Embedding de graphes temporels : temporalisation de méthodes statiques et alignement

Mounir Haddad

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THESE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPERIEURE MINES-TELECOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par
Mounir HADDAD

Embedding de Graphes Temporels : Temporalisation de Méthodes Statiques et Alignement

Thèse présentée et soutenue à Brest, le 24/02/2022
Unité de recherche : Lab-STICC, CNRS UMR 6285
Thèse N° : 2022IMTA0289

Rapporteurs avant soutenance :

Mme Christine LARGERON	Professeur, Université Jean Monnet
Mme Hamida SEBA	Maître de Conférences HDR, Université Claude Bernard Lyon 1

Composition du Jury :

Président :	M. Hocine CHERIFI	Professeur, Université de Bourgogne
Examinateurs :	M. Nicolas DUGUE	Maître de Conférences, Université du Mans
	Mme Christine LARGERON	Professeur, Université Jean Monnet
	Mme Hamida SEBA	Maître de Conférences HDR, Université Claude Bernard Lyon 1
Co-encadrante:	Mme Cécile BOTHOREL	Maître de Conférences, IMT Atlantique
Dir. de thèse :	M. Philippe LENCA	Professeur, IMT Atlantique

Invité

M. Dominique BEDART	Directeur du Développement, DSI Global Services

Cette thèse est le fruit d'une collaboration CIFRE entre l'IMT Atlantique et DSI Global Services

Remerciements

En premier lieu, c'est avec un grand plaisir que je tiens à remercier mon encadrante de thèse, Mme Cécile Bothorel, pour son accompagnement exigeant et indulgent durant ces 4 dernières années. Je lui suis très reconnaissant pour le temps qu'elle m'a consacré, pour les échanges de qualité que nous avons pu avoir, pour ses encouragements aux moments de doute, mais aussi pour ses recadrages aux moments d'égarement.

Un grand merci à mon directeur de thèse, M. Philippe Lenca, pour son soutien aux moments les plus importants de cette thèse. Ses conseils et directives ont su me sortir de ma zone de confort, notamment en m'incitant à évaluer un article ou à participer à des colloques. Je lui suis également reconnaissant pour avoir été de mon côté quand il le fallait.

En réalité, le soutien de Mme Bothorel et de M. Lenca remonte à bien avant le début de mon travail de thèse. En 2015, quand j'étais encore élève ingénieur, ils m'accordaient déjà leur confiance pour représenter Télécom Bretagne et candidater au prix du meilleur stage de la fondation Mines-Télécom. Je les remercie donc pour la confiance renouvelée durant cette thèse.

J'ai une profonde gratitude à l'égard de M. Dominique Bedart, mon encadrant côté entreprise, pour son soutien qui dure depuis plus de 4 ans et qui, je l'espère, se poursuivra encore. Je le remercie d'abord pour l'opportunité qu'il m'a donnée d'effectuer ma thèse au sein de DSI Global Services, pour m'avoir accordé le temps nécessaire aux travaux de thèse, et surtout pour sa compréhension et sa souplesse aux moments les plus difficiles. Je tiens également à adresser mes plus chaleureux remerciements à toutes les équipes de DGS que j'ai côtoyées durant ces 4 dernières années et que j'aurai plaisir à garder comme collègues à l'avenir.

Par ailleurs, je remercie les rapportrices de cette thèse Mme Christine Largeron et Mme Hamida Seba pour l'intérêt qu'elles ont porté à mon travail. J'associe également à ces remerciements M. Hocine Cherifi et M. Nicolas Dugué pour avoir accepté d'examiner mon travail de thèse. Je me réjouis grandement de la qualité du jury de thèse.

Je souhaiterais également remercier les membres de mon CSI, M. Noël Crespi et M. Nicolas Farrugia pour les échanges scientifiques que nous avons eus à différentes occasions, ainsi que pour les éclairages qu'ils ont pu m'apporter. Merci également à M. Furkan Gursoy pour le travail conjoint sur le dernier axe de recherche.

En outre, je remercie l'ANRT pour avoir aidé à financer ma thèse. Je remercie également mes structures d'accueil, à savoir mon équipe de recherche (DECIDE) et ses membres, le département LUSSI et l'IMT Atlantique.

Sur un plan personnel, un immense merci à mes parents pour leur soutien indéfectible et pour leur patience à toute épreuve. Merci également à mon frère et à ma sœur d'être présents pour moi. Enfin, je remercie Charlotte qui m'a toujours encouragé et soutenu.

Sommaire

Liste des Figures

Liste des Tableaux

Chapitre 1

Introduction

Les interactions régissent une grande partie des phénomènes de notre monde. Les structures les plus élémentaires de la matière se forment à l'issue d'interactions entre atomes. Dans un système nerveux, les neurones communiquent entre eux via des influx électriques. Les différentes espèces vivantes interagissent entre elles et avec leur environnement pour former un écosystème. Les adresses IP communiquent au sein d'un gigantesque réseau virtuel...

Si les phénomènes d'interaction existent depuis toujours, la collecte massive des données qui en résultent est très récente. En effet, il aura fallu attendre l'avènement des TIC et l'accroissement colossal de la capacité de stockage pour que nous disposions de larges bases de données, retraçant des interactions entre entités de tous types.

Les graphes sont la manière la plus courante de représenter ces données d'interaction. Dépendamment du type d'interactions considérées, les graphes correspondants peuvent être orientés ou non-orientés, pondérés ou non-pondérés. De surcroît, des interactions d'une nature plus complexe peuvent être transposées en des graphes de formats plus sophistiqués élargissant le champ des possibles. Tel est par exemple le cas pour les hypergraphes (i.e. les graphes où les arêtes peuvent mettre en relation un nombre de nœuds ≥ 2) ou encore les graphes avec métadonnées (associées aux nœuds ou aux arêtes). En l'occurrence, le présent travail de thèse traite du cas des graphes temporels : il s'agit de graphes possédant une dimension temporelle au sens où les interactions qu'ils représentent sont datées.

Sur une note globale, l'apprentissage de représentations latentes de graphes temporels est un sujet de recherche en plein essor. Au début de ce travail de thèse (i.e. en 2017), les objectifs et verrous que nous avions défini semblaient être audacieux, voire atypiques. Depuis, l'état de l'art traitant du sujet s'est fortement enrichi. Des méthodes pionnières

ont prouvé leur efficacité et se sont imposées comme des approches de référence. Nous avons tenté d'élaborer des méthodes peu ou prou originales, parfois à contre courant de la direction que prenait l'état de l'art, et non sans échecs.

Interopérabilité des approches d'apprentissage automatique

Dans le contexte de la révolution numérique et de la production exponentielle de données qui l'accompagne, les algorithmes de traitement automatique se spécialisent de plus en plus aux différents types de données qu'ils traitent. S'il est certain que des algorithmes spécialisés sont plus à même d'extraire de la valeur des données, il est tout aussi évident qu'il existe souvent des similitudes entre les différents types de données. En l'occurrence, il existe une multitude de domaines où les données produites représentent des processus séquentiels, s'inscrivant dans le temps par exemple. C'est le cas des vidéos ou encore des interactions entre individus au sein d'un réseau social. Il est donc possible d'imaginer l'adaptation de certaines techniques d'apprentissage automatique issues de la vision par ordinateur aux données d'interaction sociale. Plusieurs cas d'étude pourraient tirer profit de cette idée. Par exemple, la reconnaissance de mouvements pourrait être appliquée au cas des réseaux temporels pour la détection de motifs d'interaction particuliers ; les techniques de reconstruction de régions manquantes sur les images d'une vidéo peuvent aider à compléter les graphes présentant des données manquantes ; la génération de graphes temporels synthétiques peut tirer avantage des méthodes similaires de génération de vidéos. En substance, ce type d'approches constitue la finalité première motivant ce travail de recherche.

Cela étant dit, une transposition de techniques d'apprentissage automatique d'un type de données à un autre requiert au préalable plusieurs prétraitements. En conséquence, nous nous sommes spécifiquement intéressés aux différents verrous à lever pour permettre une telle transposition. Il s'agit là des principaux axes de recherche autour desquels se sont articulés nos travaux. Par ailleurs, la structure du présent document suit ces axes de recherche comme détaillé ci-après.

Axes de recherche et plan du document

Bien que les vidéos et les données d'interaction sociale présentent des similarités, il est évident que leurs spécificités impliquent des différences dans les méthodes de traitement. En effet, l'information représentée dans une matrice d'adjacence diffère de celle présente sur une image. Par exemple, la notion de localité n'est pas identique : l'on

peut intervertir l'ordre des individus pour une représentation en matrice d'adjacence (cela revient à un changement de référentiel), mais il est impossible de faire de même sur les pixels d'une image. À l'inverse, la représentation d'un réseau social en graphe conserve par construction la localité, mais est structurellement différente d'une image (une image étant assimilable à une matrice). Or, la notion de localité est fondamentale dans la représentation de vidéos et dans les réseaux de neurones à convolution (CNN), base des algorithmes les plus récents et performants en apprentissage profond appliqué aux vidéos. Fort de ce constat, un verrou élémentaire à lever afin de parvenir à adapter des techniques de vision par ordinateur aux données d'interaction sociale est celui de la modélisation des données. L'objectif est donc de trouver une représentation *canonique* sous une forme matricielle tout en conservant la notion de localité d'un graphe. Il s'agit là de l'objectif principal des deux premiers axes de ce travail de thèse.

Dans un premier temps, le parti pris a été de concevoir, *from scratch*, des méthodes de représentation des nœuds d'un graphe temporel, communément appelées méthodes d'embedding. Dans ce cadre, deux approches ont été proposées et sont présentées dans le chapitre 2. La première, GeoEmb (2.1), se distingue de l'état de l'art et construit des représentations géométriques en respectant certaines contraintes que lesdites représentations doivent satisfaire. À l'inverse, la deuxième approche, TemporalNode2vec 2.2, adapte un modèle d'embedding de graphes statiques de l'état de l'art au traitement de graphes temporels en utilisant un mécanisme de lissage temporel emprunté au cas d'embedding de corpus textuels. Par la suite, cette approche a été reprise pour concevoir TsTemporalNode2vec 2.3, une variante semi-supervisée adressant de manière spécifique la classification des nœuds d'un graphe temporel et améliorant la qualité des embeddings vis-à-vis de cette tâche d'inférence.

Dans un deuxième volet présenté dans le chapitre 3, l'idée poursuivie a été de proposer une méthode générique de *temporalisation* de modèles d'embedding statiques de l'état de l'art. En d'autres termes, au lieu de concevoir une nouvelle méthode d'embedding temporelle, l'objectif est d'élaborer un processus permettant d'adapter une catégorie de méthodes statiques au cas des graphes temporels. L'intérêt de cette approche réside dans la possibilité de tirer profit des avantages que chacune des méthodes *temporalisées* peut apporter, attendu que l'état de l'art compte des modèles d'embedding plus ou moins efficaces, dépendamment du cas d'application (tâche d'inférence, nature des interactions décrites par le graphe). Ce faisant, les performances de certaines tâches d'inférence (particulièrement celles qui sont sensibles à la dimension temporelle) sont améliorées de manière significative.

Le chapitre 4 a été consacré au dernier axe de recherche et à l'étude de l'alignement temporel des représentations des nœuds d'un graphe temporel. Autrement dit, afin

d'améliorer la qualité des embeddings construits par une méthode quelconque, l'idée est de corriger, quantifier et évaluer l'impact des potentiels décalages temporels entre les représentations des nœuds d'un graphe. Ainsi, des métriques de mesure des décalages temporels ont été conçues et validées sur des données synthétiques et réelles, sur des embeddings en sortie de plusieurs méthodes de l'état de l'art. De plus, une procédure de *réalignement* temporel des embeddings a été proposée, améliorant ainsi la qualité des représentations des nœuds d'un graphe temporel.

Enfin, dans le chapitre 5 de conclusion, nous récapitulons les différentes méthodes développées ainsi que les expérimentations et les contributions les plus importantes du travail de thèse. Nous ouvrons également plusieurs pistes de recherche potentielles, liées entre autres à l'application de méthodes de la vision par ordinateur aux embeddings temporels conçus ou encore aux domaines d'application des approches développées.

Dans l'ensemble, et comme illustré dans la figure 1.1, les différents axes de ce travail de thèse sont à considérer comme étant les étapes permettant de transformer un graphe temporel en une représentation comparable à une vidéo (avec une dimension temporelle et plusieurs dimensions spatiales) et compatible avec l'application d'algorithmes de vision par ordinateur.

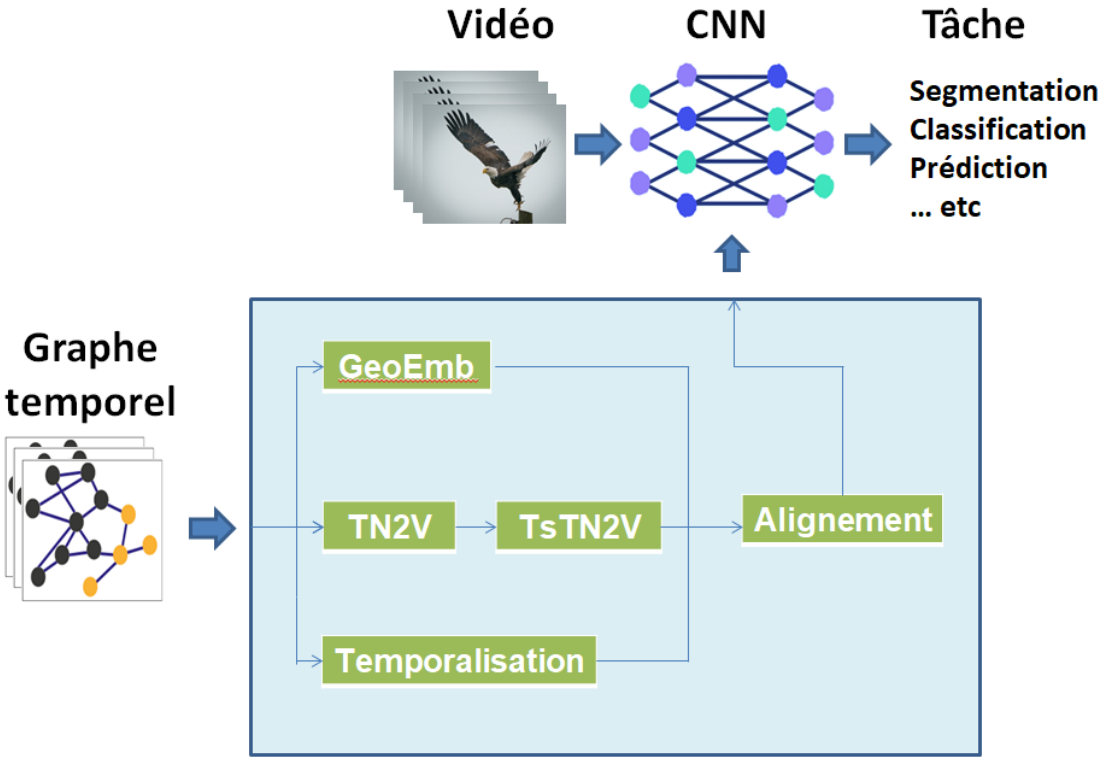


FIGURE 1.1: Fil conducteur des axes de recherche

Généralités et définitions

En apprentissage automatique, l'apprentissage de représentations est un ensemble de techniques permettant de découvrir automatiquement des représentations de données brutes, souvent dans un espace latent multidimensionnel. Cela remplace l'ingénierie manuelle de caractéristiques et permet d'automatiser l'apprentissage de caractéristiques utiles pour effectuer une tâche spécifique en aval. Dans la littérature, les techniques d'apprentissage de représentations ont été conçues pour différents types de données, les données d'interactions temporelles entre autres. Néanmoins, il existe différentes modélisations pour décrire les interactions entre des entités au cours du temps :

- Temps continu : L'évolution est modélisée à une granularité temporelle fine. Cela se traduit par un graphe temporel composé d'arêtes datées, formées de triplets $\{n_i, n_j, t\}$ (signifiant une interaction entre n_i et n_j à un instant t quelconque). Dans une telle modélisation, les graphes temporels correspondent au cas général d'un réseau d'interaction temporelle entre entités, à savoir des interactions pouvant survenir à tout moment du déroulement du graphe temporel.
- Temps discret : Dans cette modélisation, l'évolution d'un graphe temporel peut être représentée sous la forme d'une séquence de graphes statiques. En d'autres termes, le graphe temporel peut être écrit sous la forme $\{G_1, G_2, ..., G_T\}$, chacun des G_i représentant les interactions qui se produisent dans un des T *pas de temps* (i.e. tranches temporelles formant l'ensemble de l'historique du graphe).

Dans le cadre de l'apprentissage de représentations, des embeddings construits sur la base d'une modélisation en temps continu permet d'obtenir des représentations à chaque instant du déroulement temporel du graphe en entrée. Il s'agit, entre autres, de l'un des objectifs motivant la conception de GeoEmb 2.1. Néanmoins, cela est possible au prix d'une complexité de traitement très importante, souvent incompatible avec les graphes temporels issus de données du monde réel. Par conséquent, les autres travaux menés dans les différents axes de ce travail de thèse, (i.e. TN2V 2.2, la temporalisation d'autoencodeurs 3 et l'étude de l'alignement des embeddings 4) traitent d'embeddings construits sur des graphes temporels en temps discret.

Chapitre 2

Vers une Nouvelle Représentation des Graphes Temporels

L'objectif principal poursuivi dans le premier axe de ce travail de recherche est la conception de méthodes d'embedding de graphes temporels. En d'autres mots, étant donné un graphe temporel, l'idée est de créer des représentations latentes de ses nœuds à différents moments du déroulement de son historique. Dans ce cadre, deux approches différentes ont été conçues et sont exposées dans ce chapitre. La première méthode (i.e. GeoEmb), présentant des limites critiques, n'a pas été concluante. La deuxième méthode, en partie inspirée de l'état de l'art, s'avère être efficace pour certaines tâches d'inférence.

2.1 GeoEmb : une méthode géométrique exhaustive

La première approche de représentation conçue (GeoEmb) est une méthode géométrique, déterministe et bijective. En ce sens, cette méthode est assimilable à une retranscription de la donnée en entrée dans le format souhaité, à savoir une représentation vectorielle dans un espace latent multidimensionnel.

Il s'agit d'un plongement dans un espace vectoriel latent conservant le plus possible les relations entre individus du graphe temporel en entrée. Plus concrètement, à chaque instant de l'historique d'interactions, chaque nœud du graphe en entrée est représenté sous la forme d'un vecteur multidimensionnel ; un instantané du graphe est donc représenté sous la forme d'une matrice (la $i^{ème}$ ligne contient les coordonnées du $i^{ème}$ nœud). Les vecteurs associés respectivement à deux nœuds sont proches (selon une métrique géométrique à définir) d'autant plus que leur interaction mutuelle est forte : le respect de ce principe garantit la conservation de la notion de localité propre à une représentation

sous forme de graphe. Ainsi, nous ne parlons plus de poids d'un lien ou d'absence de lien entre les nœuds comme dans le cas d'un graphe, mais plutôt de distance entre les embeddings.

De nombreuses approches de l'état de l'art traitent du sujet de la représentation des nœuds de graphes sous formes vectorielles. Ces méthodes sont communément appelées *méthodes d'embedding de graphes* et couvrent les cas des graphes statiques [2–11] ou temporels [12–17] (un état de l'art plus détaillé est présenté dans la section 2.2.1). Néanmoins, la méthode GeoEmb présente d'importantes différences avec les approches de l'état de l'art. En effet, les algorithmes d'embedding de graphes temporels gèrent des pas de temps (i.e. tranches temporelles de l'historique des interactions entre les nœuds) agrégés. Cela signifie que les interactions datées sont d'abord regroupées en un nombre réduit de pas de temps, puis les embeddings des nœuds sont construits pour chaque pas de temps. En d'autres termes, avec les méthodes d'embedding temporelles de l'état de l'art, il est impossible de construire des embedding à la granularité temporelle la plus fine (i.e. à chaque nouvelle interaction). De plus, les différentes méthodes de l'état de l'art incluent une phase inhérente de réduction de la dimension : l'idée est de construire des embeddings dans un espace latent à dimension réduite (généralement <100) gardant uniquement l'information pertinente. Néanmoins, cela se fait forcément au prix d'une perte de l'information contenue dans les données en entrée.

À l'inverse, l'idée avec GeoEmb est de construire des embeddings exhaustifs dans le sens où la donnée en entrée est entièrement conservée. Au niveau temporel, cela se traduit par une mise à jour des embeddings à chaque interaction. Au niveau spatial, GeoEmb construit des embeddings dans un espace latent à dimension suffisamment grande pour contenir toute l'information en entrée. Éventuellement, une phase de réduction de la dimension peut être appliquée (en post-traitement) sur les embeddings construits en sortie de GeoEmb.

2.1.1 Initialisation et dimension d'embedding

D'abord, nous assumons qu'au tout premier instant de l'historique d'interactions, les nœuds n'ont pas de liens les uns avec les autres (dû au fait que les interactions n'ont pas encore eu lieu). Dans la représentation vectorielle, cela se traduit par une équidistance entre les embeddings. Cela implique une dimension minimale de l'espace vectoriel latent de représentation. Par exemple, il est impossible d'avoir 4 points équidistants les uns des autres dans un espace à dimension inférieure ou égale à 2 (par exemple dans le plan), la figure géométrique correspondant à une telle configuration étant une figure qui s'inscrit dans un volume, à savoir un tétraèdre régulier (cf. figure 2.1).

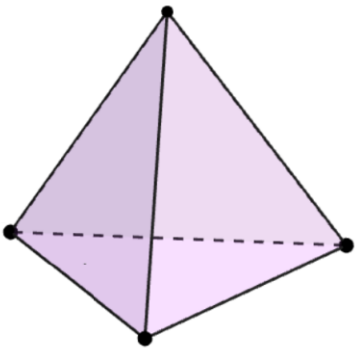


FIGURE 2.1: Tétraèdre régulier : 4 points équidistants les uns des autres s'inscrivent dans un espace de dimension ≥ 3.

D'une manière générale, la dimension minimale pour une représentation vectorielle équidistante pour N points $(p_1, p_2, \ldots, p_N)$ est de $N-1$. Par exemple, si la métrique choisie pour les distances entre les embeddings est la distance linéaire, une possible configuration des embeddings d'initialisation est :

$$\begin{aligned} p_1 &= (0,0,0,...,0) \\ p_2 &= (1,0,0,...,0) \\ p_3 &= (\frac{1}{2}, \frac{\sqrt{3}}{2}, 0, ..., 0) \\ &... \\ p_N &= (\frac{\alpha_1}{2}, \frac{\alpha_2}{3}, \frac{\alpha_3}{4}, ..., \frac{\alpha_{N-2}}{N-1}, \alpha_{N-1}) \end{aligned} \tag{2.1}$$

avec $\alpha_i = \sqrt{\frac{i+1}{2i}}$. Ce résultat peut être déduit à partir de l'égalité : $\alpha_i^2 = \frac{i^2-1}{i^2}\alpha_{i-1}^2$. La démonstration est fournie en annexe A.

La matrice d'initialisation $M_{t=0}$ peut donc être exprimée comme suit :

$$M_{t=0} = \begin{pmatrix} 0 & 0 & 0 & 0 & ... & 0 & 0 \\ 1 & 0 & 0 & 0 & ... & 0 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & ... & 0 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{6} & \frac{\sqrt{2}}{\sqrt{3}} & 0 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \frac{\alpha_3}{4} & \frac{\alpha_4}{5} & ... & \alpha_{N-2} & 0 \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \frac{\alpha_3}{4} & \frac{\alpha_4}{5} & ... & \frac{\alpha_{N-2}}{N-1} & \alpha_{N-1} \end{pmatrix} \tag{2.2}$$

À signaler que la dimension minimale de représentation ne concerne pas uniquement l'initialisation : N points sont à priori représentables dans un espace de dimension $N-1$ (sauf dans le cas de points alignés, co-planaires, co-volumiques, ..., co-hyperplanaires).

Cela étant dit, nous remarquons que cette expression présente certaines difficultés de manipulation. En effet, les composants des vecteurs tendent à s'annuler avec l'agrandissement de N :

$$\lim_{i \to \infty} \frac{\alpha_i}{i+1} = 0 \tag{2.3}$$

Moyennant l'ajout d'une dimension, nous pouvons formuler les embeddings $(p_1, p_2, \ldots, p_N)$ ainsi que la matrice $M_{t=0}$ de manière plus simple et plus intuitive :

$$\begin{aligned} p_1 &= (1, 0, ..., 0) \\ p_2 &= (0, 1, ..., 0) \\ &... \\ p_N &= (0, 0, ..., 1) \end{aligned} \tag{2.4}$$

et

$$M_{t=0} = I_N = \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & 1 \end{pmatrix} \tag{2.5}$$

Cela revient à un encodage *one-hot* pour les embeddings d'initialisation.

2.1.2 Dynamique temporelle des embeddings

Afin d'illustrer l'évolution du réseau induite par les interactions entre les nœuds, nous parcourons le déroulement de l'historique des interactions en les considérant une à une. Chaque interaction conduit à un *rapprochement* entre les représentations des nœuds concernés, à savoir ceux qui interagissent, mais également ceux qui se trouvent à proximité de ces derniers : deux nœuds n'interagissant pas directement, mais partageant un nœud *voisin* commun devraient se retrouver relativement proches dans leurs représentations vectorielles. L'idée est donc d'opérer une transformation impactant l'espace vectoriel pour rapprocher les régions latentes impliquées dans les interactions. Dépendamment de la métrique de distance considérée, ce rapprochement est opéré via une expression mathématique différente.

2.1.2.1 Rapprochement linéaire

La première métrique de distance envisagée est celle de la distance linéaire :

$$dist(n_i, n_j) = \|p_i - p_j\| \tag{2.6}$$

où $dist(n_i, n_j)$ représente la distance entre deux nœuds n_i et n_j, et p_i (respectivement p_j) représente le vecteur d'embedding de n_i (respectivement n_j) dans l'espace latent. Comme évoqué précédemment, une interaction entre 2 nœuds n_i et n_j est censée rapprocher leurs embeddings, mais également ceux des autres nœuds d'autant plus fortement qu'ils sont proches de n_i et/ou n_j. La transformation imaginée répondant à ces critères peut être formulée comme suit : pour une interaction entre n_i et n_j, la projection orthogonale de P_k (point d'embedding correspondant à un nœud n_k) sur la droite (P_i,P_j) se rapproche du milieu du segment $[P_i, P_j]$ d'un facteur ρ, avec $0 < \rho < 1$ (cf. figure 2.2).

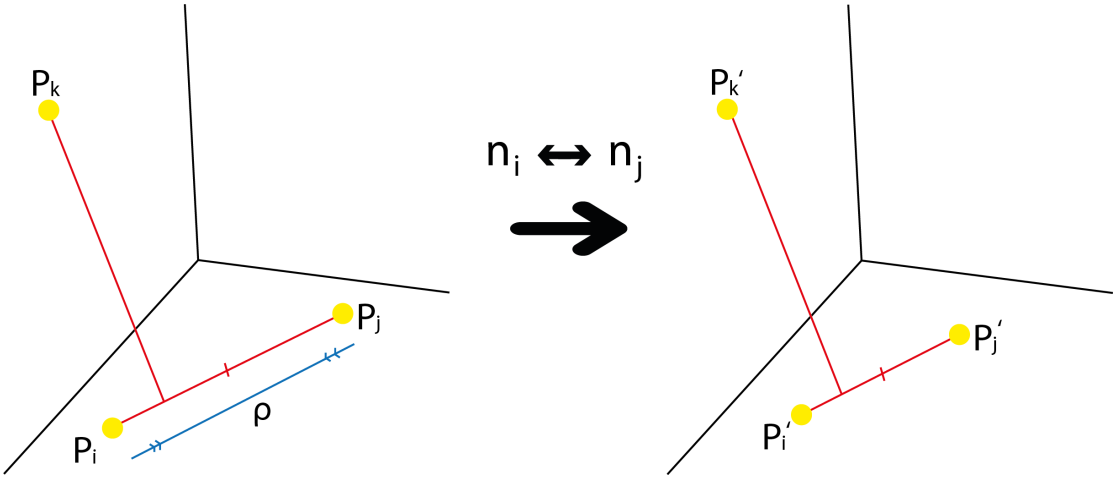


FIGURE 2.2: Effets du rapprochement linéaire impliqué par l'interaction entre n_i et n_j.

Cette transformation peut être exprimée comme suit :

$$\overrightarrow{OP'_k} = \overrightarrow{OP_k} + \rho \cdot \frac{\overrightarrow{OP_k} \cdot \overrightarrow{M_{i,j}P_i}}{\left\|\overrightarrow{M_{i,j}P_i}\right\|^2} \cdot \overrightarrow{M_{i,j}P_i} \tag{2.7}$$

où O est l'origine du référentiel, $M_{i,j}$ le milieu du segment $[P_i, P_j]$ et P'_k est le point d'embedding du nœud n_k à l'issue de l'interaction entre n_i et n_j.

Bien que cette transformation corresponde dans les grandes lignes aux contraintes fixées, elle pose néanmoins un problème. En effet, considérons le cas où la toute première interaction de l'historique concerne 2 nœuds n_i et n_j, et regardons l'impact sur l'embedding p_k d'un nœud n_k. Avant l'interaction, au vu de la matrice d'initialisation $M_{t=0}$, les distances (deux à deux) entre les embeddings de n_i, n_j, n_k sont égales à $\sqrt{2}$. Après l'interaction, les distances transformées sont :

$$\begin{aligned} \left\|\overrightarrow{P'_iP'_j}\right\| &= \rho \cdot \sqrt{2} \\ \left\|\overrightarrow{P'_kP'_i}\right\| = \left\|\overrightarrow{P'_kP'_j}\right\| &= \sqrt{\frac{3+\rho^2}{2}} \end{aligned} \tag{2.8}$$

Nous constatons que les distances latentes entre les paires de nœuds (n_i, n_k) et (n_j, n_k) sont impactées par l'interaction, bien que les nœuds n_i et n_k (ou n_j et n_k) n'aient pas été en contact, que ce soit de manière directe (i.e. via une interaction $n_i \leftrightarrow n_k$) ou indirecte (i.e. via une chaîne d'interactions $n_i \leftrightarrow n_x \leftrightarrow ... \leftrightarrow n_y \leftrightarrow n_k$). Il apparaît donc qu'une métrique de distance linéaire présente certains inconvénients.

2.1.2.2 Rapprochement angulaire

La deuxième métrique envisagée est celle de la distance angulaire :

$$dist(n_i, n_j) = |\widehat{P_iOP_j}| = \arccos\left(\frac{\overrightarrow{OP_i} \cdot \overrightarrow{OP_j}}{\left\|\overrightarrow{OP_i}\right\| \cdot \left\|\overrightarrow{OP_j}\right\|}\right) \tag{2.9}$$

Nous pouvons remarquer que les normes des vecteurs d'embedding n'ont aucun impact sur les distances. Par conséquent, nous choisissons de représenter les embeddings sous forme de vecteurs unitaires :

$$\left\|\overrightarrow{OP_i}\right\| = 1 \tag{2.10}$$

Par la même, la formulation des distances s'en trouve simplifiée :

$$dist(n_i, n_j) = \arccos\left(\overrightarrow{OP_i} \cdot \overrightarrow{OP_j}\right) \tag{2.11}$$

De la même manière que celle adoptée pour la métrique linéaire, nous modélisons la transformation à l'issue d'une interaction entre deux nœuds n_i et n_j comme suit : la projection orthogonale de P_k (correspondant à un nœud n_k) sur le plan formé par les points $\{O, P_i, P_j\}$ se rapproche angulairement de la bissectrice de l'angle $|\widehat{P_iOP_j}|$ d'un facteur ρ, avec $0 < \rho < 1$ (cf. figure 2.3).

