Apprentissage automatique appliqué au Bitcoin

Théo Rabut
Encadré par Rémy Cazabet

Résumé : Ce document expose les travaux réalisés durant mon stage de fin d'études dans le cadre du Master en Intelligence Artificielle de l'université Lyon 1. Les travaux réalisés sont centrés autour des notions de classification, d'extraction de caractéristiques dans des réseaux complexes et d'apprentissage de représentation. Le réseau étudié correspond à une modélisation des données historiques de transactions du Bitcoin. Nous cherchons à caractériser les adresses et entités qui participent à ce réseau. Nos travaux se basent sur le fait qu'une partie de ces individus sont identifiés comme participant à des tâches spécifiques.

Mots-Clés : Graphe du Bitcoin, classification d'entités, classification d'adresses, plongement de noeud, apprentissage de représentation

Abstract : This report exposes the work done during an internship as part of a Master 2 in Artificial Intelligence in the University Lyon 1. The work is centered on notions such as classification, feature engineering in complex networks and representational learning. The complex network studied correspond to a model of Bitcoin Transactional Historical Data. We aim at characterizing Bitcoin's Addresses and Entities which are known to play a role in specifics activities in this complex networks.

Keywords : Bitcoin graph, entity classification, address classification, node embedding, representationnal learning

Master Intelligence Artificielle
Université Claude Bernard Lyon 1
17 Juin 2020

Table des matières

1 Introduction **2**
1.1 Bitcoin . . . 2
1.1.1 Bitcoin et BlockChain . . . 2
1.1.2 Graphe de Transaction . . . 3
1.1.3 Bitcoin et Anonymat . . . 4
1.2 Techniques et contexte du stage . . . 5
1.2.1 Le LIRIS . . . 5
1.2.2 Environnement de travail . . . 5
1.2.3 Contraintes . . . 5
1.3 État de l'art . . . 7
1.4 Problématique . . . 11

2 Développement **12**
2.1 Classification d'adresses . . . 12
2.1.1 Classes disponibles . . . 12
2.1.2 Extraction de caractéristiques . . . 13
2.1.3 Technique d'apprentissage automatique . . . 14
2.1.4 Résultats . . . 16
2.2 Classification d'entités . . . 17
2.2.1 Extraction de caractéristiques . . . 18
2.2.2 Node Embedding . . . 19
2.3 Technique d'apprentissage automatique . . . 22
2.3.1 Résultats . . . 23

3 Conclusion **26**

1 Introduction

1.1 Bitcoin

1.1.1 Bitcoin et BlockChain

La chaîne de blocs (parfois appelée ici par sa version anglophone, Blockchain) est une technologie distribuée régie par ses utilisateurs. Elle est développée autour de la cryptographie et de la transmission d'information pour jouer le rôle d'une base de données immuable. La création d'un bloc est réalisée par des utilisateurs particuliers appelés mineurs. Il existe plusieurs approches pour la création d'un bloc mais la plus célèbre d'entre-elles est la preuve de travail. Dans la Blockchain du Bitcoin, ce travail consiste en un calcul, dont la difficulté change en fonction du temps que le réseau a mis pour produire un nombre fixe de blocs (2016). Ce calcul nécessite plusieurs entrées pour produire un identifiant (hash). L'une de ces entrées est l'identifiant du bloc le plus récent qui a été reçu par le mineur. Ce mécanisme assure la validité des blocs calculés et introduits. En effet, une fausse preuve de travail sera invalidée par le calcul du bloc suivant par le réseau et donc rejetée par ce dernier. De plus l'immuabilité des données que la Blockchain offre, est une caractéristique importante. Elle est due à l'impossibilité de modifier le contenu d'un bloc une fois qu'il a été validé par le réseau, sous peine d'avoir, dans les blocs suivant les modifications, des incohérences dans les preuves de travail produites. Les blocs constituent des zones de stockage immuables et validées par le réseau. Le contenu de ces blocs varient d'une chaîne à l'autre et les transactions qui constituent le fonctionnement de la crypto-monnaie étudiée sont encodées et incluses dans le contenu de ces blocs. En effet, les mineurs sélectionnent dans un registre des transactions qui ne se sont pas encore intégrées dans des blocs. Le contenu du bloc va en suite être utilisé (en plus d'autres éléments) dans la preuve de travail. Lorsqu'un bloc est validé, le mineur qui l'a produit, obtient une rémunération. L'augmentation de la puissance de calcul (Loi de Moore) accélère la génération de blocs et donc augmente de manière exponentielle la rémunération des mineurs. Or cette rémunération est décroissante : plus la chaîne est longue, moins la rémunération est élevée.

1.1.2 Graphe de Transaction

Un utilisateur du Bitcoin possède une clé privée qui lui permettra de générer des clés publiques et adresses publiques dans le but de recevoir ou de faire des paiements. Cette clé privée est un nombre entre 1 et 2^{256} choisi au hasard (256 bits), encodée par 51 caractères en base 58 et commençant par 5, K ou S, par exemple :

$Kzczf8E4oq8MLakhRS479gpZpSe2e6u2xErKHQNqpeFMPEK4irtc$

Il existe donc environ 1.16×10^{77} clés privées différentes possibles. C'est ce qui rend impossible (statistiquement) la génération aléatoire d'une clé privée qui a déjà été attribuée. De cette clé privée, des clés publiques peuvent être générées à partir de l'algorithme ECDSA. Cet algorithme est basé sur la cryptographie et permet de certifier qu'une clé publique appartient à une clé privée sans pouvoir produire le calcul inverse et déterminer une clé privée à partir d'une clé publique. Ces clés publiques sont utilisées pour générer des adresses qui serviront à recevoir et envoyer des paiements. Les adresses sont un hachage (SHA-256 et RIPEMD-160) de la clé privée. Les adresses sont donc utilisées pour signer des transactions, et la validité d'une signature est donnée par les couples clés publiques, clés privées. Une transaction est un ensemble d'informations représentant des paiements. Elle est notamment composée par une liste d'entrées (adresses et montants), une liste de sorties (adresses et montants) et une date correspondant à la date du bloc dans lequel la transaction a été incluse (d'autres informations constituent aussi une transaction mais n'ont pas d'intérêt particulier dans cette mise en contexte). Une transaction simplifiée est représentée par la figure 1.1. On notera l'absence des montants de chaque participation ou le manque d'information globale. Cette figure a pour but d'offrir une introduction à la modélisation du réseau par un graphe.

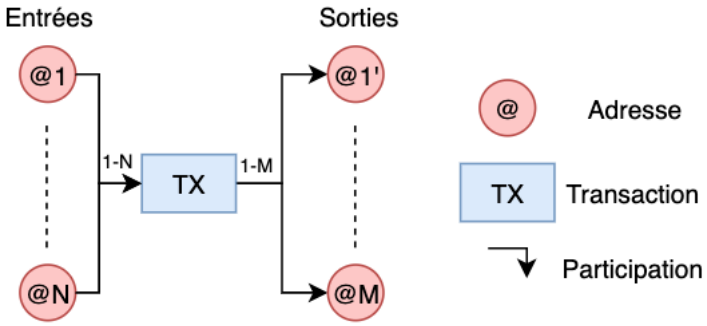

FIGURE 1.1 – Transaction simplifiée.

La modélisation d'une transaction présentée dans la figure 1.1 et la disponibilité de toute les transactions réalisées en Bitcoins permettent d'apporter une modélisation du graphe de transactions (figure 1.2) représentant l'activité globale du réseau du Bitcoin.

Le graphe représenté par la figure 1.2 est un graphe bipartie constitué par un ensemble d'adresses et un ensemble de transactions. Certaines particularités du Bitcoin rendent ce graphe complexe et difficile à analyser. On peut citer l'obli-

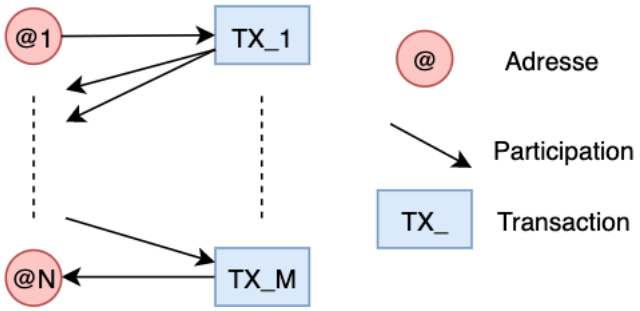

FIGURE 1.2 – Graphe de transactions et d'adresses.

gation de dépenser un montant reçu dans sa totalité (et recevoir un montant en tant que sortie pour l'excédant) ou la possibilité pour un utilisateur de posséder plusieurs adresses sans lien publiquement direct.

Une des caractéristiques du réseau du Bitcoin est qu'il est en perpétuel changement. En effet, des adresses sont très régulièrement générés et des nouvelles transactions sont produites de plus en plus fréquemment. On peut noter que l'immuabilité de la Blockchain apporte une caractéristique particulière aux changements qui sont produits sur le graphe de transactions. Ces modifications sont :

— L'ajout d'adresses (ajout de noeud)
— L'ajout de transactions (ajout d'arêtes/noeuds ou hyper-arêtes)

Il n'y aucune suppression d'arête ou de noeud possible. Cette propriété apporte une valeur non négligeable quant à la complexité et à la conception des algorithmes qui vont pouvoir être utilisés sur ce graphe.

1.1.3 Bitcoin et Anonymat

L'anonymat du Bitcoin n'est en fait qu'un pseudo-anonymat. Les utilisateurs sont liés à des adresses publiques et ce lien n'est pas disponible publiquement. Malgré cela, il est possible de déterminer le lien entre un utilisateur et une adresse en produisant une transaction vers un utilisateur et en récupérant les données après validation de la blockchain. Nous pouvons attribuer la participation de l'utilisateur à l'ensemble des transactions dans laquelle l'adresse, préalablement liée, apparaît. Ceci constitue une des principales failles dans l'anonymat du Bitcoin.

