058	0.66	0.05	0.6688	0.05	0.6688	0.05
2	0.55	0.04	0.54	0.04	0.53	0.04	0.56	0.04	0.14	0.07	0.38	0.10	0.52	0.11	0.56	0.03	0.55	0.044	0.55	0.044
3	0.44	0.07	0.41	0.08	0.42	0.088	0.46	0.061	0.057	0.041	0.32	0.047	0.34	0.14	0.46	0.05	0.43	0.07	0.43	0.07
4	0.33	0.04	0.30	0.049	0.31	0.081	0.37	0.043	0.015	0.01	0.25	0.024	0.055	0.096	0.39	0.04	0.33	0.05	0.33	0.05
5	0.28	0.028	0.239	0.027	0.20	0.092	0.32	0.026	0	0	0	0	0	0	0.35	0.02	0.28	0.027	0.28	0.027

TABLE A.6 – Scores de fmeasure pour graphes LFR à haut degré et 1 000 nœuds (jeu 2)

μ	SW	Walk.	Info.	Louv.	MCL	EigenDec	L. Prop	SpinG.	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.	M.	M.	M.	M.
0.1	0.98	0.99	0.96	0.97	0.79	0.55	0.99	0.89	0.971	0.971
0.2	0.87	0.95	0.78	0,94	0.71	0.40	0.97	0,91	0.91	0.91
0.3	0.84	0.79	0.91	0,88	0.49	0,32	0.93	0.88	0.834	0.834
0.4	0.70	0.62	0.81	0.72	0,28	0.28	0.25	0.71	0.749	0.749
0.5	0.54	0.47	0.67	0.57	0.153	0.13	0.07	0.58	0.665	0.665

Les tableaux A.7, A.8 et A.9 présentent respectivement les scores de NMI, de modularité et de F-mesure pour les graphes du jeu 3 introduit dans la section 4.3.1.

TABLE A.7 – Scores de NMI pour graphes LFR à faible densité et 1 000 nœuds (jeu 3)

Mode	SW		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
μ	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.
0.1	0.942	0.009	0.938	0.009	0.717	0.01	0.945	0.007	0.81	0.018	0.554	0.295	0.941	0.009	0.911	0.01	0.96	0.008	0.956	0.01
0.2	0.874	0.011	0.861	0.010	0.675	0.005	0.877	0.015	0.674	0.017	0.50	0.21	0.881	0.013	0.848	0.015	0.907	0.016	0.902	0.015
0.3	0.789	0.019	0.764	0.02	0.644	0.009	0.79	0.022	0.56	0.014	0.468	0.119	0.80	0.02	0.778	0.016	0.833	0.02	0.833	0.019
0.4	0.648	0.027	0.642	0.021	0.615	0.005	0.633	0.033	0.467	0.010	0.372	0.10	0.637	0.153	0.657	0.024	0.71	0.025	0.719	0.024
0.5	0.502	0.0247	0.515	0.028	0.596	0.009	0.387	0.035	0.411	0.013	0.305	0.08	0.129	0.227	0.495	0.03	0.537	0.045	0.553	0.046

TABLE A.8 – Scores de modularité pour graphes LFR à faible densité et 1 000 nœuds (jeu 3)

μ	SW		Walk.		Info.		Louv.		MCL		EigenDec		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.
1	0.855	0.003	0.858	0.003	0.239	0.007	0.859	0.003	0.632	0.048	0.505	0.262	0.853	0.007	0.853	0.004	0.858	0.003	0.858	0.003
2	0.753	0.005	0.753	0.005	0.228	0.012	0.758	0.004	0.419	0.044	0.455	0.17	0.753	0.006	0.755	0.004	0.757	0.004	0.756	0.005
3	0.644	0.004	0.639	0.005	0.223	0.009	0.66	0.003	0.196	0.039	0.424	0.102	0.653	0.003	0.657	0.003	0.655	0.005	0.655	0.005
4	0.526	0.007	0.51	0.008	0.215	0.008	0.56	0.006	0.054	0.01	0.368	0.09	0.526	0.12	0.563	0.004	0.551	0.004	0.548	0.006
5	0.40	0.008	0.383	0.009	0.21	0.01	0.469	0.006	0.005	0.004	0.34	0.082	0.117	0.207	0.485	0.006	0.455	0.004	0.447	0.008

TABLE A.9 – Scores de fmeasure pour graphes LFR à faible densité et 1 000 nœuds (jeu 3)

μ	SW	Walk.	Info.	Louv.	MCL	EigenDec	L. Prop	SpinG.	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.	M.	M.	M.	M.
0.1	0.928	0.942	0.24	0.88	0.667	0.537	0.931	0.648	0.973	0.973
0.2	0.821	0.858	0.227	0.825	0.619	0.332	0.873	0.697	0.89	0.875
0.3	0.771	0.758	0.229	0.707	0.434	0.354	0.814	0.608	0.821	0.808
0.4	0.566	0.552	0.177	0.514	0.193	0.281	0.667	0.516	0.675	0.698
0.5	0.330	0.316	0.152	0.277	0.131	0.16	0.213	0.356	0.472	0.484

Les tableaux A.10, A.11 et A.12 présentent respectivement les scores de NMI, de modularité et de F-mesure pour les graphes du jeu 4 introduit dans la section 4.3.1.

TABLE A.10 – Scores de NMI pour graphes LFR avec paramètres par défaut et 10 000 nœuds (jeu 4)

μ	SW		Walk.		Info.		Louv.		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.
0.5	0.96	0.02	0.96	0.01	0.74	0.001	0.86	0.004	0.99	0	0.71	0.001	0.959	0.007	0.999	0
0.6	0.90	0.01	0.85	0.01	0.73	0.001	0.83	0.004	.99	0.001	0.70	0.001	0.939	0.01	0.993	0.0008
0.7	0.76	0.02	0.67	0.01	0.72	0.001	0.729	0.009	0.61	0.46	0.59	0.007	0.745	0.012	0.836	0.007
0.75	0.57	.12	0.64	0.01	0.71	0.001	0.43	0.023	0	0	0.33	0.01	0.547	0.021	0.649	0.011
0.8	0.38	0.09	0.52	0.01	0.71	0.001	0.14	0.01	0	0	0.11	0.007	0.44	0.017	0.493	0.009

TABLE A.11 – Scores de modularité pour graphes LFR avec paramètres par défaut et 10 000 nœuds (jeu 4)

μ	SW		Walk.		Info.		Louv.		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.
0.5	0.49	0	0.49	0	0.07	0	0.50	0	0.49	0	0.48	0	0.49	0.0005	0.496	0.0003
0.6	0.389	.001	0.37	0.001	0.07	0	0.40	0	0.39	0	0.39	0	0.397	0.0004	0.395	0.0003
0.7	0.25	0.002	0.22	0.001	0.07	0.0003	0.30	0.001	0.19	0.14	0.30	0.0007	0.268	0.001	0.265	0.001
0.75	0.19	0.01	0.15	0.002	0.07	0.0005	0.24	0.002	0	0	0.27	0.001	0.192	0.002	0.188	0.002
0.8	0.18	0.02	0.10	0.001	0.07	0	0.22	0.006	0	0	0.26	0.001	0.142	0.001	0.14	0.001

TABLE A.12 – Scores de F-mesure pour graphes LFR avec paramètres par défaut et 10 000 nœuds (jeu 4)

μ	SW	Walk.	Info.	Louv.	L. Prop	SpinG.	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.	M.	M.
0.5	1	0.95	0.12	0.24	1	0.09	0.742	1
0.6	0.81	0.76	0.12	0.19	0.97	0.07	0.825	0.979
0.7	0.73	0.52	0.11	0.16	0.70	0.09	0.628	0.856
0.75	0.41	0.33	0.11	0.10	0.003	0.05	0.332	0.453
0.8	0.24	0.10	0.01	0.02	0.05	0.05	0.186	0.231

Les tableaux A.13, A.14 et A.15 présentent respectivement les scores de NMI, de modularité et de F-mesure pour les graphes du jeu 5 introduit dans la section 4.3.1.