Cette transformation peut être exprimée comme suit :

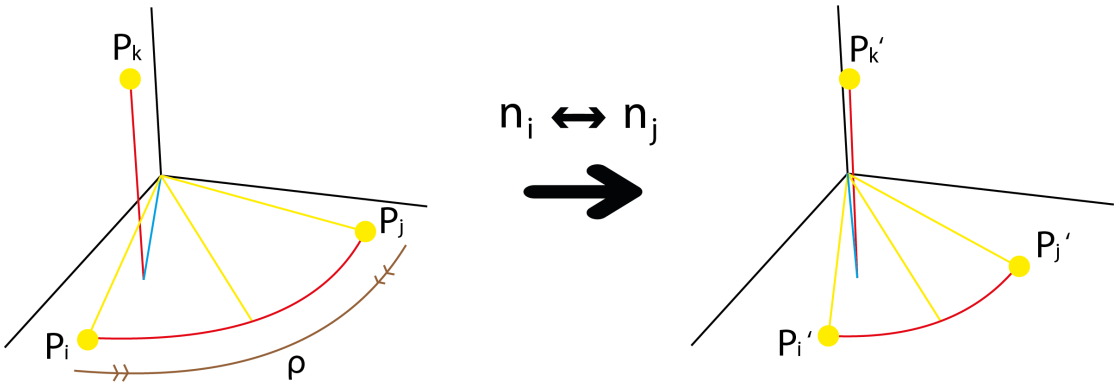


FIGURE 2.3: Effets du rapprochement angulaire impliqué par l'interaction entre n_i et n_j.

$$\begin{aligned}\overrightarrow{OP'_k} = \overrightarrow{OP_k} &+ \left\|\overrightarrow{OP_k^{\perp}}\right\| \cdot \cos\left(\rho \cdot \arccos\left(\frac{\overrightarrow{OP_k^{\perp}} \cdot \overrightarrow{IJ^{\parallel}}}{\left\|\overrightarrow{OP_k^{\perp}}\right\|}\right)\right) \cdot \overrightarrow{IJ^{\parallel}} \\ &+ \left\|\overrightarrow{OP_k^{\perp}}\right\| \cdot \sin\left(\rho \cdot \arcsin\left(\frac{\overrightarrow{OP_k^{\perp}} \cdot \overrightarrow{IJ^{\perp}}}{\left\|\overrightarrow{OP_k^{\perp}}\right\|}\right)\right) \cdot \overrightarrow{IJ^{\perp}}\end{aligned} \tag{2.12}$$

avec

$$\overrightarrow{IJ^{\parallel}} = \frac{\overrightarrow{P_iP_j}}{\left\|\overrightarrow{P_iP_j}\right\|} \qquad \overrightarrow{IJ^{\perp}} = \frac{\overrightarrow{OP_i} + \overrightarrow{OP_j}}{\left\|\overrightarrow{OP_i} + \overrightarrow{OP_j}\right\|}$$

$$\overrightarrow{OP_k^{\perp}} = \frac{\begin{pmatrix}\left(\left(\overrightarrow{OP_i} \cdot \overrightarrow{OP_k}\right) - \left(\overrightarrow{OP_i} \cdot \overrightarrow{OP_j}\right) \cdot \left(\overrightarrow{OP_j} \cdot \overrightarrow{OP_k}\right)\right) \cdot \overrightarrow{OP_i} + \\ \left(\left(\overrightarrow{OP_j} \cdot \overrightarrow{OP_k}\right) - \left(\overrightarrow{OP_i} \cdot \overrightarrow{OP_j}\right) \cdot \left(\overrightarrow{OP_i} \cdot \overrightarrow{OP_k}\right)\right) \cdot \overrightarrow{OP_j}\end{pmatrix}}{1 - \left(\overrightarrow{OP_i} \cdot \overrightarrow{OP_j}\right)^2} \tag{2.13}$$

Le vecteur $\overrightarrow{OP_k^{\perp}}$ représente la projection orthogonale de P_k sur le plan formé par les points $\{O, P_i, P_j\}$.

L'intérêt de cette transformation est que le changement des distances ne s'opère que entre les paires de nœuds impactées par l'interaction. En d'autres termes, à l'issue d'une interaction entre deux nœuds n_i et n_j, la distance entre les points d'embedding P_u et P_v est diminuée uniquement si les nœuds n_u et n_v ont chacun interagi (directement ou indirectement) avec n_i et/ou n_j. À l'inverse, un nœud n'ayant pas interagi avec n_i ou n_j verra ses distances vis-à-vis des autres nœuds rester inchangées.

2.1.3 Complexité des traitements, erreur cumulée et scalabilité

Un réseau d'interactions issu du monde réel comprend souvent des dizaines de milliers, des centaines de milliers voire des millions de nœuds, et un nombre d'interaction du même ordre de grandeur. Étant donné la modélisation d'embedding considérée, de telles volumétries impliquent un nombre très important de calculs élémentaires. En effet, à raison de N nœuds et de I interactions, le graphe temporel est représenté par une séquence de I matrices carrées de taille $N \times N$. Chacune des matrices est calculée à partir de celle qui la précède temporellement. Cela implique un total de $(I-1) \cdot N^2$ calculs de positions. Il est évident que le calcul des embeddings devient rapidement de moins en moins scalable avec l'agrandissement de N et/ou de I.

De plus, la précision des valeurs calculées des vecteurs d'embedding est impactée par le nombre d'interactions I. En effet, sous des langages de programmation Python ou R, les nombres sont en général encodés en virgule flottante sur 53 bits. Cela implique que la retranscription d'un nombre introduit une imprécision de l'ordre de 10^{-16}. Cela pose un problème de divergence de l'erreur lorsque plusieurs calculs sont enchainés les uns à la suite des autres, comme c'est le cas pour la modélisation d'embedding considérée. Empiriquement, nous constatons la perte d'approximativement un chiffre significatif derrière la virgule pour le calcul de la mise à jour des embeddings suite à 87 interactions. Il est donc nécessaire de pouvoir contrôler la précision à laquelle les calculs se font. Cela est rendu possible par le recours à des librairies du type *mpfr* pour python et *Rmpfr* pour R.

Néanmoins, le nombre important d'interactions dans un graphe temporel nécessite une précision excessivement grande (avec des nombres encodés sur plus de 3000 bits quand $I > 10^5$), ce qui ne semble pas être réalisable. Or, l'augmentation de la précision des calculs se fait au détriment du temps de traitement. En conséquence, ce type d'approches (i.e. contrôle de la précision via le recours à des librairies telles que *mpfr* ou *Rmpfr*) ne suffit pas à résoudre le problème du cumul de l'erreur des calculs en pratique.

2.1.3.1 Sous-espaces d'embedding

Une autre manière de réduire la complexité des traitements (et par conséquent de diminuer la précision nécessaire pour les traitements ainsi que le temps de traitement) est de chercher des simplifications mathématiques à la formulation de la transformation qui a lieu à l'issue d'une interaction. Certaines observations peuvent être utiles dans ce cadre.

Comme indiqué auparavant, pour la métrique de distance angulaire, seuls les nœuds impactés par une interaction voient leurs embeddings changer à l'issue de celle-ci. Cela

veut dire qu'il n'est pas nécessaire de calculer les embeddings des nœuds n'ayant jamais interagi, de manière directe ou indirecte, avec les nœuds impliqués dans l'interaction en question. Cela réduit considérablement la complexité globale des traitements, mais pas suffisamment : en général, au bout d'un certain nombre d'interactions, la plupart des nœuds du graphe sont reliés les uns aux autres directement ou indirectement (i.e. le graphe ne contient plus qu'une seule composante connexe). À partir de ce stade, la moindre interaction impacte chacun des embeddings des nœuds.

Il est néanmoins possible de pousser l'idée de restreindre les calculs aux nœuds impactés par une interaction. Imaginons le cas d'un nœud n_a qui, durant tout l'historique, n'interagit qu'avec un seul autre nœud n_A. Il est alors possible de connaître la distance entre n_a et un nœud quelconque n_B en se basant sur les distances $dist(n_a, n_A)$ et $dist(n_A, n_B)$:

$$\overrightarrow{OP_a} \cdot \overrightarrow{OP_B} = \left(\overrightarrow{OP_a} \cdot \overrightarrow{OP_A}\right) \cdot \left(\overrightarrow{OP_A} \cdot \overrightarrow{OP_B}\right) \tag{2.14}$$

Cela est dû au fait que, dans cette configuration, les plans $\{O, P_a, P_A\}$ et $\{O, P_A, P_B\}$ sont perpendiculaires. En se basant sur ce constat, imaginons le cas où, à un instant donné de l'historique, deux ensembles de nœuds $X = \{n_i^x\}$ et $Y = \{n_j^y\}$ sont connectés par un unique nœud n_b (que nous appellerons *nœud pont*). C'est-à-dire qu'il n'y a eu aucune interaction entre un nœud de X et un nœud de Y, mais que les nœuds de chacun des deux ensembles ont pu interagir entre eux, ou avec le nœud pont n_b. Alors de la même manière, les 2 sous-espaces d'embedding des 2 ensembles de nœuds $X \bigcup \{n_b\}$ et $Y \bigcup \{n_b\}$ sont perpendiculaires et vérifient, pour $n_i^x \in X$ et $n_j^y \in Y$ quelconques :

$$\overrightarrow{OP_i^x} \cdot \overrightarrow{OP_j^y} = \left(\overrightarrow{OP_i^x} \cdot \overrightarrow{OP_b}\right) \cdot \left(\overrightarrow{OP_b} \cdot \overrightarrow{OP_j^y}\right) \tag{2.15}$$

Par conséquent, à l'issue d'une interaction impliquant des nœuds d'un des deux ensembles, il n'est pas nécessaire d'opérer des calculs sur les nœuds de l'autre ensemble. Ce type de simplifications est utile car les sous-graphes reliés par des nœuds ponts sont des configurations courantes. Pour aller plus loin, il est théoriquement possible de généraliser cette démarche en considérant le cas des sous-graphes reliés par 2 nœuds ponts ou plus. Néanmoins, ce type de calculs est mathématiquement ardu et réduit l'intérêt de simplification de la démarche.

2.1.4 Résultats et limites

À des fins d'évaluation, GeoEmb a été appliqué à un jeu de données de taille réduite. Il s'agit d'une liste de co-publications datées entre les chercheurs d'une équipe de recherche (DECIDE) de l'IMT Atlantique (cf. figure 2.4). Une co-publication est considérée comme une interaction entre chercheurs.

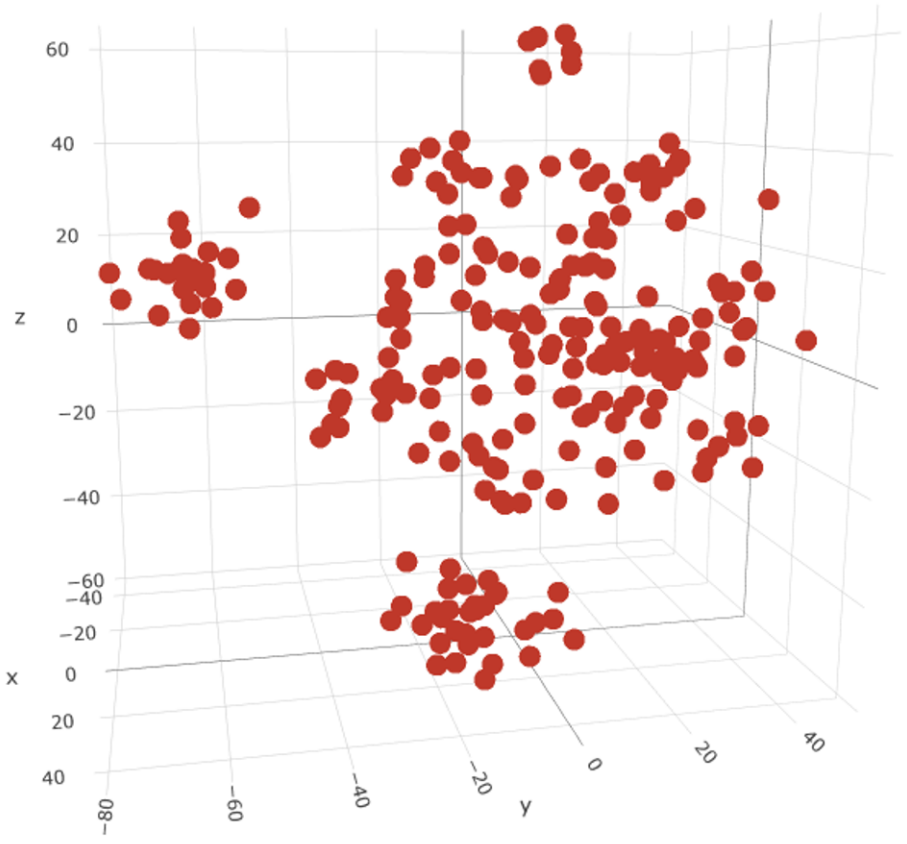


FIGURE 2.4: Exemple d'un instantané d'embeddings des chercheurs de l'équipe de recherche DECIDE. Les embeddings ont été soumis à une réduction de la dimension en utilisant t-SNE [1]. Certains clusters retrouvés correspondraient bien à des habitudes de co-publications au sein de DECIDE.

La limite principale de la méthode GeoEmb concerne la complexité des traitements. Pour les raisons expliquées auparavant, GeoEmb est limité au traitement de graphes de quelques centaines de nœuds et de quelques milliers d'interactions, et ce malgré les simplifications rendues possibles en considérant des sous-espaces d'embedding. Par conséquent, GeoEmb n'est généralement pas applicable aux graphes temporels issus de données du monde réel.

2.2 TemporalNode2vec : une méthode inspirée de l'état de l'art

La deuxième méthode conçue, TemporalNode2vec (TN2V), s'inscrit dans la lignée des approches de l'état de l'art. En effet, TN2V reprend le modèle [18], développé pour l'embedding dynamique de corpus textuels, et l'adapte au cas des graphes temporels : l'idée est de remplacer les phrases (séquences de mots) en entrée, par des séquences de nœuds obtenues en recourant à des marches aléatoires à la manière de node2vec (N2V) [6]. En ce sens, TN2V peut être considéré comme un cas d'étude du modèle [18] appliqué aux graphes temporels. Ainsi, de manière similaire à [18], le principal apport de TN2V réside dans la manière d'incorporer l'information temporelle : les embeddings des différents pas de temps sont appris conjointement, améliorant ainsi les performances des tâches de classification de nœuds. A noter que TN2V se distingue du modèle de [18] non seulement au niveau du type de données en entrée (où l'on tire avantage de la flexibilité offerte par N2V dans la conception des marches aléatoires), mais aussi lors de l'initialisation des embeddings avant leur optimisation comme cela est explicité dans la section 2.2.3.5. TN2V a fait l'objet d'un article [19] accepté à la 8$^{\text{ème}}$ édition de la " International Conference on Complex Networks and Their Applications "et publié dans ses proceedings.

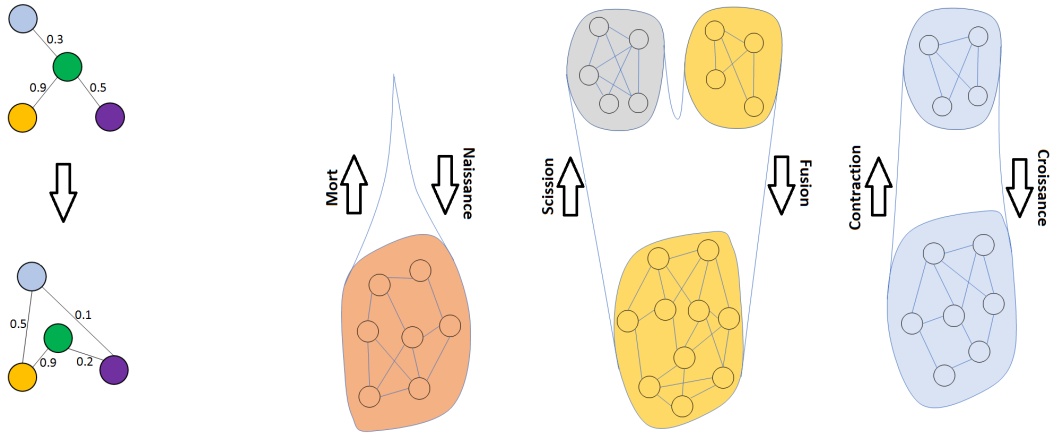
2.2.1 État de l'art

Les dernières années ont vu émerger une multitude de travaux dédiés à l'extraction des informations structurelles importantes des graphes, telles que les caractéristiques des communautés ou encore les modèles de diffusion de l'information. Cependant, afin de faciliter l'exploitation des données des graphes pour les modèles d'apprentissage automatique en aval, une étape préliminaire élémentaire est celle de la construction d'une représentation pertinente des nœuds et des arêtes. Les approches classiques d'apprentissage automatique qui s'inscrivent dans ce cadre reposent sur des heuristiques définies par l'utilisateur [20–22]. Néanmoins, ces méthodes sont chronophages en termes d'ingénierie vu qu'elles ne sont pas *data-driven*. Par ailleurs, elles ne présentent pas de garanties de performances pour des tâches de prédiction différentes de celles pour lesquelles elles ont été conçues.

Plus récemment, les méthodes d'apprentissage de représentation [23] (i.e. les méthodes d'embedding) ont gagné en popularité. Elles couvrent un large spectre de domaines de la science des données allant du traitement d'images [24] et de vidéos [25] aux séries temporelles [26]. Ces méthodes data-driven encodent les données dans une représentation

générique, indépendante des tâches d'apprentissage automatique en aval. Elles sont donc à considérer comme une étape de prétraitement. Les premières applications de l'embedding de données se sont attachées à l'exploration de textes [27]. Compte tenu des avantages de ces approches, elles ont été adaptées aux graphes, entre autres structures de données. Certains de ces modèles se basent sur la factorisation de matrices (Laplacian eigenmaps [2], GraRep [3], HOPE [4]) et d'autres techniques reprennent la notion des marches aléatoires ([5–7]). Un autre type de méthodes utilise les avancées récentes en apprentissage profond : les autoencodeurs pour DNGR [8], SDNE [9], ARGAE [28] et les réseaux de neurones à convolution comme GCN [10].

Bien que les méthodes d'embedding de graphes aient été disruptives et se soient imposées comme des références, la plupart d'entre elles se focalisent sur des graphes statiques où la structure des nœuds et des arêtes demeure fixe dans le temps, éludant la dimension temporelle qui reste fondamentale dans l'appréhension des phénomènes d'interaction. Par exemple, la prédiction du churn est moins efficace sur des graphes statiques quand les marqueurs de ce phénomène se produisent généralement dans le temps [29]. Cependant, de récents travaux intégrant le temps commencent à émerger. Par exemple, [12, 13] utilisent l'information temporelle disponible pour construire des embeddings globaux (i.e. un embedding par nœud, sur toute la durée de l'historique) plus efficaces. En l'occurrence, le problème que nous adressons est différent : l'objectif de l'embedding temporel est d'obtenir une représentation d'un graphe à chaque pas de temps et à différentes échelles. L'évolution locale des nœuds et des arêtes au fil du temps est capturée ainsi que les changements des communautés à une échelle plus globale (Figure 2.5).



(a) Échelle locale (b) Échelle globale

FIGURE 2.5: Exemples d'évolutions temporelles d'un graphe aux échelles locale et globale.

Une manière simpliste serait d'appliquer des méthodes d'embedding statiques aux

différents pas de temps indépendamment les uns des autres. Certaines approches apprennent séparément les embeddings des différents pas de temps puis recherchent une transformation linéaire optimale à appliquer aux matrices en sortie, minimisant la distance entre les embeddings consécutifs dans le temps [30]. [14] adapte N2V aux cas des graphes temporels : l'idée est d'apprendre la représentation d'un nœud à un pas de temps donné en l'initialisant avec celle du pas de temps précédent. D'autres approches adaptent de récentes techniques de deep-learning (les RNN pour [31], les GCN pour [32]) pour prendre en compte l'aspect dynamique pour l'apprentissage des représentations des nœuds. D'autre part, des avancées ont été faites via une exploration sophistiquée des voisinages des nœuds, en contraignant les marches aléatoires sur les clusters dans le cas de [33], ou en précalculant des voisinages *diffusion-based* [34]. À noter qu'il existe une multitude d'autres techniques adressant le problème d'embeddings de graphes temporels [15–17, 35–38].

La plupart de ces méthodes peuvent être sensibles à la sparsité des données : par exemple, un nœud qui n'interagit pas durant un pas de temps pourrait être plongé loin de ses embeddings précédents et suivants. À l'inverse, les embeddings en sortie de TN2V sont moins sensibles à la sparsité des interactions du fait du lissage temporel inhérent à la méthode.

Enfin, il est opportun de signaler que la conception de TN2V a précédé les travaux de l'état de l'art — adressant le problème de l'apprentissage de représentations d'un graphe temporel — cités ci-dessus, exception faite de DynamicTriad [15] et dynnode2vec [14].

2.2.2 Définition du problème

Nous considérons un graphe temporel composé de N nœuds $V = \{v_1, \dots, v_N\}$ connectés par des interactions datées $I = \{i_{j,k,t}\}$ où $i_{j,k,t}$ représente une interaction entre v_j et v_k à l'instant t. Le but de l'embedding temporel est de calculer un vecteur représentatif de chaque nœud à chaque pas de temps.

Afin d'éviter les problèmes liés à la scalabilité rencontrés par le modèle GeoEmb et explicités en 2.1.4, la dimension temporelle est fragmentée en T pas de temps. Dans le cadre de nos expérimentations, le découpage se fait de manière temporellement régulière : les différents pas de temps couvrent des intervalles de temps de durées égales (différentes stratégies de découpage temporel sont possibles, cf. 5.2.1). Dans chacun d'entre eux, nous agrégeons les interactions entre les paires de nœuds, pour former des arêtes pondérées (où les poids représentent le nombre d'interactions groupées). Cela se traduit par une transformation du graphe temporel en entrée en une séquence de

T graphes $\{G_1, G_2, \ldots, G_T\}$ où G_t est le graphe pondéré non orienté représentant les interactions du $t^{\text{ème}}$ pas de temps.

Nous définissons l'embedding temporel des nœuds comme suit : étant donné une séquence de graphes $\{G_1, \ldots, G_T\}$, nous cherchons à apprendre pour chaque pas de temps t ($t \in [\![1, T]\!]$) des fonctions de correspondance $f_t : V \to \mathbb{R}^d$. Ici, d est la dimension d'embedding ($d << N$). La sortie attendue est un ensemble de T matrices de taille $N \times d$ où la $t^{\text{ème}}$ matrice (désignée par U_t) contient les vecteurs d'embedding de tous les nœuds au pas de temps t.

2.2.3 TemporalNode2vec

Comme mentionné précédemment, TemporalNode2vec est une adaptation du modèle d'embedding dynamique de mots [18] au cas des graphes temporels. Concrètement, moyennant une marche aléatoire, nous avons transformé la donnée en entrée (i.e. séquence de graphes) en une donnée compatible avec l'application du modèle [18]. Par conséquent, les briques constitutives de TemporalNode2vec décrites dans cette section (à l'exception de 2.2.3.5) sont des rappels et des explications de notions exposées dans [6] et [18].

2.2.3.1 Marches aléatoires

Le point de départ d'une partie des approches d'embedding de graphes est la transposition de techniques d'embedding de mots, ayant prouvé par ailleurs leur efficacité en termes de qualité d'inférence par rapport aux travaux précédents. Généralement, cette adaptation est rendue possible en transformant les données des graphes en *phrases* de nœuds. DeepWalk (DW) [5] a apporté une méthode permettant de construire ces séquences de nœuds appelées *walks*, inspirée des premières applications de marches aléatoires [39]. L'idée est de simuler une *marche* probabiliste sur les nœuds d'un graphe à travers ses arêtes (non)orientées (non)pondérées. Node2vec (N2V) [6] étend les possibilités des marches aléatoires en offrant plus de flexibilité dans la manière dont les voisinages des nœuds sont explorés, en privilégiant l'exploration locale (breadth-first sampling ou BFS) ou la découverte de macro-structures (depth-first sampling ou DFS). Cette flexibilité, rendue possible grâce à l'ajout de deux hyperparamètres variables, permet d'améliorer la qualité des embeddings. Dans TN2V, nous construisons nos marches aléatoires de la même manière. Cela signifie que, étant donné une séquence de graphes temporels $\{G_1, \ldots, G_T\}$, nous constituons T ensembles de marches aléatoires. Nous obtenons alors $T \times M \times N$ séquences de nœuds de longueur l, où M est le nombre de marches partant de chaque nœud.

2.2.3.2 Les matrices $PPMI$

Comme mentionné précédemment, la notion de localité propre à une représentation sous forme de graphe est un aspect important que les vecteurs d'embedding doivent conserver. Il est possible de capturer les structures de voisinage des nœuds en utilisant des statistiques sur les fréquences auxquelles les nœuds co-apparaissent dans l'ensemble des marches aléatoires considérées [8]. Nous définissons la matrice $PPMI$ (*positive pointwise mutual information*) comme suit :

$$\begin{aligned} &\forall(v_1,\ v_2 \neq v_1,\ t) \in V^2 \times [\![1,\ T]\!] : \\ PPMI(v_1,\ v_2)_t &= max\left(0, PMI(v_1,\ v_2)_t\right) \\ &= max\left(0,\ log\left(\theta\ \frac{|v_1,\ v_2|_t^w \cdot N}{|v_1|_t \cdot |v_2|_t}\right)\right) \end{aligned} \tag{2.16}$$

où $|v_1, v_2|_t^w$ est le nombre de fois où v_1 et v_2 co-apparaissent dans l'ensemble des marches aléatoires de G_t sur une fenêtre glissante de taille w (i.e. le nombre de fois où les 2 nœuds apparaissent ensemble dans une marche aléatoire en étant espacés d'au plus $w-1$ pas), $|v_i|_t$ est le nombre d'occurrences de v_i dans l'ensemble des marches de G_t, et θ est un hyperparamètre réglable. Les matrices $PPMI$ peuvent être considérées comme des matrices temporelles des similarités entre les nœuds. D'autre part, il convient de mentionner que nous utilisons des matrices *positive* PMI afin de réduire l'instabilité des entrées des matrices PMI : les valeurs $log\left(\theta\ \frac{|v_1,\ v_2|_t^w \cdot N}{|v_1|_t \times |v_2|_t}\right)$ peuvent produire des valeurs largement négatives, assimilables à une co-apparition nulle ou quasi nulle. L'hyperparamètre θ contrôle le compromis entre la stabilité des entrées PMI et une rare co-apparition. Cette manière de borner les matrices PMI est également utilisée dans [40].

2.2.3.3 Termes de la fonction objectif

L'idée principale de plusieurs modèles d'embedding de graphes statiques [3–6, 41] est d'apprendre des matrices U_t satisfaisant :

$$u_1(t)^T \cdot u_2(t) \simeq PPMI(v_1,\ v_2)_t \tag{2.17}$$

où $u_i(t)$ est le vecteur d'embedding de v_i au pas de temps t. N2V utilise conjointement différentes techniques (*CBOW* ou *skip-gram*, *negative sampling*) pour satisfaire implicitement (2.17). Dans le cas de l'embedding de mots, [40] observe que cela est équivalent à la factorisation *low-rank* des matrices PMI décalée d'une constante (θ dans l'équation 2.16). Par conséquent, le terme statique L_{St} de la fonction objectif peut

être formulé comme suit :

$$L_{St} = \sum_{t=1}^{T} \left\| PPMI_t - U_t \, U_t^T \right\|_F^2 \tag{2.18}$$

Un important verrou de l'apprentissage des embeddings d'un graphe temporel est appelé l'alignement : les vecteurs d'embedding des différents pas de temps ne s'inscrivent a priori pas dans le même espace latent. Par exemple, L_{St} est invariant par rapport aux différentes rotations et réflexions des matrices U_t :

$$\begin{aligned} L_{St} &= L_{St}(U_1, \ \ldots \ , U_T) \\ &= L_{St}(U_1 \, R_1, \ \ldots \ , U_T \, R_T) \end{aligned} \tag{2.19}$$

où $R_t \in O(d)$ et $O(d)$ est le groupe orthogonal (i.e. le groupe des rotations/réflexions) de dimension d. Il est donc nécessaire d'ajouter un terme de lissage L_{Sm} à notre fonction objectif :

$$L_{Sm} = \sum_{t=2}^{T} \| U_t - U_{t-1} \|_F^2 \tag{2.20}$$

Après l'ajout d'un dernier terme pénalisant les larges valeurs (*low-rank data fidelity*) comme adopté dans [18, 42], la fonction objectif finale à minimiser s'exprime comme suit :

$$\begin{aligned} L = \ & L_{St} \ + \tau \, L_{Sm} \ + \lambda \, L_{LR} \\ = & \sum_{t=1}^{T} \left\| PPMI_t - U_t \, U_t^T \right\|_F^2 + \tau \ \sum_{t=2}^{T} \| U_t \ - \ U_{t-1} \|_F^2 + \lambda \ \sum_{t=1}^{T} \| U_t \|_F^2 \end{aligned} \tag{2.21}$$

2.2.3.4 Optimisation de la fonction objectif

Les graphes temporels issus du monde réél contiennent généralement des dizaines de milliers de nœuds, voire plus. Cela signifie que les matrices $PPMI$ peuvent ne pas tenir en mémoire, bien qu'elles soient éparses. Par conséquent, il est judicieux de procéder à des simplifications de la fonction objectif avant de chercher à la minimiser. De la même manière que [18], nous introduisons des matrices W_t et un hyperparamètre γ pour casser la symétrie de la factorisation $PPMI$. Ainsi, nous pouvons écrire notre fonction objectif modifiée comme suit :

$$\begin{aligned} & \sum_{t=1}^{T} \left\| PPMI_t - U_t \, W_t^T \right\|_F^2 \ + \lambda \ \sum_{t=1}^{T} \| U_t \|_F^2 + \lambda \ \sum_{t=1}^{T} \| W_t \|_F^2 \\ & + \tau \ \sum_{t=2}^{T} \| U_t \ - \ U_{t-1} \|_F^2 + \tau \ \sum_{t=2}^{T} \| W_t \ - \ W_{t-1} \|_F^2 \ + \ \gamma \ \sum_{t=1}^{T} \| U_t \ - \ W_t \|_F^2 \end{aligned} \tag{2.22}$$

La minimisation de (2.22) pour U_t (ou pour W_t) peut alors simplement être réalisée par annulation du gradient. Concrètement, l'objectif est de minimiser l'expression $U_t \cdot A = B$ relativement à U_t, avec :

$$\begin{aligned} A &= W_t^T\, W_t + (\lambda + 2\tau + \gamma)\, I \\ B &= PPMI_t\ W_t + \gamma\, W_t + \tau\, (U_{t-1} + U_{t+1}) \end{aligned} \tag{2.23}$$

pour $t \in [\![2,\, T-1]\!]$ et avec des constantes ajustées dans le cas de $t \in \{0,\, T\}$. Afin de gérer le cas des matrices $PPMI$ de grande taille, nous utilisons l'algorithme de descente de coordonnées par blocs (*block coordinate descent*) afin de résoudre $U_t \cdot A = B$. Ce faisant, à chaque itération, seuls des blocs de $PPMI_t$ sont chargés en mémoire et seuls des blocs de U_t (et alternativement de W_t) sont mis à jour.

2.2.3.5 Initialisation des embeddings temporels

Comme mentionné précédemment, les matrices $PPMI$ peuvent être vues comme des matrices de similarités entre les paires de nœuds. Elles peuvent donc être utilisées pour l'initialisation des embeddings pour accélérer la convergence de l'étape de minimisation de la fonction objectif. Pour $t \in [\![1, T]\!]$, nous initialisons U_t et W_t en récupérant certaines des lignes de $PPMI_t$. Nous choisissons les d lignes qui maximisent la variance après normalisation (en norme $L1$) : une variance élevée signifie une meilleure discrimination entre les nœuds.