Un grand nombre d'utilisateurs utilisent les Bitcoins en passant par des logiciels divers et variés. L'utilisation du Bitcoin se caractérise par une mise en place complexe. Par exemple, un service bien connu est celui des portefeuilles virtuels, ceux-ci vont gérer les mécanismes particuliers du protocole pour que l'interface du réseau (actions nécessaires à son utilisation) soit intuitive et à la portée des non-initiés. Ces logiciels tierces peuvent parfois, volontairement ou à leurs insus, transmettre des informations (e.g. comportements particuliers, données privées) de leurs utilisateurs. Le pseudo-anonymat du Bitcoin n'est donc pas une garantie, la conception de son protocole assure, à minima, un certain

anonymat mais il existe des failles exploitables. La communauté scientifique s'applique à étudier à quel point des solutions sont capables de remettre en cause cet anonymat, notamment grâce aux avancées les plus récentes en apprentissage automatique.

1.2 Techniques et contexte du stage

1.2.1 Le LIRIS

Ce stage a été réalisé au laboratoire LIRIS. Ce laboratoire de recherche est supporté par l'université Claude Bernard Lyon 1, l'université Lyon 2, l'INSA Lyon et le CNRS. J'ai eu l'occasion d'intégrer l'équipe "Data Mining et Machine Learning (DM2L)" dont le champ de compétence est la science des données. Ce champ de compétence est en adéquation avec ma formation en Intelligence Artificielle puisque plusieurs des mouvements actuels de l'intelligence artificielle sont dirigés par la donnée. J'ai pu participer au projet BITUNAM, Bitcoin User Network Analysis and Mining, qui a pour but d'analyser les transactions réalisées en Bitcoin (ou d'autres crypto-monnaies majeures) pour les caractériser. Ce projet est financé par l'Agence Nationale de Recherche pour 3 ans. J'ai eu l'occasion de travailler avec mon encadrant et un doctorant dont le sujet est proche du mien.

1.2.2 Environnement de travail

Les "notebook" sont des environnements à la mode dans le monde de la programmation. Un notebook est un environnement python qui est composé d'une suite de cellule qui peuvent s'exécuter indépendemment. L'état de l'environnement python est modifié par l'exécution des cellules, ce qui est pratique pour garder en mémoire des données. J'ai eu à disposition une machine distante et un accès SSH. JupyterHub offre une expérience de partage de ressources et d'environnement multi-utilisateurs, accessible par tout les navigateurs (VPN-Lyon1). J'ai pu avoir accès à cet environnement à domicile.

1.2.3 Contraintes

Les données correspondant à l'historique de transactions du Bitcoin sont massives. L'historique de transactions sur lequel j'ai pu travailler est constitué de :

- $\sim$ 170M d'adresses
- $\sim$ 150M de transactions

Sous deux formats :

- JSON, un objet JSON pour chaque transaction avec les détails pour chaque entrée et pour chaque sortie.
- CSV, plusieurs formats de fichier CSV. Ces fichiers ont été utilisés pour alimenter une base de donnée Neo4J. Certains de ces fichiers offrent une manipulation plus rapide que les fichiers JSON pour des tâches spécifiques.

Ces données correspondent à l'historique sur la période 01/03/2009 - 09/08/2016. La notoriété du Bitcoin a connu un pic en 2017 et depuis la quantité de transactions produites par le réseau est beaucoup plus importantes. En effet, la figure 1.3 (www.blockchain.com) nous montre l'évolution du nombre de transactions confirmées par jours.

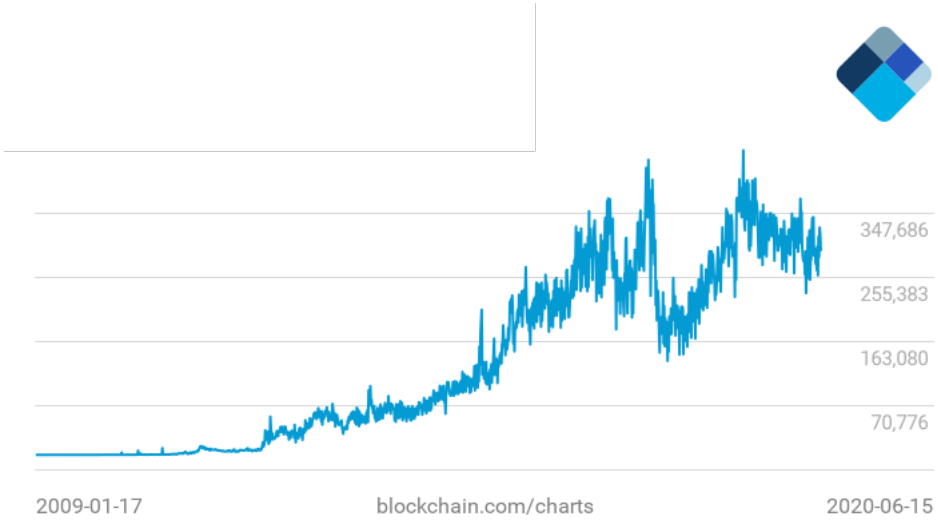

FIGURE 1.3 – Nombre de transactions confirmées par jours.

La figure 1.4 (www.blockchain.com) nous montre l'évolution de la taille en octet que représente une des versions les plus condensées de l'historique du Bitcoin.

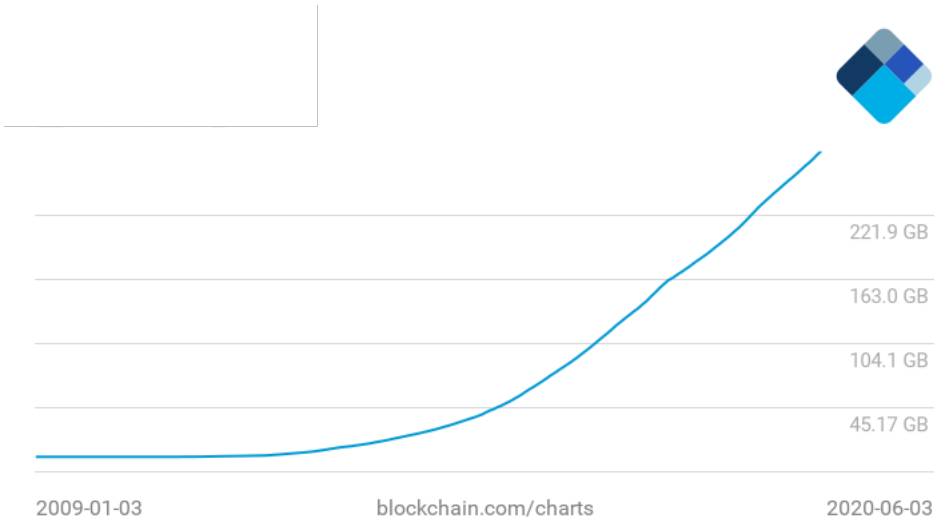

FIGURE 1.4 – Taille de la Blockchain Bitcoin.

Le nature et la quantité des données que j'ai eu à manipuler nécessitent donc une architecture particulière. En effet, j'ai été confronté à des problématiques

algorithmiques que je n'ai pas eu l'occasion de voir auparavant. J'ai du garder en mémoire vive des fichiers de plus 40 Go ou passer par des tables de hashage (natif pour certaines structures python) pour réduire la complexité de certains algorithmes. Certains de ces problèmes ne m'étaient pas totalement inconnu mais la nécessité de mettre en oeuvre des astuces pour manipuler efficacement des quantités importantes de données l'était bel et bien.

Les travaux réalisés dans le cadre de Bitunam, offrent à l'équipe un accès à un supercalculateur dédié aux travaux informatiques extrêmes et de haute performance. Le machine distante qui m'a été fourni a été suffisante. Certaines techniques aurait pu être compliqué à utiliser sur la machine distante que j'ai eu à disposition (l'apprentissage profond notamment), et donc l'accès à un outil comme le supercalculateur Jean Zay aurait pu être une ressource indispensable.

1.3 État de l'art

Regroupement d'adresse et entités
Les attaques au pseudo-anonymat du Bitcoin sont nombreuses et leurs angles varient selon les disciplines. Une des attaques les plus effectives est sans doute celle du regroupement d'adresses (Clustering) par heuristique [1]. Le regroupement d'adresses permet de lier un groupe d'adresses entre elles pour représenter une unique entité autoritaire (Cluster) sur celles-ci. L'heuristique la plus fiable et la plus utilisée est celle de la dépense commune [2]. Deux ou plusieurs adresses qui ont participé en entrée d'une transaction appartiennent au même utilisateur. Les heuristiques plus récentes sont toutes dépendantes de la dépense commune (sur-couche). La fiabilité de cette première est analysée et discutée par [3], qui la décrit comme étant d'une fiabilité très importante.

Réponse de la communauté et intérêt pour nos travaux
Cependant il existe, de la part de la communauté des utilisateurs, des tentatives d'amélioration de l'anonymat du protocole Bitcoin (CoinJoin, ZeroCoin, ...). CoinJoin [4] en est une qui a été conçue pour agréger plusieurs transactions en une seule sans porter à défaut le protocole sous-jacent. Coinjoin est un essai cherchant à contrer l'heuristique de la dépense commune mais nécessite l'utilisation d'un tiers de confiance qui se chargera du lien entre ses utilisateurs, il peut être de nature purement communicative (chat IRC, email, etc) ou être un logiciel tiers. En principe, Coinjoin une couche supplémentaire du protocole Bitcoin. Mais l'utilisation de celle-ci apporte un compromis :

— Contrer l'heuristique de la dépense commune
— Passer par un logiciel tiers (basé sur la confiance en celui-ci)

Coinjoin affecte l'utilisation de l'heuristique de la dépense commune mais affaibli trop peu nos travaux. Le compromis impliqué par l'utilisation de Coinjoin induit qu'une démocratisation s'avère complexe.