TABLE A.13 – Scores de NMI pour graphes LFR avec paramètres à haut degré et 10 000 nœuds (jeu 5)

Mode	SW		Walk.		Info.		Louv.		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C..	M.	E.C..	M.	E.C.
0.1	0.977	0.012	0.969	0.005	0.981	0.004	0.994	0.002	0.99	0.0006	0.89	0.012	0.983	0.001	0.984	0.001
0.2	0.95	0.02	0.90	0.01	0.958	0.01	0.989	0.003	0.998	0.001	0.873	0.015	0.952	0.003	0.956	0.002
0.3	0.91	0.02	0.77	0.02	0.90	0.014	0.979	0.005	0.99	0.003	0.86	0.014	0.89	0.018	0.89	0.019
0.4	0.78	0.02	0.619	0.024	0.80	0.018	0.88	0.009	0.90	0.011	0.79	0.012	0.738	0.02	0.738	0.22
0.5	0.57	0.03	0.51	0.021	0.67	0.024	0.72	0.02	0.34	0.34	0.66	0.013	0.50	0.033	0.505	0.036
0.6	0.27	0.03	0.45	0.02	0.55	0.02	0.39	0.03	0	0	0.43	0.012	0.31	0.03	0.297	0.041

TABLE A.14 – Scores de modularité pour graphes LFR avec paramètres à hauts degrés et 10 000 nœuds (jeu 5)

Mode	SW		Walk.		Info.		Louv.		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C..	M.	E.C..	M.	E.C.
1	0.86	0.002	0.867	0.003	0.866	0.003	0.872	0.002	0.872	0.002	0.854	0.0032	0.872	0.0003	0.871	0.0002
2	0.76	0.003	0.75	0.004	0.76	0.005	0.77	0.002	0.77	0.002	0.75	0.003	0.767	0.001	0.767	0.001
3	0.65	0.004	0.62	0.006	0.645	0.0058	0.672	0.002	0.672	0.002	0.65	0.0034	0.657	0.005	0.657	0.005
4	0.548	0.007	0.48	0.007	0.53	0.006	0.57	0.002	0.57	0.004	0.558	0.003	0.53	0.006	0.536	0.006
5	0.42	0.012	0.33	0.009	0.42	0.007	0.47	0.003	0.138	0.21	0.46	0.002	0.39	0.01	0.39	0.01
6	0.30	0.012	0.20	0.007	0.32	0.004	0.36	0.005	0	0	0.38	0.001	0.28	0.008	0.285	0.008

TABLE A.15 – Scores de fmeasure pour graphes LFR avec paramètres à hauts degrés et 10 000 nœuds (jeu 5)

Mode	SW	Walk.	Info.	Louv.	L. Prop	SpinG.	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.	M.	M.
0.1	0,95	1	0,99	0,75	1	0,13	0.55	0.55
0.2	0,66	0,97	0,99	0,44	0,98	0,24	0.586	0.749
0.3	0,45	0,9	0,96	0,42	0,98	0,19	0.586	0.744
0.4	0,36	0,70	0,90	0,28	0,88	0,09	0.680	0.680
0.5	0,29	0,56	0,8	0,19	0,32	0,16	0.48	0.466
0.6	0,13	0,34	0,67	0,14	0,003	0,12	0.585	0.551

Les tableaux A.16, A.17 et A.18 présentent respectivement les scores de NMI, de modularité et de F-mesure pour les graphes du jeu 6 introduit dans la section 4.3.1.

TABLE A.16 – Scores de NMI pour graphes LFR à faible densité et 10 000 nœuds (jeu 6)

Mode	SW		Walk.		Info.		Louv.		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C..	M.	E.C..	M.	E.C.
0.1	0.9624	0.002	0.959	0.0008	0.815	0.001	0.876	0.003	0.965	0.001	0.681	0.0009	0.960	0.010	0.975	0.001
0.2	0.9217	0.0038	0.912	0.005	0.786	0.003	0.816	0.004	0.929	0.004	0.64	0.001	0.93	0.03	0.945	0.004
0.3	0.867	0.005	0.847	0.009	0.763	0.002	0.749	0.005	0.889	0.004	0.59	0.004	0.877	0.006	0.906	0.003
0.4	0.773	0.016	0.77	0.004	0.74	0	0.644	0.006	0.823	0.005	0.526	0.004	0.814	0.003	0.844	0.004
0.5	0.677	0.004	0.687	0.009	0.731	0	0.49	0.008	0.722	0.007	0.422	0.007	0.73	0.009	0.747	0.006

TABLE A.17 – Scores de modularité pour graphes LFR à faible densité et 10 000 nœuds (jeu 6)

Mode	SW		Walk.		Info.		Louv.		L. Prop		SpinG.		WalkMiner-s		WalkMiner-s NoModOpt.	
	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C.	M.	E.C..	M.	E.C..	M.	E.C.
1	0.889	0.0007	0.892	0.001	0.243	0.002	0.899	0.001	0.891	0.001	0.864	0.001	0.894	0.001	0.893	0.001
2	0.786	0.0011	0.787	0.001	0.23	0.002	0.80	0.001	0.791	0.001	0.772	0.001	0.793	0.001	0.792	0.0007
3	0.679	0.002	0.67	0.003	0.22	0.002	0.70	0.001	0.691	0.002	0.682	0.001	0.694	0.001	0.692	0.001
4	0.564	0.0015	0.546	0.001	0.219	0	0.61	0.001	0.595	0.001	0.594	0.0009	0.594	0.001	0.590	0.001
5	0.437	0.001	0.404	0.003	0.218	0.001	0.517	0.001	0.501	0.002	0.517	0.002	0.489	0.001	0.487	0.001

TABLE A.18 – Scores de fmeasure pour graphes LFR avec paramètres par défaut et 10 000 nœuds (jeu 6)

Mode	SW	Walk.	Info.	Louv.	L. Prop	SpinG.	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.	M.	M.
0.1	0.9424	0.947	0.322	0.36	0.963	0.088	0.881	0.970
0.2	0.863	0.764	0.21	0.296	0.829	0.07	0.733	0.882
0.3	0.702	0.663	0.20	0.26	0.68	0.073	0.867	0.867
0.4	0.623	0.566	0.175	0.166	0.801	0.08	0.764	0.764
0.5	0.366	0.374	0.183	0.115	0.579	0.062	0.546	0.546

Les tableaux A.19, A.20 et A.21 présentent respectivement les scores de NMI, de modularité et de F-mesure pour les graphes du jeu 7 introduit dans la section 4.3.1.