Algorithme 1: TemporalNode2vec

Données en entrée: $(\{G_1, \ldots, G_T\}, V, d, l, w, p, q, \theta, \lambda, \gamma, \tau, iter)$
pour $t \leftarrow [\![1, T]\!]$ **faire**
  $marches = simulerMarches(G_t, l, p, q)$;
  $PPMI_t = calculerPPMI(marches, w, \theta)$;
fin
$\{U_1, \ldots, U_T, W_1, \ldots, W_T\} = initialiser(PPMI, d)$;
pour $i \leftarrow [\![1, iter]\!]$ **faire**
  pour $t \leftarrow [\![1, T]\!]$ **faire**
    pour $b \leftarrow batch(V)$ **faire**
      $U_t[b] = mettreAJour(W_t[b], \lambda, \gamma, \tau, PPMI_t)$;
      $W_t[b] = mettreAJour(U_t[b], \lambda, \gamma, \tau, PPMI_t)$;
    fin
  fin
fin
retourner $\{U_1, \ldots, U_T\}$

2.2.4 Expérimentations

2.2.4.1 Méthodes de référence

Comme mentionné précédemment, TN2V a plusieurs hyperparamètres : $\{l,\ w,\ p,\ q,\ \theta,\ \lambda,\ \gamma,\ \tau\}$. Il est donc difficile d'établir un jeu de valeurs pour chacun des hyperparamètres et de tester toutes les combinaisons possibles selon une *grid-search*. Pour contourner ce problème, nous choisissons de rechercher les paramètres optimaux un par un, en fixant les autres (des stratégies plus élaborées pour le tuning des hyperparamètres sont évoquées en 2.4.2.1). Le tableau 2.1 résume les valeurs testées.

Paramètre	**Valeurs testées**
l	{20, 40, 60, 80}
w	{4, 7, 10, 13, 16}
p	{0.2, 0.5, 1, 2, 5}
q	{0.2, 0.5, 1, 2, 5}
θ	{0.05, 0.25, 1, 4, 20}
λ	{0, 0.1, 0.3, 1, 3, 10}
γ	{0.1, 0.3, 1, 3, 10}
τ	{0, 0.1, 0.3, 1, 3, 10}

TABLEAU 2.1: Valeurs testées pour les différents hyperparamètres de TN2V.

Pour évaluer les performances de TN2V, nous considérons 3 méthodes de référence de l'état de l'art. Il est opportun de signaler, comme mentionné auparavant, que les travaux de conception de TN2V et de l'évaluation de ses performances ont précédé la majorité des approches d'embedding temporel de l'état de l'art.

- **DeepWalk** (DW) [5] : il s'agit de l'une des premières méthodes d'apprentissage de représentations des nœuds d'un graphe statique inspirée du traitement automatique du language et utilisant des marches aléatoires sur les arêtes. DW a deux hyperparamètres : la longueur de la marche l et la taille de la fenêtre de co-apparition w. Pour nos expérimentations, nous testons une grid-search selon $(l, w) \in \{20, 40, 60, 80\} \times \{4, 8, 12, 16\}$.
- **Node2vec** (N2V) [6] : N2V étend DW et permet plus de flexibilité quant à la découverte du voisinage des nœuds en ajoutant deux hyperparamètres : le paramètre de retour (*return*) p et le paramètre d'entrée-sortie (*in-out*) q. Pour nos expérimentations, nous gardons les valeurs de l et w donnant les meilleures performances dans les tests de DW, et effectuons une grid-search selon $(p, q) \in \{0.5, 1, 1.5, 2, 5\}^2$.

- **DynamicTriad** (DT) [15] : DynamicTriad (DT) est une approche d'embedding de graphes temporels qui porte sur la façon dont les triades de nœuds se ferment, i.e. comment une paire de nœuds partageant un nœud voisin commun peut se connecter via une arête aux pas de temps suivants. Cette méthode a deux hyper-paramètres : le poids de fermeture de triades β_0 et le paramètre de lissage temporel β_1. Pour nos expérimentations de comparaison, nous nous référons aux jeux de paramètres préconisés par les auteurs de DT et considérons donc une grid-search selon $(\beta_0, \beta_1) \in \{0.01, 0.1, 1, 10\}^2$.

En ce qui concerne la dimension d'embedding, nous choisissons $d = 48$ pour des raisons d'espace mémoire. Des valeurs plus petites de d sont testées dans la section 2.2.5.

2.2.4.2 Jeux de données

Afin de comparer les performances des différents modèles d'embedding, nous avons récolté 3 jeux de données issus du monde réel. Ces jeux de données doivent répondre à certains critères. D'abord, il doit s'agir de séquences de graphes datés. Par ailleurs, les jeux de données doivent inclure des métadonnées correspondant à la vérité de terrain de l'appartenance des nœuds à des communautés, à chaque pas de temps. Une partie de l'évaluation des différents modèles consiste à découvrir à quel point les embeddings en sortie préservent ces communautés.

- **AMiner** : Ce jeu de données[1] se compose de 51k chercheurs et de 624k relations de co-publication. Il se divise en 17 graphes pondérés datés, où un poids représente le nombre de co-publications communes entre une paire de chercheurs au cours d'un pas de temps. Étant donné que les articles sont publiés dans des conférences traitant de différents domaines de recherche, il est possible d'attribuer des labels aux auteurs : à chaque pas de temps, nous supposons qu'un chercheur appartient à une communauté (i.e. un domaine de recherche) si la majorité de ses articles sont publiés dans des conférences connexes. D'autre part, pour les expérimentations ayant trait à l'analyse de la dimension d'embedding, nous extrayons de ce jeu de données 3 échantillons de tailles différentes Smp_A, Smp_B et Smp_C (cf. tableau 2.2).

- **Yelp** : Ce jeu de données est un extrait du Yelp Challenge Dataset[2]. Il retrace les commentaires d'internautes sur des commerces (tels que des restaurants et des centres commerciaux). Nous considérons les utilisateurs et les commerces

[1] Nous utilisons la version dérivée d'ArnetMiner [43] extraite par [15].
[2] https://www.yelp.com/dataset/challenge

comme étant les nœuds du graphe temporel et les commentaires comme des arêtes. Les commerces sont classées par catégories (nous considérons uniquement les huit catégories les plus représentées). Comme les catégories de commerces ne changent pas au fil du temps (communautés statiques), nous attribuons des labels aux utilisateurs : à chaque pas de temps, un utilisateur est supposé appartenir à une catégorie si la majorité de ses commentaires sont postés sur des commerces liés à cette catégorie. Enfin, le graphe temporel est fragmenté en une séquence de 17 graphes avec un total de 744k arêtes entre 38k nœuds.

- **Tmall** : Ce jeu de données est un extrait des ventes sur Tmall.com[3] 6 mois avant l'événement "Double 11 Day" en 2014. Il reprend les interactions des acheteurs avec des produits (i.e. les clics sur la page du produit, l'ajout au panier, l'achat ou l'ajout aux favoris). Les produits sont associés à des catégories. Pour nos expérimentations, nous ne considérons que les cinq catégories les plus représentées. Nous attribuons des labels aux utilisateurs comme décrit pour le jeu de données Yelp. Nous obtenons un graphe temporel divisé en 10 graphes datés, composé de 2,9M d'arêtes entre 27k nœuds.

Jeu de données	nœuds	arêtes	pas de temps
AMiner	51 060	624 381	17
Smp_A	16 546	12 826	
Smp_B	9 244	5 884	
Smp_C	2 300	1 230	
Yelp	38 475	744 195	17
Tmall	27 039	2 976 392	10

TABLEAU 2.2: Les différents jeux de données et leurs échantillons considérés dans les expérimentations d'évaluation des modèles d'embedding.

2.2.4.3 Tâches d'inférence

Nous comparons notre approche avec les méthodes de référence en exécutant différentes tâches d'inférence sur la base de leurs embeddings respectifs, par rapport à la métrique du F1 score. L'idée est que de bons scores d'inférence devraient refléter des embeddings plus pertinents et plus fidèles à la donnée en entrée et, par voie de conséquence, un modèle d'embedding plus performant. Les tâches d'inférence suivantes ont été expérimentées :

- Classification de nœuds (nc) : un classifieur (régression logistique) est entrainé sur les embeddings des nœuds afin de prédire leurs labels. Cette opération est

[3]https://tianchi.aliyun.com/competition/entrance/231576/information

réalisée sur les différents pas de temps indépendamment les uns des autres. Pour les 3 jeux de données considérés, 5k paires {embedding, label} constituent le jeu d'entrainement, choisies aléatoirement sur tous les embeddings des nœuds aux différents pas de temps. Les scores d'évaluation sont mesurés sur le jeu de test, à savoir toutes les autres paires {embedding, label}.

- Prédiction de classe de nœuds (np) : cette tâche est similaire à la classification de nœuds, à la différence que nous nous basons sur l'embedding courant d'un nœud pour prédire son label au pas de temps suivant.

- Reconstruction de liens (er) : pour cette tâche, l'objectif est de déterminer si une arête existe entre une paire de nœuds, étant donné la distance euclidienne entre leurs embeddings. Le modèle de classification binaire employé est une régression logistique et chaque pas de temps est examiné indépendamment des autres. Pour tous les jeux de données considérés, le jeu d'entrainement est constitué en choisissant aléatoirement (vis-à-vis de i, j et t) 10k paires de la forme {embeddings de n_i et de n_j en t, booléen d'existence d'une arête entre n_i et n_j en t}. Les scores d'évaluation sont mesurés sur le jeu de test, à savoir toutes les autres paires non comprises dans le jeu d'entrainement.

- Prédiction de liens (ep) : de même que pour la reconstruction de liens, l'objectif pour cette tâche est de déterminer si une paire de nœuds est connectée par une arête à un pas de temps, en se basant sur la distance entre leurs deux embeddings au pas de temps précédent.

2.2.4.4 Résultats de comparaison

Le tableau 2.3 détaille les résultats comparatifs entre les différentes méthodes considérées. Il en ressort que TN2V présente des performances meilleures que les autres modèles pour les tâches liées à la classification des nœuds (jusqu'à 14,2% en F1 score). Cela signifie que le fait d'apprendre les représentations de différents pas de temps de concert peut améliorer les performances globales. Cela pourrait être dû au fait que nous forçons la continuité temporelle des embeddings. D'autre part, TN2V est moins efficace dans les tâches d'inférence liées aux arêtes[4]. Nous interprétons ces résultats comme suit : TN2V capture les structures temporelles globales plutôt que les structures locales. En effet, les tâches de reconstruction/prédiction d'arêtes portent sur les paires de nœuds indépendamment du reste du graphe, tandis que les tâches de classification

[4]Une optimisation plus fine des hyperparamètres (p et q en particulier) pourrait améliorer les scores des tâches de reconstruiction et prédiction d'arêtes (cf. section 2.4.2.1).

évaluent la possibilité de l'appartenance d'un nœud à des macro-structures (les communautés). TN2V semble donc être plus approprié pour détecter les communautés et capturer leur évolution. En ce qui concerne l'efficacité de DT en prédiction d'arêtes, cela peut s'expliquer par l'idée principale de la méthode : DT porte sur la dynamique des arêtes, plus particulièrement sur le processus d'ouverture/fermeture des triades aux pas de temps suivants. Enfin, il est intéressant de noter les bonnes performances de N2V pour la tâche de reconstruction d'arêtes. Cette spécificité peut trouver son explication dans le fait que cette tâche d'inférence est agnostique à la dimension temporelle : l'information permettant de statuer sur la présence ou l'absence d'une arête entre 2 nœuds provient des embeddings du seul pas de temps courant, minimisant ainsi l'apport qu'une méthode temporelle peut avoir.

Jeu de données	Modèle	nc	np	er	ep
AMiner	DW	0.698	0.673	0.926	0.758
	N2V	0.770	0.758	**0.984**	0.806
	DT	0.744	0.740	0.924	**0.873**
	TN2V	**0.890**	**0.852**	0.861	0.749
Yelp	DW	0.298	0.254	0.945	0.757
	N2V	0.297	0.261	**0.988**	0.796
	DT	0.256	0.247	0.937	**0.916**
	TN2V	**0.315**	**0.281**	0.932	0.811
Tmall	DW	0.944	0.688	0.848	0.688
	N2V	0.965	0.700	**0.913**	0.728
	DT	0.563	0.533	0.778	**0.733**
	TN2V	**0.966**	**0.773**	0.773	0.680

TABLEAU 2.3: Scores des modèles.

2.2.4.5 Visualisation des embeddings

La figure 2.6 montre un exemple de projection sur un plan des embeddings d'un pas de temps sur AMiner, suite à une réduction de la dimensionnalité via t-SNE [1]. Ne sont considérés que les trois domaines de recherche les plus représentés (computer architecture, data mining et computing theory). Globalement, nous pouvons observer que les embeddings se regroupent selon leurs labels. D'autre part, la région au centre du plan concentre des embeddings de nœuds appartenant à plusieurs communautés : cela peut être expliqué par les effets de bord introduits par t-SNE du fait de la perte d'information induite par la réduction de la dimension inhérente à t-SNE.

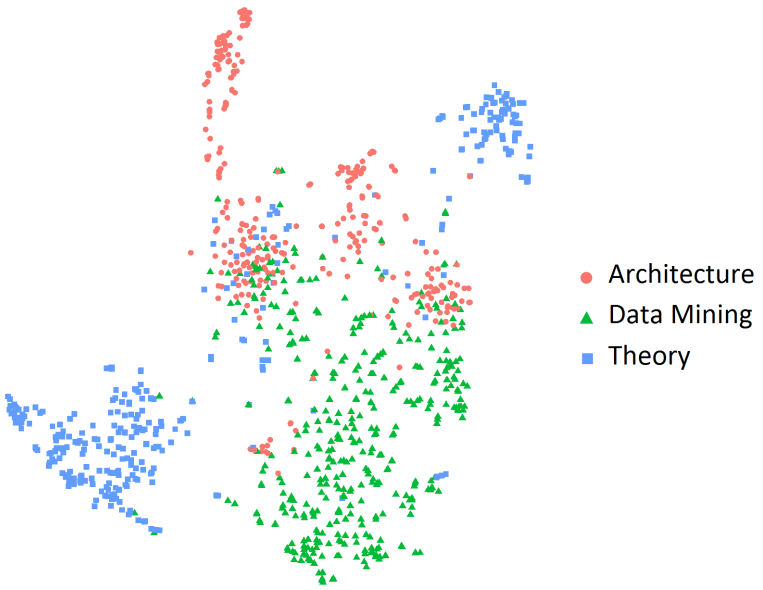


FIGURE 2.6: Exemple d'embeddings de TN2V sur un pas de temps (le 13$^{\text{ème}}$) du jeu de données AMiner. La dimension a été réduite à 2 en utilisant t-SNE.

2.2.4.6 Influence des hyperparamètres

Il est intéressant d'étudier l'influence que peut avoir chacun des hyperparamètres de TN2V sur les performances globales des tâches d'inférence, et donc sur la fiabilité des embeddings temporels. D'après la figure 2.7, nous pouvons remarquer que les paramètres les plus impactants sont q (le paramètre d'entrée-sortie de la marche aléatoire) et θ (le paramètre de compromis stabilité PMI / rare co-apparition). Dans le cas de q, ce constat signifie que la manière d'explorer les voisinages des nœuds lors de la construction des marches aléatoires revêt une importance primordiale pour la consistance des embeddings. La forte influence de la valeur de θ traduit quant à elle le caractère essentiel du compromis entre, d'un coté, la précision de la mesure des co-apparitions des paires de noeuds dans les marches aléatoires, et de l'autre, une variance raisonnable des valeurs dans les matrices $PPMI$.

Par ailleurs, nous pouvons relever deux observations intéressantes.

- D'abord, plus grande est la valeur de la taille de fenêtre de co-apparition w meilleurs sont les embeddings. Cela peut confirmer l'idée que TN2V est plus à même d'identifier et caractériser les structures temporelles globales. En effet, quand la taille de la fenêtre est grande, les voisinages des nœuds ne sont pas restreints à leurs entourages immédiats.
- La seconde observation concerne la longueur de la marche aléatoire l. Nous pouvons remarquer que de plus petites marches conduisent à des embeddings plus

performants. Cela peut potentiellement s'expliquer par le fait que de courtes marches produisent des séquences plus équilibrées. Par exemple, si nous imaginons une région dense du graphe de départ (dense en termes d'arêtes et de poids associés), une longue marche aléatoire peut y être *piégée*. Inversement, les régions les moins denses du graphe peuvent être relativement sous-représentées sur une longue marche aléatoire.

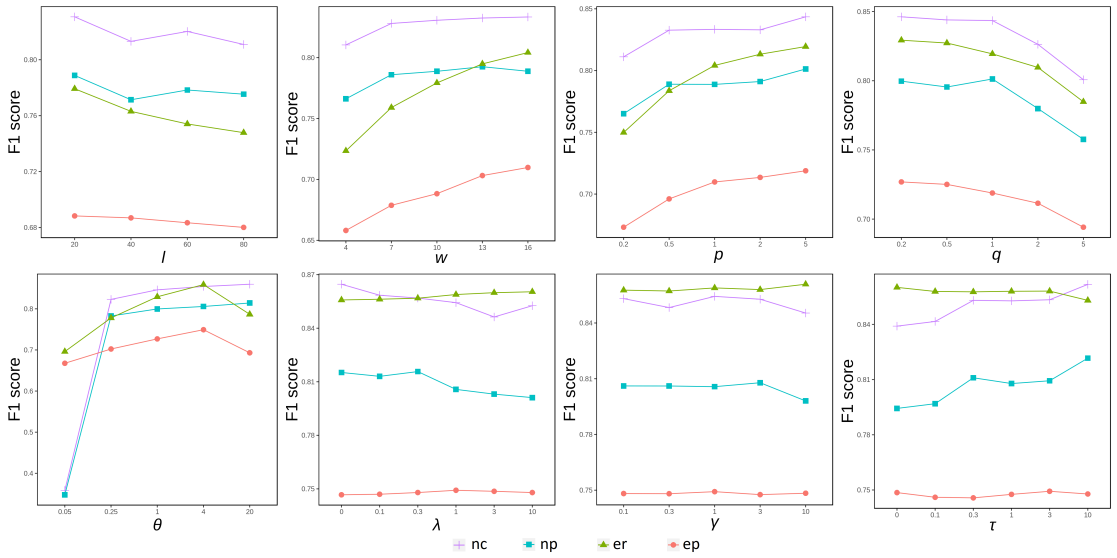


FIGURE 2.7: Analyse de l'effet des hyperparamètres de TN2V sur la classification de nœuds (nc), la prédiction de classes de nœuds (np), la reconstruction d'arêtes (er) et la prédiction d'arêtes (ep).

2.2.4.7 Alignement temporel des embeddings

L'alignement est une notion importante de l'embedding de graphes temporels. Par exemple, c'est une notion nécessaire pour visualiser correctement une séquence d'embeddings : la position des nœuds ne doit pas être volatile d'un pas de temps à un autre. Afin de mettre à l'épreuve les différentes méthodes vis-à-vis de l'alignement, les embeddings sont soumis à un test consistant à prédire le changement de classe des nœuds, sur la base du déplacement des embeddings sur deux pas de temps consécutifs.

En effet, les communautés constituant la vérité de terrain des jeux de données en entrée ont une signification sémantique que nous pouvons raisonnablement supposer être stable au cours du temps (les domaines de recherche pour AMiner, les catégories de produits et de commerces pour Yelp et TMall respectivement). Par conséquent, si l'alignement est respecté, les emplacements des communautés dans l'espace latent d'embedding devraient également être stables, ou du moins continus. Ainsi, pour les méthodes produisant des embeddings bien alignés, la trajectoire d'un nœud dans l'espace d'embedding contient une information sur sa communauté aux pas de temps suivants. Cela signifie que le

déplacement d'un nœud (la norme du vecteur de différence entre ses embeddings à deux pas de temps consécutifs) devrait être un bon indicateur sur son potentiel changement de communauté dans le cas d'embeddings bien alignés.

Le tableau 2.4 montre les résultats de cette expérimentation. Nous remarquons que, exception faite du dataset Yelp, TN2V produit des embeddings plus alignés que les autres méthodes. Cela peut trouver son explication dans le fait que les embeddings sont lissés temporellement (cf. équation 2.20). D'autre part, et sur une note plus globale, il n'est pas surprenant de constater que les embeddings des méthodes temporelles sont plus alignés que ceux des méthodes statiques.

Modèle	**AMiner**	**Yelp**	**Tmall**
DW	0.489	0.404	0.407
N2V	0.483	0.411	0.431
DT	0.528	**0.481**	0.499
TN2V	**0.544**	0.461	**0.568**

TABLEAU 2.4: F1 score de la prédiction du changement de classe d'un nœud en se basant sur son déplacement latent entre deux pas de temps consécutifs.

Le chapitre 4 est consacré à l'alignement temporel des embeddings. Les décalages temporels des espaces latents consécutifs sont qualifiés et quantifiés. L'impact des désalignements est également analysé.

2.2.5 Dimension d'embedding

La dimension de l'espace latent cible est un hyperparamètre important : la qualité d'un modèle d'embedding réside dans les performances d'inférence, mais aussi dans l'efficacité du processus de réduction de la dimension. Afin d'évaluer TN2V vis-à-vis de cet aspect, nous étudions l'influence de ce paramètre pour les différentes méthodes considérées pour la tâche de classification de nœuds. La figure 2.8 montre les résultats de cette expérimentation pour AMiner : alors que TN2V permet d'obtenir de bonnes performances avec un nombre limité de *features* (à partir de $d = 10$), les autres approches nécessitent une dimension d'embedding conséquente et sont instables lorsque la dimension augmente.

De plus, nous notons une sorte de saturation concernant la dimension d'embedding : il semble que 20 features soient suffisantes pour caractériser un nœud. Cette valeur ($d = 20$) pourrait correspondre à la *dimension latente intrinsèque* du graphe temporel, c'est-à-dire la dimension minimale d'un espace latent capable de capturer toute l'information structurelle présente dans le graphe. Elle est liée à la nature des données en entrée et ne dépend pas de leur taille (en nombre de nœuds/arêtes) [44]. Pour confirmer cette idée, nous réalisons une expérimentation complémentaire. La figure 2.9

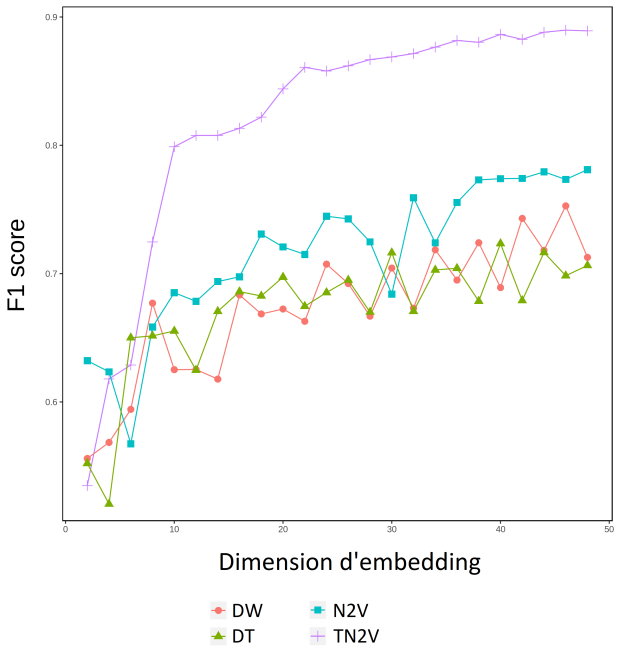


FIGURE 2.8: Efficacité du processus de réduction de la dimension pour les différents modèles d'embedding sur AMiner.

montre l'impact de la dimension d'embedding sur le score d'inférence, pour la tâche de classification de nœuds, sur les trois échantillons de AMiner de tailles différentes (i.e. Smp_A, Smp_B et Smp_C, décrits dans le tableau 2.2). Nous remarquons que les formes des courbes sont similaires. En d'autres termes, le rapport entre l'information capturée à une certaine dimension d'embedding et l'information *maximale* qu'il est possible de capturer à la saturation ne dépend pas de la taille de l'échantillon.

2.3 Task-specific TemporalNode2vec

Comme constaté précédemment en section 2.2.5, il est possible d'extraire un nombre restreint de features décrivant fidèlement les nœuds d'un graphe temporel. Cela signifie que les dimensions de l'espace d'embedding latent contiennent suffisamment d'informations pour effectuer les différentes tâches d'inférence, attendu que les embeddings sont construits de manière agnostique vis-à-vis des tâches d'apprentissage automatique en aval. Cependant, à ce stade, la distribution de l'information utile à chaque tâche sur les dimensions latentes reste inconnue. Par exemple, dans un scénario simpliste, certaines des features retenues peuvent être pertinentes pour les tâches de reconstruction d'arêtes et inutiles pour celles de classification des nœuds. Dans une telle configuration, nous pourrions nous contenter d'un plus petit nombre de features pour la classification de nœuds. En se basant sur cette idée, nous proposons une variante de TN2V que nous appelons

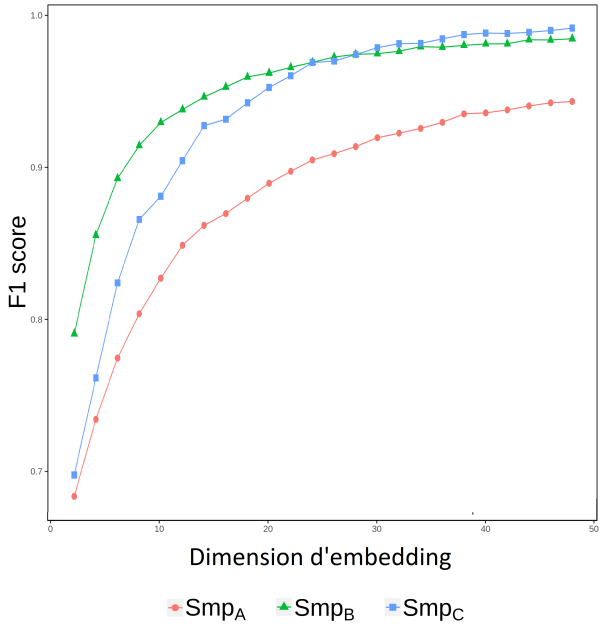


FIGURE 2.9: Découverte de la dimension latente intrinsèque de AMiner : saturation à partir de $d = 20$ pour les 3 échantillons définis en 2.2.

TsTemporalNode2vec (TsTN2V) pour *task-specific TemporalNode2vec* : il s'agit d'un algorithme d'embedding semi-supervisé dédié aux tâches de classification de nœuds, rendu possible grâce à une légère modification de la fonction objectif de TN2V. TsTN2V a fait l'objet d'un article [45] accepté et présenté à la 1ère édition de la conférence nationale " French Regional Conference on Complex Systems ".

2.3.1 Le modèle TsTemporalNode2vec

En plus d'un graphe temporel, TsTN2V prend également en entrée une séquence de labels d'une partie des nœuds dudit graphe $S = \{s_{i,c,t}\}$ où $s_{i,c,t}$ signifie que le nœud v_i appartient à la communauté c au pas de temps t.

Afin de concevoir TsTN2V, nous modifions la fonction objectif de TN2V en forçant la proximité des embeddings pour les nœuds qui appartiennent à une même communauté. Ceci peut être réalisé en ajoutant un terme à la fonction objectif :

$$L_{ts} = \sum_{(i,j,t)\in S_{pos}} \|u_i(t) - u_j(t)\|^2 \tag{2.24}$$

où $S_{pos} = \{(i, j, t)\}$ est dérivé à partir de S et signifie que les nœuds v_i et v_j appartiennent à la même communauté au pas de temps t. Cependant, l'introduction de L_{ts} à la fonction objectif de TN2V ne serait pas commode : ce terme porte sur des vecteurs plutôt que des matrices, contrairement aux autres termes de la fonction objectif. En

outre, les optimisations décrites dans la section 2.2.3.4 ne seraient plus valides. En place et lieu, nous choisissons d'incorporer la donnée supplémentaire de supervision (co-appartenance de nœuds à des communautés) dans les matrices $PPMI$. En effet, il est possible de majorer les entrées correspondant aux paires de nœuds appartenant à une même communauté, dans la mesure où les $PPMI$ peuvent être considérées comme des matrices de similarités temporelles entre les nœuds. Nous introduisons donc les matrices S_{sc} et $SPMI$:

$$S_{sc}(v_i,\ v_j)_t = \begin{cases} 1 & \text{si } (i,j,t) \in S_{pos} \\ 0 & \text{sinon} \end{cases} \tag{2.25}$$

$$SPMI(v_i,\ v_j)_t = PPMI(v_i,\ v_j)_t + \alpha \cdot m_t \cdot S_{sc}(v_i,\ v_j)_t \tag{2.26}$$

où m_t est la médiane des valeurs non nulles de $PPMI_t$ et α est un hyperparamètre contrôlant la majoration des valeurs de $PPMI$. Nous utilisons m_t comme *normalisation* de sorte que α soit de l'ordre de grandeur 1.

Par ailleurs, il est pertinent de noter que les labels datés S englobent également des informations utiles sur les paires de nœuds appartenant à des communautés différentes. De la même manière que S_{pos}, nous dérivons S_{neg} de l'ensemble S. Il est alors possible d'incorporer S_{neg} dans les matrices S_{sc} et $SPMI$:

$$S_{sc}(v_i,\ v_j)_t = \begin{cases} 1 & \text{si } (i,j,t) \in S_{pos} \\ -1 & \text{si } (i,j,t) \in S_{neg} \\ 0 & \text{sinon} \end{cases} \tag{2.27}$$

$$SPMI(v_i,\ v_j)_t = max\Big(0,\ PPMI(v_i,\ v_j)_t + \alpha \cdot m_t \cdot S_{sc}(v_i,\ v_j)_t\Big) \tag{2.28}$$

Ensuite, nous remplaçons les matrices $PPMI$ par les $SPMI$ dans la fonction objectif et son optimisation exprimées respectivement dans les équations (2.21) et (2.22). Les étapes suivantes de TN2V restent inchangées.

2.3.2 Expérimentations

Afin d'évaluer les performances de TsTN2V, nous soumettons ses embeddings en sortie aux tâches de classification des nœuds et de prédiction de classes de nœuds pour les jeux de données décrits dans la section 2.2.4.2.

2.3.2.1 Ratio de labellisation

TsTN2V est un modèle d'embedding semi-supervisé. Par conséquent, il faut définir et fixer le ratio de labellisation, i.e. la part de labels temporels sur laquelle le modèle se

basera pour son apprentissage. Pour les jeux de données que nous considérons, il est important de noter que les labels des nœuds ne sont pas égaux en termes de *quantité d'information* qu'ils fournissent : certains nœuds interagissent sur toute la durée du graphe temporel tandis que d'autres sont actifs à de rares pas de temps. Par conséquent, il semble utile d'évaluer les contributions des nœuds en information apportée. Pour chaque nœud v_i, nous définissons la quantité Q_i :

$$Q_i = \frac{q_i}{\sum\limits_{v_j \in V} q_j} \tag{2.29}$$

où q_i représente le nombre de pas de temps où le nœud v_i est actif. Ensuite, nous pouvons fixer un ratio de labellisation r en choisissant un échantillon aléatoire de nœuds RS_r satisfaisant :

$$\sum_{v_j \in RS_r} Q_j \simeq r \tag{2.30}$$

2.3.2.2 Valeurs testées

Outre les hyperparamètres de TN2V, TsTN2V en possède un supplémentaire (i.e. α), comme décrit dans les équations (2.26) et (2.28). Pour nos expérimentations, nous conservons les valeurs des hyperparamètres de TN2V donnant les meilleures performances sur la classification des nœuds et effectuons une grid-search selon $(d,\ r,\ \alpha) \in \{2,\ 4,\ 7,\ 10,\ 15,\ 20,\ 25\} \times \{0.1,\ 0.25,\ 0.5\} \times \{0.1,\ 0.33,\ 0.67,\ 1\}$.

2.3.2.3 Analyse et interprétation des résultats

Le tableau de figures 2.10 présente les résultats de l'expérimentation. Globalement, TsTN2V améliore les performances de la classification de nœuds. C'est particulièrement le cas pour les petites dimensions d'embedding. Cette constatation confirme l'idée selon laquelle le task-specific embedding capture autant d'informations pertinentes — pour la tâche d'inférence adressée — que possible dans les dimensions latentes d'embedding. De plus, à mesure que la dimension d'embedding augmente, l'écart entre les performances des approches task-agnostic et task-specific tend à se réduire : pour de grandes valeurs de d, les informations pertinentes sont capturées par les deux méthodes.