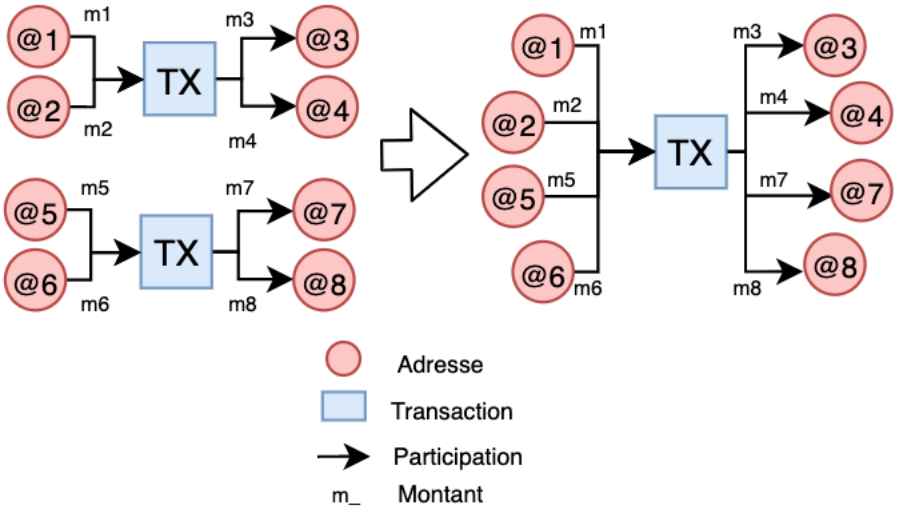

FIGURE 1.5 – CoinJoin - Agrégation de deux transactions.

Association d'adresses et d'utilisateurs

Lorsqu'une adresse est connue comme étant régie par un utilisateur particulier, il est possible d'associer le cluster (groupe d'adresses préalablement liées par heuristiques) à cet utilisateur particulier [1]. Le site WalletExplorer.com [5] présente une quantité importante d'adresses reliées à des utilisateurs. Ces utilisateurs sont classifiées en 5 catégories :

- Exchanges : places d'achat et de vente de Bitcoin.
- Pools : organisations qui utilisent des ressources de ses participants pour miner des Bitcoins.
- Old/Historic : données historiques, organisations qui n'exercent plus.
- Services : organisation fournissant des services de natures différentes (e.g. Portefeuille virtuel, Service de paiement, ...).
- Gambling : place de jeux.

Il est intéressant de noter que ce site utilise l'heuristique de dépense commune pour associer cet ensemble massif d'adresses ($\sim$ 20 millions) à des utilisateurs. Il n'y a aucune adresse qui est attribuée à plusieurs utilisateurs. Cette information ne permet pas de prouver l'efficacité de cette heuristique mais renforce l'idée que plusieurs adresses apparaissant en entrée dans une même transaction appartiennent à une seule entité. Il est possible d'extraire des nouveaux éléments à catégoriser (couple adressse - classe) sur des forums ou d'autres sites qui présentent ces informations. Malgré qu'il est évidement important d'utiliser le maximum d'éléments pour entraîner un modèle de classification, il est indispensable d'utiliser un système de classe harmonieux où chaque classe est relativement bien fournie.

Classification des adresses

La classification d'adresse est un exercice ayant pour but de prédire la classe des adresses Bitcoin. Connu comme une tâche d'apprentissage supervisé, la classification se base sur la définition de modèles de prédiction. Ces modèles sont

modifiés et entraînés grâce à des observations déjà classifiées dans le but de prédire la classe d'une nouvelle observation. La classification d'adresses est une tâche qui est beaucoup étudiée [6–9]. Pour produire un apprentissage performant, les différents modèles se basent sur les caractéristiques des adresses. Ces caractéristiques peuvent être obtenus par diverses méthodes. Nous nous intéresserons à certains d'entre elles.

Création du graphe d'entité

Un utilisateur du Bitcoin peut posséder plusieurs adresses (sans limite) donc les adresses perdent de leurs pouvoirs représentatif quant aux entités qui les régissent. L'idée est donc d'agréger ces adresses dans le graphe pour modéliser des transactions entre entités et non plus entre pseudonymes (adresses).

Le graphe d'entité est construit à partir du graphe bipartie en utilisant l'heuristique de dépense commune. Il est possible d'utiliser des techniques plus avancées comme la détection de communauté [10]. Il est intéressant de noter que des algorithmes s'appliquant au réseau Bitcoin doivent être adaptées à un réseau complexe. On préférera des algorithmes incrémentaux puisqu'ils est impossible de retirer une adresse ou un transaction du réseau Bitcoin une fois qu'elle a été validée. L'heuristique utilisée est fiable et nous permet, en plus d'agréger les noeuds adresses en noeud entité, de transformer les noeuds transactions en arêtes transactions. Cette transformation est explicitée par la figure 1.6. Cette figure montre que deux adresses (@1 et @2) en entrée seront regroupé en un cluster (C1) tandis que les adresses 3 et 4 ne peuvent pas être regroupées donc il y aura deux clusters en plus (C2 et C3))

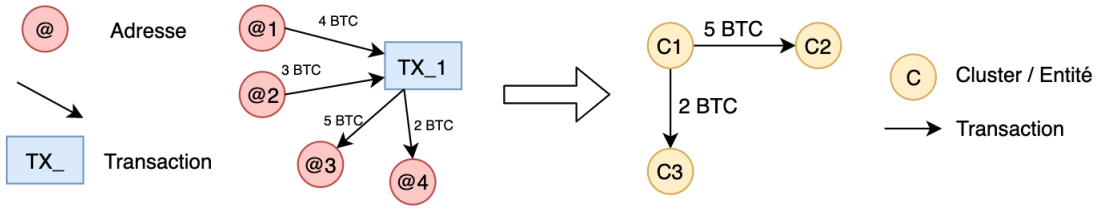

FIGURE 1.6 – Graphe Bipartie vers Graphe d'entité.

Classification de cluster (ou entité)

La classification de noeud dans un graphe est accompagnée d'une extraction de caractéristiques. C'est un travail fastidieux qui doit être produit en parallèle avec l'acquisition de savoir expert sur le contexte du graphe. Ces caractéristiques sont rarement présentées de manière exhaustive dans la littérature. Certains travaux se sont concentrés sur la quantité pour ensuite affiner leurs modèles [11], d'autres utilisent des caractéristiques apportées par l'agrégation d'adresses en clusters [12] [13] [14] [15]. Certains travaux cherchent à catégoriser les caractéristiques utilisées pour la classification en se basant sur leurs méthodes d'extraction ou les sources de leurs extractions par exemple. Ces catégories de caractéristiques peuvent être complémentaires ou parfois redondantes, il est in-

téressant de chercher à identifier ces liens entres les caractéristiques pour éviter de passer à côté d'une information ou de surcharger nos modèles. Un motif (dans la théorie des graphes) est un sous-ensemble d'un graphe, composé de noeuds et d'arêtes. La fouille de motif est une méthode qui consiste à énumérer tout les types de motifs présent dans un graphe et à les compter. Cette approche est beaucoup utilisée pour caractériser des noeuds dans un graphe comme celui du Bitcoin [11, 15, 16]. Mais le graphe du Bitcoin est un réseau complexe de plusieurs centaines de millions de noeuds, de ce fait les techniques employées pour la fouille de motifs se doivent d'être performantes et réalisables. Malgré cela, elle est une approche intéressante puisqu'elle permet de modéliser des comportements particuliers quant aux utilisations du Bitcoin.

Embedding

L'embedding est une technique issue du monde du traitement du langage naturel et rentre dans le cadre de l'apprentissage de représentation. Le plongement est une opération qui associe à des arêtes, à des noeuds ou même à des graphes entiers, des valeurs dans un autre espace (euclidien généralement). On parle alors de graph embedding pour un plongement de graphe ou node embedding pour un plongement de noeud. Ces opérations peuvent, avec des manipulations particulières, prendre en compte des propriétés tout aussi particulières. Ces propriétés dans le milieu des graphes sont issues de trois sources [17] :

- Proximité dans le graphe (distance)
- Équivalence structurelle
- Caractéristiques des noeuds

Certains travaux [18, 19] ont produit une taxonomie des différentes méthodes, techniques et sources d'informations utilisées par les algorithmes d'embedding. Ces travaux permettent, entre autres, d'avoir un premier aperçu sur le type de représentations qui peuvent être apprises et sur quel type de graphe. On peut évoquer la séparation des techniques en 3 sous-groupes :

- Les techniques basées sur les marches aléatoire et inspirées du milieu du traitement du langage naturel.
- Les techniques basées sur l'apprentissage profond et par convolution.
- Les techniques basées sur la factorisation de matrices (d'adjacence ou Laplacienne).

Les graphes générés à partir de l'historique du Bitcoin sont complexes et certains ne sont pas propices (hypergraphe, dirigé, pondéré, taille, ...) à l'utilisation de la majorité des techniques d'embedding. Cependant, il est possible de réduire, d'agréger ou de transformer un graphe pour obtenir une représentation des noeuds grâce à une approche par apprentissage. La tâche de classification est motivée par la démonstration de l'efficacité des caractéristiques utilisées. Ces caractéristiques peuvent utilisées pour d'autres tâches comme la détection de fraude [20, 21] ou la prédiction de la valeur du Bitcoin [22].

1.4 Problématique

La motivation du travail réalisé durant ce stage est apportée par les notions d' Apprentissage de représentation pour une caractérisation de noeuds dans un réseau dynamique et complexe comme celui du Bitcoin. Il sera expliqué dans une première partie le travail réalisé pour caractériser et classifier les adresses en se basant sur des travaux existants. Puis dans un second temps, sera présenté le travail réalisé pour la caractérisation des entités du Bitcoin et la mise en place d'un protocole de classification de ces entités. L'extraction de caractéristiques des entités a été décomposée en plusieurs catégories selon leurs sources ou leurs méthodes d'extraction avec pour objectif de déterminer l'importance de chacune d'elles dans la prise de décision de la prédiction.