TABLE A.19 – Scores de NMI pour graphes LFR avec paramètres par défaut et 100 000 nœuds

Mode	Walk.	Info.	Louv.	L. Prop	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.
0.1	0.988	0.711	0.999	0.999	0.996	0.967
0.2	0.904	0.672	0.994	0.999	0.944	0.944
0.3	0.793	0.661	0.975	0.99	0.835	0.835
0.4	0.651	0.648	0.895	0.944	0.62	0.62

TABLE A.20 – Scores de modularité pour graphes LFR avec paramètres par défaut et 100 000 nœuds

Mode	Walk.	Info.	Louv.	L. Prop	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.
1	0.894	0.10	0.895	0.892	0.895	0.880
2	0.771	0.121	0.796	0.796	0.779	0.779
3	0.639	0.121	0.696	0.696	0.626	0.625
4	0.485	0.122	0.595	0.596	0.436	0.427

TABLE A.21 – Scores de fmeasure pour graphes LFR avec paramètres par défaut et 100 000 nœuds

Mode	Walk.	Info.	Louv.	L. Prop.	WalkMiner-s	WalkMiner-s NoModOpt.
	M.	M.	M.	M.	M.	M.
0.1	0.985	0.0632	1	1	1	0.970
0.2	0.878	0.025	1	1	0.978	0.978
0.3	0.802	0.023	0.597	1	0.906	0.906
0.4	0.467	0.019	0.56	0.938	0.704	0.704

Bibliographie

[1] Abdullah Almaatouq, Ahmad Alabdulkareem, Mariam Nouh, Erez Shmueli, Mansour Alsaleh, Vivek K Singh, Abdulrahman Alarifi, Anas Alfaris, and Alex Sandy Pentland. Twitter : who gets caught ? observed trends in social micro-blogging spam. In *Proceedings of the 2014 ACM conference on Web science*, pages 33–41. ACM, 2014.

[2] 2013 State of Social Media Spam. Technical report, Nexgate, 2013.

[3] Rachel Gwillim and Amado Schwegel. A new look at spam, by the numbers, mar 2010.

[4] Jamie Bartlett and Carl Miller. The state of the art : A litterature review of social media intelligence capabilities for counter-terrorism. (November), 2013.

[5] Gabriel Weimann. Al Qaeda Has Sent You A Friend Request. *Communication*, pages 1–13, 2011.

[6] Nasrullah Sheikh, Zekarias Kefato, and Alberto Montresor. Gat2Vec : Representation Learning for Attributed Graphs. *Computing*, pages 1–23, 2018.

[7] Omar Jaafor and Babiga Birregah. Social engineering threat assessment using a multi-layered graph-based Model. *Lecture Notes in Social Networks*, 17, 2016.

[8] Omar Jaafor and Babiga Birregah. Sponge walker : Community Detection in Large Directed Social Networks using Local Structures and Random Walks. In *ENICS*, Duisburg, 2017.

[9] Omar Jaafor and Babiga Birregah. *Approche multimodale pour la détection d'anomalies sur les réseaux sociaux numériques*. PhD thesis, Université de Technologie de Troyes.

[10] Omar Jaafor and Babiga Birregah. KNN-LC : Classification in unbalanced datasets using a KNN based algorithm and local centralities. In *Data Driven Modeling for Sustainable Engineering*, Accra, 2017. Springer.

[11] O. Jaafor and B. Birregah. Collective classification in social networks. *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2017*, 2017.

[12] Zhana Kuncheva and Giovanni Montana. Community detection in multiplex networks using locally adaptive random walks. *arXiv preprint arXiv :1507.01890*, 1(September) :1–7, 2013.

[13] Stanley Wasserman and Katherine Faust. *Social Network Analysis : Methods and Applications*, volume 8. 1994.

[14] Liaoruo Wang, Tiancheng Lou, Jie Tang, and John E Hopcroft. Detecting Community Kernels in Large Social Networks. In *Data Mining (ICDM)*, pages 784–793, 2011.

[15] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 29(3) :93, 2008.

[16] Omar Jaafor and Babiga Birregah. TweetCapt application de collecte et de visualisation en temps réel de données massives provenant de Twitter ref : IDDN.FR.001.370001.000.S.P.2016.000.10000, 2014.

[17] Omar Jaafor, Babiga Birregah, Charles Perez, and Marc Lemercier. Privacy Threats from Social Networking Service Aggregators. In *Cybercrime and Trustworthy Computing Conference (CTC)*. IEEE Computer Society, 2014.

[18] Giuseppe Labianca. Stephen P. Borgatti, Ajay Mehra, Daniel J. Brass. Network Analysis in the Social Sciences. *science*, 323(April) :892–896, 2009.

[19] John Scott. *Social networks : Critical concepts in sociology*. 2002.

[20] Jure Leskovec. Dynamics of Large Networks. *Technology*, 27344(September) :384, 2008.

[21] Milgram Stanley. The small world problem. *Phychology Today*, 1(1) :61–67, 1967.

[22] Nicole B Ellison Danah m. boyd. Social Network Sites : Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 13 :1, 2007.

[23] Piyush Mangukiya. Social media by the numbers, apr 2016.

[24] Mohammad Behdad, Luigi Barone, Mohammed Bennamoun, and Tim French. Nature-inspired techniques in the context of fraud detection. *IEEE Transactions on Systems, Man and Cybernetics Part C : Applications and Reviews*, 42(6) :1273–1290, 2012.

[25] Chao Yang, Robert Harkreader, Jialong Zhang, Seungwon Shin, and Guofei Gu. Analyzing spammers' social networks for fun and profit : a case study of cyber criminal ecosystem on twitter. *WWW '12 : Proceedings of the 21st international conference on World Wide Web*, pages 71–80, 2012.

[26] Saptarshi Ghosh, Bimal Viswanath, Farshad Kooti, Naveen Kumar Sharma, Gautam Korlam, Fabricio Benevenuto, Niloy Ganguly, and Krishna Phani Gummadi. Understanding and combating link farming in the twitter social network. *Proceedings of the 21st international conference on World Wide Web - WWW '12*, page 61, 2012.

[27] Chao Yang, Robert Harkreader, and Guofei Gu. Empirical evaluation and new design for fighting evolving twitter spammers. *IEEE Transactions on Information Forensics and Security*, 8(8) :1280–1293, 2013.

[28] Kevin D. Mitnick and William L. Simon. *The Art of Deception : Controlling the Human Element in Security*. 2003.

[29] KOOBFACE INSIDE A CRIMEWARE NETWORK, nov 2010.