Concernant le ratio de labellisation, les résultats montrent que fixer r à 0.1 est suffisant pour améliorer le F1 score. Autrement dit, la labellisation de 10% des nœuds permet d'améliorer significativement les performances d'inférence. En revanche, il semble que

des valeurs plus élevées de r (0.25 et 0.5) n'apportent pas plus d'amélioration des performances (respectivement, un gain moyen de F1 score de 0.8% et 1.4% par rapport à $r = 0.1$). En d'autres termes, un faible ratio de nœuds labllisés suffit pour comprendre le type de communautés que les embeddings doivent préserver dans l'espace latent.

Enfin, l'hyperparamètre α semble avoir un impact significatif sur les performances. Par exemple, des valeurs élevées de α entraînent de mauvaises performances pour le jeu de données AMiner, tandis que fixer α à 0.33 serait le meilleur choix pour le jeu de données Yelp. Concernant le jeu de données Tmall, il semble que la valeur de α n'ait pas un effet significatif sur le score de classification. Des recherches supplémentaires sont nécessaires

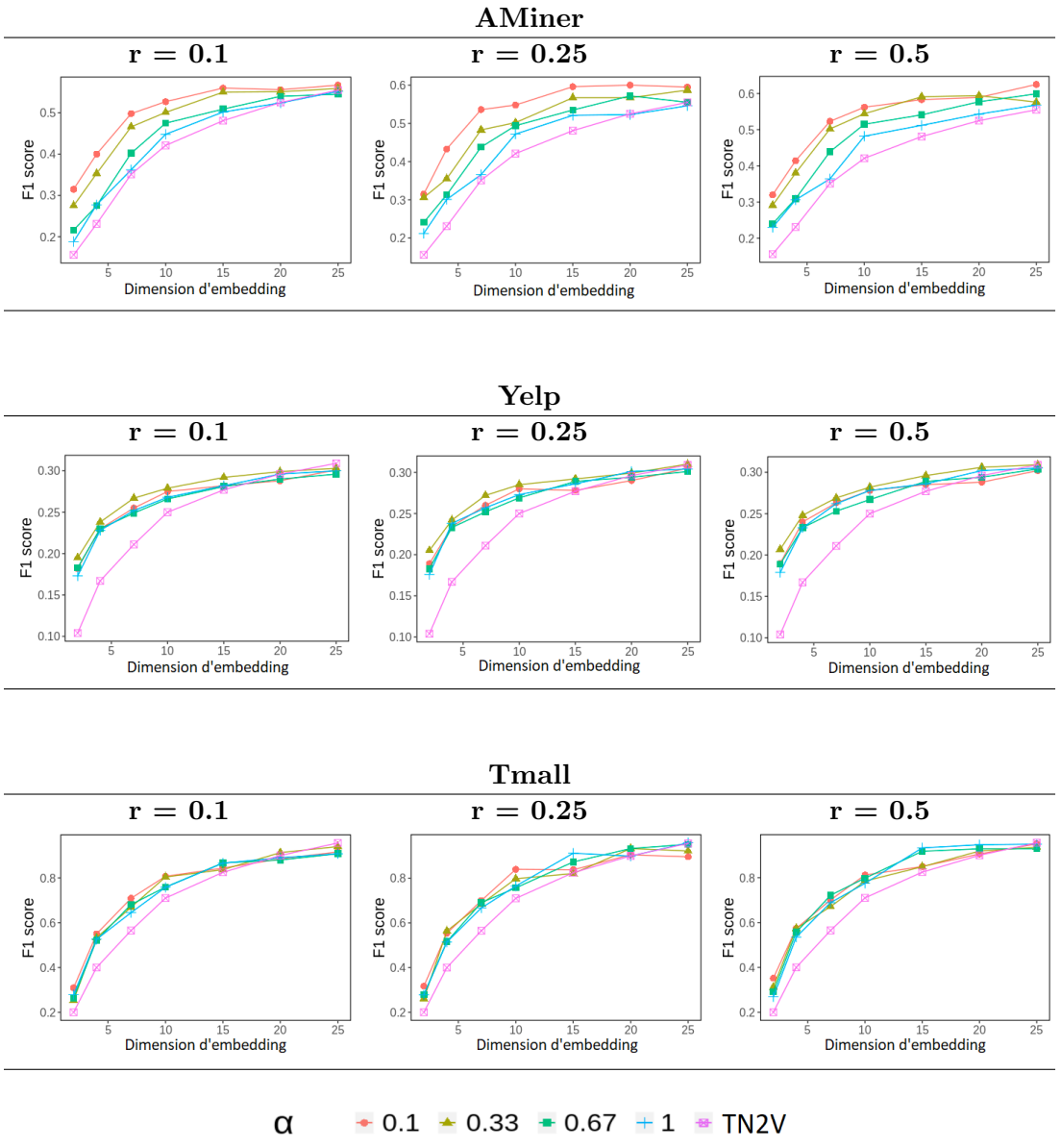


FIGURE 2.10: Comparaison des résultats d'inférence de TsTN2V et TN2V sur la classification de nœuds.

pour expliquer ce point. Globalement, il semble que la valeur optimale de α et son effet dépendent fortement du jeu de données considéré.

2.3.3 Contexte d'application de TsTemporalNode2vec

Le task-specific embedding de graphes temporels peut être utile dans plusieurs situations. En effet, il est possible de tirer parti des métadonnées partiellement voire totalement portant sur l'appartenance des nœuds à des communautés. Par ailleurs, lorsque de telles données ne sont pas disponibles, il est possible d'effectuer une étape de labellisation manuelle consistant à marquer une petite partie des paires de nœuds appartenant probablement à la même communauté ou probablement appartenant à des communautés différentes, en se basant sur une heuristique définie par l'utilisateur. Dans ce cas, le task-specific embedding offre plus de flexibilité : il permet de pointer spécifiquement les communautés ou les types de communautés que nous souhaitons relever.

2.4 Conclusions, perspectives et travaux connexes

Tel que nous l'imaginons, l'apprentissage de représentations d'un graphe temporel est la première étape de traitement en vue de l'adaptation de nombre de techniques de vision par ordinateur au cas des données issues d'interactions sociales. En effet, en sortie de la phase d'embedding, les données se présentent d'une manière comparable à une vidéo, i.e. une séquence de matrices représentant les nœuds d'un graphe, où la notion de voisinage au sens de distance entre points dans un espace multidimensionnel coïncide avec la notion de voisinage au sens d'un graphe. Deux approches ont été conçues.

2.4.1 GeoEmb

GeoEmb (section 2.1), prend à contre-pied les mécanismes généralement utilisés dans l'état de l'art et tente de trouver une représentation exhaustive d'un graphe temporel, tant au niveau de l'espace (dimension d'embedding suffisamment grande pour contenir toute l'information disponible) que du temps (une représentation à chaque moment de l'historique, sans aucune agrégation temporelle). Paradoxalement, ces éléments motivant la conception de GeoEmb constituent en même temps les points de blocage de la méthode. En effet, les calculs élémentaires à effectuer pour obtenir la représentation du graphe temporel en entrée à un instant donné ne sont pas scalables en nombre de nœuds ou en nombre d'instantanés (i.e. nombre de moments distincts où ont lieu les

interactions) et, qui plus est, est récursif au sens où chaque matrice de représentations dépend de celle qui la précède dans le temps (et donc de toutes celles qui la précédent).

Mis à part le problème de la scalabilité, d'autres points potentiellement bloquants demeurent en suspens. Par exemple, la conception exposée de GeoEmb présente un défaut majeur, celui de la convergence temporelle des vecteurs de représentation des nœuds d'un graphe temporel : en effet, plus on avance dans le temps (i.e. dans le déroulement de l'historique des interactions du graphe temporel), plus les embeddings des nœuds se rapprochent (selon la distance euclidienne en 2.1.2.1 ou selon la distance angulaire en 2.1.2.2). Par exemple, cela implique que, dans le cas des graphes temporels denses (en termes de nombre d'arêtes et de force des poids), les embeddings des différents nœuds peuvent se retrouver concentrés sur une région restreinte de l'espace de plongement latent à partir d'un certain moment du déroulement de l'historique des interactions. Ce problème pourrait être contrebalancé par un mécanisme d'espacement global forcé, à chaque nouvelle interaction : l'idée serait que chaque rapprochement impliqué par une interaction soit compensé par un éloignement global de chaque paire d'embeddings de telle sorte que la distance moyenne entre les paires d'embeddings demeure constante au cours du temps. Cette idée pourrait être facilement mise en place dans le cas où la métrique considérée est celle de la distance euclidienne (en multipliant la matrice représentative d'un instantané par le facteur approprié), mais risquerait d'être mathématiquement ardue à formaliser pour le cas de distance angulaire.

Par ailleurs, au vu du fait que GeoEmb n'inclut pas de réduction de la dimension, cette étape reste à effectuer sur les représentations temporelles en sortie. Pour ce faire, il existe plusieurs méthodes telles que l'ACP [46], les autoencodeurs [47, 48] ou encore t-SNE [1]. Néanmoins, les différentes techniques citées s'appliquent à une matrice unique. Autrement dit, il est possible d'appliquer une réduction de la dimension de la représentation d'un graphe temporel sur chacun des instantanées de manière indépendante. En procédant de la sorte, l'information relative à la nature séquentielle est détériorée du fait de la potentielle introduction d'un *décalage en alignement* (cf. chapitre 4). Afin de remédier à ce décalage, des approches de réduction de la dimension adaptées aux données séquentielles existent et pourraient être appliquées en sortie de GeoEmb [49–51].

2.4.2 TemporalNode2vec

À l'inverse de GeoEmb, l'approche TN2V (section 2.2) adapte différents mécanismes de l'état de l'art. Dans un premier temps, nous reprenons la technique de marches aléatoires telle que formalisée dans N2V [6]. Ensuite, une fois que le graphe temporel

en entrée est transformé en séquences de nœuds, nous adaptons un mécanisme de lissage temporel des embeddings issu du traitement automatique du langage. Ce faisant, TN2V est à même d'encoder l'information inhérente à chaque nœud / pas de temps dans une représentation vectorielle, particulièrement adaptée aux tâches relatives à la classification de nœuds. D'autre part, TN2V est globalement plus efficace en termes de réduction de la dimension, phase intrinsèque aux procédés d'embedding. Enfin, la variante TsTN2V permet d'adresser spécifiquement la classification de nœuds. Ainsi, il est possible d'améliorer les performances de cette tâche d'inférence tout en réduisant la dimension d'embedding. Plusieurs améliorations sont envisageables afin d'affiner TN2V.

2.4.2.1 Tuning de TemporalNode2vec

TN2V posséde un nombre important d'hyperparamètres $\{l, w, p, q, \theta, \lambda, \gamma, \tau\}$. Cela complique considérablement la tâche de tuning du modèle. En effet, tester 3 valeurs pour chacun d'entre eux selon une grid-search conduirait à plus de 6k jeux de paramètres. Ce problème a été partiellement contourné lors des expérimentations menées en section 2.2.4.1 en optimisant les hyperparamètres un à un et en fixant les autres. Néanmoins, cette manière de procéder présente des limites évidentes : il n'y a aucune garantie de trouver un optimum.

Il existe des manières plus sophistiquées, notamment l'optimisation bayésienne [52]. L'idée est de construire un modèle probabiliste d'estimation du retour de la fonction objectif, en fonction des hyperparamètres en entrée. Ce modèle bayésien se base sur un nombre réduit d'observations (i.e. jeux de paramètres et retours de la fonction objectif associés) pour déterminer le comportement de la fonction de correspondance entre entrées et sorties de la fonction objectif. Plusieurs travaux [53, 54] ont démontré que l'optimisation bayésienne permettrait de trouver plus rapidement un *bon* jeu de hyperparamètres. Par ailleurs, cela permettrait de trouver des hyperparamètres optimaux sur des intervalles continus, au lieu de valeurs discrètes comme ce serait le cas pour une grid-search.

Une autre manière de tuning de TN2V pourrait se faire via une descente de gradient. En effet, lors de la construction des embeddings, il est possible de forcer le modèle à apprendre certains hyperparamètres optimaux, au même titre que les embeddings eux-mêmes. Il faudrait néanmoins poser des contraintes aux valeurs que pourraient prendre les hyperparamètres appris. Par exemple, τ doit être strictement positif vu que ce paramètre correspond au poids imposé au lissage temporel des embeddings. À noter que l'apprentissage automatique de paramètres optimaux ne peut concerner que ceux qui entrent en jeu dans la fonction objectif exprimée dans l'équation 2.22, à savoir

$\{\theta, \lambda, \gamma, \tau\}$. Les autres paramètres ne pourraient faire partie des poids à apprendre car ils sont utilisés pour la construction des marches aléatoires, phase ayant lieu avant l'apprentissage des embeddings. Néanmoins, il serait possible de combiner l'optimisation bayésienne des hyperparamètres ayant trait à la construction des marches aléatoires et l'apprentissage de ceux impliqués dans la fonction objectif.

2.4.2.2 Cohérence temporelle des marches aléatoires

La construction des marches aléatoires est une étape importante de TN2V et sa variante task-specific. En effet, l'ensemble des marches constitué se doit de conserver le plus fidèlement possible l'information présente dans le graphe temporel en entrée. Par exemple, dans le cadre des graphes statiques, le fait que N2V offre plus de flexibilité aux processus de marches par rapport à DW permet d'améliorer la qualité des embeddings produits.

Telles que définies dans TN2V, les marches sont construites sur chacun des pas de temps du graphe temporel à la manière de N2V. En d'autres termes, les graphes correspondant aux différents pas de temps sont considérés comme des graphes statiques. Cela signifie que, sur chaque pas de temps, l'information temporelle est perdue. Il est possible de remédier à celà en incorporant cette information temporelle utile, améliorant potentiellement ainsi les marches aléatoires. En particulier, l'approche proposée par CTDNE [12] permettrait de composer des marches cohérentes dans le temps. Autrement dit, sur chaque pas de temps, chacune des marches effectuées pourrait être constituée tout en respectant la séquentialité présente dans les interactions en entrée : un pas d'une marche aléatoire allant du nœud n_1 à n_2 n'est permis que si l'interaction correspondante $\{n_1, n_2\}$ a lieu ultérieurement à celle du pas précédent (i.e. celui menant au nœud n_1). Ce procédé est assimilable à un filtre appliqué à l'ensemble des marches aléatoires où ne seraient conservées que celles qui sont *temporellement valides*.

À noter que ce type de filtrage pourrait dans certaines conditions être trop restrictif. Le nombre de marches pourrait être significativement réduit dépendamment du nombre d'interactions ayant lieu au cours du pas de temps considéré. Par ailleurs, les marches n'auraient pas forcément la même longueur. Plus critique encore, certains nœuds pourraient être sous représentés dans les marches temporellement cohérentes : par exemple, dans le cas extrême où un nœud n'interagirait qu'une seule fois, à la toute fin du pas de temps considéré, il serait présent au plus une fois dans une marche (dans le dernier pas de la marche). Il serait néanmoins possible de permettre aux marches aléatoires de passer d'un pas de temps à un autre afin de réduire ces effets de bord tel que proposé

dans l'approche [55]. Cependant, cela remettrait en cause le découpage en pas de temps du graphe temporel.

2.4.2.3 TsTN2V spécifique à d'autres tâches d'inférence

Dans la conception de la TsTN2V, le choix a été fait de concevoir une variante spécifique à la classification de nœuds. Nous avons démontré que cela pouvait aider à améliorer les performances tout en optimisant la capacité de réduction de la dimension des embeddings. TsTN2V peut simplement être adapté à la prédiction de classes de nœuds : en effet, il suffirait de substituer l'information contenue dans les matrices $SPMI$ et S_sc (équations 2.25, 2.26, 2.27, 2.28) au pas de temps t par celle du pas de temps $t+1$.

D'autre part, il est également possible de concevoir une variante TsTN2V spécifique à la tâche de prédiction de liens. De manière similaire à celle dédiée à la classification de nœuds, l'idée serait de majorer (respectivement minorer) les entrées de la matrice $PPMI$ pour les paires de nœuds connectés par un lien (respectivement déconnectés) au pas de temps suivant. À noter qu'une variante spécifique à la tâche de prédiction de liens pourrait ne pas se révéler concluante : la prise en compte de l'information supplémentaire relative à la présence/absence de liens au pas de temps suivant est potentiellement redondante avec le lissage temporel inhérent à TN2V.

À noter que, l'approche de modification des matrices $PPMI$ ne pourrait être déclinée pour adresser la tâche de reconstruction de liens : en effet, les matrices $PPMI$ contiennent par essence l'information sur la présence ou non d'arêtes entre nœuds.

Chapitre 3

Temporalisation d'Autoencodeurs de Graphes Statiques

3.1 Temporalisation : idée générale et travaux connexes

Comme mentionné dans la section 2.2.1, il existe une multitude de méthodes d'embedding de graphes temporels [14–17, 30–38]. Ces techniques améliorent les performances sur les tâches d'inférence qui sont sensibles à la dimension temporelle. Le chapitre 2 présentait la méthode d'embedding TN2V, une version de N2V *temporalisée* (i.e. adaptée au cas des graphes temporels). Ce faisant, nous avons pu tirer profit des avantages que peut présenter N2V sur le traitement des graphes statiques. Par ailleurs, bien que N2V soit globalement performant, il est évident que les autres approches d'embedding statiques peuvent être plus efficaces sur certains jeux de données ou sur certaines tâches d'inférences. Par conséquent, il serait intéressant de concevoir une approche générique pour temporaliser différentes méthodes statiques.

Ce chapitre présente un travail de temporalisation d'une catégorie de méthodes d'embedding statiques, à savoir les autoencodeurs prenant des matrices d'adjacence de graphes en entrée. Les autoencodeurs sont des modèles couvrant un large spectre de domaines de l'apprentissage automatique : traitement d'images, traitement automatique du langage, des séries temporelles et, dans le cas qui nous intéresse, des graphes. Il s'agit de réseaux de neurones comportant deux composantes : un encodeur et un décodeur. L'encodeur convertit la donnée en entrée en une représentation latente de dimension réduite et le décodeur reconstruit le plus fidèlement possible la donnée en entrée à partir de la représentation latente apprise.

À cette fin de temporalisation d'autoencodeurs, nous reprenons et adaptons la notion de matrice de *supra-adjacence* d'un graphe temporel, notion utilisée dans plusieurs travaux [56–58]. Cela consiste en une opération transformant une séquence de graphes (les pas de temps d'un graphe temporel par exemple) en un *supra-graphe* statique dont les *supra-nœuds* sont des paires {nœud, pas de temps} du graphe temporel en entrée. Les *supra-arêtes* de ce supra-graphe sont transposées depuis le graphe initial : par exemple, une arête entre les nœuds v_1 et v_2 au pas de temps t se traduit par une supra-arête entre les supra-nœuds $\{v_1, t\}$ et $\{v_2, t\}$. À ce stade, le supra-graphe est composé de T composantes déconnectées, où T est le nombre de pas de temps. Ensuite, pour connecter les paires de supra-nœuds n'appartenant pas à la même composante, des supra-arêtes pondérées supplémentaires sont créées. Dans la suite de ce chapitre, *arêtes temporelles* et *poids temporels* feront respectivement référence à ces supra-arêtes supplémentaires et aux poids qui leurs sont associés.

La principale contribution de ce travail réside dans la manière dont ces arêtes temporelles sont créées. En effet, là où d'autres approches basées sur les matrices de supra-adjacence [56–58] attribuent des poids fixes aux supra-arêtes liant les paires de supra-nœuds, nous attribuons des poids appris, rendant la méthode plus *data-driven*. Le supra-graphe ainsi construit est plus adapté aux tâches d'inférence relatives à la classification/prédiction de nœuds ou à la reconstruction/prédiction d'arêtes.

La présente méthode de temporalisation d'autoencodeurs a fait l'objet d'un article [59] accepté et présenté à la " 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining ".

3.2 Temporalisation d'autoencodeurs

L'idée clé de l'approche présentée réside dans la façon dont nous adaptons et modifions légèrement la structure des autoencodeurs pour apprendre les poids optimaux à attribuer aux arêtes temporelles.

3.2.1 Couche d'entrée ajoutée

Étant donné un autoencodeur de graphe produisant des embeddings statiques et prenant une matrice d'adjacence en entrée, nous insérons d'abord une couche préliminaire connectée aux données d'entrée, dont le rôle est de former la matrice de supra-adjacence. Concrètement, cette couche construit une matrice diagonale par blocs de la forme $(N{\cdot}T){\times}(N{\cdot}T)$ à partir de la séquence des T matrices d'adjacence d'entrée de taille $N{\times}N$,

où N est le nombre des nœuds V du graphe initial. Ensuite, des cellules supplémentaires entraînables (i.e. des neurones ajustables par l'autoencodeur), correspondant aux poids des arêtes temporelles, sont introduites dans la matrice diagonale par blocs, comme le montre la figure 3.1b.

Par souci de simplicité du modèle, nous ne considérons pas la possibilité de créer des arêtes temporelles entre chaque paire de supra-nœuds. Les arêtes temporelles autorisées sont de la forme : $\{v, t_i\}$, $\{v, t_j\}$ (figures 3.1a et 3.1c), c'est-à-dire les supra-arêtes entre un nœud et lui-même à différents pas de temps. Autrement, le nombre de poids entraînables serait beaucoup plus important, ce qui augmenterait le temps d'apprentissage.

Plusieurs variantes de création d'arêtes temporelles sont considérées et sont expliquées ci-après.

3.2.1.1 Arêtes temporelles orientées / non-orientées

Une fois la matrice de supra-adjacence construite, toutes ses cellules sont de la même nature, indépendamment du fait qu'elles correspondent à des arêtes temporelles apprises ou à des supra-arêtes dérivées du graphe initial. Cela signifie que, au-delà de leur caractère latent, les arêtes temporelles peuvent également être interprétées de manière similaire aux autres arêtes : si tel est le cas, une arête temporelle entre $\{v, t_i\}$ et $\{v, t_j\}$ signifierait qu'il existe une influence mutuelle entre les états du nœud v à t_i et t_j.

Cela dit, un nœud qui s'influencerait lui-même dans des pas de temps antérieurs ne semble pas être une configuration réaliste. Par conséquent, une possibilité d'éviter cette situation consiste à utiliser des arêtes temporelles orientées, par exemple une arête orientée de $\{v, t_i\}$ à $\{v, t_j\}$ avec $t_i < t_j$. Cela se traduirait par une matrice de supra-adjacence asymétrique où seule la partie triangulaire inférieure de la matrice est peuplée de poids temporels appris.

Sinon, nous pouvons toujours considérer des matrices de supra-adjacence symétriques. Dans ce cas, une connexion non orientée entre $\{v, t_i\}$ et $\{v, t_j\}$ pourrait être interprétée comme une contrainte de lissage temporel forçant la continuité des embeddings d'un nœud dans le temps, plutôt que comme une influence mutuelle entre deux états d'un nœud à des pas de temps différents.

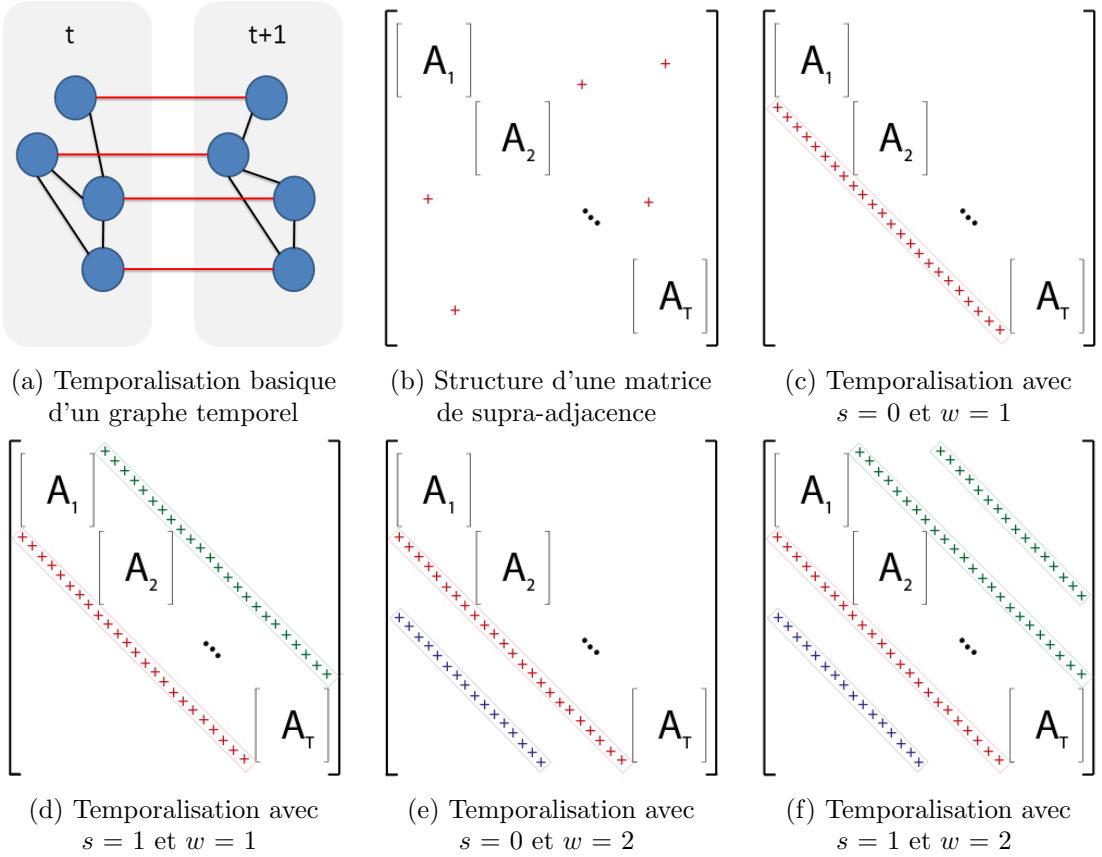
Nous définissons l'hyperparamètre s qui contrôle les deux possibilités : s est égal à 0 ou 1 lorsque nous imposons respectivement des arêtes temporelles orientées ou non-orientées (resp. figures 3.1c et 3.1d).

3.2.1.2 Fenêtre temporelle

L'intuition première derrière la méthode de temporalisation d'autoencodeurs présentée réside dans le fait que l'embedding d'un nœud devrait être conditionné par ses interactions ainsi que par ses états à d'autres pas de temps. La manière la plus directe d'y parvenir est de créer des arêtes temporelles entre chaque paire de supra-nœuds consécutifs, à savoir $\{v,\, t\}$, $\{v,\, t+1\}$ (avec $(v,t) \in V \times [\![1,\, T-1]\!]$).

Cependant, il est possible de laisser à l'autoencodeur temporalisé la possibilité de construire des schémas d'évolution temporels plus sophistiqués et complexes en permettant aux arêtes temporelles de couvrir des intervalles de temps plus longs. À cette fin, nous définissons l'hyperparamètre de la fenêtre temporelle w : pour une valeur donnée de w, l'ensemble des arêtes temporelles à ajouter à la matrice de supra-adjacence est : $\{v,\, t\}$, $\{v,\, t+i\}$ pour $(v,i,t) \in V \times [\![1,\, w]\!] \times [\![1,\, T-1]\!]$ avec $t+i \leqslant T$ (figure 3.1e).

Il est également possible de combiner des arêtes temporelles non orientées avec une fenêtre temporelle $w > 1$ comme le montre la figure 3.1f.



(a) Temporalisation basique d'un graphe temporel

(b) Structure d'une matrice de supra-adjacence

(c) Temporalisation avec $s = 0$ et $w = 1$

(d) Temporalisation avec $s = 1$ et $w = 1$

(e) Temporalisation avec $s = 0$ et $w = 2$

(f) Temporalisation avec $s = 1$ et $w = 2$

FIGURE 3.1: Exemples de matrices de supra-adjacence. A_i représente l'adjacence au $i^{\text{ème}}$ pas de temps et les " + " marquent les positions des poids temporels sur la matrice de supra-adjacence.

3.2.2 Modification de la couche en sortie

Afin d'obtenir des embeddings fiables, la dernière couche des autoencodeurs de graphes consiste généralement en une comparaison entre la matrice d'adjacence en entrée et une matrice de mesure de similarité entre paires d'embeddings, souvent le produit scalaire. Dans le cadre de la temporalisation, il ne semble pas être approprié de comparer la matrice de supra-adjacence dans sa globalité aux similarités de toutes les paires de supra-nœuds. En effet, les embeddings sont pertinents quand ils sont capables de préserver les structures du graphe en entrée. Ainsi, dans la mesure où les arêtes temporelles sont des artefacts ajoutés, elles ne devraient pas être prises en compte dans la dernière couche de l'autoencodeur temporalisé. Par conséquent, dans la dernière couche, la comparaison est faite entre la matrice de supra-adjacence sans arêtes temporelles et les similarités des paires de supra-nœuds appartenant au même pas de temps. Cela équivaut à comparer la séquence des T matrices d'adjacence initiales aux similarités des paires de nœuds pour chaque pas de temps.

3.2.3 Embeddings temporels

Un autoencodeur temporalisé apprend des représentations latentes des supra-nœuds. Cela signifie que, étant donné un graphe temporel composé de N nœuds sur T pas de temps, un autoencodeur temporalisé retourne en sortie $N \cdot T$ vecteurs d'embedding, un pour chaque supra-nœud. Ensuite, il est possible de manière évidente de remanier cette matrice d'embeddings en une séquence de T matrices de forme $N \times d$, où d est la dimension d'embedding. Chacune de ces T matrices représente les embeddings des nœuds du graphe temporel initial à un pas de temps donné. Ainsi, en sortie d'un autoencodeur temporalisé, nous obtenons des données de même forme que celles issues d'une méthode temporelle d'embedding.

3.3 Cadre expérimental

3.3.1 Autoencodeurs temporalisés

Comme indiqué dans la section 3.2, les méthodes d'embedding compatibles avec la technique de temporalisation présentée doivent répondre à certains critères : elles doivent consister en des réseaux de neurones prenant une matrice d'adjacence en entrée. Nous considérons 5 modèles, présentés dans les articles suivants :

- SDNE [9] : Ce modèle construit les embeddings en utilisant un réseau de neurones visant à préserver conjointement les proximités de premier et second ordres des nœuds d'un graphe statique. Outre la dimension d'embedding d, SDNE possède deux hyperparamètres α et ν ainsi que le paramètre du nombre de couches cachées et leurs tailles respectives. Dans les expérimentations à venir, nous utilisons les valeurs par défaut courantes pour α et ν et nous ne considérons qu'une seule couche cachée d'une taille égale à $2d$.

- GAE / VGAE [60] : Cet article décrit deux méthodes différentes avec comme idée principale d'utiliser un réseau à convolution de graphes comme encodeur et un produit scalaire de paires d'embeddings comme décodeur. La différence entre les variantes GAE et VGAE réside dans le fait que la seconde apprend à faire correspondre les données en entrée à une distribution plutôt qu'à un vecteur. Les embeddings sont ensuite obtenus en prenant un échantillon aléatoire de la distribution apprise. Dans les expérimentations de temporalisation, nous utilisons les valeurs par défaut pour les hyperparamètres des deux modèles (i.e. le *learning rate*, le *decay* et le *dropout*). Pour la couche cachée, nous fixons sa taille au double de la dimension d'embedding.

- ARGAE / ARGVAE [28] : Dans ce travail, les modèles GAE et VGAE ont été repris et modifiés en utilisant la régularisation adversariale (*adversarial regularization* [61]) pour forcer les embeddings à correspondre à une distribution préalable. De manière similaire à [60], deux variantes sont conçues, ARGAE et sa version variationnelle ARGVAE. Dans les expérimentations à venir, nous conservons les mêmes valeurs d'hyperparamètres que celles utilisées pour GAE/VGAE.

Le tableau 3.1 reporte les implémentations reprises ainsi que les différentes valeurs d'hyperparamètres utilisées pour chacun des autoencodeurs temporalisés.

GAE/VGAE		**ARGAE/ARGVAE**		**SDNE**	
github.com/tkipf/gae		github.com/GRAND-Lab/ARGA		github.com/shenweichen/GraphEmbedding	
Hyperparamètre	Valeur	Hyperparamètre	Valeur	Hyperparamètre	Valeur
epochs	100	epochs	100	epochs	100
lr	0.01	lr	0.05	nb hid.	2
hid. 1	32	hid. 1	32	hid. 1	32
hid. 2	16	hid. 2	16	hid. 2	16
dropout	0	dropout	0	α	10^{-6}
w. decay	0	w. decay	0	β	5
		disc. out	0	ν_1	10^{-5}
		disc. lr	0.001	ν_2	10^{-4}

TABLEAU 3.1: Implémentations et valeurs d'hyperparamètres des autoencodeurs temporalisés.