2 Développement

2.1 Classification d'adresses

Il existe beaucoup de travaux qui traitent des sujets similaires à la classification des adresses du Bitcoin. En effet, les fraudes, le blanchiment d'argent et les anomalies sont célèbres dans le Bitcoin, donc ces sujets captent une attention particulière de la communauté scientifique. Pour ce travail, je me suis basé sur l'état de l'art pour avoir une approche performante en reproduisant certaines méthodes. Cependant, les individus déjà classifiés que nous avons ne sont pas considérés comme des benchmarks il est donc difficile de reproduire des expériences. La performance de la classification d'adresse possède une importance secondaire. En effet, les résultats de cette dernière, ont pour but d'être utilisés pour la classification des entités qui sera présentée dans une partie suivante de ce rapport.

Dans un premier temps, je me suis familiarisé avec le contexte Bitcoin. Pour ce faire, j'ai analysé les données déjà classifiées qui ont servi pour l'entraînement des différents modèles.

2.1.1 Classes disponibles

Les différents travaux que nous avons étudié se basent tous sur WalletExplorer.com [6], [15]. Chacun de ces travaux prend en compte une période différente. Il est cohérent qu'un travail produit en 2017 ne prenne pas en compte les données transactionnelles de 2018, mais il serait intéressant d'avoir une forme de consensus sur l'utilisation d'un ensemble défini d'individus déjà classifié.

Il existe plusieurs sources pour obtenir des individus classifiés. On notera :

— Walletexplorer.com.
— Divers forums, parfois des utilisateurs partagent des adresses, ou notifient qu'une adresse appartient à un escroc.
— Certains scientifiques partagent les informations et classes qu'ils ont pu obtenir au fil de leurs travaux [1].

Les classes résultantes sont au nombre de 12 : Abuse, Faucet, User, Wallet, Old/historic, Market, Exchanges, Mining, Gambling, Thief, Mixing Service, Services. Certaines de ces classes sont redondantes (e.g. : Abuse et Thief) et certaines sont sous-représentées ou n'ont aucun intérêt dans une classification multi-classe (Old/Historic). Nous nous intéresserons donc à produire une harmonisation de ces classes pour passer de 12 classes à 5 classes :

— Exchanges, zone d'échange de (crypto-)monnaie.
— Services, multitude de service différents (service de paiement, de portefeuille virtuel, ...)
— Market, zone d'achat de biens ou de services
— Mining, adresses participantes à l'activité de minage de blocs
— Gambling, zone de paris et de jeux d'argent.

Les nombre d'individu est spécifié dans la table 2.1. "Services" est surreprésenté puisque cette classe est constituée de beaucoup de services différents, notamment les services de porte-feuille qui représente une proportion non-négligeable de ces adresses.

Classes	Nombres d'adresses disponibles
Services	13858384
Exchanges	5127995
Market	2257394
Gambling	2016844
Mining	87990
Total	23348607

TABLE 2.1 – Représentation des classes disponibles.

2.1.2 Extraction de caractéristiques

L'apprentissage supervisé est souvent accompagné d'une extraction de caractéristiques fastidieuse. La liste suivante présente les caractéristiques calculées.

- Nombre de participations en entrée dans des transactions
- Nombre de participations en sortie dans des transactions
- Nombre d'adresses uniques qui ont été en entrée lorsque l'adresse considérée était en sortie (prédécesseurs)
- Nombre d'adresses uniques qui ont été en sortie lorsque l'adresse considérée était en entrée (successeurs)
- Nombre d'adresses uniques qui ont été en entrée lorsque l'adresse considérée était en entrée (Adresses soeurs en entrée)
- Nombre d'adresses uniques qui ont été en sortie lorsque l'adresse considérée était en sortie (Adresses soeurs en sortie)
- Nombre de schéma de "Réciprocité" (figure 2.1), le nombre de successeurs (temps t0) qui sont aussi prédécesseurs (temps t $>$ t0)
- Nombre de schéma d'anti-réciprocité (figure 2.1), le nombre de prédécesseurs (temps t0) qui sont aussi des successeurs (temps t $>$ t0).

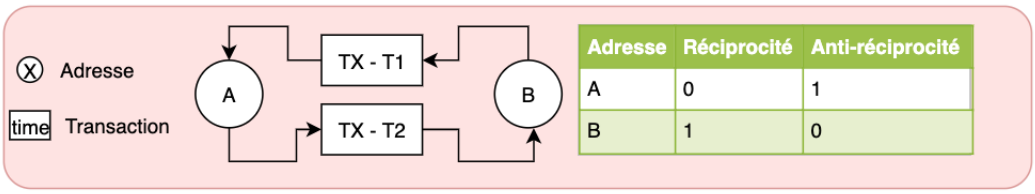

FIGURE 2.1 – Pattern de réciprocité et d'anti-réciprocité.

Ces schémas, ou motifs, ont une notion temporelle, un schéma de réciprocité, indique qu'une adresse B a envoyé des bitcoins à A à un temps T1 et que A a envoyé des bitcoins à B à un temps T2. Cela veut dire qu'un des successeurs de B sera un de ses prédécesseurs. Inversement

pour l'anti-reciprocité, où l'on cherche à dire si un des prédécesseurs de A sera un de ses successeurs. Nous comptons alors le nombre de motifs ainsi calculés.
- Balance, nombre de Bitcoins reçus, moins le nombre de Bitcoins envoyés.
- Nombre de Bitcoin envoyés
- Nombre de Bitcoin reçus
- Nombre de dollars envoyés
- Nombre de dollars reçus
- Durée de vie (secondes) de l'adresse (Date dernière action - date première action)

Ces caractéristiques sont extraites via plusieurs sources. Une première est induite par les informations relatives au protocole Bitcoin (e.g. durée de vie d'une adresse). D'autres caractéristiques viennent de l'analyse du graphe représentant le réseau du Bitcoin.

Certaines caractéristiques comportent des valeurs en Bitcoin. Il est intéressant de noter qu'une adresse qui a envoyé 1000 Bitcoins en 2009 et une autre qui a envoyé 1000 Bitcoin en 2016, n'ont pas manipulé la même somme en monnaie courante. Il est donc intéressant de produire une transformation des valeurs en Bitcoin vers des valeurs en Dollars. Pour cela, nous avons pu récupérer des données extraites d'une zone d'échange bien connue : Bitstamp. Un historique des transactions internes à la zone d'échange est public. J'ai donc récupéré tous les échanges de Bitcoin contre des Dollars pour extraire un prix du Bitcoin en dollars sur la période étudiée.

2.1.3 Technique d'apprentissage automatique

La classification d'adresse a été considéré comme une tâche d'arrière plan. Malgré cela, il est indispensable de construire un protocole clair et efficace pour établir un outil de classification performant sous les contraintes que nous avons et les caractéristiques disponibles. Scikit-learn [23] est un module python en sur-couche du module SciPy qui est une bibliothèque de module python à usage scientifique. Scikit-learn est open-source et est maintenu par une communauté compétente pour être à la pointe des performances en apprentissage automatique. C'est ce module que nous utiliserons pour toutes les tâches de classification.

Pour établir ce protocole, nous nous sommes inspirés de la littérature sur le sujet [6–9].

Puis, dans l'objectif de définir quels outils sont les plus performants, nous nous sommes intéressés aux métriques classiques utilisées en apprentissage automatique :

- Rappel
$$Rappel = \frac{Vrai\,positif}{Vrai\,positif + Faux\,Negatif}$$
- Précision
$$Prcision = \frac{Vrai\,positif}{Vrai\,positif + Faux\,positif}$$

— Accuracy

$$Accuracy = \frac{Vraipositif + VraiNegatif}{Nombre\ d'individu}$$

— F-Score

$$F\_score = 2 \times \frac{Prcision \times Rappel}{Prcision + Rappel}$$

Ces mesures se basent sur une matrice de confusion (figure 2.2), dans laquelle les colonnes représentent la vraie classe d'un individu et les lignes représentent les classes prédites.

	Classe Positive	Classe Négative
Classe Predite Positive	Vrai Positif	Faux Positif
Classe Predite Négative	Faux Négatif	Vrai Négatif

FIGURE 2.2 – Matrice de Confusion binaire.

La figure 2.2 est une matrice de confusion pour une tâche de classification binaire. Pour une classification multi-classe, on doit créer une table de confusion pour chacune des classes et construire les mesures par rapport à chacune des tables. La classification multi-classe implique un choix dans l'utilisation de ces métriques. Il existe plusieurs approches pour adapter ces mesures aux problèmes multi-classe. Une d'entre elles est de faire une moyenne pondérée par le nombre d'individu de chaque classe sur la population totale. Une moyenne simple ne prend pas en compte l'équilibre entre chaque classe. Les formules présentées ci-dessus ont été adaptées à la classification multi-classe pour prendre en compte le non-équilibre de la représentation de chaque classe.

Un point important de l'évaluation d'un outil est l'interprétabilité des résultats en rapport à la tâche. Par exemple, l'apprentissage profond est un outil de généralisation performant, mais n'apporte aucune explication quant à l'importance des propriétés utilisées pour la classification. À l'inverse, les techniques dérivées des arbres de décision possèdent une capacité de généralisation moins importante mais offrent des informations supplémentaires sur la prise de décision.

Nous nous intéresserons aux différentes mesures notées ici pour l'évaluation de nos modèles. En effet, nous ne possédons pas d'intérêt particulier à prendre en compte la proportion de résultats positifs réels, qui a été identifiée correctement (Rappel) ou la proportion d'identifications positives qui était effectivement correcte (Précision).