[30] Andrew Silke. Holy Warriors. *European Journal of Criminology*, 5(1) :99–123, 2008.

[31] Jytte Klausen. Tweeting the Jihad : Social media networks of Western foreign fighters in Syria and Iraq. *Studies in Conflict and Terrorism*, 38(1) :1–22, 2015.

[32] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. Who falls for phish ? A Demographic Analysis of Phishing Susceptibility and Effectiveness of Interventions. *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, pages 373 – 382, 2010.

[33] Derek Kvedar, Michael Nettis, and Steven P Fulton. the Use of Formal Social Engineering Techniques To Identify Weaknesses During a Computer vulnerability competition. *Journal of Computing Sciences in Colleges*, 26(2) :80–87, 2010.

[34] Priya Kaul and Deepak Sharma. Study of Automated Social Engineering, its Vulnerabilities, Threats and Suggested Countermeasures. *International Journal of Computer Applications*, 67(7) :13–16, 2013.

[35] Katharina Krombholz, Heidelinde Hobel, Markus Huber, and Edgar Weippl. Advanced social engineering attacks. *Journal of Information Security and Applications*, pages 1–10, 2014.

[36] Kiran Malagi, Akshata Angadi, and Karuna Gull. A Survey on Security Issues and Concerns to Social Networks. 2(5), 2013.

[37] Ralph Gross and Alessandro Acquisti. Information revelation and privacy in online social networks (Facebook case). *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.

[38] Markus Jakobsson. Modeling and Preventing Phishing Attacks. *Lecture Notes in Computer Science*, 3570(578) :1–19, 2005.

[39] Tom N. Jagatic, Nathaniel a. Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10) :94–100, 2007.

[40] Julie Tuva and Engebrethsen Smith. ISLAMIC STATE AND SOCIAL MEDIA : ETHICAL CHALLENGES AND POWER RELATIONS - ANALYSIS. *Eurasia Review : A journal of news and analysis*, 2015.

[41] SHANE SCOTT. From Minneapolis to ISIS : An American's Path to Jihad, 2015.

[42] Dominique Cardon. Le design de la visibilité : un essai de cartographie du web 2.0. *Réseaux*, 152 :93–137, 2008.

[43] Twitter. Interface de recherche historique.

[44] Twitter. API Streaming.

[45] D M Boyd and N B Ellison. Social network sites : Definition, history, and scholarship. *IEEE Engineering Management Review*, 38(3) :16, 2010.

[46] M LESK and M PEYRADE. Introduction à la théorie des graphes. *cat.inist.fr*, 2012.

[47] B LATOUR, P MAUGUIN, and G TEIL. Théorie des graphes. *cat.inist.fr*, 2010.

[48] B Bollobás. *Graph theory*. 1982.

[49] Matteo Magnani and Luca Rossi. Multi-Stratum Networks : toward a unified model of on-line identities. *arXiv preprint arXiv :1211.0169*, pages 1–18, 2012.

[50] Alessio Cardillo, Massimiliano Zanin, Jesús Gómez-Gardeñes, Miguel Romance, Alejandro J. García del Amo, and Stefano Boccaletti. Modeling the multi-layer nature of the European Air Transport Network : Resilience and passengers re-scheduling under random failures. *European Physical Journal : Special Topics*, 215 :23–33, 2013.

[51] Charles Perez, Babiga Birregah, and Marc Lemercier. A smartphone-based online social network trust evaluation system. *Social Network Analysis and Mining*, 3(4) :1293–1310, 2013.

[52] S. Boccaletti, G. Bianconi, R. Criado, C. I. del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendiña-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1) :1–122, 2014.

[53] Mikko Kivelä, Alexandre Arenas, Marc Barthelemy, James P. Gleeson, Yamir Moreno, and Mason a. Porter. Multilayer Networks. *arXiv*, page 37, 2014.

[54] Douglas Hawkins. *Identification of Outliers*. 1980.

[55] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description : a survey. In *Data Mining and Knowledge Discovery*, jul 2014.

[56] Stephen Ranshous, Shitian Shen, Danai Koutra, Christos Faloutsos, Nagiza F Samatova, Computer Science, Mathematics Division, and Oak Ridge. Anomaly Detection in Dynamic Networks : A Survey. pages 1–27, 2014.

[57] Deepayan (Carnegie Mellon University) Chakrabarti. AutoPart : Parameter-Free Graph Partitioning. In *Knowledge Discovery in Databases : PKDD 2004. PKDD 2004. Lecture Notes in Computer Science*. 2004.

[58] Jing Gao, Feng Liang, Wei Fan, Chi Wang, Yizhou Sun, and Jiawei Han. On community outliers and their efficient detection in information networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, 2010.

[59] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 1346–1355, 2014.

[60] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data*, 10(1) :1–51, 2015.

[61] Sun Jimeng, Qu Huiming, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2005.

[62] Carlos Castillo, Debora Donato, and Vanessa Murdock. Know your Neighbors : Web Spam Detection using the Web Topology. *Framework*, pages 423–430, 2007.

[63] Hanbo Dai. Anomaly Detection on Social Data. *Dissertations and Theses Collection (Open Access)*, 2013.

[64] Leman Akoglu, Mary McGlohon, and Christos Faloutsos. OddBall : Spotting anomalies in weighted graphs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6119 LNAI(PART 2) :410–421, 2010.

[65] Taher H. Haveliwala. Topic-sensitive pagerank : A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 2003.

[66] Taher H Haveliwala. Topic-Sensitive PageRank Link-Based Scoring (HITS) Link-Based Scoring (PageRank) Original PageRank Topic-Sensitive PageRank Topic-Sensitive PageRank. *Search*, 2002.

[67] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1 Introduction and Motivation 2 A Ranking for Every Page on the Web. *World Wide Web Internet And Web Information Systems*, 54(1999-66) :1–17, 1998.

[68] Emmanuel Müller, Ira Assent, Patricia Iglesias, Yvonne Mülle, and Klemens Böhm. Outlier ranking via subspace analysis in multiple views of the data. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2012.

[69] Huajing Li, Zaiqing Nie, Wang-Chien Lee, Lee Giles, and Ji-Rong Wen. Scalable community discovery on textual data with relations. *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, page 1203, 2008.

[70] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. APATE : A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 2015.

[71] Sofus A Macskassy and Foster Provost. Suspicion scoring based on guilt-by-association, collective inference, and focused data access 1. *New York*, (February), 2005.

[72] Sofus a Macskassy and Foster Provost. Classification in Networked Data : A Toolkit and a Univariate Case Study. *Journal of Machine Learning Research*, 8(December 2004) :935–983, 2007.

[73] Nesreen K. Ahmed, Ryan A. Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L. Willke, and Hoda Eldardiry. A Framework for Generalizing Graph-based Representation Learning Methods. 2017.

[74] Urvesh Bhowan, Mark Johnston, and Mengjie Zhang. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics*, 42(2) :406–421, 2012.

[75] Wenqian Shang, Houkuan Huang, Haibin Zhu, Yongmin Lin, Youli Qu, and Zhihai Wang. A novel feature selection algorithm for text categorization. *Expert Systems with Applications*, 33(1) :1–5, 2007.