Pour les besoins de la temporalisation, deux hyperparamètres supplémentaires, s et w, sont nécessaires, comme décrit dans le section 3.2.1. Les tests sont réalisés sur une grid-search selon $(s, w) \in \{0, 1\} \times \{1, 2, 3\}$.

Dans la suite de ce chaptire, un autoencodeur temporalisé sera noté *TT* (*trained temporalization*), par exemple SDNE_TT ou GAE_TT.

3.3.2 Méthodes de référence

Pour évaluer l'approche de temporalisation, nous comparons les performances des autoencodeurs temporalisés à d'autres modèles d'embedding statiques et temporels.

Comme modèles statiques, nous considérons DW [5], N2V [6] ainsi que les autoencodeurs statiques originaux à temporaliser, à savoir SDNE, GAE, VGAE, ARGAE et ARGVAE. De plus, pour challenger l'étape où les poids temporels sont appris, nous temporalisons chacune des méthodes d'embedding statiques en utilisant des poids d'arêtes temporelles fixes (non entraînables), de manière similaire aux autres méthodes basées sur la notion de matrices de supra-adjacence : [56–58, 62]. Dans ce contexte, différentes stratégies d'assignation de poids temporels fixes sont considérées :

- Attribuer à toutes les arêtes temporelles un poids identique, égal à la valeur maximale de tous les poids du graphe temporel en entrée (tous pas de temps confondus).
- Attribuer à toutes les arêtes temporelles un poids égal à la valeur moyenne des poids du graphe temporel en entrée.
- Pour chaque nœud, attribuer à toutes les arêtes temporelles qui lui sont associées la valeur maximale des poids de toutes ses arêtes dans le graphe en entrée.
- Pour chaque nœud, attribuer à ses arêtes temporelles la valeur moyenne de ses poids dans le graphe en entrée.

Dans ce qui suit, cette méthode de *temporalisation fixe* sera notée *FT*, par exemple SDNE_FT ou GAE_FT.

Pour les méthodes de référence d'embedding temporel, nous utilisons DT [15] et TN2V [19]. Ci-dessous, les jeux d'hyperparamètres testés pour les différentes méthodes de référence considérées dans la comparaison :

- DW : avec wl et ws représentant respectivement la longueur de la marche et la taille de la fenêtre glissante, une grid-search selon $(wl, ws) \in \{40, 80, 120\} \times \{3, 5, 7\}$.

- N2V : nous conservons les valeurs de wl et ws donnant les meilleures performances pour DW, puis nous effectuons une grid-search selon $(p, q) \in \{0.5, 1, 2\}^2$.
- DT : une grid-search selon $(\beta_0, \beta_1) \in \{0, 01, 0, 1, 1, 10\}^2$.
- TN2V : nous reprenons les valeurs d'hyperparamètres et la procédure explicitées en section 2.2.4.1

3.3.3 Jeux de données et tâches d'inférence

Afin de comparer les performances des différents algorithmes, nous reprenons et modifions les 3 jeux de données présentés en section 2.2.4.2. Dans un souci de temps de traitement de la temporalisation, nous extrayons, pour chacun des 3 jeux de données, une partie des nœuds (ceux qui interagissent le plus) et nous ne considérons qu'une partie des pas de temps. Le tableau 3.2 présente les extraits sur lesquels les méthodes ont été comparées.

Jeu de données	nœuds	arêtes	pas de temps
AMiner	2 385	11 371	8
Yelp	2 445	2 839	7
Tmall	2 586	4 152	8

TABLEAU 3.2: Les différents jeux de données considérés dans les expérimentations d'évaluation de la temporalisation d'autoencodeurs.

Étant donné que les trois jeux de données considérés ont un nombre de nœuds du même ordre de grandeur et environ une douzaine de communautés *ground-truth*, nous considérons une dimension d'embedding $d = 16$ pour chacun d'entre eux. À noter que la dimension d'embedding optimale dépend moins du nombre de nœuds du graphe en entrée (comme mentionné en section 2.2.5) que de la nature des données et du nombre de communautés ground-truth (ainsi que le remarque [11]). Néanmoins, le choix de la dimension d'embedding est un aspect en dehors du cadre du travail sur la temporalisation d'autoencodeurs.

Par ailleurs, les tâches d'inférence que nous utilisons pour l'évaluation des différents modèles sont ceux définis en section 2.2.4.3 à savoir : la classification de nœuds (notée nc), la prédiction de classe de nœuds (np), la reconstruction d'arêtes (er) et la prédiction d'arêtes (ep).

3.4 Expérimentations et résultats

3.4.1 Apport de la temporalisation

D'abord, nous comparons les scores d'inférence des autoencodeurs originaux à leurs versions temporalisées fixe (FT) et entraînée (TT). La figure 3.2 montre les résultats de cette expérimentation. Pour chaque modèle, tâche d'inférence et jeu de données, sont représentés uniquement les meilleurs F1 scores (sur les différents jeux d'hyperparamètres de temporalisation pour TT et sur les différentes stratégies d'assignation de poids temporels fixes pour FT).

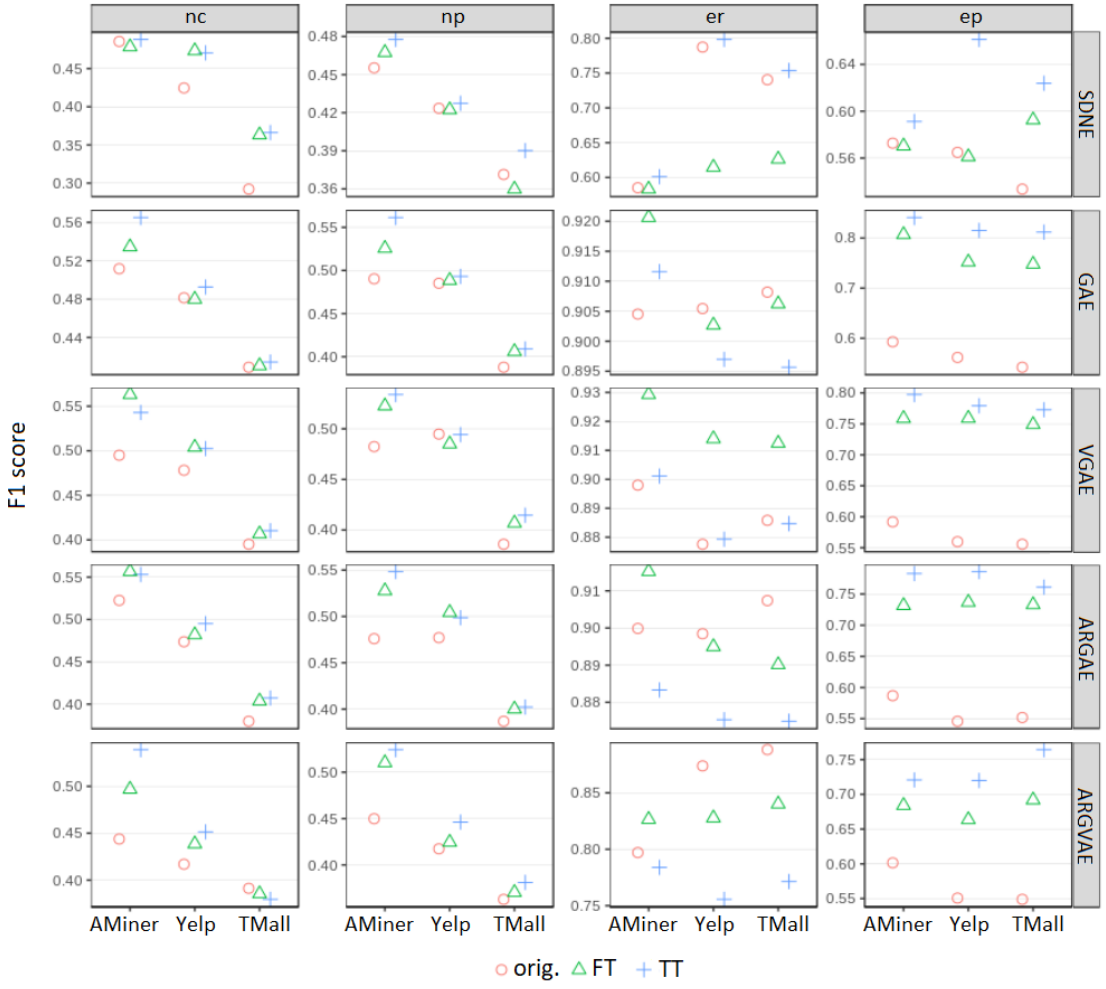


FIGURE 3.2: Améliorations apportées par la temporalisation des autoencodeurs.

Comme anticipé, les autoencodeurs originaux sont moins efficaces que les modèles temporalisés. De plus, nous pouvons observer une différence notable dans le classement des performances des modèles entre les tâches sensibles à la dimension temporelle (c'est-à-dire la prédiction des nœuds et des arêtes) et celles qui y sont insensibles (classification de

nœuds et reconstruction d'arêtes). En effet, pour la tâche de prédiction d'arêtes et, dans une moindre mesure pour la tâche de prédiction de classe de nœud, la temporalisation entraînée (TT) présente de meilleures performances par rapport à la temporisation fixe (FT). L'autoencodeur statique original est surpassé par les deux variantes de temporalisation. D'abord, cela signifie que la temporalisation des autoencodeurs, de manière fixe ou entraînée, améliore l'efficacité de l'embedding. D'autre part, l'apprentissage des poids temporels apporte une amélioration supplémentaire au processus de temporalisation. Ceci peut être expliqué par la capacité de la temporalisation entraînée à capturer à la fois les structures spatiales et les modèles d'évolution temporels du graphe en entrée. Par ailleurs, l'amélioration de la temporalisation est mineure et non systématique lorsqu'il s'agit de la classification des nœuds et de la reconstruction d'arêtes.

D'un autre coté, nous constatons que les différents autoencodeurs réagissent globalement de la même manière au processus de temporalisation : les améliorations apportées aux tâches sensibles à la dimension temporelle concernent toutes les méthodes d'embedding considérées et ne semblent pas modifier le classement des modèles en F1 score.

3.4.2 Réutilisation des poids temporels appris

Ensuite, nous examinons la possibilité de réutiliser les poids temporels appris dans le processus de temporalisation des autoencodeurs. L'objectif est de déterminer si ces poids temporels sont propres à l'autoencodeur pour lequel ils ont été conçus, ou s'ils peuvent être utilisés au-delà de ce cadre. À cette fin, nous concevons des versions temporalisées (fixes et entraînées) de DW et N2V comme suit :

- DW_FT et N2V_FT : temporalisation fixe selon les 4 différentes stratégies d'assignation des poids temporels (selon la procédure décrite dans 3.3.2).
- DW_TT et N2V_TT : temporalisation fixe (non entraînable) où nous reprenons les poids temporels appris à travers l'autoencodeur temporalisé donnant le meilleur F1 score pour chaque tâche d'inférence et chaque jeu de données.

La figure 3.3 présente les résultats obtenus. Globalement, la temporalisation de DW et N2V augmente fortement les performances pour les tâches d'inférence sensibles ou non à la dimension temporelle. En outre, DW_TT (resp. N2V_TT) présente une amélioration, souvent très faible, mais néanmoins systématique, par rapport à DW_FT (resp. N2V_FT). Cela confirme l'intuition préalable selon laquelle les poids temporels appris capturent des informations sur les schémas d'évolution temporels des nœuds et peuvent donc être utilisés à d'autres fins que celles pour lesquelles ils ont été conçus.

Par ailleurs, une explication possible de la faible différence entre les performances des variantes FT et TT réside dans le fait que le F1 score est déjà très élevé (généralement supérieur à 0,9 pour les tâches liées aux arêtes).

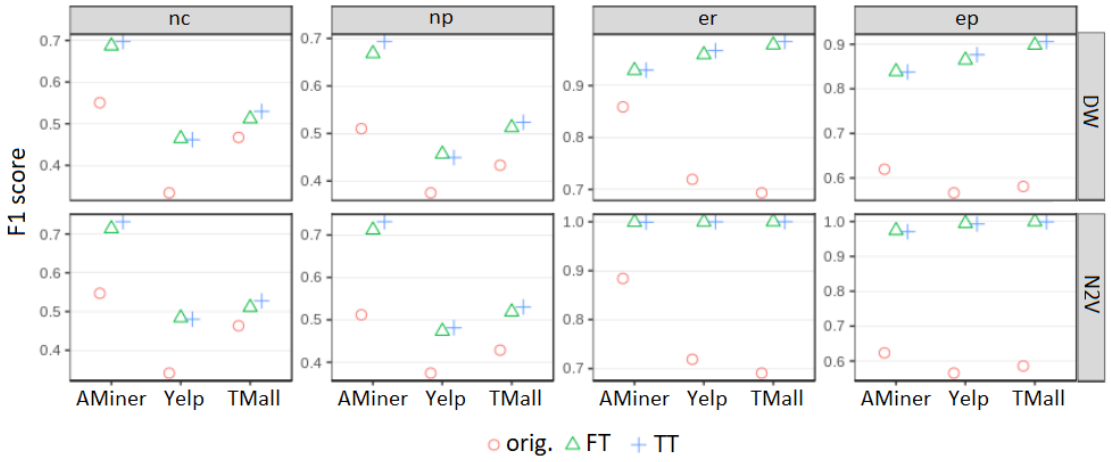


FIGURE 3.3: Temporalisation de DW et N2V à l'aide de poids temporels entrainés et fixes. Pour DW_FT et N2V_FT, sont uniquement reportés les meilleurs F1 scores sur les différentes stratégies de fixation de poids.

3.4.3 Comparaison aux méthodes de référence

Afin de mettre à l'épreuve la méthode de temporalisation, nous comparons les performances des autoencodeurs temporalisés aux approches d'embedding temporelles selon le cadre décrit dans la section 3.3.2. Le tableau 3.3 résume les résultats obtenus : pour chaque méthode, jeu de données et tâche d'inférence, les embeddings donnant les meilleures performances sont reportés.

La remarque la plus intéressante concerne la version temporelle de N2V. En effet, N2V_TT surpasse toutes les autres méthodes (y compris les méthodes temporelles), pour tous les jeux de données et toutes les tâches d'inférence. DW_TT affiche également de bonnes performances, notamment pour les jeux de données AMiner et Yelp. Cela corrobore les observations présentées dans la section 3.4.2 concernant la pertinence des poids temporels appris. Par ailleurs, les autoencodeurs temporalisés présentent des résultats disparates : par exemple, GAE_TT donne des résultats supérieurs à DT sur les tâches relatives aux nœuds et de meilleurs scores que TN2V sur les tâches liées aux arêtes. D'autre part, les résultats de SDNE_TT sont relativement faibles, bien que la temporalisation les ait améliorés.

Il convient de noter que, comme décrit dans le cadre expérimental en section 3.3.1, seules les valeurs par défaut courantes pour les hyperparamètres des autoencodeurs ont été utilisées, contrairement à DT, TN2V, DW_TT, et N2V_TT. Par conséquent, il est

Modèle	**AMiner**				**Yelp**				**Tmall**			
	nc	np	er	ep	nc	np	er	ep	nc	np	er	ep
SDNE_TT	0.49	0.48	0.601	0.591	0.47	0.43	0.798	0.661	0.37	0.39	0.754	0.624
GAE_TT	0.57	0.56	0.912	0.841	0.49	0.49	0.897	0.815	0.41	0.41	0.896	0.812
VGAE_TT	0.54	0.53	0.901	0.797	0.50	0.49	0.879	0.779	0.41	0.41	0.885	0.773
ARGAE_TT	0.55	0.55	0.883	0.783	0.50	0.50	0.875	0.786	0.41	0.40	0.875	0.761
ARGVAE_TT	0.54	0.52	0.784	0.721	0.45	0.45	0.756	0.720	0.38	0.38	0.771	0.764
DT	0.56	0.54	**0.997**	**0.927**	0.48	0.45	0.979	**0.957**	0.37	0.37	**0.988**	**0.931**
TN2V	0.61	0.60	0.886	0.747	**0.53**	0.49	0.871	0.813	**0.52**	**0.48**	0.883	0.800
DW_TT	**0.70**	**0.69**	0.930	0.838	**0.53**	**0.52**	**0.985**	0.906	0.46	0.45	0.967	0.877
N2V_TT	**0.73**	**0.73**	**0.999**	**0.971**	**0.53**	**0.53**	**0.999**	**0.999**	**0.48**	**0.48**	**0.999**	**0.993**

TABLEAU 3.3: Autoencodeurs temporalisés vs. modèles d'embedding temporels.

vraisemblablement possible d'améliorer les scores des autoencodeurs temporalisés en optimisant leurs hyperparamètres.

3.4.4 Analyse des hyperparamètres de temporalisation

Par la suite, nous analysons l'impact des hyperparamètres de temporalisation sur les scores d'inférence (figure 3.4). En ce qui concerne l'hyperparamètre s contrôlant la symétrie de la matrice de supra-adjacence, nous constatons que, en dehors de la tâche de reconstruction des arêtes, la prise en compte des arêtes temporelles non orientées est plus avantageuse. Une raison possible justifiant ce résultat pourrait être la nécessité de forcer une forte continuité temporelle entre les embeddings de pas de temps différents pour obtenir de meilleures performances, comme expliqué dans la section 3.2.1.1. En outre, en ce qui concerne la fenêtre temporelle, nous remarquons que, dans l'ensemble, les valeurs plus grandes de w conduisent à de meilleurs scores d'inférence. Cela pourrait être dû au fait qu'une fenêtre temporelle w plus large permet de construire des modèles d'évolution temporels plus sophistiqués, comme souligné en section 3.2.1.2.

3.4.5 Analyse des poids temporels

Enfin, nous nous intéressons à l'interprétation des poids temporels. L'idée est de comprendre la valeur apprise d'un poids temporel en fonction de certaines propriétés des supra-nœuds qu'il connecte. Étant donné une supra-arête entre $\{v, t_i\}$ et $\{v, t_j\}$, nous définissons 3 caractéristiques qui nous semblent être, a priori, des *features* explicatives potentielles :

- Changement d'adjacence ($|\Delta A|$) : la distance euclidienne entre le vecteur d'adjacence de v au pas de temps t_i et celui à t_j. $|\Delta A| = ||Adj_j(v) - Adj_i(v)||$

- Variation du degré pondéré (ΔSW) : la différence entre la somme des poids des arêtes de v aux pas de temps t_i et t_j. $\Delta\text{SW} = \sum poids_j(v) - \sum poids_i(v)$
- Degré pondéré cumulé (SWS) : l'addition des sommes des poids de toutes les arêtes que v possède dans les pas de temps t_i et t_j. $\text{SWS} = \sum poids_i(v) + \sum poids_j(v)$

Après avoir défini ces features, nous calculons les corrélations partielles de Spearman entre la variable cible (i.e. les poids temporels appris) et les features explicatives. La figure 3.5 montre les résultats de cette expérimentation où nous représentons les corrélations ainsi que leurs p-value correspondantes. Différentes observations peuvent être effectuées.

D'abord, nous remarquons que les corrélations sont différentes entre, d'une part, SDNE, ARGAE et ARGVAE et, d'autre part, GAE et VGAE.

- Pour SDNE, ARGAE, et ARGVAE, les features ΔSW et SWS ont globalement une forte corrélation positive avec les poids temporels appris : des corrélations partielles supérieures à 0.31, avec une p-value inférieure à 10^{-187}. De même, il existe

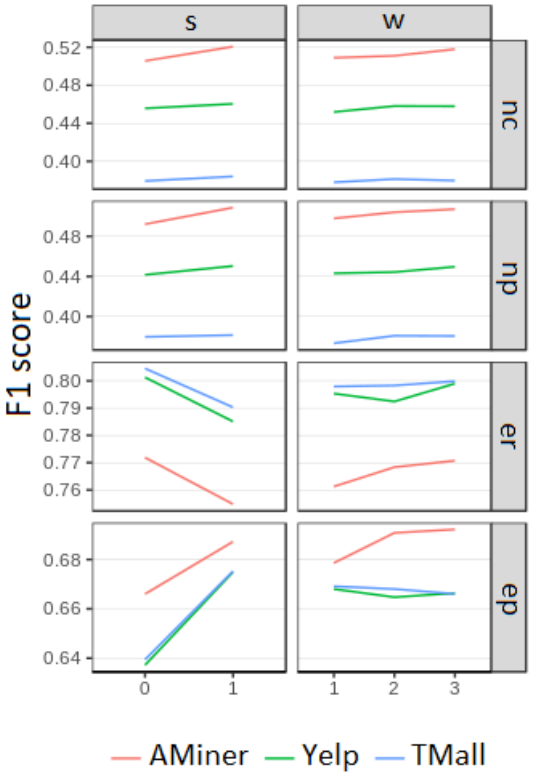


FIGURE 3.4: L'impact des hyperparamètres de temporalisation s et w. Pour l'évaluation de l'influence de s (respectivement w), nous reportons le meilleur score d'inférence pour $w \in \{1, 2, 3\}$ (respectivement pour $s \in \{0, 1\}$).

une corrélation négative moins prononcée entre $|\Delta A|$ et les poids temporels (environ -0.1), avec des p-values inférieures à 10^{-23}. Autrement dit, ces corrélations signifient qu'un nœud v qui change en termes de adjacence (i.e. changement de voisinage) entre deux pas de temps t_i et t_j, ou dont les poids diminuent globalement de t_i et t_j, ou dont les poids sont globalement faibles à t_i et t_j, a généralement un poids temporel relativement faible entre ses supra-nœuds à t_i et t_j. La figure 3.6 résume ces différents scenarii.

- En ce qui concerne GAE et VGAE, il semble que les observations ci-dessus, faites sur les autres autoencodeurs temporalisés, soient toujours appropriées, mais avec quelques anomalies notables. D'abord, les corrélations partielles entre $|\Delta A|$ et les poids temporels sur AMiner sont positives contrairement aux autres jeux de données et aux autres autoencodeurs. Par ailleurs, pour ΔSW et SWS, les corrélations partielles sont significativement plus faibles par rapport aux 3 autres autoencodeurs. De plus, les p-values des corrélations partielles pour les features explicatives sont globalement moins marquées.

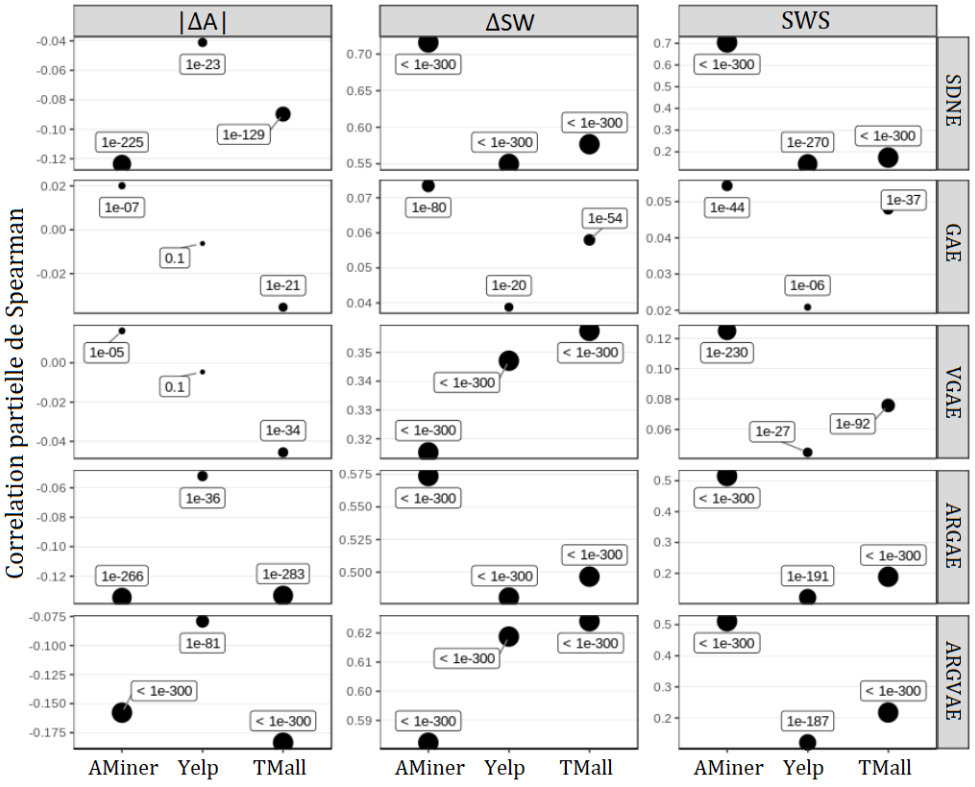


FIGURE 3.5: Analyse des poids temporels : corrélations partielles de Spearman p-values associées entre les poids temporels appris et les features explicatives des nœuds. Plus les p-values sont petites, plus la taille des points est grande (les valeurs p-values sont encadrées).

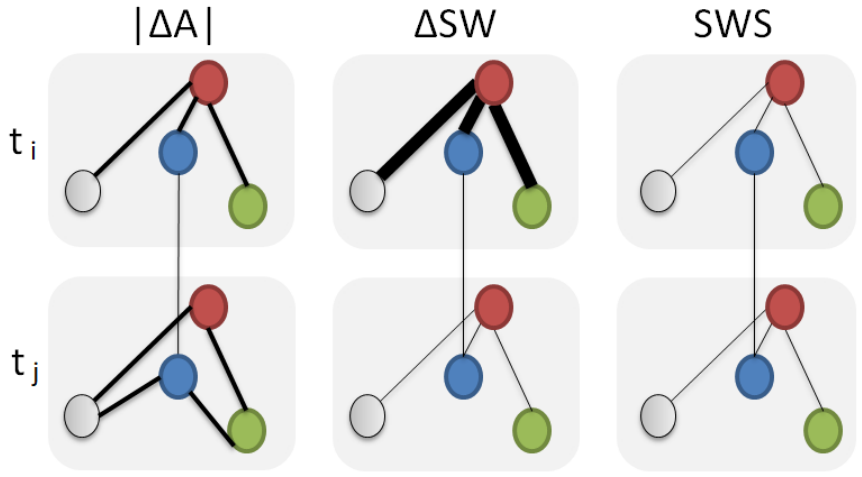


FIGURE 3.6: Situations qui ont tendance à produire de faibles poids temporels (représentés par un trait fin) : un nœud qui change de voisinage (a), ou qui voit son activité baisser (b), ou encore dont l'activité est faible dans la durée (c).

Pour résumer, bien que les poids temporels soient appris dans le cadre d'un processus assimilable à une *boîte noire* — au sens où il est difficile de connaître la logique conduisant la temporalisation à attribuer une certaine valeur à un poids temporel —, il est possible d'interpréter statistiquement et partiellement les résultats obtenus vis-à-vis des propriétés des nœuds du graphe temporel initial. Ce constat n'est pas étonnant. En effet, et comme démontré dans la section 3.4.2, les poids temporels contiennent une information utile sur les structures du graphe temporel en entrée, indépendante de l'autoencodeur temporalisé et du processus de temporalisation. À l'échelle locale, il est probable que cette information capturée soit directement liée à certaines caractéristiques spatio-temporelles des nœuds, telles que celles que nous avons définies (à savoir $|\Delta A|$, ΔSW et SWS).

3.5 Conclusions et perspectives

Dans ce chapitre, nous avons présenté une méthode de temporalisation d'autoencodeurs de graphes ayant pour but l'adaptation d'une catégorie de méthodes d'embedding statiques au cas des graphes temporels. Cela est rendu possible en modifiant le concept de matrices de supra-adjacence pour que certains de ses éléments puissent être entraînés, à savoir les poids temporels exprimant la relation temporelle entre un nœud et lui-même à différents pas de temps. La structure des autoencodeurs est légèrement modifiée pour apprendre les poids optimaux à attribuer aux arêtes temporelles. Ce faisant, les autoencodeurs temporalisés construisent des embeddings plus précis pour différentes tâches d'inférence sensibles à la dimension temporelle. Par ailleurs, il apparaît que

les poids temporels appris revêtent une utilité qui dépasse le cadre de l'autoencodeur pour lequel ils ont été conçus, dans la mesure où ils peuvent être employés pour temporaliser d'autres méthodes d'embedding statiques et améliorer considérablement leurs performances. D'autre part, les expérimentations menées démontrent des corrélations intéressantes entre les poids temporels appris et certaines caractéristiques temporelles des nœuds du graphe en entrée.

3.5.1 Incorporation de métadonnées

Dans les expérimentations menées, 5 autoencodeurs ont été temporalisés. Or il est opportun de rappeler que la méthode de temporalisation présentée est générique et qu'il est possible de l'adapter à tout autoencodeur de type réseau de neurones prenant en entrée une matrice d'adjacence. En ce sens, l'autoencodeur temporalisé est à considérer comme un hyperparamètre en soi, au même titre que s et w.

Par ailleurs, des méthodes d'embedding plus sophistiquées peuvent également être temporalisées. En l'occurrence, si l'autoencodeur considéré permet de prendre en compte certaines métadonnées supplémentaires des nœuds, la version temporalisée qui en résulte le permet également. C'est par exemple le cas pour GAE/VGAE [60] et ARGAE/ARGVAE [28]. De plus, la temporalisation présentée permet de prendre en compte des métadonnées dynamiques des nœuds. En effet, les états d'un nœud à différents pas de temps étant transposés en des supra-nœuds différents, il est possible de faire correspondre à chacun d'entre eux des métadonnées différentes.

3.5.2 Temporalisation d'autres types de GNN

Telle que présentée, la méthode décrite est définie pour temporaliser des autoencodeurs avec pour objectif premier d'apprendre des représentations temporelles de graphes. Il est néanmoins possible d'adapter et d'appliquer le processus de temporalisation à des GNN (*graph neural networks*) d'autres types, adressant des problèmes différents de celui de l'apprentissage de représentations. Par exemple, il est possible de temporaliser des modèles de génération de graphes tels que [63, 64]. Ce type de méthodes composées de réseaux de neurones, prennent en entrée des matrices d'adjacence de graphes statiques et retournent des graphes générés aux propriétés similaires à ceux en entrée. En utilisant la méthode de temporalisation sur de telles approches, l'idée serait de générer un graphe temporel simulé semblable à celui en entrée.

3.5.3 Temporalisation *lightweight*

Au vu des résultats présentés en section 3.4.5, une analyse approfondie visant à découvrir les corrélations existant entre les différentes caractéristiques des nœuds et les poids temporels appris pourrait déboucher sur une méthode de temporalisation *lightweight*, plus rapide et plus générique. En effet, l'apprentissage d'un autocodeur temporalisé est une opération relativement lourde : sur les jeux de données considérés, cela prend environ 90 minutes sur une machine à 24 cœurs et 64 Go de RAM. De plus, si les avantages de la temporalisation semblent être évidents sur les autoencodeurs temporalisés, la réutilisation des poids temporels sur d'autres méthodes d'embedding statiques apporte des améliorations mineures par rapport à la temporalisation fixe, comme le suggère la section 3.4.2. Par conséquent, il est possible d'envisager la conception d'une méthode de temporalisation définie par l'utilisateur : l'idée serait de créer des poids temporels en se basant sur des caractéristiques spatio-temporelles des nœuds (à l'instar de celles que nous avons définies en 3.4.5) et en se référant aux corrélations partielles découvertes. Un tel processus serait une méthode plus simple (en scalabilité et en temps de traitement) pour temporaliser tout type d'approche d'embedding statique (de manière similaire à DW_TT et N2V_TT) sans aucune contrainte sur sa structure ou son type de données d'entrée, contrairement à la méthode de temporalisation proposée.