L'évaluation d'un apprentissage se fait en séparant l'ensemble des données en deux sous-groupes :

— Données d'apprentissage

— Données de test

Le premier sous-ensemble sera utilisé pour que le modèle apprenne à classifier correctement des individus. Un modèle est composé d'une quantité plus ou moins importante de paramètres (dépendante du modèle en question). Durant l'apprentissage, les paramètres seront optimisés pour correspondre aux données utilisées à cette étape. L'ensemble de tests est fait pour vérifier la qualité de l'apprentissage. Il est nécessaire d'avoir des données qui n'ont pas déjà été présentées au modèle. Ce concept est appelé validation croisée. Il existe plusieurs techniques de validation croisée mais nous en utiliserons une en particulier. La technique du "K-fold" permet d'éviter d'incorporer un biais dans la séparation de l'ensemble des données en deux échantillons (apprentissage / test). Cette technique consiste à séparer l'ensemble des données en K parties. Chaque partie va jouer le rôle de la validation une fois lorsque toutes les autres parties vont être utilisées pour l'apprentissage. Il faut donc un nombre K particulier. Si K est trop grand, les différentes parties seront trop petites pour la validation. Si K est trop petit, le biais peut toujours être induit par la séparation. Les résultats seront calculés durant chaque validation, on étudiera principalement, l'écart type et la moyenne de ces résultats.

2.1.4 Résultats

Une approche naïve pour cette classification serait de classifier toutes les adresses dans la classe la plus représentée. Ici, il s'agirait de la classe "Services". De cette manière, on classifierait correctement 59% des individus. Ce qui nous intéresse est donc d'obtenir des scores plus élevés qu'une approche naïve. Nous nous inspirons des travaux existant sur la classification d'adresse [6, 9, 15]. Les deux modèles utilisés sont :

— Adaptative Boosting (Adaboost).

— forêt aléatoire décisionnelle (Random Forest).

Tous les articles cités utilisent plus de deux outils de classification. Nous avons noté les performances de chacun et choisi ceux pour qui nous portons un intérêt particulier quant à leurs utilisabilités. En effet, certains articles utilisent des techniques, comme l'apprentissage profond [24] ou de GradientBoosting [9]. Ces outils seront utilisés dans la suite du stage. En effet, un apprentissage grâce à ces modèles prend un temps bien supérieur ou ne permet pas d'interpréter la prise de décision.

Nous avons utilisé un K-Fold de taille trois en s'assurant que les classes étaient correctement représentées dans chaque partie.

Algorithmes	Accuracy	Précision	Rappel	F1-Score	Durée d'apprentissage
Adaboost	74.19 %	74.43 %	74.19 %	74.07 %	2769 s
Random Forest	88.05 %	88.47 %	88.05 %	88.15 %	6503 s

Table 2.2 – Performance de la classification des adresses.

L'importance de chaque caractéristique dans la prise de décision est un premier indicateur de l'expressivité de chaque type de caractéristiques. On peut voir sur la figure 2.3 l'importance (ordonnée) de chaque caractéristique (abscisse) pour la prise de décision du modèle de forêt aléatoire décisionnelle. Le pouvoir de différenciation de la durée de vie d'une adresse et des caractéristiques basées sur l'étude du graphe (e.g `successeurs` : nombre de successeurs uniques, `sibl_out` : nombre d'adresses uniques qui étaient en sortie quand l'adresse considérée l'était aussi) indique qu'une approche temporelle, ou une approche autour de la théorie des graphes (motifs, embedding, etc.) peut être intéressante pour caractériser les entités.

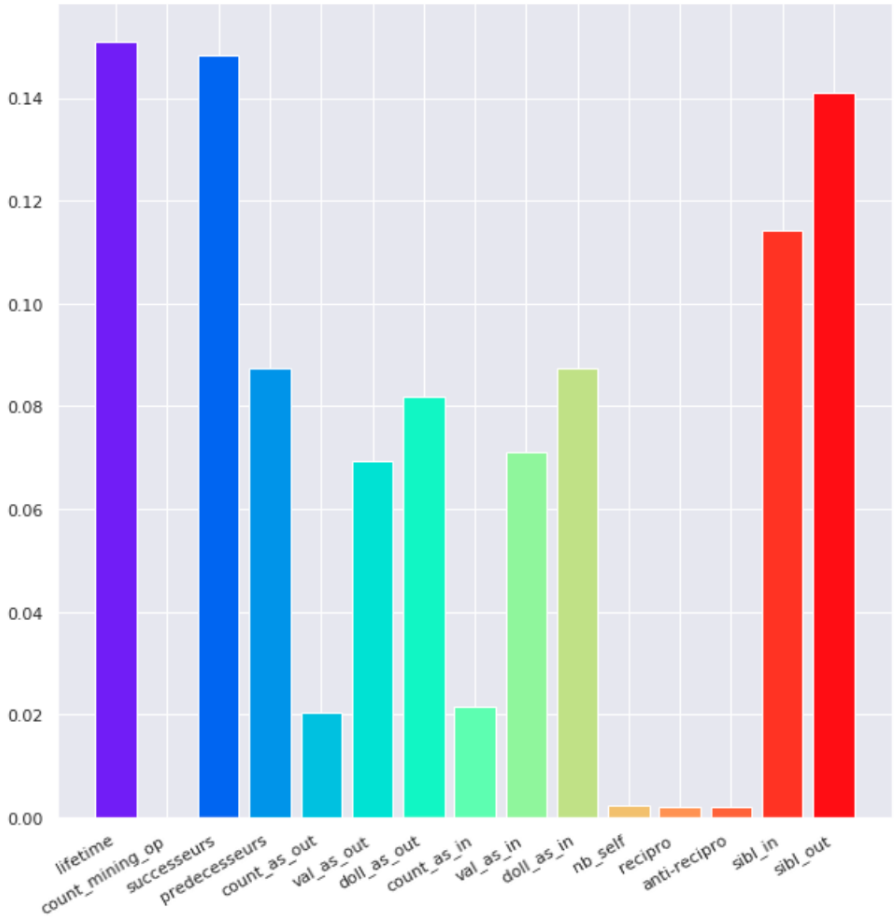

FIGURE 2.3 – Importance de chaque caractéristique dans la prise de décision.

2.2 Classification d'entités

La classification d'entités est basée sur le graphe d'entités présenté dans la section 1.3. Ce graphe est composé de :

— $\sim$ 70 Millions de noeuds
— $\sim$ 450 Millions d'arêtes

L'harmonisation des classes pour les entités nous contraint le plus. En effet, il est impossible d'entraîner un modèle avec trop peu d'individu dans des classes. Des travaux [12] utilisent des techniques pour équilibrer les classes en générant de nouveaux individus dans les classes sous-peuplées. Nous nous intéresserons aux 5 classes présentées dans 2.1.1. La table 2.3 explicite la représentation de chaque classe.

Classes	Nombres d'entités disponibles
Services	84
Exchanges	260
Market	42
Gambling	97
Mining	37
Total	523

TABLE 2.3 – Représentation des classes disponibles.

2.2.1 Extraction de caractéristiques

Nous avons catégorisé les caractéristiques en plusieurs parties distinctes représentant leurs sources ou leurs méthodes d'extraction.

Statistiques et données inhérentes au Bitcoin

— Nombre de Bitcoin reçus
— Nombre de Bitcoin envoyés
— Nombre d'adresses
— Nombre de frais de transactions payés
— Nombre de dollars payés en frais de transactions
— Nombre de Bitcoin payés en frais de transactions

Statistiques sur séries temporelles

Ces données sont des données issues des transactions (montants payées, reçues, ...) transformées pour les avoir sous une forme temporelle. Après évaluation de la complexité de l'analyse de ces séries temporelles et le bénéfice ajouté par rapport à l'utilisation (simple) des données brutes, nous avons choisis de ne pas continuer dans ce sens. On citera tout de même les deux types de séries temporelles créées et l'utilité d'un module python comme TSfresh [25](extraction automatique de statistiques sur des séries temporelles).

— Évolution de la quantité des bitcoins possédés pour chaque entité.
— Évolution des frais de transactions payés pour chaque entité.

Données basées sur la modélisation en graphe

Ces données sont simples et efficaces. Il est possible d'extraire des informations supplémentaires en utilisant des mesures plus avancées de la théorie des graphes (e.g. mesure de centralité) ou en produisant des statistiques sur les voisins de noeuds considérés (e.g. degré moyen des voisins).

— Nombre de participations en entrée dans des transactions (degré en tant qu'entrée)
— Nombre de participations en sortie dans des transactions (degré en tant que sortie)
— Nombre de successeurs uniques
— Nombre de prédécesseurs uniques
— Nombre de transactions internes (ou l'entité est en entrée et en sortie)
— Valeurs en Bitcoin des transactions internes
— Valeurs en Dollars des transactions internes
— Nombre de participations en entrée dans des transactions externes (ou l'entité n'est pas en sortie)
— Nombre de participations en sortie dans des transactions (ou l'entité n'est pas en entrée)

Caractéristiques de compositions
Cette approche proposée par les travaux [14,15] consiste à utiliser les proportions du résultats de la classification des adresses appartenant à chaque entité. Pour chaque entité, on comptera le nombre d'adresses qui ont été classifiées dans chaque classe, puis on normalisera ces données.

2.2.2 Node Embedding

L'extraction de caractéristiques présentée dans la partir 2.2.1 est performante. Notamment la fouille de motifs qui n'a pas été utilisé dans le cadre de ce stage mais qui est largement utilisée par la communauté pour caractériser des comportements particuliers. La motivation pour utiliser des techniques issues de l'apprentissage de représentation se trouve dans la nouveauté d'une telle approche et dans sa capacité de généralisation. La notion d'embedding a été introduite dans la section 1.3. Malgré cela, on abordera une définition plus en détail du concept de node embedding, puis on explicitera le choix de l'algorithme utilisé sur les entités du Bitcoin et sa mise en place.

Concept
Un embedding est une représentation d'un élément dans un autre espace de définition que celui dont il provient. Pour les techniques de plongement de noeud, chaque noeud d'un graphe aura une représentation dans un espace. Cet espace, pour des fins analytiques sera un espace euclidien, plus propice à l'utilisation des outils de classification et plus expressif. La figure 2.4 schématise ce concept. On passe d'un ensemble de noeud à un ensemble de vecteur de dimension K.