[76] Songbo Tan. Neighbor-weighted K-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4) :667–671, 2005.

[77] T.M. Padmaja, N. Dhulipalla, R.S. Bapi, and P.R. Krishna. Unbalanced data classification using extreme outlier elimination and sampling techniques for fraud detection. *15th International Conference on Advanced Computing and Communications (ADCOM 2007)*, pages 511–516, 2007.

[78] Bartosz Krawczyk, Michal Wozniak, and Gerald Schaefer. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing Journal*, 14(PART C) :554–562, 2014.

[79] Yanmin Sun, Mohamed S. Kamel, Andrew K.C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12) :3358–3378, 2007.

[80] C.Y. Lee, M.R. Yang, L.Y. Chang, and Z.J. Lee. A Hybrid Algorithm Applied to Classify Unbalanced Data. *Vasa*, pages 618–621, 2008.

[81] Chou Yuan Lee and Zne Jung Lee. A novel algorithm applied to classify unbalanced data. *Applied Soft Computing Journal*, 12(8) :2481–2485, 2012.

[82] Shuo Wang Shuo Wang and Xin Yao Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 324–331, 2009.

[83] Giovanna Menardi and Nicola Torelli. Training and assessing classification rules with unbalanced data. pages 1–28, 2010.

[84] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. RUSBoost : A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics Part A :Systems and Humans*, 40(1) :185–197, 2010.

[85] Mikel Galar, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera. EUSBoost : Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12) :3460–3471, 2013.

[86] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE : Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16 :321–357, 2002.

[87] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter*, 6(1) :20, 2004.

[88] Cong Rui Ji Li and Zhi Hong Deng. Mining frequent ordered patterns without candidate generation. *Proceedings - Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007*, 1 :402–406, 2000.

[89] Nathalie Japkowicz. Supervised versus unsupervised binary-learning by feedforward neural networks. *Machine Learning*, 42(1-2) :97–122, 2001.

[90] Jianping Zhang, E Bloedorn, L Rosen, and D Venese. Learning rules from highly unbalanced data sets. *Proc. Fourth IEEE International Conference on Data Mining ICDM '04*, pages 571–574, 2004.

[91] M E J Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23) :8577–8582, 2006.

[92] M E J Newman and M Girvan. Finding and evaluating community structure in networks. *Physical review E*, pages 1–16, 2004.

[93] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 10008(10) :6, 2008.

[94] Santo Fortunato and Marc Barthélemy. Resolution limit in community detection. *Pnas*, 104(1) :36–41, 2007.

[95] Zhenping Li, Shihua Zhang, Rui Sheng Wang, Xiang Sun Zhang, and Luonan Chen. Quantitative function for community detection. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 77(3), 2008.

[96] P Pons and M Latapy. Computing communities in large networks using random walks. *Lect Notes Comput Sc*, 3733 :284–293, 2005.

[97] Lovro Subelj, Nees Jan Van Eck, and Ludo Waltman. Clustering scientific publications based on citation relations : A systematic comparison of different methods. *PLoS ONE*, 11(4) :1–23, 2016.

[98] Lefebvre Etienne Blondel Vincent, Guillaume Jean-Loup, Lambiotte Renaud. The Louvain method for community detection in large networks. *Journal of Statistical Mechanics : Theory and Experiment*, 2008.

[99] Zied Yakoubi and Rushed Kanawati. LICOD : A Leader-driven algorithm for community detection in complex networks. *Vietnam Journal of Computer Science*, 1(4) :241–256, 2014.

[100] Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. Overlapping correlation clustering. *Knowledge and Information Systems*, 35(1) :1–32, 2013.

[101] Usha Nandini Raghavan, Reka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. pages 1–12, 2007.

[102] Santo Fortunato. Community detection in graphs, 2010.

[103] Fragkiskos D. Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks : A survey. *Physics Reports*, 533(4) :95–142, 2013.

[104] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. pages 1–5, 2007.

[105] Lingjian Yang, Jonathan C. Silva, Lazaros G. Papageorgiou, and Sophia Tsoka. Community Structure Detection for Directed Networks Through Modularity Optimisation. *Algorithms*, 9(4) :1–10, 2016.

[106] Youngdo Kim, Seung Woo Son, and Hawoong Jeong. Finding communities in directed networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 81(1) :1–9, 2010.

[107] Venu Satuluri and Srinivasan Parthasarathy. Symmetrizations for Clustering Directed Graphs. *Proceedings of the 14th International Conference on Extending Database Technology*, (i) :343–354, 2011.

[108] Christine Klymko, David Gleich, and Tamara G. Kolda. Using Triangles to Improve Community Detection in Directed Networks, 2014.

[109] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences of the United States of America*, 105(4) :1118–23, 2008.

[110] D Zhou, B Schölkopf, and T Hofmann. Semi-Supervised Learning on Directed Graphs. *Adv. in Neur. Inf. Proc. Syst. (NIPS)*, 17 :1633–1640, 2005.

[111] Santa Agreste, Pasquale De Meo, Giacomo Fiumara, Giuseppe Piccione, Sebastiano Piccolo, Domenico Rosaci, Giuseppe M. L. Sarne, and Athanasios Vasilakos. An empirical comparison of algorithms to find communities in directed graphs and their application in Web Data Analytics. *IEEE Transactions on Big Data*, XX(c) :1–1, 2016.

[112] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 80(1) :1–9, 2009.

[113] Jungeun Kim and J.-G. Lee. Community detection in multi-layer graphs : A survey. *SIGMOD Record*, 44(3) :37–48, 2015.

[114] Michele Berlingerio, Fabio Pinelli, and Francesco Calabrese. ABACUS : Frequent pattern mining-based community discovery in multidimensional networks. *Data Mining and Knowledge Discovery*, 27(3) :294–320, 2013.

[115] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Physical Review X*, 5(1) :1–11, 2015.

[116] Lei Tang, Xufei Wang, and Huan Liu. Uncovering groups via heterogeneous interaction analysis. *Proceedings - IEEE International Conference on Data Mining, ICDM*, (December) :503–512, 2009.

[117] Brigitte Boden, Stephan Günnemann, Holger Hoffmann, and Thomas Seidl. Mining coherent subgraphs in multi-layer graphs with edge labels. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1258–1266, 2012.

[118] Piotr Bródka, Tomasz Filipowski, and Przemysław Kazienko. An Introduction to Community Detection in Multi-layered Social Network. *Communications in Computer and Information Science*, 278 :185–190, 2013.

[119] Manel Hmimida and Kanawati Rushed. COMMUNITY DETECTION IN MULTIPLEX NETWORKS : A SEED-CENTRIC APPROACH. *Discrete and Continuous Dynamical Systems*, X(0) :1–33, 2014.

[120] Wangqun Lin, Xiangnan Kong, Philip S. Yu, Quanyuan Wu, Yan Jia, and Chuan Li. Community detection in incomplete information networks. *Proceedings of the 21st international conference on World Wide Web - WWW '12*, pages 341–350, 2012.