Chapitre 4

Alignement et Stabilité Temporels des Embeddings

En apprentissage de représentations, les dimensions du référentiel d'embedding latent ne revêtent, a priori, pas de signification physique précise. En effet, les coordonnées d'un objet représenté dans l'espace latent, i.e. son vecteur d'embedding, ne contiennent pas d'information de manière autonome : la donnée pertinente réside dans la position relative du vecteur d'embedding par rapport à ceux des autres objets représentés. En d'autres termes, l'information utile se trouve dans les distances entre les embeddings. Or, ces distances sont invariantes vis-à-vis de certains types de transformations de l'espace latent d'embedding. Ainsi, dans le cas de l'apprentissage de représentation des nœuds d'un graphe statique, il est possible de construire différents embeddings fournissant exactement les mêmes informations, par exemple en fonction des conditions initiales d'une méthode d'apprentissage de représentation. Généralement, cela ne pose pas de problème dans le cas de graphes statiques et de tâches d'inférence insensibles à la dimension temporelle dans la mesure où un seul espace latent est suffisant pour représenter tous les nœuds du graphe.

Contrairement aux structures statiques, les systèmes dynamiques contiennent également des informations temporelles. Par exemple, le sens d'un mot peut changer au cours du temps et les métadonnées des nœuds d'un graphe peuvent évoluer. Dans le premier cas, la différence entre les vecteurs d'embedding d'un mot à deux moments différents peut indiquer sa dérive sémantique. Dans le second, l'évolution des frontières de décision pour une tâche de classification de nœuds, apprises à des pas de temps différents, pourrait donner une indication sur les potentiels changements d'appartenance aux communautés. Cependant, pour étudier cette évolution temporelle à l'aide d'embeddings, il est

nécessaire de disposer d'un système de référence de l'espace latent qui soit fixe dans le temps.

Dans le cadre d'un graphe temporel donné, les différences entre les représentations apprises pour différents pas de temps peuvent être imputables à des changements réels dans le graphe ou à des transformations qui ne modifient pas les distances entre nœuds dans l'espace latent. Dans le premier cas, ces changements sont liés à la *stabilité* du graphe dans le temps tandis que dans le second, ils correspondent à un défaut d'*alignement* des embeddings. Il s'agit là d'un artefact introduit par la méthode d'apprentissage de représentation employée. Par exemple, de très légères modifications dans les données en entrée sont susceptibles de produire des embeddings radicalement différents d'un pas de temps à un autre lorsque les embeddings desdits pas de temps sont appris de manière indépendante à l'aide d'une méthode statique : en effet, d'infimes changements en entrée peuvent modifier de manière significative les régions dans lesquelles s'inscrivent les embeddings appris. Dans un cas plus extrême, les mêmes données d'entrée peuvent donner lieu à des embeddings différents en raison de l'aspect non déterministe de nombreuses méthodes d'embedding. En résumé, la stabilité caractérise la dynamique du système lui-même, tandis que le *désalignement* caractérise les changements parasites dans l'espace latent. Dans la suite de ce chapitre, des embeddings d'un graphe temporel sont dits *alignés* lorsque les différences entre les embeddings de pas de temps consécutifs reflètent les changements réels du graphe plutôt que des transformations propres à l'espace latent.

Dépendamment du contexte, les questions liées à la stabilité et à l'alignement des embeddings d'un graphe temporel peuvent être d'une importance primordiale. Tel est par exemple le cas dans les tâches d'inférence sensibles à la dimension temporelle (e.g. prédiction d'arêtes), dans l'évolution temporelle ou encore dans la visualisation dynamique. Bien que cette problématique soit connue dans l'état de l'art, elle n'a pas été définie formellement, explorée théoriquement ou analysée empiriquement de manière suffisante.

Le travail exposé dans ce chapitre présente plusieurs contributions. L'alignement d'embeddings temporels est défini de manière formelle. Il est décomposé et ses différents éléments constitutifs sont expliqués et interprétés. De nouvelles métriques appropriées à la mesure de l'alignement et de la stabilité sont élaborées, puis les fondements mathématiques sous-jacents sont justifiés. Un ensemble d'expérimentations sur des données synthétiques puis réelles est mené afin de démontrer la robustesse et l'utilité des métriques conçues. En parallèle, une méthode de *réalignement* des embeddings temporels est conçue. Les représentations en sortie de plusieurs méthodes d'embeddings, statiques et temporelles, sur plusieurs jeux de données, sont analysées à travers le prisme des métriques proposées, puis l'apport du réalignement est évalué.

Ce travail d'analyse de l'alignement et de la stabilité d'embeddings temporels de graphes a été mené conjointement avec un doctorant de l'université du Bosphore (Istanbul, Turquie) et a fait l'objet d'un article [65] soumis à la revue scientifique " Neurocomputing ".

4.1 Contexte et travaux connexes

L'alignement est l'un des principaux inconvénients de l'application de méthodes d'embedding statiques à l'apprentissage de représentation de graphes temporels. En effet, des embeddings calculés de manière indépendante d'un pas de temps à un autre ne se conforment a priori à aucune cohérence temporelle : les distances euclidiennes entre paires d'embeddings sont invariantes vis-à-vis de certaines opérations comme la translation, la rotation ou la réflexion. Autrement dit, les embeddings de pas de temps consécutifs calculés à partir d'une méthode statique peuvent être décalés ou pivotés l'un par rapport à l'autre (comme le montre la figure 4.1) en raison du fait qu'ils sont placés dans des espaces latents a priori différents. Dans une telle situation, les embeddings temporels sont dits *désalignés*.

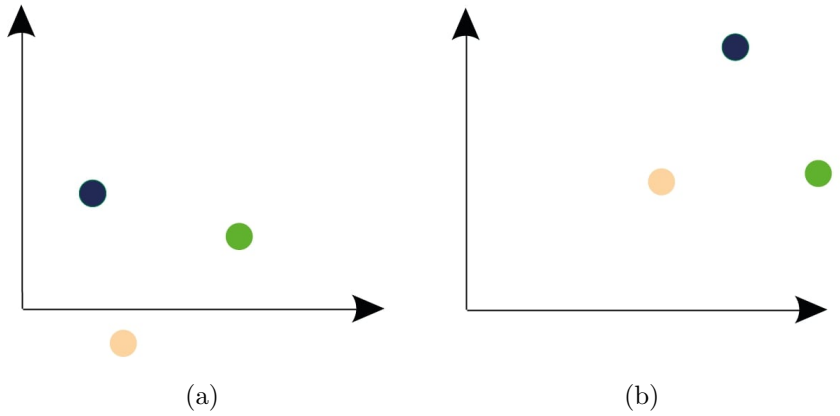


FIGURE 4.1: Exemple de désalignement. (a) Échantillon d'embedding au pas de temps t. (b) Les embeddings des mêmes nœuds au pas de temps $t+1$. Bien que les distances soient préservées entre les deux pas de temps, les vecteurs d'embedding sont différents : l'évolution est entièrement induite par une translation et une rotation.

À l'inverse, les approches d'embedding temporelles créent des embeddings lissés dans le temps et, par conséquent, des embeddings mieux alignés que les approches statiques. Néanmoins, ces méthodes n'adressent généralement pas les problèmes liés à l'alignement de manière directe. Certains travaux de l'état de l'art mentionnent cette problématique sans pour autant proposer une méthode explicite pour la traiter : [66, 67]. Dans l'embedding de mots, [30] utilise une régression localement linéaire pour forcer l'alignement d'un mot central par rapport à son voisinage, dans différents pas de

temps. Cependant, cette opération doit être appliquée pour chacun des mots de manière indépendante. Certaines études [68, 69] utilisent l'analyse procrustéenne orthogonale [70] : l'idée est de trouver et d'appliquer la matrice orthogonale (i.e. matrice de rotation et/ou réflexion) qui fait correspondre le mieux les embeddings de deux pas de temps consécutifs, via une décomposition en valeurs singulières d'une certaine matrice tirée des embeddings. Il convient de noter que l'analyse procrustéenne orthogonale n'offre pas la possibilité de trouver des translations possibles entre des embeddings de différents pas de temps. Cela peut potentiellement être un inconvénient quand la métrique de distance considérée dans l'espace latent d'embedding est la distance euclidienne plutôt que le produit scalaire ou la similarité cosinus. Ainsi, d'autres travaux [71–74] utilisent l'analyse procrustéenne généralisée [75] où les opérations de translation et de mise à l'échelle optimales sont autorisées en plus de la rotation et de la réflexion.

Nous considérons que deux ensembles d'embeddings sont alignés quand il n'existe pas de mouvement global (translation, rotation, réflexion) entre les vecteurs qui les composent. Outre son aspect pratique, cette définition découle d'une observation que nous pouvons remarquer : sous toutes leurs formes possibles (graphes, matrices d'adjacence, listes d'arêtes...), les données en entrée des méthodes d'embedding représentent généralement des interactions entre nœuds. Or ces données ne sauraient contenir d'informations sur un éventuel mouvement de masse, d'un pas de temps à un autre, qui concernerait les nœuds dans leur globalité, étant donné qu'il n'existe pas de relations *inter-pas-de-temps* entre nœuds différents (i.e. arêtes reliant deux nœuds différents à deux pas de temps différents). De ce fait, tout mouvement global d'embeddings dans le temps est imputable au désalignement. Sur cette base, nous définissons également le réalignement et la mesure de l'alignement comme étant les processus de découverte et de quantification des transformations linéaires optimales rapprochant le plus possible les embeddings de deux pas de temps.

Il existe quelques travaux de l'état de l'art abordant le problème de la mesure de l'alignement des embeddings. [76] et [77] définissent une quantité de stabilité (ou de continuité) temporelle censée mesurer le *désalignement* entre les embeddings de deux pas de temps. L'idée générale est d'analyser la corrélation entre le changement de voisinage d'un nœud et son déplacement dans le temps dans l'espace latent. Cependant, une telle quantité englobe deux notions connexes mais différentes, à savoir l'alignement et la stabilité. Telle que nous la concevons, la stabilité reflète les changements qui peuvent se produire entre les embeddings de pas de temps consécutifs et qui ne sont pas liés à des décalages en alignement. Pour une méthode produisant des embeddings idéalement alignés, la stabilité est directement liée à la dynamique du réseau considéré, c'est-à-dire à son évolution structurelle à différentes échelles. Il convient également de noter que l'instabilité peut également être causée par la méthode d'embedding utilisée en raison du

caractère aléatoire du processus de création de la représentation. En définitive, plusieurs problématiques demeurent relativement sous-étudiées dans l'état de l'art, telles que la dissociation entre l'alignement et la stabilité des embeddings temporels et la définition de grandeurs valides, pertinentes et utiles pour les mesurer.

4.2 Alignement et stabilité

Comme mentionné précédemment, plusieurs types d'opérations pouvant s'insinuer entre deux matrices d'embedding consécutives peuvent causer un désalignement. C'est le cas des translations, des rotations et des réflexions. Par conséquent, une mesure appropriée de l'alignement devrait quantifier l'amplitude de chacune de ces opérations. En effet, essayer de mesurer l'alignement à l'aide d'une seule valeur peut masquer différentes situation : par exemple, une rotation d'un certain angle θ et une translation de vecteur directeur $\vec{t}$ pourraient produire la même mesure d'alignement ; de telles équivalences ne semblent pas être pertinentes. Au lieu de cela, nous choisissons de concevoir une mesure d'alignement pour chacune des opérations qui entrent en jeu dans l'alignement.

4.2.1 Erreur en translation

Cette mesure vise à quantifier le déplacement global moyen entre deux embeddings consécutifs. Plus précisément, nous nous intéressons au déplacement du centre de gravité dans le temps. Étant donné deux matrices d'embeddings consécutives $E^t = \{e_i^t,\ i \in [\![1,\ N]\!]\}$ et $E^{t+1} = \{e_i^{t+1},\ i \in [\![1,\ N]\!]\}$, nous définissons le déplacement global comme étant :

$$t_{glob} = \left\| o^{t+1} - o^t \right\| \quad avec \quad o^t = \frac{\sum_{i=1}^{N} e_i^t}{N} \tag{4.1}$$

À signaler que t_{glob} n'est pas une grandeur objectivement représentative en raison de sa dépendance à l'espace d'embedding. Il faut donc la normaliser par rapport à certaines distances caractéristiques des embeddings dans l'espace latent. En l'occurrence, nous choisissons les rayons des embeddings, i.e. les distances moyennes respectives aux centres de gravité :

$$t_{norm} = \frac{t_{glob}}{r^t + r^{t+1}} \quad \text{avec} \quad r^t = \frac{\sum_{i=1}^{N} \left\| e_i^t - o^t \right\|}{N} \tag{4.2}$$

Par ailleurs, il est intéressant de noter que, comme présenté dans la figure 4.2, t_{norm} est égal à 1 lorsque les embeddings des deux pas de temps sont *tangents* en termes de rayon.

La grandeur t_{norm} prend ses valeur dans $\mathbb{R}^+$. Afin de borner l'erreur de translation finale ξ_{tr}, nous définissons :

$$\xi_{tr} = \frac{t_{norm}}{t_{norm} + 1} \tag{4.3}$$

Par conséquent, l'erreur de translation ξ_{tr} est comprise entre 0 et 1.

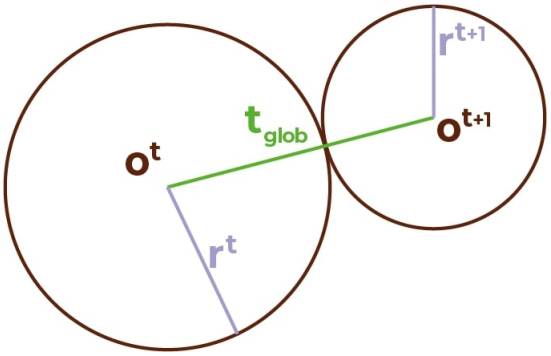


FIGURE 4.2: Embeddings tangents en termes de rayon. o^t, r^t et t_{glob} désignent respectivement le centre de gravité des embeddings au pas de temps t (défini dans l'équation 4.1), leur rayon (équation 4.2) et la translation globale (équation 4.1). Dans ce cas, t_{norm} est égal à 1 et l'erreur de translation ξ_{tr} est égale à 0.5.

Enfin, après avoir mesuré l'erreur en translation, et afin de corriger le désalignement relatif à la translation, nous centrons les deux embeddings comparés en déportant leurs centres de gravité à l'origine de l'espace latent d'embedding. Nous obtenons ainsi les embeddings centrés C^t et C^{t+1} :

$$C^t = \{c_i^t\} \quad \text{avec} \quad c_i^t = e_i^t - o^t \quad \text{pour} \quad i \in [\![1,\ N]\!]. \tag{4.4}$$

4.2.2 Erreur en rotation

Étant donné deux embeddings centrés de pas de temps consécutifs C^t et C^{t+1}, il est possible de trouver la matrice optimale de rotation et/ou réflexion (i.e. matrice orthogonale) R à appliquer à C^t pour faire correspondre, de la manière la plus proche possible, les embeddings des deux pas de temps. Pour ce faire, nous pouvons recourir à l'analyse procrustéenne orthogonale [70] :

$$R = argmin_\Omega \left\| C^t\ \Omega - C^{t+1} \right\| \tag{4.5}$$

Afin de quantifier la rotation/réflexion entre les embeddings, nous comparons R à la matrice identité I : dans le cas idéal où $R = I$, C^t et C^{t+1} sont d'emblée correctement

orientées l'une par rapport à l'autre. Nous définissons l'erreur en rotation ξ_{rot} comme suit :

$$\xi_{rot} = \frac{\|R - I\|_F}{2\sqrt{d}} = \frac{\sqrt{2\,d - 2\,Tr(R)}}{2\sqrt{d}} \tag{4.6}$$

où d est la dimension d'embedding, $Tr(R)$ est la trace de la matrice R et $\|R - I\|_F$ représente la norme Frobenius de la matrice $R - I$. L'égalité entre les deux expressions de ξ_{rot} formulées dans l'équation 4.6 est démontrée dans l'annexe B. En divisant par $2\sqrt{d}$, nous ramenons ξ_{rot} à varier entre 0 à 1. À ce stade, nous pouvons nous poser la question de la capacité de ξ_{rot} à quantifier le niveau de rotation/réflexion que représente R. Afin de répondre à cette question, nous introduisons R_{can}, la forme canonique de R. En effet, chaque matrice orthogonale peut être décomposée en a rotations élémentaires et écrite sous la forme d'une matrice diagonale par blocs, en utilisant la décomposition de Schur par exemple :

$$R_{can} = Q^T R Q = \begin{bmatrix} R_1 & & & & & \\ & \ddots & & & & \\ & & R_a & & & \\ & & & \pm 1 & & \\ & & & & \ddots & \\ & & & & & \pm 1 \end{bmatrix} \text{ avec } R_j = \begin{bmatrix} cos(\theta_j) & -sin(\theta_j) \\ sin(\theta_j) & cos(\theta_j) \end{bmatrix} \tag{4.7}$$

et Q une matrice orthogonale. Plus précisément, comme le montre le tableau 4.1, la forme de R_{can} dépend de la parité de la dimension d'embedding d et du signe du déterminant de R (i.e. si R est une rotation seulement ou une rotation couplée à une réflexion) :

	Dimension impaire	**Dimension paire**
Rotation uniquement det(R) = 1	$\begin{bmatrix} R_1 & & & \\ & \ddots & & \\ & & R_{\frac{d-1}{2}} & \\ & & & 1 \end{bmatrix}$	$\begin{bmatrix} R_1 & & \\ & \ddots & \\ & & R_{\frac{d}{2}} \end{bmatrix}$
Rotation/réflexion det(R) = -1	$\begin{bmatrix} R_1 & & & \\ & \ddots & & \\ & & R_{\frac{d-1}{2}} & \\ & & & -1 \end{bmatrix}$	$\begin{bmatrix} R_1 & & & \\ & \ddots & & \\ & & R_{\frac{d-2}{2}} & \\ & & & 1 \\ & & & & -1 \end{bmatrix}$

TABLEAU 4.1: Forme canonique d'une matrice orthongonale. La forme dépend de la parité de d et du déterminant de R.

Il est possible de considérer R_{can} comme une expression de R dans une base où elle peut être décomposée en a rotations élémentaires (i.e. planaires) occurant dans des plans orthogonaux les uns par rapport aux autres (avec $\{\theta_j, j \in [\![1,\ a]\!]\}$ comme angles de rotation), et éventuellement une réflexion. Les deux matrices R et R_{can} ont la même erreur de rotation ξ_{rot} car elles partagent la même trace. De plus, vu que la trace de R_{can} est liée aux angles θ_j, ξ_{rot} peut s'écrire comme suit :

$$\xi_{rot} = \frac{\sqrt{2d - 4\sum_{j=1}^{A} cos(\theta_j) - 2\mu}}{2\sqrt{d}} \quad \text{avec} \quad \mu = \begin{cases} 1 & \text{si } d \text{ est impair et } \det(R) = 1 \\ -1 & \text{si } d \text{ est impair et } \det(R) = -1 \\ 0 & \text{si } d \text{ est pair} \end{cases} \tag{4.8}$$

Ainsi, nous pouvons observer que plus les angles de rotations élémentaires θ_j sont faibles, plus la valeur de ξ_{rot} est faible. De même, les réflexions augmentent l'erreur de rotation.

Après avoir mesuré l'erreur de rotation/réflexion, nous procédons à la rotation de la matrice d'embedding C^t selon la matrice R afin d'obtenir des embeddings consécutifs centrés et correctement orientés :

$$C_r^t = C^t\, R \tag{4.9}$$

Nous considérons les matrices bien orientées centrées C_r^t et C^{t+1} comme étant les versions alignées des embeddings en entrée.

4.2.3 Erreur en échelle

Cette mesure vise à examiner le changement temporel d'échelle des embeddings alignés. À cette fin, nous définissons l'erreur en échelle sur la base des rayons des embeddings, i.e. les distances moyennes à l'origine du référentiel latent :

$$\xi_{sc} = \frac{|r^t - r^{t+1}|}{r^t + r^{t+1}} \tag{4.10}$$

Cette grandeur est comprise entre 0 et 1. En particulier, ξ_{sc} est nulle lorsque les embeddings ont exactement le même rayon. Au contraire, ξ_{sc} s'approche de 1 lorsque les rayons des embeddings sont très différents. Une fois l'erreur en échelle mesurée, nous normalisons les embeddings centrés correctement orientés vis-à-vis de leurs rayons respectifs :

$$G^t = \frac{C_r^t}{r^t} \quad \text{et} \quad G^{t+1} = \frac{C^{t+1}}{r^{t+1}} \tag{4.11}$$

À noter que l'erreur en échelle n'est pas considérée comme faisant partie de l'alignement. En effet, contrairement à la translation, la rotation et la réflexion, les matrices de distance entre paires d'embeddings ne sont pas invariantes en cas de changement d'échelle. Cependant, le changement d'échelle peut indiquer l'évolution de la densité des embeddings et, par conséquent, le changement de la force des interactions entre les nœuds du graphe en entrée. Il apparaît donc qu'il est opportun de pouvoir mesurer cet effet.

4.2.4 Erreur de stabilité

Étant donné deux embeddings centrés, correctement orientés et normalisés, il est intéressant d'examiner l'évolution des embeddings dans le temps. En effet, la différence entre les caractéristiques respectives de $G^t = \{g_i^t\}$ et $G^{t+1} = \{g_i^{t+1}\}$ ne peut être attribuée à un éventuel désalignement. Ainsi, la comparaison entre G^t et G^{t+1} donne des informations sur la dynamique structurelle du graphe représenté, telle que la discontinuité temporelle des embeddings ou le bruit existant entre des pas de temps consécutifs. Nous définissons l'erreur de stabilité comme suit :

$$\xi_{st} = \frac{\sum_{i=1}^{N} \frac{\left\|g_i^{t+1} - g_i^t\right\|}{\left\|g_i^t\right\| + \left\|g_i^{t+1}\right\|}}{N} \tag{4.12}$$

L'erreur de stabilité ξ_{st} varie entre 0 à 1. Elle est égale à 0 si et seulement si $G^t = G^{t+1}$.

Bien que l'erreur de stabilité soit calculée après les processus de mesure de l'alignement et de mise à l'échelle, elle apporte des informations supplémentaires sur la pertinence des autres mesures définies. Par exemple, dans le cas extrême où nous comparons deux matrices d'embedding générées aléatoirement, nous pouvons toujours obtenir ξ_{tr}, ξ_{rot} et ξ_{sc}. Cependant, ces valeurs ne sont pas pertinentes car elles ne sont pas porteuses de sens. L'erreur de stabilité peut aider à relever ces cas, en produisant une valeur relativement élevée.

4.3 Expérimentations sur des données synthétiques

4.3.1 Procédure de simulation de données

Afin de démontrer la pertinence et les caractéristiques de chacune des mesures d'erreur proposées, nous concevons et réalisons des expérimentations sur des données synthétiques (pour lesquelles la vérité de terrain est par conséquent connue). Nous explorons le comportement de chacune des mesures d'erreur suite à différentes combinaisons de transformations (translations, rotations, réflexions, changements d'échelle, introductions de bruit) appliquées à une matrice initiale E^t pour créer une matrice transformée E^{t+1} (E^t et E^{t+1} correspondent respectivement aux embeddings des pas de temps t et $t+1$). Les mesures d'erreur sont ensuite calculées pour étudier la stabilité et l'alignement des deux embeddings. L'idée est de mettre en évidence les corrélations entre les transformations introduites et les erreurs mesurées pour l'alignement, l'échelle et la stabilité, et ce afin de valider la pertinence de ces différentes mesures et leur capacité à refléter les effets qu'elles sont supposées révéler.

4.3.1.1 Simulation d'une matrice d'embedding initiale

Afin de synthétiser les embeddings au pas de temps t, nous créons une matrice de telle sorte que chacune de ses cellules soit un nombre aléatoire choisi uniformément dans l'intervalle $[0, 1]$. D'autre part, afin de découvrir l'impact des différents paramètres entrant en jeu sur les différentes mesures d'erreur, nous faisons varier la dimension d'embedding ainsi que le nombre de nœuds ; en d'autres termes, nous faisons varier la taille des matrices synthétiques. Ainsi, à la création d'une matrice d'embedding initiale, d et N sont choisis de manière aléatoire et uniforme respectivement dans les ensembles $[\![2,\ 32]\!]$ et $[\![10,\ 10000]\!]$.

4.3.1.2 Simulation des transformation linéaires

Dépendamment de l'expérimentation menée, différentes transformations (translation, rotation/réflexion, changement d'échelle) sont opérées sur la matrice synthétique du pas de temps t pour en obtenir une autre correspondant aux embeddings au pas de temps $t+1$. L'idée est de simuler une transformation contrôlée par un facteur d'amplitude : plus ce facteur est grand, plus l'amplitude de la transformation est importante.

Les translations appliquées sont opérées dans des directions aléatoires, de sorte que le rapport entre le rayon de la matrice initiale et la norme du vecteur de translation soit égal au facteur de décalage imposé.

Le changement d'échelle est réalisé de telle sorte que le centre de gravité soit préservé et que le rapport entre les rayons des matrices transformée et initiale (rayon tel que défini dans l'équation 4.2) soit égal au facteur d'échelle imposé.

Pour concevoir une matrice de rotation/réflexion, nous tirons profit de la forme canonique que peut prendre une matrice orthogonale (cf. tableau 4.1) et de l'expression de l'erreur de rotation qui en découle (explicitée dans l'équation 4.8). L'idée est que, plus le facteur de rotation est grand, plus les angles de rotation élémentaires sont proches de π. Par ailleurs, un choix aléatoire détermine si une réflexion est également appliquée à la matrice de rotation créée. L'annexe C explique dans le détail la procédure suivie.

4.3.1.3 Modèle du bruit introduit

Afin d'évaluer la robustesse des différentes métriques évaluées vis-à-vis du bruit, nous concevons un modèle temporel de bruit basé sur des marches aléatoires dans l'espace d'embedding. L'idée est de simuler le comportement que l'embedding d'un nœud peut avoir dans l'espace latent au cours du temps, du fait de phénomènes chaotiques indépendants de l'évolution *réelle* du nœud (régie pas ses interactions). Le principe intuitif à satisfaire est le suivant : un embedding change de position dans l'espace latent d'autant plus fortement que le temps avance.

À chaque pas de la marche aléatoire, la matrice d'embedding E est mise à jour selon l'équation suivante :

$$E \leftarrow E + \frac{1}{2\sqrt{d}} \cdot U \tag{4.13}$$

où U est une matrice de bruit de même taille que E, dont les valeurs sont échantillonnées uniformément dans l'intervalle $[-1, 1]$. Cela garantit que les directions des pas de la marche sont aléatoires. Le terme $1/\sqrt{d}$ permet de s'assurer que le facteur de bruit est équivalent indépendamment de la taille de la matrice d'embedding. Le terme $1/2$ est un choix pratique qui permet de vérifier que la taille des pas n'est ni trop petite ni trop grande. En effet, 25 pas de marche aléatoire donnent lieu à un bruit ajouté ayant approximativement le même rayon que l'embedding original :

$$\mathbb{P}(\sum_{k=1}^{25} \frac{u_k}{2} > 1) \simeq \mathbb{P}(\sum_{k=1}^{25} \frac{u_k}{2} < 1) \simeq 50\% \tag{4.14}$$

où chaque u_k représente un nombre aléatoire uniformément tiré dans l'intervalle [-1, 1]. En conséquence, nous fixons le nombre de 25 pas comme étant une unité pour le facteur

de bruit. Autrement dit, un facteur de bruit de 1 correspond en moyenne à un bruit de même rayon que l'embedding original. En définitive, le résultat de l'introduction d'un bruit de facteur b à une matrice d'embedding E est :

$$E + \frac{r}{2\sqrt{d}} \cdot \sum_{k=1}^{\lceil 25 \cdot b \rceil} U_k \tag{4.15}$$

où $\lceil x \rceil$ représente la partie entière supérieure de x et r représente le rayon de la matrice d'embedding E tel que défini dans l'équation 4.2.

4.3.2 Validation des erreurs conçues

4.3.2.1 Validation de l'erreur en échelle

Dans le cas où les embeddings de deux pas de temps sont identiques, à la différence près que l'un est une version agrandie (i.e. dilatée) ou réduite (i.e. contractée) de l'autre, nous pouvons nous attendre à ce que l'erreur d'échelle reflète l'ampleur de la différence d'échelle. Afin de le vérifier, nous créons des matrices d'embedding synthétiques en leur appliquant un changement d'échelle compris dans l'intervalle $[10^{-3}, 10^{3}]$, tel que décrit dans 4.3.1.2. La figure 4.3 présente les résultats de cette expérimentation.

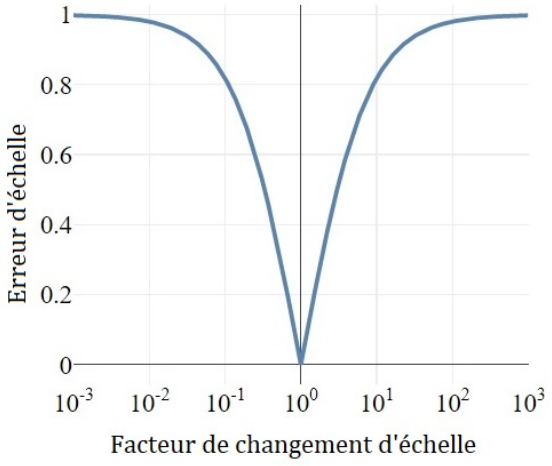


FIGURE 4.3: Corrélation entre un changement d'échelle imposé et l'erreur en échelle associée.

Nous observons que l'erreur d'échelle est de 0 lorsque les deux embeddings ont la même échelle (c'est-à-dire lorsque le facteur de changement d'échelle est à 1 et que, par conséquent, les rayons des embeddings sont égaux). L'erreur tend vers 1 d'autant plus que le facteur de changement d'échelle s'éloigne *logarithmiquement* de 1, dans un

sens ou dans l'autre (dilatation ou contraction). Par ailleurs, au vu de la régularité de la courbe, l'erreur en échelle est visiblement indépendante de la dimension d'embedding ou du nombre de nœuds.

4.3.2.2 Validation de l'erreur en translation

D'une manière similaire, la situation où une matrice d'embedding est translatée par rapport à celle qui la suit dans le temps devrait produire une erreur de translation corrélée à l'écart entre les deux matrices. Afin de le vérifier, nous créons une paire de matrices d'embedding en appliquant une translation de l'une par rapport à l'autre (cf. procédure 4.3.1.2) avec un facteur de décalage compris dans $[10^{-3}, 10^{3}]$. La figure 4.4 montre les résultats de cette expérimentation.

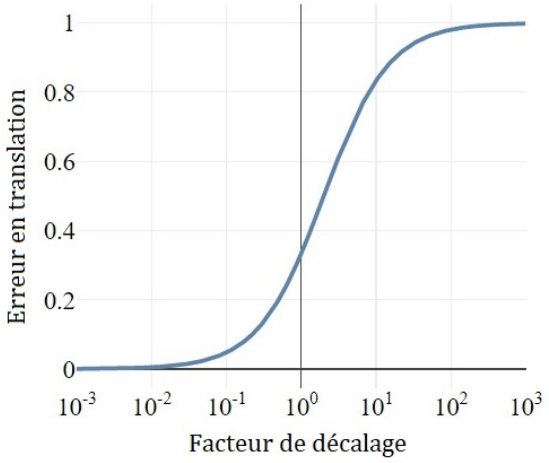


FIGURE 4.4: Corrélation entre une translation contrôlée par un facteur de décalage et l'erreur en translation associée.

Nous constatons que l'erreur en translation est nulle lorsque le facteur de décalage tend vers 0 (i.e. dans le cas où il n'y a pas de translation), que l'erreur augmente avec le facteur de décalage et qu'elle est indépendante de la taille de la matrice d'embedding (dimension et nombre de nœuds).

4.3.2.3 Combinaison de translation et de changement d'échelle

Ensuite, nous étudions le comportement des mesures d'erreur en translation et en échelle pour des transformations combinant ces deux effets. Des embeddings initiaux sont créés, puis un décalage suivi d'un changement d'échelle sont appliqués. Dans le cas de l'erreur d'échelle, nous nous attendons à ce qu'elle soit indépendante de la translation dans la mesure où elle concerne les rayons des embeddings et non la position des embeddings

dans l'espace latent. Cela est confirmé par la figure 4.5a : l'erreur en échelle augmente si et seulement si le facteur d'échelle s'éloigne de 1.