Les différentes approches pour extraire un embbeding de chaque noeud ont été explicité dans la section 1.3. Nous nous sommes intéressés à l'une d'entre elles : les marches aléatoires.

Marche Aléatoire et Skip-Gram
Les marches aléatoires sont utilisées en couple avec un algorithme basé sur le plongement lexical (word embedding). Le plongement de mots est une technique analogue au plongement de noeuds. On caractérise les mots dans un espace

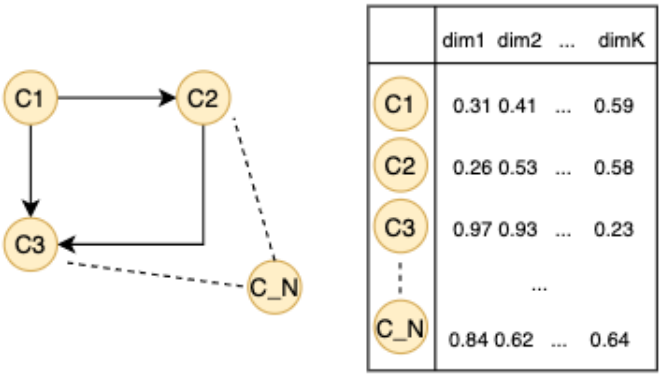

FIGURE 2.4 – Node embedding.

euclidien. L'algorithme utilisé est Skip-Gram [26]. Skip-Gram est un algorithme qui apporte du contexte à un embedding. Il est composé d'un réseau de neurones multi-couche. En analogie avec les mots (noeuds), les marches aléatoires sont des phrases. Une marche aléatoire de taille N est réalisée en partant d'un noeud du graphe et en choisissant aléatoirement un voisin de ce noeud, puis en choisissant un voisin du voisin préalablement sélectionné, etc, jusqu'à avoir un chemin de taille N. Il est possible d'introduire des biais différents dans le choix du prochain voisin. Certains travaux [27,28], placent leurs biais en fonction de la distance par rapport au noeud source et en fonction de la propriété que l'algorithme souhaite capturer. La figure 2.5 illustre un graphe simple (à gauche) et deux marches aléatoires pour le noeud C1 (à droite).

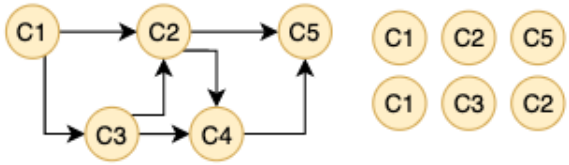

FIGURE 2.5 – 2 Marches aléatoire C1.

Ces marches aléatoires sont donc utilisées pour l'apprentissage du modèle basé sur Skip-Gram. Cette approche par marche aléatoire a inspiré beaucoup de variantes différentes.

Struc2vec

Struc2vec [28] est une approche principalement inspirée de node2vec [27] et de rolx [29]. L'idée directrice est de capturer l'identité structurelle d'un noeud dans son embedding en utilisant les marches aléatoires et Skip-Gram. Les trois concepts autour de struc2vec sont :

— La similarité des noeuds dans l'embedding doit être indépendante de leurs proximités dans le graphe, de leurs connecticités (i.e. est-ce qu'un chemin peut relier les noeuds entre eux), et de leurs attributs ou labels.

— La construction d'une hiérarchie de similarité. Dans les niveaux les plus bas, deux noeuds sont similaires si leurs degrés sont similaires, tandis qu'aux niveaux les plus haut, la similarité est définie sur l'ensemble du réseau.
— La construction d'un graphe, qui induit les marches aléatoire à apporter un contexte structurel à l'embedding.

L'avantage d'utiliser struc2vec plutôt qu'un autre algorithme d'embedding se retrouve dans sa capacité à produire un embedding qui dépend de la structure des noeuds, et par conséquent de leurs positions dans le graphe.

Implémentation

Une implémentation de struc2vec est disponible [30]. Malgré quelques succinctes modifications, la mise en place de struc2vec fut rapide. La complexité des algorithmes d'embedding est généralement élevée (quadratique, cubique, ...). Naïvement, struc2vec a une complexité cubique au nombre de noeud du graphe. Plusieurs optimisations sont présentées dans l'article original et sont implémentées dans le code disponible que nous utilisons. Nous utilisons donc les trois optimisations disponibles. La complexité de struc2vec est toujours quadratique au nombre de noeud. Nous sommes donc contraints de réduire la taille de notre graphe en supprimant un grand nombre d'arêtes et de noeuds. Cette réduction peut améliorer les performances de l'embedding puisqu'elle permet d'inclure un biais dans la structure du graphe, qui sera par la suite traité par struc2vec. Le choix des noeuds, qui seront inclus dans le graphe, doit permettre d'avoir un échantillon représentatif du réseau complet en utilisant les données déjà classifiées. La version finale de nos différents essais comporte :

— N noeuds correspondant aux noeuds contenant le plus d'adresses.
— N noeuds correspondant aux noeuds ayant le plus participé à des transactions en entrées.
— N noeuds correspondant aux noeuds ayant le plus participé à des transactions en sorties.
— O noeuds aléatoires du réseau.
— Tout les noeuds déjà classifiés.

Ces ensembles de noeuds ne sont pas indépendant. On notera pour $N = 1000$ que le graphe sans les noeuds aléatoires comporte environ 2000 noeuds (pour 3 ensembles différents).

Nous conservons toutes les arêtes entre ces noeuds. Le graphe final est un multi-graphe. Une entité a pu envoyer plusieurs fois des Bitcoins vers une autre entité. Nous avons donc agrégé ce multi-graphe en graphe pondéré. Nous avons trois types de graphe où seul les poids des arêtes diffèrent :

— Nombre de transactions entre deux entités
— Somme des Bitcoin des transactions entre deux entités
— Somme en Dollars dans transactions entre deux entités

Cas simple
Le graphe présenté par la figure 2.6 est un graphe miroir où deux noeuds centraux (1, 37) relient les deux graphes identiques.

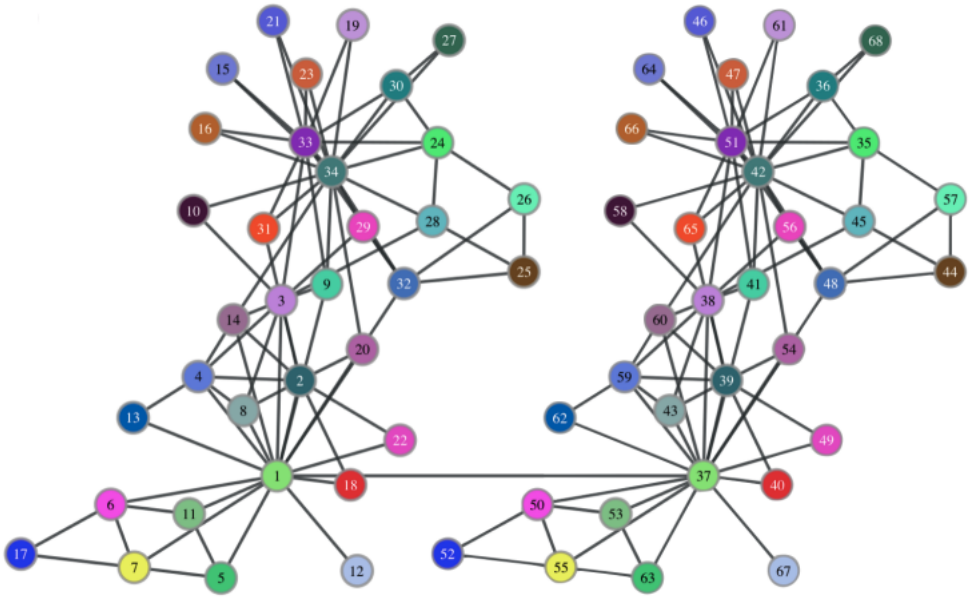

FIGURE 2.6 – Mirror Karate Club Network, image reproduite de [28].

La figure 2.7 présente un embedding de dimensions 2 pour chaque noeud du graphe. Cette figure comporte en abscisse, une première dimension et en ordonnée, la deuxième dimension.

L'intérêt que nous portons à struc2vec vient du fait que des noeuds structurelement similaires seront proches dans l'embedding. Par exemple :

— les noeuds 1 et 37 sont identiques, donc proches dans l'embedding.
— les noeuds 34 et 42 sont identiques, donc proches dans l'embedding.
— les noeuds 1, 34, 37 et 42 ont des rôles centraux dans le graphe, donc proches dans l'embedding.

2.3 Technique d'apprentissage automatique

La tâche de classification sert à évaluer la performance des caractéristiques que nous avons extrait. Les principaux concepts et idées des tâches de classification ont été présentés en 2.1.3. On utilisera quatre mesures différentes pour évaluer les performances de nos modèles de classification.