[121] Salvatore Ruggieri and Franco Turini. Machine Learning and Knowledge Discovery in Databases. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8190(September) :249–253, 2013.

[122] a P Dempster, N M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1) :1–38, 1977.

[123] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2) :207–216, 1993.

[124] David M Blei, Andrew Y Ng, and Michael I Jordan. 10.1162/Jmlr.2003.3.4-5.993. *Cross-Ref Listing of Deleted DOIs*, 1(4-5) :993–1022, 2000.

[125] Belo Horizonte, Arlei Silva, Mohammed J. Zaki, Wagner Meira Jr., and Wagner Meira. Mining Attribute-structure Correlated Patterns in Large Attributed Graphs. *Proceedings of the VLDB Endowment*, pages 466–477, 2012.

[126] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. Efficient Community Detection in Large Networks using Content and Links. pages 1–21, 2012.

[127] Xufei Wang, Lei Tang, Huiji Gao, and Huan Liu. Discovering overlapping groups in social media. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 569–578, 2010.

[128] Zhongying Zhao, Shengzhong Feng, Qiang Wang, Joshua Zhexue Huang, Graham J. Williams, and Jianping Fan. Topic oriented community detection through social objects and link analysis in social networks. *Knowledge-Based Systems*, 26 :164–173, 2012.

[129] Liping Jing, Michael K. Ng, and Joshua Zhexue Huang. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Transactions on Knowledge and Data Engineering*, 19(8) :1026–1041, 2007.

[130] Karsten Steinhaeuser and N.V. Chawla. Community detection in a large real-world social network. *Social Computing, Behavioral Modeling, and Prediction*, pages 168–175, 2008.

[131] Yuan Zhang, Elizaveta Levina, and Ji Zhu. Community Detection in Networks with Node Features. pages 1–16, 2015.

[132] The Anh Dang and Emmanuel Viennet. Community Detection based on Structural and Attribute Similarities. *International Conference on Digital Society (ICDS)*, (c) :7–14, 2012.

[133] David Combe, Christine Largeron, Mathias Gery, and Elod Egyed-Zsigmond. I-Louvain : An attributed graph clustering method. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9385 :181–192, 2015.

[134] Juan David Cruz, Cecile Bothorel, and Francois Poulet. Entropy based community detection in augmented social networks. *2011 International Conference on Computational Aspects of Social Networks CASoN*, (OCTOBER) :163–168, 2011.

[135] Lihua Zhou and K. Lu. Detecting Communities with Different Sizes for Social Network Analysis. *The Computer Journal*, 58(9) :bxu087–, 2014.

[136] Lian Duan, Willian Nick Street, Yanchi Liu, and Haibing Lu. Community detection in graphs through correlation. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, pages 1376–1385, 2014.

[137] Hédia Zardi, Lotfi Ben Romdhane, and Zahia Guessoum. Efficiently mining community structures in weighted social networks. *International Journal of Data Mining, Modelling and Management*, 8(1) :32, 2016.

[138] J. C. Delvenne, S. N. Yaliraki, and M. Barahona. Stability of graph communities across time scales. pages 1–6, 2008.

[139] Jure Leskovec and Christos Faloutsos. 10.1.1.61.8589.Pdf.

[140] Tianyi Wang, Yang Chen, Zengbin Zhang, Tianyin Xu, Long Jin, Pan Hui, Beixing Deng, and Xing Li. Understanding graph sampling algorithms for social network analysis. *Proceedings - International Conference on Distributed Computing Systems*, pages 123–128, 2011.

[141] N. Kim, C. Laing, S. Elmetwaly, S. Jung, J. Curuksu, and T. Schlick. Graph-based sampling for approximating global helical topologies of RNA. *Proceedings of the National Academy of Sciences*, 111(11) :4079–4084, 2014.

[142] A Arenas, J Duch, A. Fernández, and S. Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9(6), 2007.

[143] Manuel Graña and Emilio Corchado. A survey of multiple classifier systems as hybrid systems. 16 :3–17, 2014.

[144] Qiang Yang, Xindong Wu, Pedro Domingos, Charles Elkan, Johannes Gehrke, Jiawei Han, David Heckerman, Daniel Keim, Jiming Liu, David Madigan, Gregory Piatetsky-Shapiro, Vijay V Raghavan, Rajeev Rastogi, Salvatore J Stolfo, Alexander Tuzhilin, and Benjamin W Wah. 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology and Decision Making*, 5(4) :597–604, 2006.

[145] Gary King and Langche Zeng. Logistic Regression in Rare Events Data T. 2016.

[146] Steve Lawrence, Ian Burns, Andrew Back, Ah Chung Tsoi, and C. Lee Giles. Neural Network Classification and Prior Class Probabilities. (299) :299–313, 1998.

[147] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying Support Vector Machines to Imbalanced Datasets. *Lnai*, 3201 :39–50, 2004.

[148] Julián Luengo, Alberto Fernández, Salvador García, and Francisco Herrera. Addressing data complexity for imbalanced data sets : Analysis of SMOTE-based oversampling and evolutionary undersampling. *Soft Computing*, 15(10) :1909–1936, 2011.

[149] Gary M Weiss and Foster J Provost. Learning When Training Data are Costly : The Effect of ClassDistribution on Tree Induction. *J. Artif. Intell. Res. (JAIR)*, 19 :315–354, 2003.

[150] Michael Pazzani, Christoper Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. Reducing Misclassification Costs. *Icml*, pages 217–225, 1994.

[151] Jianping Zhang and Inderjeet Mani. kNN Approach to Unbalanced Data Distributions : A Case Study involving Information Extraction. *Workshop on Learning from Imbalanced Datasets II ICML Washington DC 2003*, pages 42–48, 2003.

[152] Eibe Frank and Remco R. Bouckaert. Naive bayes for text classification with unbalanced classes. *PKDD'06 Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases*, pages 503–510, 2006.

[153] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3) :368–386, 2013.

[154] A. Ramanan, S. Suppharangsan, and M. Niranjan. Unbalanced decision trees for multiclass classification. In *ICIIS 2007 - 2nd International Conference on Industrial and Information Systems 2007, Conference Proceedings*, pages 291–294, 2007.

[155] T. Eitrich, A. Kless, C. Druska, W. Meyer, and J. Grotendorst. Classification of highly unbalanced CYP450 data of drugs using cost sensitive machine learning techniques. *Journal of Chemical Information and Modeling*, 47(1) :92–103, 2007.

[156] Yves Grandvalet, Johnny Mariéthoz, and Samy Bengio. A probabilistic interpretation of SVMs with an application to unbalanced classification. *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 467–474, 2006.

[157] Claudia Plant, Christian Böhm, Bernhard Tilg, and Christian Baumgartner. Enhancing instance-based classification with local density : A new algorithm for classifying unbalanced biomedical data. *Bioinformatics*, 22(8) :981–988, 2006.

[158] Salvador García and Francisco Herrera. Evolutionary Under-Sampling for Classification with Imbalanced Data Sets : Proposals and Taxonomy. (x).