Cependant, l'erreur en translation ne dépend pas exclusivement de la distance entre deux embeddings dans l'espace latent. En effet, cette mesure dépend également des rayons des embeddings en question comme exprimé dans l'équation 4.2. La figure 4.5b confirme ce comportement et montre que l'erreur de translation est proche de 0 lorsque le facteur de décalage est proche de 0. D'autre part, cette erreur augmente avec le facteur de décalage et, dans le même temps, diminue lorsque le facteur de changement d'échelle est important.

Par ailleurs et plus généralement, la régularité des motifs observés dans la figure 4.5 confirme visuellement la conclusion précédente selon laquelle les deux mesures sont indépendantes de la taille des embeddings. Elles peuvent donc être généralisées à des graphes comportant des nombres de nœuds quelconques et à différentes dimensions d'embedding.

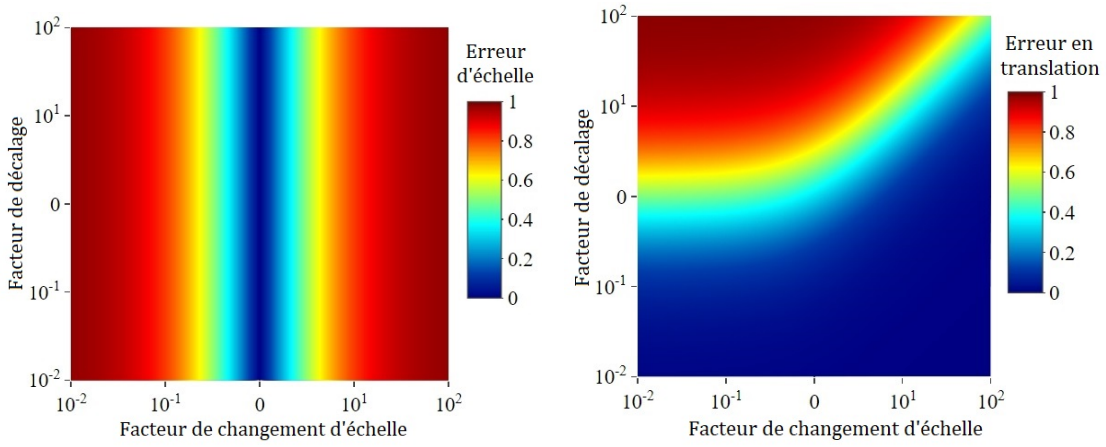


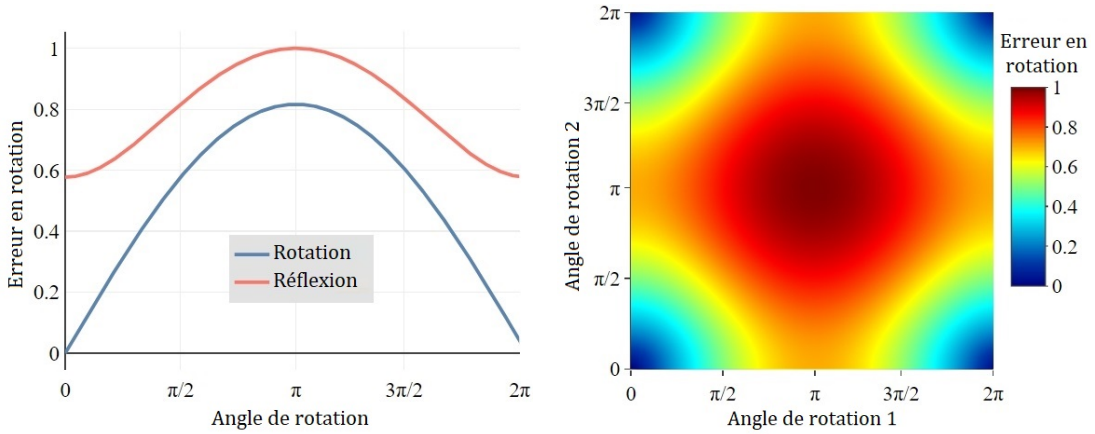
FIGURE 4.5: Erreurs en échelle et en translation en fonction de la combinaison d'un changement d'échelle et d'un décalage. (a) Facteurs de changement d'échelle et de décalage, et erreurs en échelle correspondantes. (b) Facteurs de changement d'échelle et de décalage, et erreur en translation correspondante.

4.3.2.4 Validation de l'erreur en rotation

Comme explicité dans le tableau 4.1, toute matrice de rotation rapprochant les embeddings de deux pas de temps consécutifs peut être représentée sous une forme canonique avec au plus $d/2$ rotations et éventuellement une réflexion. L'équation 4.8 suggère que l'erreur de rotation est directement liée à la magnitude des angles de rotation élémentaires, présents dans la forme canonique de la matrice orthogonale. Afin de vérifier

cela, nous créons un grand nombre de matrices de rotation selon la procédure décrite dans 4.3.1.2 en faisant varier le facteur de rotation.

La figure 4.6a illustre l'erreur de rotation obtenue pour ces différentes matrices en dimension $d = 3$ en distinguant les cas de présence / absence de réflexion. Nous observons que l'erreur est maximale lorsque l'angle de rotation est égal à π. Par ailleurs, l'erreur de rotation est plus importante lorsqu'une réflexion est appliquée. Nous vérifions également la corrélation entre angles de rotation élémentaires et erreur en rotation en dimension $d = 4$. Nous nous restreignons au cas d'une matrice orthogonale sans réflexion, composée donc de 2 rotations élémentaires. Comme le montre la figure 4.6b, l'erreur de rotation est maximale lorsque les deux angles sont égaux à π, et minimale lorsque les deux angles sont nuls.



(a) (b)

FIGURE 4.6: Erreur de rotation en fonction de l'amplitude de la rotation / réflexion. (a) Pour $d = 3$, l'angle de rotation couplé ou non à une réflexion et l'erreur de rotation associée. (b) Pour $d = 4$, les deux angles de rotation et l'erreur de rotation correspondante.

Globalement, la figure 4.6 montre la relation claire entre, d'un coté, les angles de rotation et une éventuelle réflexion, et de l'autre, l'erreur de rotation, confirmant visuellement les attentes intuitives et les fondements mathématiques de cette mesure.

4.3.2.5 Robustesse des mesures d'erreur vis-à-vis du bruit

Nous étudions le comportement des erreurs en translation, en rotation/réflexion et en échelle à l'issue de leurs transformations respectives et de l'ajout du bruit.

La figure 4.7a montre que, en plus de l'amplitude du changement d'échelle appliqué, l'erreur en échelle est également impactée par l'amplitude du bruit introduit. Plus précisément, dans le cas d'une contraction des embeddings (i.e. facteur de changement d'échelle < 1) combinée à un bruit additionnel de forte amplitude, l'erreur en échelle

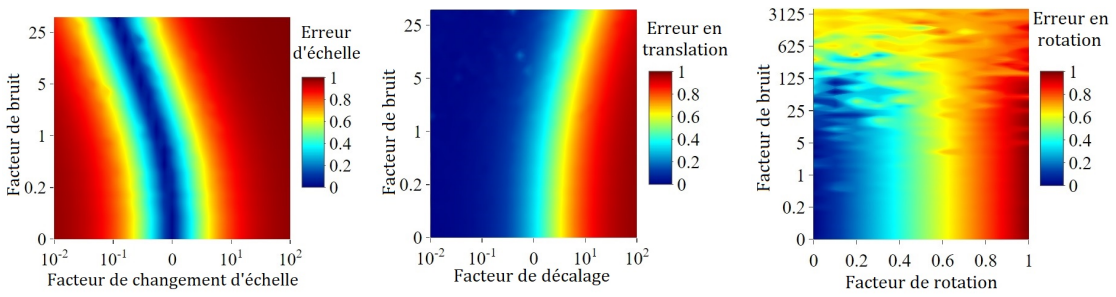
est plus faible. En effet, au fur et à mesure que l'amplitude du bruit augmente, le déplacement des embeddings par rapport à leurs positions de départ est plus important, faisant croître le rayon moyen des embeddings, et donc leur échelle. Une telle augmentation compense la contraction, expliquant ainsi le comportement observé dans la figure.

Cet effet de compensation est également à l'origine du comportement observé dans la figure 4.7b illustrant le lien entre l'amplitude du bruit introduit et l'erreur en translation. L'augmentation du rayon impacte le terme de normalisation qui est utilisé dans le calcul de l'erreur en translation (i.e. $r^t + r^{t+1}$ dans l'équation 4.2). Par conséquent, dans le cas d'un bruit de forte amplitude, l'erreur de translation est réduite. Dans l'ensemble, ce comportement résultant de l'augmentation du rayon est dû à la manière dont est modélisé le bruit dans les expérimentations menées plutôt qu'aux propriétés des mesures proposées et n'est visible de manière flagrante que dans le cas d'un bruit additionnel non réaliste car trop important.

L'introduction d'un bruit de magnitude importante transforme toute matrice d'embedding en une matrice à caractère aléatoire. Comme le montre la figure 4.7c, dans le régime caractérisé par de grands facteurs de bruit, l'erreur en rotation converge vers une grande valeur même si les facteurs de rotation initiaux sont différents. À noter que cette valeur n'est pas la limite supérieure que peut atteindre l'erreur de rotation. En effet, sa valeur limite est obtenue pour une matrice de rotation/réflexion particulière, à savoir l'inverse de la matrice identité (des angles de rotation élémentaires à π en plus d'une réflexion quand d est impair). Dans le régime caractérisé par des niveaux de bruit modérés et plus réalistes, l'erreur de rotation est capable de faire la distinction entre le bruit et la structure originale et produit des valeurs directement liées au facteur de rotation imposé. Par exemple, sachant qu'un facteur de bruit égal à 1 produit un bruit aléatoire additionnel ayant des valeurs du même ordre de grandeur que la matrice d'embedding, le résultat démontre la capacité de l'erreur de rotation à capturer les différences de rotation même sous des quantités considérables de bruit.

4.3.2.6 Validation de l'erreur de stabilité

Contrairement aux trois mesures précédentes, l'erreur de stabilité évalue les changements structurels relatifs, ou en d'autres termes, les mouvements des nœuds par rapport aux autres et des communautés par rapport aux autres. Cette erreur est donc mesurée à l'issue du réalignement et de la normalisation (mise à échelle commune) d'une paire d'embeddings consécutifs. Par conséquent, une évolution pouvant être entièrement expliquée par une combinaison des transformation précédentes (translation,



(a) (b) (c)

FIGURE 4.7: Effet combiné du bruit et du changement d'échelle, de la translation, ou de la rotation/réflexion sur les mesures d'erreurs associées : (a) changement d'échelle, (b) translation, (c) rotation / réflexion.

rotation/réflexion, changement d'échelle) devrait produire une erreur de stabilité nulle. Afin de valider la pertinence de la mesure de stabilité proposée, nous introduisons des changements structurels sur les embeddings synthétiques en leur additionnant du bruit, tel que modélisé en section 4.3.1.3 : l'ajout de bruit modifie de manière aléatoire les distances entre les embeddings des nœuds.

D'abord, nous vérifions que l'erreur de stabilité est robuste face à différentes tailles d'embedding. Pour ce faire, nous créons des matrices initiales d'embedding de différentes tailles (différents N et d), puis nous modifions leur structure en introduisant du bruit avec un facteur fixé à 1 (embeddings en $t+1$). Ensuite, nous observons l'erreur de stabilité entre les paires d'embeddings, tel que montré dans la figure 4.8. À l'exception du régime caractérisé par de très faibles valeurs de N/d, l'erreur de stabilité est stable vis-à-vis du nombre de nœuds et de la dimension d'embedding. Le régime à faible valeur de N/d n'est pas représentatif de graphes issus de cas d'usage réels dans la mesure où l'apprentissage de représentation vise à apprendre des embeddings satisfaisant $d << N$. De plus, dès que le nombre de nœuds augmente modestement pour dépasser 100 (comme c'est le cas dans des scénarii réalistes), nous atteignons le régime général où la mesure de stabilité est robuste par rapport à la taille de la matrice d'embedding.

Ensuite, nous nous intéressons au lien entre l'erreur de stabilité et le facteur de bruit (cf. figure 4.9). Au premier abord, nous remarquons une relation directe entre l'amplitude du changement dans la structure d'embedding (i.e. le facteur de bruit) et la mesure de l'erreur de stabilité. Par ailleurs, d'autres constatations peuvent êtres effectuées. D'un coté, l'absence de fluctuations importantes confirme la robustesse de l'erreur de stabilité vis-à-vis de la taille de la matrice d'embedding. D'autre part, nous observons que l'erreur de stabilité n'atteint pas sa valeur maximale (i.e. 1) et ne dépasse pas 0.75. En réalité, la limite supérieure de l'erreur de stabilité n'est atteinte que dans des cas très extrêmes qu'il est pratiquement impossible de reproduire avec des embeddings et du bruit aléatoires (cf. annexe D).

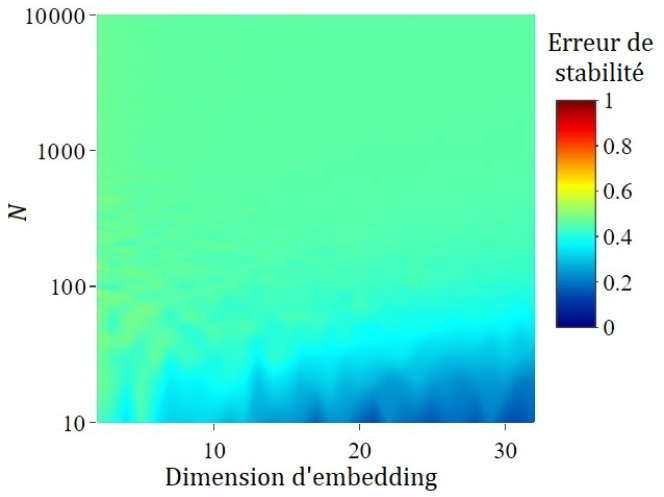


FIGURE 4.8: Robustesse de l'erreur de stabilité vis-à-vis de la dimension d'embedding et du nombre de nœuds.

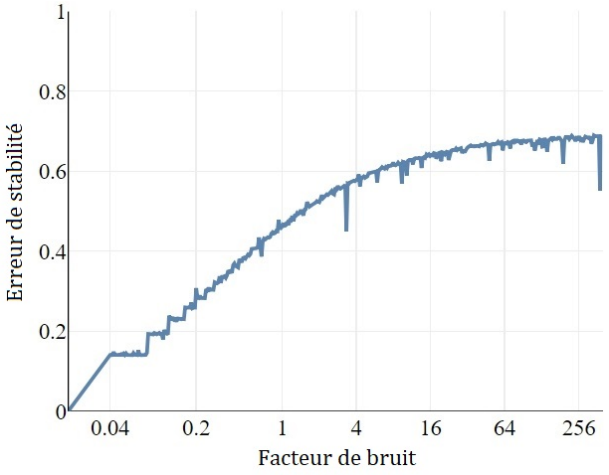
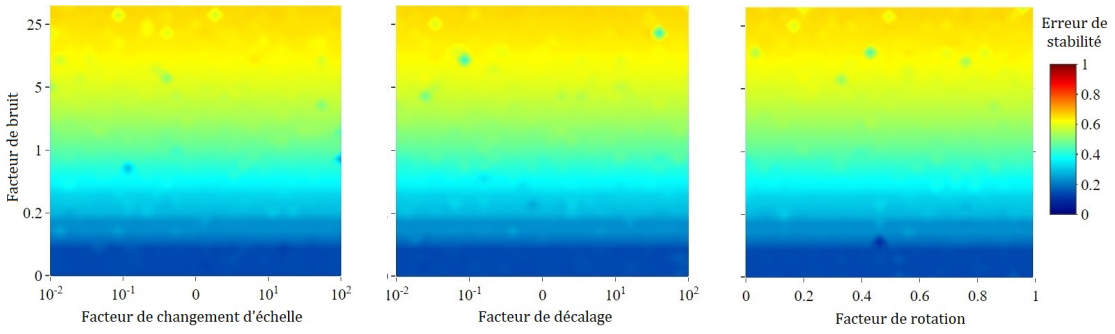


FIGURE 4.9: Relation entre l'amplitude du bruit synthétique introduit et l'erreur de stabilité.

En présence à la fois de changements structurels obtenus par l'introduction de bruit, et d'autres transformations linéaires ne modifiant pas la structure relative (désalignement et changement d'échelle), nous pouvons nous attendre à ce que l'erreur de stabilité soit liée uniquement à l'amplitude du bruit et indépendante des transformations linéaires opérées. Afin de vérifier cela, nous créons des matrices initiales (embeddings en t) auxquelles nous appliquons du bruit et une des transformations linéaires (embeddings en $t+1$).

La figure 4.10 présente les résultats de ces expérimentations pour chacune des transformations considérées (translation, rotation / réflexion et changement d'échelle). Nous constatons, comme anticipé, que l'erreur de stabilité varie uniquement en fonction de

l'amplitude des changements introduits dans la structure relative du graphe (i.e. bruit) et reste indépendante vis-à-vis des autres transformations.



(a) (b) (c)

FIGURE 4.10: Effet combiné du bruit et des transformations linéaires sur l'erreur de stabilité. (a) Facteurs de changement d'échelle et de bruit, et erreur de stabilité correspondante. (b) Facteurs de translation et de bruit, et erreur de stabilité correspondante. (c) Facteurs de rotation et de bruit, et erreur de stabilité correspondante.

Il convient de rappeler que l'alignement est pertinent dans les cas où il est possible de trouver un modèle structurel de correspondance entre deux embeddings. Une erreur de stabilité importante indique que deux embeddings sont structurellement si différents qu'il n'est pas utile d'étudier l'alignement et d'interpréter les résultats des autres mesures d'erreur. Par conséquent, le comportement des erreurs en échelle, en translation et en rotation n'est pas pertinent, significatif ou utile en pratique lorsque l'amplitude du bruit additionnel est trop importante.

En résumé, les mesures d'erreurs proposées pour l'alignement, le changement d'échelle et la stabilité sont capables de saisir l'existence et l'amplitude des différences introduites par les changements ou transformations structurels respectifs. Elles sont robustes vis-à-vis de la dimension d'embedding et du nombre de nœuds. Par ailleurs, chacune d'entre elles reflète directement et — presque — uniquement les changements qu'elle est censée mesurer. L'ensemble de ces quatre mesures peut donc être utilisé sur des graphes issus de cas réels pour étudier l'existence et le degré de stabilité et d'alignement ainsi que pour réaligner des embeddings temporels.

4.4 Expérimentations sur des données réelles

Nous réalisons des expérimentations sur des graphes temporels issus de 7 jeux de données du monde réél à l'aide de 13 méthodes d'embedding de graphe, statiques et temporelles. D'abord, nous mesurons les erreurs d'alignement et de stabilité de ces embeddings.

4.4.1 Jeux de données

En utilisant les dépôts de données [78, 79] ainsi que d'autres ressources, nous avons identifié plusieurs graphes temporels présentant des métadonnées (fixes ou dynamiques) sur l'appartenance des nœuds à des communautés. Pour chacun des jeux de données, les prétraitements appropriés ont été effectués, notamment afin de s'assurer que l'ensemble des nœuds est fixe sur chacun des pas de temps. Un récapitulatif des jeux de données considérés est présenté dans le tableau 4.2.

- Le graphe **Dutch-School** [80] concerne le réseau d'amitié entre les élèves d'un établissement secondaire aux Pays-Bas. Les pas de temps correspondent aux 4 trimestres de l'année 2003/2004. Les labels fixes des nœuds sont le sexe, l'âge, l'ethnie et la religion. Les nœuds ont également un label dynamique correspondant à l'existence d'incidents de comportement (délinquance) durant le trimestre.

- Le graphe pondéré **Freshmen** [81] désigne le réseau d'amitié entre les étudiants de première année du département de sociologie d'une université aux Pays-Bas, pendant l'année scolaire 1998/1999. Le poids d'une arête correspond à la *force* de l'amitié liant 2 étudiants. Les labels des nœuds fixes sont le sexe, le programme et si l'étudiant est fumeur.

- Le graphe **Sp-Hospital** [82] retrace les contacts directs entre les patients et le personnel de santé dans un hôpital en France sur 5 jours en 2010. Le poids de chaque arête correspond à la durée de l'interaction entre 2 personnes sur une journée. Les nœuds ont pour labels fixes leur statut (patient ou poste de travail le cas échéant).

- Le graphe **Sp-Primary-School** [83] montre les contacts directs sur 2 jours consécutifs dans une école primaire en France en 2009. Les poids des arêtes représentent la durée d'une interaction sur une journée. Les nœuds ont des labels fixes : le sexe et la classe.

- Le graphe **Sp-Highschool** [84] présente les contacts directs entre les élèves d'un lycée en France sur 5 jours en 2013. Les arêtes sont pondérées (durée d'interaction entre deux lycéens), et les labels des nœuds (classe et sexe) sont fixes.

- Le graphe **Yahoo** [85] retrace les communications entre un échantillon d'utilisateurs de Yahoo Messenger sur 4 semaines. L'ensemble des données est fourni par le programme Yahoo Webscope. Les données sont traitées de telle sorte que chaque poids d'arête représente le nombre de jours sur une semaine où deux internautes ont échangé des messages. Chaque nœud possède deux labels

dynamiques correspondant à une localisation précise et une autre plus grossière, obtenues à partir des codes postaux d'envoi des messages.

- Enfin, nous extrayons un échantillon de 10k nœuds du graphe **AMiner** [43] (cf. section 2.2.4.2), répartis sur 4 pas de temps de 3 ans chacun.

Nom	T	N	Pondéré	Labels fixes	Labels dynamiques
Dutch-School	4	25	Non	Sexe, age, ethnie, religion	Délinquance
FreshMen	4	30	Oui	Sexe, age, programme, fumeur	
Sp-Hospital	2	51	Oui	Statut	
Sp-Primary-School	2	232	Oui	Classe, sexe	
Sp-High-School	5	244	Oui	Classe, sexe	
AMiner	4	10565	Oui		Domaine de recherche
Yahoo	4	46049	Oui		Localisations

TABLEAU 4.2: Jeux de données. T et N désignent respectivement le nombre de pas de temps et le nombre de nœuds.

4.4.2 Méthodes d'embedding

Nous considérons différentes méthodes d'embedding de graphes, statiques et temporelles. En ce qui concerne les méthodes statiques, nous utilisons des méthodes à base de marches aléatoires, à savoir N2V[6], DW[5] ; des méthodes de factorisation de matrices (GF [41], LAP[2], HOPE[4]) ; une méthode basée sur la réduction de dimension (LLE[86]) ; une méthode d'embedding de *grands graphes* (LINE[87]) et un autoencodeur SDNE[9]. Nous appliquons ces méthodes statiques à chaque pas de temps de manière indépendante.

Pour les méthodes temporelles, nous employons DT[15], GloDyNE[16], et TN2V[19]. De plus, nous considérons des versions temporalisées de DW (DW_FT) et N2V (N2V_FT) (*fixed temporalization* telle que décrite dans 3.4.2, avec des poids temporels fixés au poids maximal rencontré sur toutes les arêtes du graphe temporel).

À noter que, pour les méthodes d'embedding nécessitant des hyperparamètres, nous choisissons les valeurs par défaut populaires : nous faisons ce choix dans la mesure où l'objectif poursuivi dans ce travail sur l'alignement ne réside pas dans la comparaison et le tuning des méthodes d'embedding.

4.4.3 Expérimentations

4.4.3.1 Mesures d'alignement et de stabilité

La figure 4.11 montre les erreurs en translation, en rotation/réflexion, en échelle et en stabilité produites par chaque méthode sur toutes les paires de pas de temps consécutifs des différents jeux de données.

Pour la rotation, nous remarquons que les méthodes statiques engendrent des erreurs plus importantes que les méthodes temporelles. Cependant, les erreurs de rotation produites par les méthodes temporelles ne sont pas négligeables, à l'exception de la méthode GloDyNE.

Dans le cas de l'erreur en translation, les méthodes basées sur des marches aléatoires produisent des erreurs plus importantes, de même que SDNE et LINE. D'autre part, à l'exception des versions temporalisées, les méthodes temporelles produisent généralement des erreurs de translation beaucoup plus faibles que les méthodes statiques, LLE mis à part.

En ce qui concerne l'erreur en échelle, nous observons que la plupart des méthodes produisent généralement de faibles erreurs ; les méthodes temporalisées, et en particulier DW_FT, produisent quant à elles des erreurs relativement plus importantes.

Enfin, pour la stabilité, comme on pouvait s'y attendre, il n'existe pas de différences significatives entre les méthodes statiques et temporelles. Cela peut signifier que les défauts d'alignement constituent une des principales différences entre les embeddings produits par des méthodes statiques comparées aux temporelles. Toutefois, les méthodes temporelles produisent généralement des erreurs de stabilité légèrement plus faibles, probablement en raison du transfert d'informations entre les pas de temps lors du processus de construction des embeddings. Cela a pour effet de lisser temporellement les embeddings en les rapprochant entre pas de temps consécutifs.

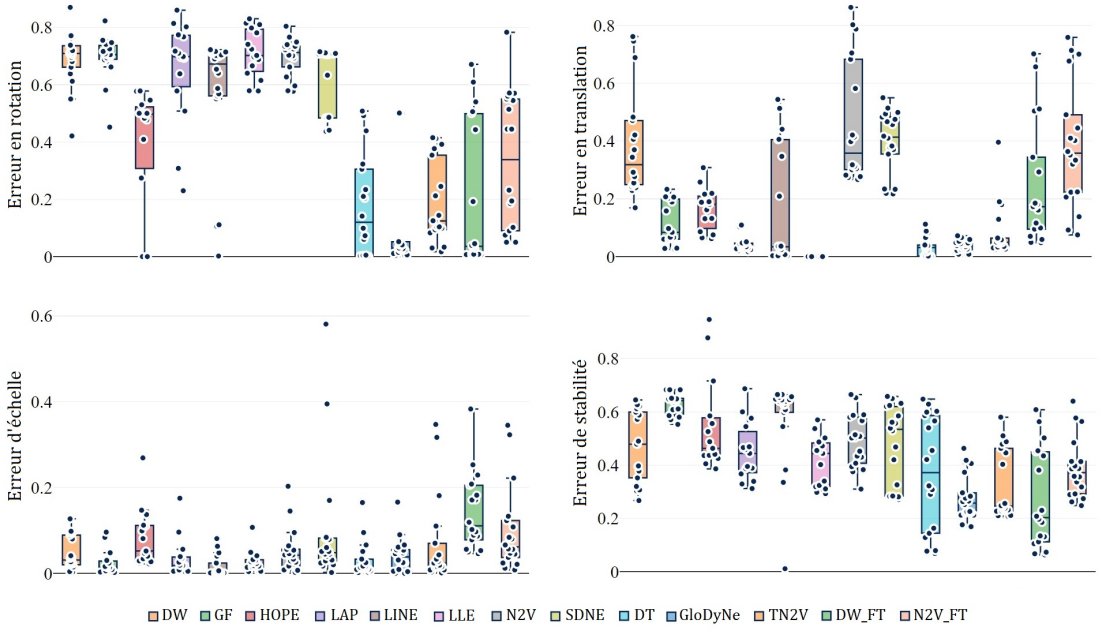


FIGURE 4.11: Erreurs produites par les méthodes d'embedding. Dans un souci de lisibilité, l'ordre des *boîtes à moustache* correspond à celui de la légende et les méthodes sont classées en fonction de leur caractère statique ou temporel.

Dans l'ensemble, cette analyse empirique montre que, globalement, les méthodes statiques produisent de fortes erreurs de rotation et de translation. Ce résultat n'est pas surprenant dans la mesure où les embeddings sont construits sur les différents pas de temps de manière indépendante. Cependant, les erreurs d'alignement (i.e. les erreurs de rotation et de translation) produites par les méthodes temporelles ne sont généralement pas négligeables. Les embeddings en sortie de ces méthodes sont donc perfectibles à cet égard.

4.4.3.2 Impact du réalignement sur les performances d'inférence

Dans cette expérimentation, nous nous intéressons à la potentielle amélioration des scores de classification de nœuds à l'issue du réalignement des embeddings. De manière intuitive, en utilisant les versions originales et réalignées des embeddings comme éléments d'entrée, nous construisons des modèles de classification de nœuds et nous observons leurs performances sur les embeddings du pas de temps suivant. De cette façon, nous pouvons nous rendre compte de l'apport du réalignement sur les performances. Plus concrètement nous menons deux opérations en parallèle.

- D'abord, pour chaque label de jeu de données, nous apprenons à un classifieur (SVM) de prédire l'appartenance d'un nœud en fonction de son embedding au pas de temps t. Ensuite, nous reprenons le classifieur construit pour prédire les labels (labels au pas de temps $t + 1$ quand ceux ci sont dynamiques) sur la base des embeddings au pas de temps $t + 1$.
- D'un autre coté, nous réalignons toutes les paires d'embeddings consécutifs (en translation et en rotation/réflexion), puis nous réexécutons la procédure précédente.

À l'issue de ces deux opérations, nous obtenons, pour chaque couple {paire d'embeddings consécutifs, label}, deux scores correspondant aux versions originales et alignées des embeddings.

À ce stade, il convient de noter que tous les labels ne peuvent pas être prédits par les données du graphe en entrée. Par exemple, dans un contexte idéal où la mixité des genres serait parfaite, les interactions d'un individu ne sauraient être conditionnées par son sexe. Il serait donc judicieux de ne pas considérer ces cas de figure, où les habitudes d'interactions ne sont pas le reflet des labels des nœuds. Pour ce faire, nous considérons uniquement les cas présentant des scores d'inférence meilleurs que celles d'un *dummy classifier* (i.e. un classifieur ayant un niveau de performance relativement faible). Avec a et b représentant respectivement les scores du modèle appris et du dummy classifier,

nous filtrons les cas où la condition suivante n'est pas satisfaite : $a > 0.25(1-b)+b$. En d'autres termes, le modèle doit capturer au moins 25% des informations que le dummy classifier n'a pas su exploiter. La figure 4.12 montre la différence absolue des scores de précision de classification avant et après le réalignement des embeddings, en séparant les méthodes statiques des temporelles.

La figure 4.12a présente les cas qui sont exclus. D'abord, il n'y a pas d'amélioration significative de la précision de classification dans les cas écartés et les valeurs sont autour de 0 selon une distribution normale. Ceci est conforme aux attentes selon lesquelles : si le modèle n'est pas meilleur pour la tâche de classification qu'un dummy classifier, alors le label est faiblement (voire nullement) corrélé aux interactions ; il ne devrait donc pas y avoir d'améliorations après réalignement.

La figure 4.12b présente les cas filtrés (cas meilleurs qu'un dummy classifier). Nous observons que, dans la plupart des cas, il y a une amélioration de la précision de classification après alignement. Les détériorations de performances suite au réalignement sont rares et constituent des exceptions vraisemblablement imputables à certains effets aléatoires.

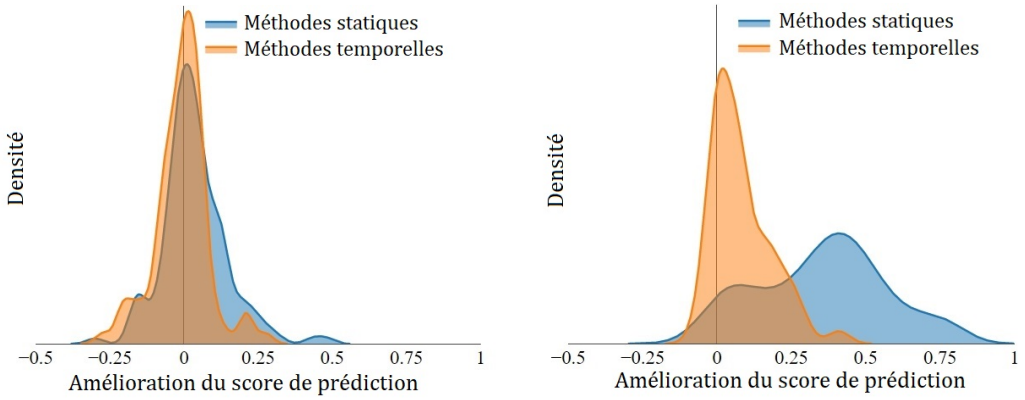


FIGURE 4.12: Distribution de l'amélioration du score de prédiction après réalignement. (a) Cas exclus sur la base de la comparaison avec un dummy classifier. (b) Cas retenus sur la même base.