— Rappel
— Précision
— Accuracy
— F-Score

Nous séparerons les données déjà classifiées disponibles avec la technique du "K-fold" en s'assurant une représentation adéquate pour chaque classe. On se

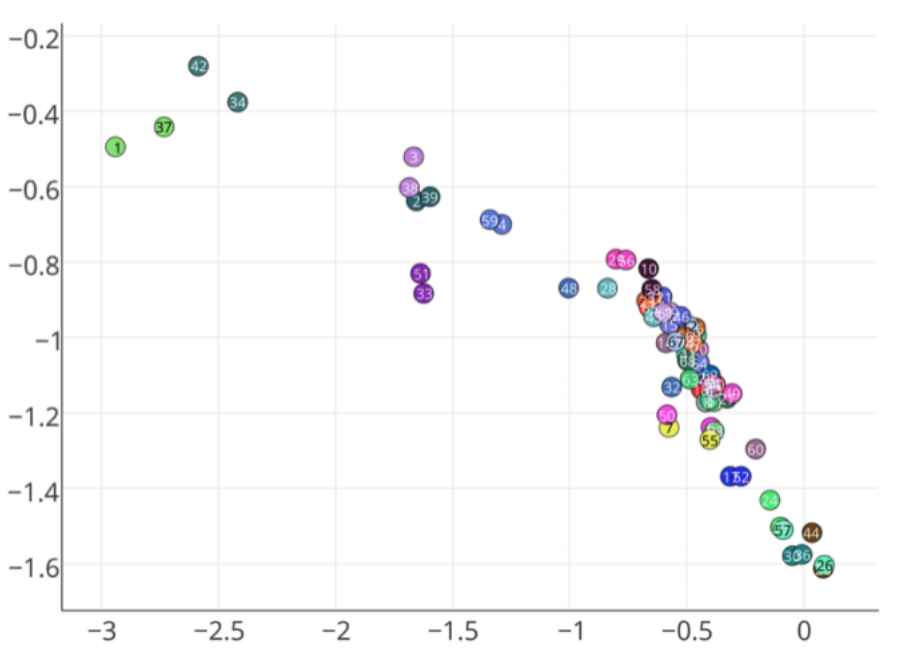

FIGURE 2.7 – struc2vec Embedding Mirror Karate Club 2 dimensions, image reproduite de [28].

basera sur l'état de l'art [11,13,15] pour la sélection des modèles de classification. Mais on portera tout de même un regard particulier sur l'explicabilité de la prise de décision des modèles.

2.3.1 Résultats

Un des critères de sélection du modèle de classification des adresses est la durée d'apprentissage. Les entités étant moins nombreuses, il est possible de tester plus facilement plusieurs modèles. Dans un premier temps, nous catégoriserons les types de données et évaluerons la capacité des modèles à produire une généralisation durant l'apprentissage sur ces différentes catégories. Puis dans un second temps, nous nous intéresserons aux différents embedding produits. Certains algorithmes ne permettent pas de prendre en compte le non-équilibre dans la représentation des classes, nous faisons donc le choix de ne pas les utiliser. On utilisera par conséquent les modèles suivants :

— Algorithme de Bagging.
— Algorithme d'Adaptative Boosting.
— Algorithme de forêt aléatoire décisionnelle.
— Algorithme de Boosting de gradient.

Évaluation

L'évaluation de nos modèles est un travail en cours à l'heure de la rédaction de ce rapport. Nous avons produit des visualisations des résultats de l'apprentissage pour mieux comprendre celui-ci. La figure 2.8 représente l'écart-type et la moyenne de chaque score pour chaque outil de classification utilisé. Ce type de graphique, en plus des valeurs numériques des scores, nous permet d'évaluer la performance de chaque modèle.

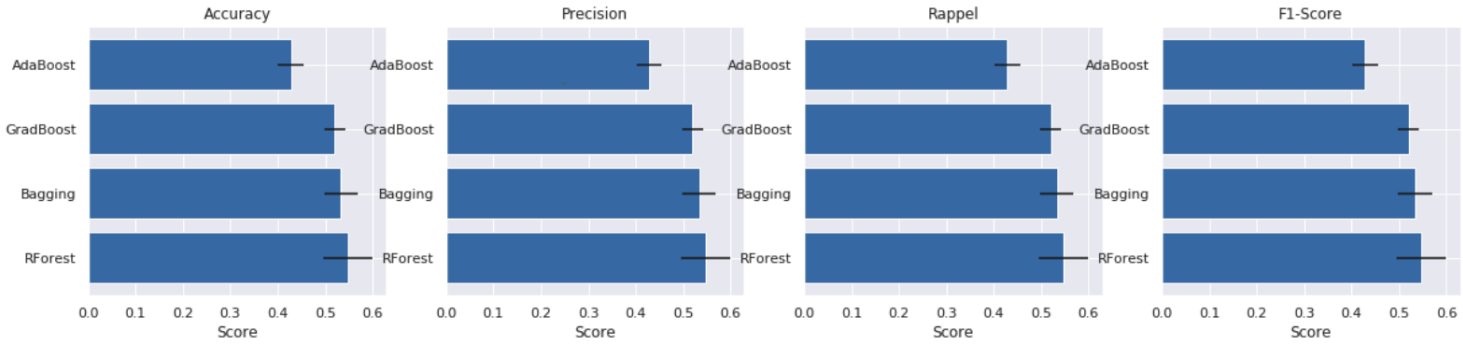

FIGURE 2.8 – Exemple de la visualisation utilisé pour l'évaluation des différents modèles.

L'évaluation des performances de chaque type de caractéristiques pour la classification est réalisée grâce à des figures comme l'annexe 1. Ces figures permettent d'identifier quel jeu de données (couleurs) offre la meilleure valeur pour chaque score.

Évaluation de l'embedding

Les embeddings présentés ici ont été fait en 128 dimensions. Nous utilisons donc une réduction de dimensions pour identifier visuellement des structures se dégageant de l'embedding. Nous utiliserons deux outils :

— Analyses en composantes principales (PCA)
— t-SNE

Ces outils incorporent un biais qu'il est naturel de prendre en compte. En effet t-SNE possède un paramètre qui peut dégager des structures intéressantes. Les modèles de classification s'exercent sur plusieurs dimensions. Nous sommes contraints à une première évaluation, qui peut ne pas apporter d'information. L'annexe 2 représente une visualisation d'une réduction de dimensions d'un embedding (de 128 dimensions vers 2 dimensions).

Les résultats obtenus (présentés dans la table 2.4) nous montrent que le meilleur embedding (Embedding Dollars : Multi-graphe réduit en sommant la valeurs en dollars de toute les transactions ayant eu lieu entre chaque couple d'entités) permet de différencier des classes. En effet, 54.86 % des prédictions sont correctes. En revanche, nous pouvons noter une précision faible, ce qui s'explique par le fait qu'un modèle préférera prédire un nombre trop important de fois la classe la plus représentée. En d'autres termes, la précision répond à

la question "Pour chaque classe, combien d'éléments classifiés dans cette classe sont correctement classifiés ?".

Jeux de données	Accuracy	Précision	Rappel	F-Score
Embedding Dollars	54.86 %	44.31 %	54.85 %	49.25 %
Embedding Bitcoin	52.87 %	58.05 %	52.96 %	50.19 %
Embedding Nombre de transactions	54.59 %	44.69 %	54.60 %	45.78 %
Données sans embedding	59.77 %	57.74 %	59.77 %	57.85 %

TABLE 2.4 – Résultats classification embedding seul.

Les résultats obtenus (présentés dans la table 2.5) nous montrent l'amélioration obtenue en utilisant les embeddings en plus des caractéristiques calculées auparavant. Il est intéressant de noter que les embeddings sont réalisés dans un grand nombre de dimensions (128) en rapport au nombres de caractéristiques de base (20). On peut donc considérer que les embeddings perturbent les modèles utilisés, mais ce point reste à clarifier durant la suite du stage.

Jeux de données	Accuracy	Précision	Rappel	F-Score
Embedding Dollars + Donéees	55.74 %	54.82 %	55.74 %	62.67 %
Embedding Bitcoin + Données	53.44 %	43.17 %	53.44 %	53.40 %
Embedding Nombre de transactions + Données	58.04 %	46.75 %	58.04 %	52.66%
Données sans embedding	59.77 %	57.74 %	59.77 %	57.85 %

TABLE 2.5 – Résultats classification embedding et données.

3 Conclusion

Le travail réalisé et celui toujours en cours me permettent d'acquérir des connaissances autour de sujet diverses, notamment la notion de graphe embedding. L'utilisation de struc2vec a été un choix difficile à prendre puisqu'il existe beaucoup d'autres approches pour la caractérisation de noeuds. L'une de celles qui ont capté notre attention est la création de graphe pour chaque noeud à caractériser ("ego-network") en prenant en compte uniquement les voisins, puis à produire un embedding sur le graphe entier. La génération de graphe particulier pour produire un embedding est une approche complexe puisque cette génération constitue un hyper-paramètre de l'embedding produit. En effet, notre méthode d'échantillonage des noeuds offre un biais dans la caractérisation mais peut être étudiée plus en profondeur, notamment en apportant un fondement mathématique plus poussé. De plus, les motifs sont largement utilisés par la communauté pour des tâches de classification. Ils ont un pouvoir de caractérisation de l'usage réalisé dans le réseau Bitcoin. On a pu constater dans nos travaux que des approches simples par l'analyse du graphe offre un pouvoir expressif important dans la classification. Il serait intéressant d'évaluer nos modèles avec des caractéristiques issues d'une fouille de motif plus avancée que celle réalisée. Les différents protocoles construits peuvent être des outils intéressant pour l'équipe travaillant sur le projet ANR BITUNAM. Il est donc essentiel de laisser des travaux maintenables et reutilisables. Ce stage est une première introduction dans le monde de la recherche. Malgré le contexte durant lequel j'ai pu effectuer mon stage (COVID-19), j'ai pu comprendre certaines mécaniques et aspects de ce monde qui m'étaient peu familiers.

Bibliographie

[1] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M Voelker, and Stefan Savage. A fistful of bitcoins : characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, 2013.

[2] Satoshi Nakamoto. Bitcoin : A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[3] Martin Harrigan and Christoph Fretter. The unreasonable effectiveness of address clustering. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 368–373. IEEE, 2016.

[4] Greg Maxwell. Coinjoin : Bitcoin privacy for the real world. In *Post on Bitcoin forum*, 2013.

[5] Walletexplorer.com.

[6] Stephen Ranshous, Cliff A Joslyn, Sean Kreyling, Kathleen Nowak, Nagiza F Samatova, Curtis L West, and Samuel Winters. Exchange pattern mining in the bitcoin transaction directed hypergraph. In *International Conference on Financial Cryptography and Data Security*, pages 248–263. Springer, 2017.

[7] Zhen Zhang, Tianyi Zhou, and Zhitong Xie. Bitscope : Scaling bitcoin address de-anonymization using multi-resolution clustering.

[8] Jiaqi Liang, Linjing Li, Weiyun Chen, and Daniel Zeng. Targeted addresses identification for bitcoin with network representation learning. In *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 158–160. IEEE, 2019.