[159] José A. Sáez, Julián Luengo, Jerzy Stefanowski, and Francisco Herrera. SMOTE-IPF : Addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. *Information Sciences*, 291(C) :184–203, 2015.

[160] Hong Gao, Degen Huang, Yuansheng Yang, and Shuang Li. Chinese chunking using ESVM-KNN. *2006 International Conference on Computational Intelligence and Security, ICCIAS 2006*, 1(2) :731–734, 2007.

[161] Yoav Freund and Re Robert E Schapire. Experiments with a New Boosting Algorithm. *International Conference on Machine Learning*, pages 148–156, 1996.

[162] Zhu Yu-quan Ou and Ji-shun Chen Geng. Dynamic weighting ensemble classifiers based on cross-validation. pages 309–317, 2011.

[163] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. SMOTEBoost : Improving Prediction. *Lecture Notes in Computer Science*, 2838 :107–119, 2003.

[164] Karthik Subbian, Charu C. Aggarwal, Jaideep Srivastava, and Vipin Kumar. Rare class detection in networks. *SIAM International Conference on Data Mining 2015, SDM 2015*, pages 406–414, 2015.

[165] Juhua Hu, De-Chuan Zhan, Xintao Wu, Yuan Jiang, and Zhi-Hua Zhou. Pairwised Specific Distance Learning from Physical Linkages. *ACM Transactions on Knowledge Discovery from Data*, 9(3) :1–27, 2015.

[166] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk : Online Learning of Social Representations. 2014.

[167] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting Semi-Supervised Learning with Graph Embeddings. 48, 2016.

[168] Xiangnan Kong, Bokai Cao, Philip S. Yu, Ying Ding, and David J. Wild. Meta Path-Based Collective Classification in Heterogeneous Information Networks. pages 1–18, 2013.

[169] David Jensen, Jennifer Neville, and Brian Gallagher. Why collective inference improves relational classification. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '04*, page 593, 2004.

[170] Przemyslaw Kazienko and Tomasz Kajdanowicz. Label-dependent node classification in the network. *Neurocomputing*, 75(1) :199–209, 2012.

[171] Bokai Cao, Mia Mao, Siim Viidu, and Philip S. Yu. HitFraud : A Broad Learning Approach for Collective Fraud Detection in Heterogeneous Information Networks. (1), 2017.

[172] Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. Using ghost edges for classification in sparsely labeled networks. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08*, page 256, 2008.

[173] Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. Combining collective classification and link prediction. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 381–386, 2007.

[174] Yinfeng Zhao, Lei Li, and Xindong Wu. Link prediction-based multi-label classification on networked data. *Proceedings - 2016 IEEE 1st International Conference on Data Science in Cyberspace, DSC 2016*, pages 61–68, 2017.

[175] Priyesh Vijayan, Shivashankar Subramanian, and Balaraman Ravindran. Multi-label collective classification in multi-attribute multi-relational network data. *ASONAM 2014 - Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 509–514, 2014.

[176] Xiaoxiao Shi, Jean-Francois Paiement, David Grangier, and Philip S Yu. Learning from Heterogeneous Sources via Gradient Boosting Consensus. *Sdm*, pages 331–342, 2012.

[177] Christine Preisach and Lars Schmidt-Thieme. Ensembles of relational classifiers. *Knowledge and Information Systems*, 14(3) :249–272, 2008.

[178] Hoda Eldardiry and Jennifer Neville. An analysis of how ensembles of collective classifiers improve predictions in graphs. *Conference of Information and Knowledge Management*, pages 1567–1571, 2012.

[179] Miroslav Kubat, Robert C. Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3) :195–215, 1998.

[180] Omar Jaafor and Babiga Birregah. Multi-layered graph-based model for social engineering vulnerability assessment. In *ASONAM*, Paris, France, 2015. IEEE Computer Society.

[181] D Cardon. Le design de la visibilité. *Réseaux*, (6) :93–137, 2009.

[182] J H Kietzmann, K Hermkens, I P McCarthy, and B S Silvestre. Social media ? Get serious ! Understanding the functional building blocks of social media. *Business Horizons*, 54(3) :241–251, 2011.

[183] Stolen credit card details available for £1 each online, 2015.

[184] J Golbeck and M Rothstein. Linking social networks on the web with foaf : A semantic web case study. In *Proceedings of the 23rd national conference on Artificial intelligence*, volume 2, pages 1138–1143, 2008.

[185] Elie Raad, Richard Chbeir, and Albert Dipanda. User profile matching in social networks. *Proceedings - 13th International Conference on Network-Based Information Systems, NBiS 2010*, pages 297–304, 2010.

[186] Matthew Rowe and Fabio Ciravegna. Disambiguating identity through social circles and social data. *CEUR Workshop Proceedings*, 351 :80–93, 2008.

[187] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5) :75–174, 2010.

[188] Christian Persson, Ludvig Bohlin, Daniel Edler, and Martin Rosvall. Maps of sparse Markov chains efficiently reveal community structure in network flows with memory. (1) :1–7, 2016.

[189] Zahra Ghaffaripour, Alireza Abdollahpouri, and Parham Moradi. A Multi-objective Genetic Algorithm for Community Detection in Weighted Networks. (September) :193–199, 2016.

[190] Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F. Samatova. Anomaly detection in dynamic networks : A survey. *Wiley Interdisciplinary Reviews : Computational Statistics*, 7(3) :223–247, 2015.

[191] Joe H. Ward. Hierarchical Grouping to Optimize an Objective Function, 1963.

[192] Neil Fore and Guozhu Dong. A Contrast Pattern Based Clustering Algorithm. In *Contrast Mining : Concepts, algorithms and applications*, pages 197–2016. 2013.

[193] Hong Cheng, Xifeng Yan, Jiawei Han, and Philip S Yu. Direct Discriminative Pattern Mining Classification Effective. *2008 IEEE 24th International Conference on Data Engineering*, pages 169–178, 2008.

[194] Hua Fu Li, Hsin Yun Huang, Yi Cheng Chen, Yu Jiun Liu, and Suh Yin Lee. Fast and memory efficient mining of high utility itemsets in data streams. *Proceedings - IEEE International Conference on Data Mining, ICDM*, (December 2008) :881–886, 2008.

[195] Ying Liu, Wei-keng Liao, and Alok Choudhary. A Two-Phase Algorithm for Fast Discovery of High Utility Itemsets. pages 689–695, 2005.

[196] Show Jane Yen, Chia Ching Chen, and Yue Shi Lee. A fast algorithm for mining high utility itemsets. *Behavior Computing : Modeling, Analysis, Mining and Decision*, pages 229–240, 2012.

[197] Dong Xin, Hong Cheng, Xifeng Yan, and Jiawei Han. Extracting Redundancy-Aware Top-K Patterns. In ACM, editor, *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[198] Xiuzhen Zhang, Guozhu Dong, and Kotagiri Ramamohanarao. Information-based classification by aggregating emerging patterns. In *Intelligent Data Engineering and Automated Learning (IDEAL)*, pages 48–53, 2000.