[9] Yu-Jing Lin, Po-Wei Wu, Cheng-Han Hsu, I-Ping Tu, and Shih-wei Liao. An evaluation of bitcoin address classification based on transaction history summarization. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 302–310. IEEE, 2019.

[10] Cazabet Remy, Baccour Rym, and Latapy Matthieu. Tracking bitcoin users activity using community detection on a network of weak signals. In *International conference on complex networks and their applications*, pages 166–177. Springer, 2017.

[11] Marc Jourdan, Sebastien Blandin, Laura Wynter, and Pralhad Deshpande. Characterizing entities in the bitcoin blockchain. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 55–62. IEEE, 2018.

[12] Mikkel Alexander Harlev, Haohua Sun Yin, Klaus Christian Langenheldt, Raghava Mukkamala, and Ravi Vatrapu. Breaking bad : De-anonymising entity types on the bitcoin blockchain using supervised machine learning. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[13] Anil Gaihre, Santosh Pandey, and Hang Liu. Deanonymizing cryptocurrency with graph learning : The promises and challenges. In *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 1–3. IEEE, 2019.

[14] Francesco Zola, Jan Lukas Bruse, Maria Eguimendia, Mikel Galar, and Raul Orduna Urrutia. Bitcoin and cybersecurity : temporal dissection of blockchain data to unveil changes in entity behavioral patterns. *Applied Sciences*, 9(23) :5003, 2019.

[15] Francesco Zola, Maria Eguimendia, Jan Lukas Bruse, and Raul Orduna Urrutia. Cascading machine learning to attack bitcoin anonymity. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 10–17. IEEE, 2019.

[16] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[17] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Homophily, structure, and content augmented network representation learning. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 609–618. IEEE, 2016.

[18] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding : Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9) :1616–1637, 2018.

[19] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance : A survey. *Knowledge-Based Systems*, 151 :78–94, 2018.

[20] Thai Pham and Steven Lee. Anomaly detection in bitcoin network using unsupervised learning methods. *arXiv preprint arXiv :1611.03941*, 2016.

[21] Patrick Monamo, Vukosi Marivate, and Bheki Twala. Unsupervised learning for robust bitcoin fraud detection. In *2016 Information Security for South Africa (ISSA)*, pages 129–134. IEEE, 2016.

[22] Alex Greaves and Benjamin Au. Using the bitcoin transaction graph to predict the price of bitcoin. *No Data*, 2015.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas,

A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[24] Wei Shao, Hang Li, Mengqi Chen, Chunfu Jia, Chunbo Liu, and Zhi Wang. Identifying bitcoin users using deep neural network. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 178–192. Springer, 2018.

[25] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307 :72–77, 2018.

[26] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.

[27] Aditya Grover and Jure Leskovec. node2vec : Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[28] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec : Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394, 2017.

[29] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx : structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239, 2012.

[30] https ://github.com/steenfatt/struc2vec.

Glossaire

Apprentissage de représentation Ensemble de technique qui permettent de représenter des objets par des valeurs plus propices à l'analyse et l'observation.. 11

Cluster Groupe d'individu. Ici, groupe d'adresses régit par une entité. Cluster est aussi appelé entité dans ce document.. 7

Clustering Concept issue de l'apprentissage automatique qui consiste à regrouper des individus en cluster. Ici, regroupement (ou agrégation) d'adresses en cluster.. 7

Embedding Plongement. Pour un ou plusieurs objets, association de valeurs d'un autre espace de définition que celui d'origine de ces objets.. 10

PCA T- Distributed Stochastic Neighbor Embedding, algorithme de réduction de dimension. 24

t-SNE T- Distributed Stochastic Neighbor Embedding, algorithme de réduction de dimension. 24

Annexe 1

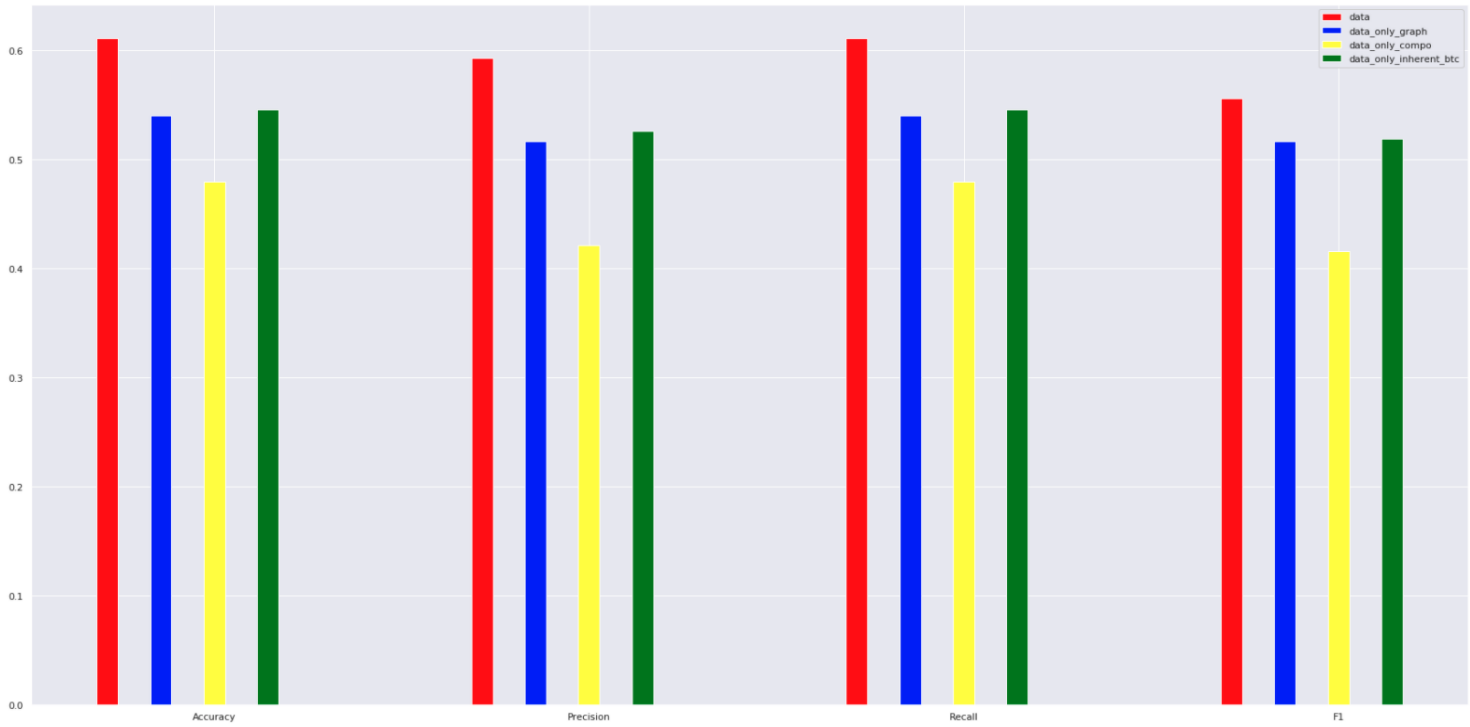

FIGURE 1 – Évaluation des jeux de données pour la classification des entités.

Annexe 2

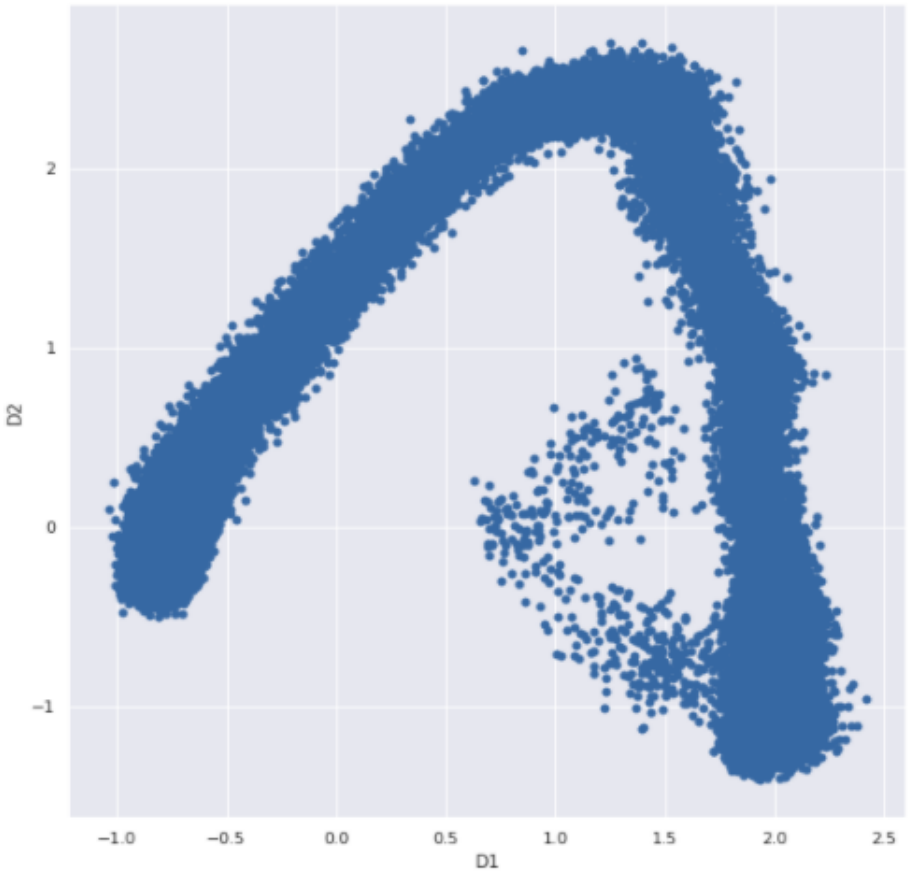

FIGURE 2 – Visualisation 2 dimensions d'un embedding de 128 dimensions.