[199] Warren S. Sarle, Anil K. Jain, and Richard C. Dubes. Algorithms for Clustering Data. *Technometrics*, 1990.

[200] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. *33rd European Conference on IR Research, ECIR 2011*, pages 338–349, 2011.

[201] Chong Wang, John Paisley, and David M. Blei. Online Variational Inference for the Hierarchical Dirichlet Process. *Icais*, 15 :752–760, 2011.

[202] Stuart P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2) :129–137, 1982.

[203] I Dhillon and D Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1) :143–175, 2001.

[204] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, (33) :452–473, 1977.

[205] Tamás Nepusz, Andrea Petróczi, László Négyessy, and Fülöp Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 77(1) :1–13, 2008.

[206] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. The paradigm-shift of social spambots : Evidence, theories, and tools for the arms race. 2017.

[207] Satu Elisa Schaeffer. Graph clustering by flow simulation. *Computer Science Review*, 1(september 1969) :27–64, 2007.

[208] Joerg Reichardt and Stefan Bornholdt. Statistical Mechanics of Community Detection. pages 1–16, 2006.

[209] M. E.J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 74(3), 2006.

[210] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 97–104, 2006.

[211] Babiga Birregah and Omar Jaafor. TVG-OSN : A time varying graph model for online social network dynamics analysis. In *European Network Intelligence Conference*, Karlskrona, Sweden, 2015. IEEE Computer Society.

[212] Linton C. Freeman. Centrality in social networks conceptual clarification, 1978.

[213] Stephen P. Borgatti, Kathleen M. Carley, and David Krackhardt. On the robustness of centrality measures under conditions of imperfect data. *Social Networks*, 28(2) :124–136, 2006.

[214] Elizabeth Costenbader and Thomas W. Valente. The stability of centrality measures when networks are sampled. *Social Networks*, 25(4) :283–307, 2003.

[215] Phillip Bonacich and Paulette Lloyd. Eigenvector-like measures of centrality for asymmetric relations. *Social Networks*, 23(3) :191–201, 2001.

[216] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson, and Gianluca Bontempi. Calibrating probability with undersampling for unbalanced classification. In *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, pages 159–166, 2016.

[217] Vincent G. Sigillito, Simon P. Wing, Larrie V. Hutton, and Kile B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, 10(3) :262–266, 1989.

[218] Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious Collective Classification. *J. Mach. Learn. Res.*, 10 :2777–2836, 2009.

[219] David Jensen, Jennifer Neville, and Michael Hay. Avoiding Bias when Aggregating Relational Data with Degree Disparity. *Proceedings of the Twentieth International Conference on Machine Learning*, 20(1) :274, 2003.

[220] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. *Data Mining and Knowledge Discovery Handbook*. 2010.

[221] He Yong Wang. Combination approach of SMOTE and biased-SVM for imbalanced datasets. In *Proceedings of the International Joint Conference on Neural Networks*, 2008.

[222] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. 2001.

[223] Sheng Chen, Haibo He, and Edwardo A. Garcia. RAMOBoost : Ranked minority oversampling in boosting. *IEEE Transactions on Neural Networks*, 21(10) :1624–1642, 2010.

[224] Hongyu Guo and Herna L. Viktor. Learning from Imbalanced Data Sets with Boosting and Data Generation : The DataBoost-IM Approach. *ACM SIGKD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1) :30–39, 2004.

[225] Y Zhou, H Cheng, and J X Yu. Graph clustering based on structural/attribute similarities. *Proceedings of the VLDB Endowment*, 2009.

[226] Haithum Elhadi and Gady Agam. Structure and attributes community detection : comparative analysis of composite, ensemble and selection methods. *Proceedings of the 7th Workshop on Social Network Mining and Analysis*, 13 :10 :1—-10 :7, 2013.

[227] Ryan A Rossi, Rong Zhou, and Nesreen K Ahmed. Deep Inductive Network Representation Learning. *Proceedings of the 3rd International Workshop on Learning Representations for Big Networks (WWW BigNet)*, 1 :8, 2018.

[228] Kyumin Lee, Brian David Eoff, and James Caverlee. Seven Months with the Devils : A Long-Term Study of Content Polluters on Twitter. *Icwsm 2011*, pages 185–192, 2011.

[229] David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML2002)*, pages 259–266, 2002.

Omar JAAFOR

Doctorat : Optimisation et Sûreté des Systèmes

Année 2018

Approches multimodales pour la détection d'anomalies sur les réseaux sociaux numériques

L'émergence des réseaux sociaux numériques tels que *Twitter*, *Facebook* ainsi que les blogs et forums ont permis pour la première fois d'effectuer des analyses à grande échelle sur le comportement humain. Alors que dans le passé, la majorité des études centrées sur l'humain était orientée terrain, ces réseaux ont permis le développement de méthodes statistiques qui analysent des traces de l'activité humaine.

Ces plateformes ont également permis aux utilisateurs malicieux d'atteindre des millions de personnes. Ainsi, des groupes radicaux utilisent les réseaux sociaux pour recruter des jihadistes. Des revendeurs de cartes de crédits volées ont également pu avoir accès à un marché important grâce aux réseaux sociaux. Aussi, des spammeurs peuvent utiliser des robots qui polluent le contenu de ces réseaux.

Cette thèse concerne la détection de ces utilisateurs malicieux qui ont un comportement atypique sur ces réseaux sociaux numériques en prenant en compte les différents modes d'interaction et de similarité (*Retweets*, Mentions, Similarité d'URL, similarité du texte, etc.). Nous avons développé des méthodes semi-supervisées ainsi que des méthodes non-supervisées (basées sur la détection de communautés) afin de détecter ces utilisateurs.

Mots clés : détection des anomalies – réseaux sociaux – intelligence collective – classification automatique – exploration de données – *boosting* (algorithmes).

Multi-Modal Approaches for Anomaly Detection in Social Networks

The emergence of on-line social networks such as Twitter, Facebook as well as blogs and forums has allowed for the first time to carry out large-scale analyzis of human interactions. Whereas in the past, the majority of human-centered studies were field-oriented, the emergence of these networks allowed the development of statistical methods which took as entry traces of human activity.

The emergence of these social networks has nevertheless been accompanied by the growth of malicious users who now have access to platforms enabling them to communicate with millions of users. Radical groups use social networks to recruit jihadists. Resellers of stolen credit cards are now able to access a large market through social networks. Also, spammers can use robots that pollute the content of these networks.

This thesis concerns the detection of malicious users who behave in an anomalous manner in on-line platforms using their different modes of interactions and similarity (Retweets, Mentions, URL similarity, text similarity). We have developed semi-supervised methods as well as unsupervised methods (based on community detection) to detect malicious users.

Keywords: anomaly detection – social networks – collective intelligence – automatic classification – data mining – boosting (algorithms).

Thèse réalisée en partenariat entre :

RÉGION CHAMPAGNE ARDENNE

bpifrance

utt UNIVERSITÉ DE TECHNOLOGIE TROYES

Ecole Doctorale "Sciences pour l'Ingénieur"