D'autre part, le tableau 4.3 fournit un aperçu plus détaillé des améliorations de la précision de classification sur les trois plus grands jeux de données utilisés dans cette étude. Dans les colonnes, nous indiquons la tâche de classification (ensemble de données, variable cible et pas de temps consécutifs) ainsi que la nature des embeddings en entrée : (O) pour les originaux et (A) pour les réalignés. Sur les lignes, nous énumérons séparément les méthodes d'embedding statiques et temporelles. Par ailleurs, nous indiquons les améliorations moyennes (O-A) par rapport aux méthodes (colonne la plus à droite) ou par rapport à la tâche de classification (dans les lignes du bas, après les

méthodes d'embedding statiques et temporelles). Globalement, nous confirmons que les améliorations obtenues pour les méthodes d'embedding statiques sont plus importantes que pour les temporelles et que cela est dû au fait que les méthodes statiques produisent des erreurs d'alignement plus fortes et donc davantage de possibilités d'amélioration. En particulier, lorsque nous réalignons les embeddings en sortie des méthodes statiques, nous observons une nette amélioration de la précision de la classification pouvant atteindre 90%. Dans le cas des embeddings de méthodes temporelles, nous notons toujours des gains de performance significatifs et réguliers dans la précision de la classification, la différence atteignant jusqu'à 40%. À noter que dans des cas extrêmes, lorsque les embeddings sont très désalignés, la précision de la classification est à 1%, mais peut atteindre plus de 70% une fois que les mêmes embeddings sont réalignés. En somme, nous pouvons conclure que les méthodes d'embedding, statiques mais aussi temporelles, tirent profit du réalignement.

	Sp-Highschool								AMiner						Yahoo												O-A
	classe								domaine recherche						local. grossière						local. précise						
	0-1		1-2		2-3		3-4		0-1		1-2		2-3		0-1		1-2		2-3		0-1		1-2		2-3		
	O	A	O	A	O	A	O	A	O	A	O	A	O	A	O	A	O	A	O	A	O	A	O	A	O	A	
DW	.01	.55	.13	.66	.12	.74	.09	.90	.34	.68	.24	.69	.35	.69	.17	.63	.17	.64	.18	.64	.03	.43	.03	.44	.04	.44	.48
GF	.12	.16	.09	.11	.12	.13	.11	.13	.26	.58	.27	.56	.26	.60													.15
HOPE	.12	.11	.14	.13	.13	.15	.09	.15	.36	.36	.26	.27	.13	.21													.02
LAP	.14	.80	.24	.65	.01	.73	.07	.80	.21	.46	.26	.37	.06	.26													.44
LINE	.11	.14	.22	.17	.14	.12	.16	.13	.33	.51	.27	.48	.15	.52	.16	.59	.17	.62	.21	.59	.03	.51	.03	.54	.05	.50	.26
LLE	.23	.78	.01	.65	.24	.74	.02	.81	.31	.51	.29	.43	.25	.39													.42
N2V	.08	.86	.07	.52	.02	.74	.01	.60	.24	.70	.25	.70	.22	.71	.16	.69	.21	.70	.18	.70	.02	.56	.02	.59	.03	.58	.55
SDNE	.12	.46	.18	.55	.08	.50	.10	.51	.33	.38	.19	.31	.16	.28	.20	.31	.21	.35	.19	.39	.05	.16	.04	.21	.04	.26	.21
O-A	.37		.30		.37		.42		.23		.22		.26		.38		.39		.39		.38		.41		.41		

DT	.18	.24	.14	.14	.17	.21	.19	.19	.52	.55	.49	.53	.53	.57	.68	.68	.75	.75	.72	.72	.61	.61	.69	.69	.65	.65	.02
GloDyNE	.96	.96	.92	.91	.92	.91	.93	.95	.65	.70	.66	.69	.62	.70	.62	.71	.63	.74	.63	.73	.42	.60	.45	.65	.45	.63	.08
TN2V	.89	.93	.56	.75	.77	.85	.90	.87	.49	.58	.39	.54	.34	.44	.33	.36	.33	.37	.33	.37	.16	.20	.17	.21	.17	.20	.06
DW_FT	.59	.83	.48	.89	.52	.72	.52	.31	.66	.70	.64	.69	.44	.69	.57	.71	.57	.73	.48	.66	.34	.58	.37	.62	.27	.51	.17
N2V_FT	.99	.99	.99	.99	.99	.99	.99	.99	.64	.66	.66	.66	.45	.70	.63	.70	.66	.73	.55	.71	.46	.57	.50	.62	.32	.60	.08
O-A	.07		.12		.06		-.04		.04		.05		.14		.07		.08		.09		.11		.12		.15		

Tableau 4.3: Précision de la classification avant et après réalignement.

4.4.3.3 Corrélations entre erreurs d'alignement et performances d'inférence

À la lueur de ces résultats (i.e. le réalignement améliore la précision de tâches d'inférence), nous examinons la relation entre les erreurs en alignement et l'ampleur de l'amélioration de la précision de la classification. À cette fin, nous calculons les corrélations partielles de Spearman entre l'amélioration absolue et les erreurs en alignement. La corrélation partielle entre l'amélioration de la précision et l'erreur en translation est de 0.42 (p-value $< 10^{-6}$). Pour l'erreur en rotation/réflexion, elle est de 0.65 (p-value $< 10^{-6}$). Le fait que les corrélations soient positives et les p-value très faibles

établit la relation entre les erreurs en alignement et la quantité d'amélioration apportée en réalignant les embeddings.

Ensuite, nous nous intéressons à la relation entre un désalignement contrôlé introduit et la précision de la classification de nœuds. Pour chaque paire de pas de temps consécutifs, nous alignons les embeddings correspondants. Ensuite, similairement à la procédure utilisée en 4.3.1.2, nous introduisons deux désalignements (en translation et en rotation/réflexion), contrôlés chacun par un facteur. Enfin, sur la base de ces paires d'embeddings, nous calculons les scores de prédiction pour la classification de nœuds, en suivant la procédure décrite en 4.4.3.2 et en utilisant plusieurs modèles de classification (SVM, k-NN et régression logistique).

La figure 4.13 expose les résultats de cette expérimentation sur les embeddings de N2V et pour le jeu de données Sp-Highschool (dans un souci de lisibilité de la représentation des résultats, nous considérons uniquement un seul jeu de données et une seule méthode d'embedding). Globalement, nous observons que plus les rotations et translations appliquées sont importantes en amplitude, plus la précision de la classification est faible. Nous remarquons également que la meilleure précision est obtenue lorsque la rotation et la translation appliquées sont faibles, voire nulles. Cela signifie que, au-delà de l'amélioration des performances, la manière dont l'alignement est réalisé est idéale pour la tâche de classification des nœuds de Sp-Highschool. Un autre point intéressant réside dans la similarité entre les résultats des différents classifieurs considérés. Ainsi, ces résultats ne dépendent pas du modèle de classification utilisé. Par ailleurs, cela justifie a posteriori le choix préalable d'employer uniquement SVM comme modèle de classification.

4.4.3.4 Précision de la prédiction par rapport à l'erreur en stabilité.

Enfin, nous analysons la relation entre la stabilité des embeddings et la précision de la prédiction pour la classification de nœuds. Par définition, l'erreur de stabilité est directement liée aux changements entre les embeddings consécutifs des nœuds qui ne sont pas attribuables à un défaut d'alignement. Cela signifie que, lorsque l'erreur en stabilité est importante, les frontières discriminant les différents emplacements des labels des nœuds dans l'espace latent d'embedding changent de manière significative dans le temps. Par conséquent, le score de la prédiction du label d'un nœud sur la base de son embedding à un pas de temps t, en utilisant un modèle entraîné sur les données du pas de temps $t-1$ (cf. procédure 4.4.3.2), devrait être corrélé à l'erreur en stabilité.

La figure 4.14 montre les résultats de cette expérimentation sur les ensembles de données comportant plus de 50 nœuds. Globalement, nous constatons que, plus l'erreur de

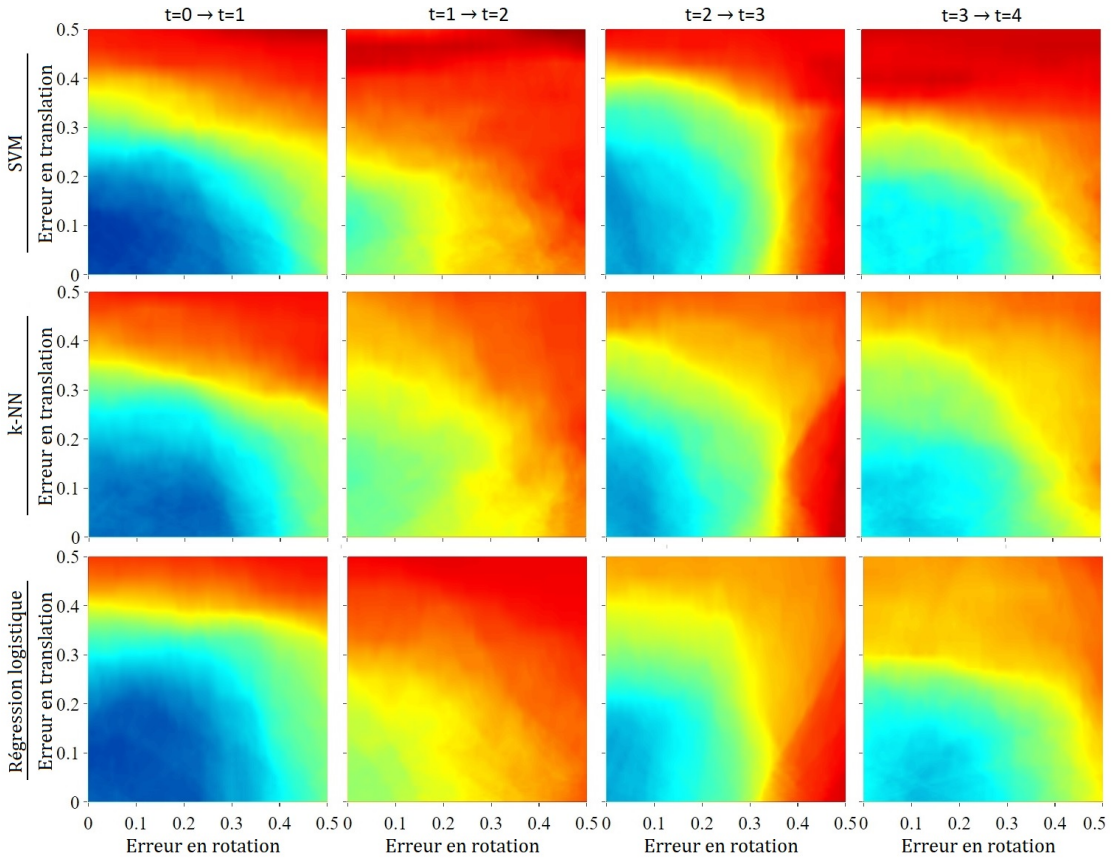


FIGURE 4.13: Effet du désalignement synthétique sur la précision de la prédiction pour Sp-Highschool, sur les embeddings de N2V. Chaque ligne concerne un classifieur différent et chaque colonne présente une paire de pas de temps consécutifs.

stabilité est élevée, moins la tâche de prédiction est précise, confirmant ainsi l'intuition de départ. Cependant, nous observons également que les points sont plus ou moins dispersés autour de la ligne de régression linéaire. Une possible explication réside dans les disparités existant entre les différents jeux de données, ou entre les différents algorithmes d'embedding. Par exemple, il existe des cas où le classifieur est relativement performant quand bien même l'erreur de stabilité est relativement élevée. Cela est peut-être dû au fait que les clusters représentant les labels sont suffisamment distincts pour compenser les déplacements des embeddings des nœuds d'un pas de temps à un autre. D'autre part, lorsque les clusters sont trop proches, voire se chevauchent, les erreurs de classification peuvent être importantes bien que l'erreur de stabilité soit faible.

4.5 Conclusions et perspectives

Dans ce chapitre, nous avons présenté une méthode de qualification et de quantification de l'alignement d'embeddings temporels consécutifs. Pour ce faire, ont été conçues et

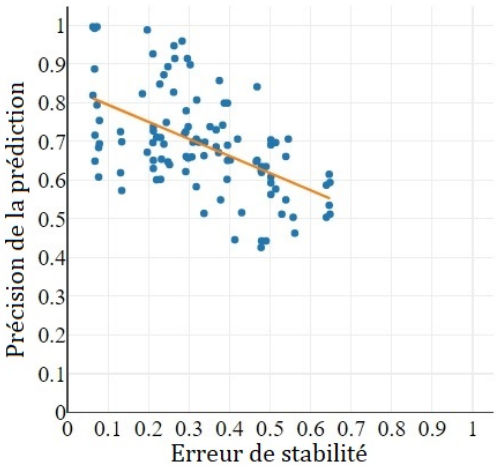


FIGURE 4.14: Précision de prédiction par rapport à l'erreur en stabilité des embeddings. Pour tous les jeux de données (dont le nombre de nœuds est supérieur à 50) et toutes les méthodes d'embedding considérées, la mesure de stabilité et la précision de prédiction sont rapportées pour chaque paire d'embeddings consécutifs.

justifiées mathématiquement plusieurs mesures relatives aux erreurs en translation, en rotation/réflexion et en échelle ainsi qu'une mesure de stabilité temporelle. Par ailleurs, nous avons proposé un processus de réalignement des embeddings temporels. Afin de démontrer la validité des mesures développées, nous avons mené différentes simulations à l'aide de données synthétiques, prouvant que chacune des mesures capture le type de désalignement pour lequel elle est conçue. Nous avons ensuite confronté la procédure proposée à des jeux de données du monde réel, en utilisant plusieurs méthodes d'embedding statiques et temporelles. Globalement, nous avons confirmé que, comme prévu, les méthodes temporelles génèrent des embeddings mieux alignés que les méthodes statiques. De plus, les expérimentations réalisées montrent que le réalignement des embeddings améliore généralement les performances de prédiction pour les tâches d'inférence sensibles à la dimension temporelle. D'autre part, la méthode de réalignement conçue semble être optimale pour éliminer les effets liés aux transformations de désalignement (translation, rotation et réflexion) pour les différentes tâches d'inférence considérées. Enfin, nous avons démontré la corrélation existant entre les performances d'inférence et la mesure de stabilité proposée.

4.5.1 Nombre de pas de temps

Dans les différentes expérimentations menées, nous nous sommes restreints à l'analyse de l'alignement des embeddings de deux pas de temps consécutifs. Néanmoins, dans le cas général, les graphes temporels contiennent plus de deux pas de temps. Cela signifie que

les processus de mesure de l'alignement et de réalignement des embeddings temporels doivent être opérés conjointement sur les différents pas de temps :

- Concernant la mesure de l'alignement, le processus appliqué aux différentes paires successives d'embeddings permet d'obtenir en sortie une série de $T - 1$ jeux de mesures (où T est le nombre de pas de temps). Il est possible d'exploiter ces informations telles quelles, attendu qu'elles renseignent sur l'évolution du désalignement des embeddings tout au long du déroulement de la dimension temporelle du graphe en entrée. D'un autre coté, si l'objectif est d'obtenir une seule mesure du désalignement global des embeddings temporels, il est possible d'agréger les différentes mesures, par exemple en les moyennant. À noter que choisir le maximum de chacune des mesures pourrait être une option à privilégier dans certains cas. En effet, cela pourrait par exemple mieux rendre compte d'un fort désalignement occurrant à une unique reprise.

- Réaligner les paires d'embeddings consécutifs indépendamment les unes des autres ne réglerait pas le problème de l'alignement. Afin de réaligner les embeddings temporels d'un graphe de plusieurs pas de temps, il faut suivre un ordre particulier. Une solution possible est de réaligner les embeddings du deuxième pas de temps vis-à-vis du premier, puis du troisième vis-à-vis du deuxième, et ainsi de suite. Ce faisant, nous nous assurons qu'une paire d'embeddings correspondants à des pas de temps quelconques t_1 et t_2 (avec $(t_1, t_2) \in [\![1,\, \mathrm{T}]\!]^2$) sont correctement alignés.

4.5.2 Métrique de l'espace latent

Un point important ayant trait au domaine de validité des processus de quantification de l'alignement et de réalignement réside dans la métrique de distance à considérer entre embeddings dans l'espace latent. En effet, les différentes mesures et opérations entrant en jeu dans la méthode présentée sont valables dans le cas où la distance entre une paire d'embeddings à un pas de temps donné est la distance euclidienne. Néanmoins, certaines méthodes d'apprentissage de représentation produisent des embeddings qu'il convient de comparer selon d'autres métriques de distance ou de similarité.

- Dans le cas où la métrique à utiliser est celle de la similarité en produit scalaire, il apparaît que la translation ne saurait être un effet imputable à un éventuel désalignement. En effet, si nous opérons une translation sur les embeddings d'un pas de temps, la matrice de similarité paire à paire entre les différents embeddings s'en trouve altérée. Par conséquent, dans un tel contexte, il ne convient pas de mesurer l'erreur en translation, ou de réaligner vis-à-vis de la translation. À

noter que dans ce cas, les embeddings ne sont pas à considérer comme des points d-dimensionnels dans l'espace latent ; ils portent plutôt les caractéristiques de vecteurs d-dimensionnels.

- Dans un autre scénario où la métrique à utiliser est celle de la similarité cosinus, et de la même manière que pour le cas du produit scalaire, les similarités entre embeddings ne sont pas invariantes à la translation. De ce fait, l'erreur en translation et le réalignement en translation ne sont pas porteurs de sens. À l'inverse, les similarités cosinus entre les paires d'embeddings restent invariables lorsqu'un changement d'échelle est opéré. Par conséquent, un changement d'échelle serait attribuable à un désalignement, contrairement au cas général présenté où la distance entre embeddings est euclidienne (comme mentionné dans la section 4.2.3). Nous notons que, dans le cas où la métrique à considérer est celle de la similarité cosinus, cela est assimilable à une distance angulaire entre embeddings. En d'autres termes, les embeddings peuvent être vus comme étant des vecteurs dont la norme est accessoire, potentiellement tous unitaires.

4.5.3 Généralisation à d'autres types de données

Telle que présentée, la méthode de mesure et d'imposition de l'alignement traite des graphes temporels. Néanmoins, notre travail peut être facilement adapté à d'autres types de données. En effet, il pourrait s'agir de toute séquence de matrices de même taille, chacune représentant un ensemble de vecteurs dans un espace multidimensionnel. Tel est par exemple le cas pour des embeddings temporels de mots où l'objectif pourrait être de mettre en évidence les glissements sémantiques temporels de certains mots ou groupes de mots en reprenant les mesures développées après avoir réaligné les embeddings des pas de temps consécutifs. De la même manière, la méthode présentée pourrait être utilisée sur des embeddings de paragraphes ou de documents avec pour application possible de découvrir l'évolution des sujets traités dans des discours par exemple.

Pour aller plus loin, il n'est pas nécessaire que les données à étudier présentent une dimension temporelle ; les données à traiter doivent être *a minima* séquentielles. En d'autres termes, elles doivent présenter une notion d'ordonnancement, pas obligatoirement temporel. Nous pourrions par exemple analyser un objet multidimensionnel présenté sur une séquence d'images successives afin de déterminer les changements de point de vue d'image, ou encore comparer deux objets différents comportant des points communs dans le but de distinguer leurs particularités. À noter qu'une telle approche serait néanmoins conditionnée par une étape préalable d'identification de repères clés des objets analysés sur les différentes images (coins, features particulières).

Chapitre 5

Conclusions et Pistes de Recherche

Dans ce travail de thèse traitant principalement de la représentation latente de graphes temporels, plusieurs contributions ont été apportées. D'abord, nous avons présenté deux méthodes d'embedding de graphes temporels. La première, basée sur des considérations et contraintes géométriques tente de retranscrire les nœuds d'un graphe dans un espace multidimensionnel en conservant l'information en entrée dans son intégralité. La deuxième [19] adapte une approche statique de référence au cas des graphes temporels et présente de bons résultats d'inférence pour les tâches de classification de nœuds. Par la suite, nous avons modifié cette approche pour adresser spécifiquement la tâche de classification [45], améliorant ainsi les performances et l'efficacité de réduction de la dimension du processus d'apprentissage de représentation.

Dans un deuxième axe, nous avons conçu une méthode générique de temporalisation d'autoencodeurs de graphes statiques [59] en reprenant et en modifiant la notion de matrices de supra-adjacence. Ce faisant, les performances des tâches d'inférence sensibles à la dimension temporelle sont significativement améliorées.

Enfin, dans un dernier volet, nous avons étudié l'alignement temporel des embeddings [65]. Dans ce cadre, nous avons conçu des processus de quantification et de correction des désalignements, puis nous avons examiné leur impact sur la consistance des embeddings et leur fiabilité quant aux tâches d'inférence en aval.

Bien que les différents travaux menés présentent plusieurs intérêts, différentes applications et pistes de recherches, théoriques et appliquées, restent à explorer.

5.1 Applications et pistes de recherche

5.1.1 Techniques de vision par ordinateur adaptées aux graphes temporels

Comme évoqué dans l'introduction, l'idée de départ du présent travail de thèse était celle d'étudier dans quelle mesure et dans quelles conditions il était possible d'adapter des techniques de vision par ordinateur au cas des graphes d'interaction temporels. À cette fin, un verrou élémentaire à lever est celui de la représentation de la donnée à traiter : en effet, pour permettre la transposition de techniques d'un type de données à un autre, il faudrait au préalable s'assurer que les deux types de données aient des formats comparables, en l'occurrence une séquence d'instantanés sous forme de matrices où la notion de localité est similaire à celle d'une image. Les techniques d'embedding répondent à cette contrainte.

Ainsi, les différents travaux présentés ont pour principal objectif de concevoir des embeddings temporels de nœuds reflétant aussi fidèlement que possible les profils d'interactions des nœuds d'un graphe temporel. Il s'est d'abord agit de construire les embeddings (en utilisant TN2V 2.2.3 ou des autoencodeurs temporalisés 3.2) pour ensuite les consolider en gommant les potentiels désalignements qu'ils peuvent inclure. En conséquence, en sortie de ces différents processus, l'information initiale est transformée en une donnée assimilable à une vidéo à d dimensions spatiales, où d est la dimension d'embedding.

À noter que les réseaux de neurones à convolution (CNN), brique essentielle des récents modèles d'apprentissage profond pour les approches de vision par ordinateur, sont théoriquement à même de traiter des données, quelle que soit leur dimension spatiale. Dans leur version bi-dimensionnelle, le principe de base des CNN est de parcourir chaque partie d'une image (selon des *carrés bi-dimensionnels* souvent de taille 3×3 pixels) afin d'en tirer des informations caractéristiques. Dans le cas où les données en entrée sont de dimension supérieure à 2, le principe reste le même : il ne s'agit alors plus de carrés mais de cubes (pour $d = 3$) ou d'hypercubes (pour $d > 3$). Par exemple, des CNN tri-dimensionnels sont appliqués pour le traitement de données LIDAR [88, 89], d'images hyperspectrales [90] ou encore d'images médicales $3D$ issues d'IRM ou de PET scans [91]. Il existe même quelques rares travaux [92] appliquant des CNN $4D$ à des données LIDAR et IRM temporelles (3 dimensions spatiales et une dimension temporelle) et considérant le temps comme une dimension quelconque, au même titre que les 3 autres dimensions spatiales.

Néanmoins, la complexité des traitements augmente de manière exponentielle vis-à-vis de la dimensionnalité des données en entrée. En pratique, il n'est pas réaliste d'appliquer

des CNN à des données de dimension 6 ou 7 avec une puissance de traitement raisonnable. Attendu que les embeddings temporels de graphes sont représentables dans des espaces latents de dimension $d > 10$ (comme démontré en section 2.2.5), il est évident qu'il est nécessaire de réduire drastiquement la dimensionnalité des embeddings. C'est d'ailleurs tout le sens de l'approche TsTN2V, où la dimension est réduite au prix de la perte de la généralité des embeddings au regard de la tâche d'inférence en aval. Par ailleurs, il existe des approches intéressantes pouvant aider à traiter de manière éparse, et donc plus légère en termes de complexité, les embeddings temporels des nœuds. Il s'agit des convolutions éparses [93, 94]. En effet, au vu de la répartition en nuages de points des embeddings comme nous pouvons le remarquer dans la figure 2.6, il semble inutile d'analyser toutes les régions de l'espace latent. En lieu et place, il est possible de se contenter d'examiner les parties occupées par les embeddings. Moyennant ces différentes solutions de réduction de la dimension (task-specific embedding et convolution éparse), il est envisageable de transposer les techniques de vision par ordinateur aux embeddings de graphes temporels.

Sur cette base, plusieurs applications sont possibles. Pour n'en citer que quelques unes :

- La classification et reconnaissance de mouvements sur des vidéos [95, 96] peut être adaptée pour détecter des motifs d'interaction particuliers entre les nœuds d'un graphe temporel et des évolutions caractéristiques, tels que le buzz ou l'émergence de nouvelles communautés par exemple.

- En utilisant des réseaux génératifs adversariaux pour vidéos (VGAN) [97], il est possible de générer des évolutions plausibles d'un graphe temporel à des fins de prédiction et de simulation.

- Les techniques de réduction de flou dans les vidéos (*video deblurring*) [98, 99] pourraient être appliquées aux embeddings temporels de nœuds afin de les consolider, en favorisant leur stabilité temporelle et en réduisant les bruits inhérents au processus de création des embeddings. Dans le même esprit, afin d'améliorer la résolution des embeddings (i.e. la précision du positionnement des embeddings dans l'espace latent), il est possible de reprendre les techniques de *video super-resolution* [100].

- Il est également envisageable d'adapter les techniques de reconstruction de régions manquantes sur certaines images d'une vidéo [101], ou d'interpolation d'instantanés (i.e. images) manquants dans une vidéo [102] au cas des embeddings temporels de graphes présentant des données incomplètes.

- Enfin, une application originale serait celle de l'adaptation des deepfakes [103] aux embeddings de graphes temporels. Dans le cas des *faceswap*, l'idée pourrait être de

simuler des évolutions factices de graphes, par exemple en calquant un phénomène de buzz avéré observé sur un réseau social sur un autre graphe temporel.

5.1.2 Alignement temporel d'embeddings : un cas d'étude

Outre l'aspect théorique de l'analyse de l'alignement des embeddings d'un graphe temporel, des cas pratiques d'application peuvent être adressés avec les méthodologies présentées. Un cas original serait par exemple l'étude des discours de personnalités politiques afin de vérifier certaines théories politiques du type : la société et — par conséquent — les hommes politiques se *droitisent*.

Afin de vérifier cela, nous pourrions créer un graphe temporel biparti où les nœuds pourraient être des politiciens ou des mots et concepts partisans : plutôt de gauche (comme écologie, répartition des richesses, violences policières, woke) ou plutôt marqués à droite (immigration, ensauvagement, grand remplacement). Dans ce graphe, où le découpage temporel pourrait par exemple être annuel, une arête illustrerait le fait qu'un politicien a prononcé un mot partisan durant un pas de temps. Le poids d'une arête pourrait être le nombre d'occurrences du mot en question dans tous les discours d'un politicien sur une année.

Sur la base de ce graphe temporel, des embeddings correspondants aux politiciens et aux concepts peuvent être construits. Ensuite, nous pouvons réaligner les embeddings, mais uniquement vis-à-vis des concepts politiques : concrètement, nous cherchons les transformations optimales (i.e. translation, rotation et réflexion) pour le réalignement des embeddings des concepts politiques, puis nous les appliquons aux embeddings de tous les nœuds du graphe temporel biparti, y compris ceux correspondant aux politiciens. Ce faisant, nous forçons la constance temporelle globale des représentations des concepts politiques dans l'espace latent tout au long de l'historique du graphe temporel. À l'inverse, nous laissons libre court aux embeddings des politiciens vis-à-vis de ce réalignement partiel.

Ensuite, il est possible d'analyser l'évolution temporelle des embeddings des politiciens au cours du temps : si droitisation il y a, nous devrions observer un glissement opérant sur ces embeddings (probablement selon une translation) et les rapprochant des zones de l'espace latent marquées à droite, i.e. les zones où se trouvent les concepts politiques de droite.

À noter que les mesures des erreurs en alignement, y compris celle en translation, ne fourniraient pas d'information utile pour la vérification de l'hypothèse de droitisation

étant donné que ces erreurs quantifient l'amplitude des mouvements des embeddings dans l'espace latent et non leurs directions et sens.

Enfin, il convient de signaler que la mise à l'épreuve de la théorie de droitisation peut également concerner les partis et formations politiques. En effet, il est possible de créer un nœud correspondant à un parti dans le graphe temporel construit. Une autre manière de procéder serait de considérer un parti politique comme la communauté de nœuds des politiciens qui y sont adhérents.

5.1.3 Profondeur des méthodes d'embedding de graphes temporels

Dépendamment de leurs types, les différentes méthodes d'embedding sont des modèles d'apprentissage automatique de profondeur variable. Par exemple, dans le cas de TN2V, nous pouvons considérer le processus d'apprentissage des embeddings comme étant peu profond (i.e. *shallow method*), étant donné qu'il n'y pas de couche cachée et que les poids en sortie sont les seuls paramètres à optimiser. À l'inverse, certains autoencodeurs présentent un nombre relativement important de couches cachées et peuvent, par conséquent, être considérés comme relevant de l'apprentissage profond. Tel est par exemple le cas de SDNE [9] — et de sa version temporalisée — où il est possible de définir le nombre de couches intermédiaires ainsi que le nombre de poids qu'elles contiennent.

Dans le cadre des graphes statiques, de récents travaux [104] traitant du sujet des GNN (*graph neural networks*) tendent à prouver que, plus les réseaux de neurones utilisés sont profonds (i.e. en nombre de couches cachées), moins ils sont performants. Cela serait dû à plusieurs phénomènes : le *over-smoothing* [105] qui consiste en la convergence des caractéristiques de l'ensemble des nœuds en un vecteur unique en conséquence de la superposition de plusieurs couches de convolution de graphes, ou encore l'effet de goulot d'étranglement au niveau des neurones d'un modèle, se traduisant par une sur-accumulation des informations provenant d'un nombre important de neurones voisins [106].

Une piste de recherche intéressante serait de vérifier si ces observations concernent également les graphes temporels. En effet, il est envisageable que, afin de traiter convenablement une séquence de graphes statiques, il faille des modèles plus sophistiqués, et donc plus profonds, pour gérer la dimension temporelle ; autrement, les GNN temporels pourraient être également sujets aux phénomènes de détérioration de performance quand leur structure est profonde.

Afin de valider ou d'infirmer ces différentes hypothèses, il est possible de mettre à profit la méthode de temporalisation décrite dans le chapitre 3. Pour un autoencodeur donné,

l'idée serait de comparer la profondeur nécessaire pour le traitement de graphes statiques d'un coté, puis de graphes temporels de l'autre (via l'emploi d'une version temporalisée du même autoencodeur). En effectuant cette opération sur plusieurs jeux de données statiques et temporels, et sur plusieurs autoencodeurs, il serait possible de répondre à la question posée, à savoir : faut-il des GNN — plus — profonds pour traiter des graphes temporels ?

5.2 Sujets de recherche connexes

Dans les différents travaux et pistes de recherche présentés, certains aspects du traitement de graphes temporels n'ont pas été abordés, bien que leur importance soit de premier ordre. Entre autres, le découpage temporel d'un graphe en pas de temps ou encore la dynamique de l'ensemble des nœuds (au sens apparition et disparition de nœuds au cours du temps).

5.2.1 Découpage temporel

À l'exception notable du modèle GeoEmb 2.1 (où les interactions sont traitées une à une, selon leur ordonnancement temporel), toutes les approches que nous avons développées sont conditionnées par une étape préalable de découpage temporel du graphe en entrée. En effet, les méthodes d'embedding et d'analyse de l'alignement présentées ne remettent pas en cause et ne challengent pas cette opération d'agrégation. À l'évidence, la pertinence des choix que nous avons faits et les qualités / défauts intrinsèques aux méthodes conçues sont indépendantes du découpage temporel, attendu que cette étape est en dehors du cadre adressé par nos travaux. Néanmoins, il est tout aussi évident que l'amélioration des performances des tâches d'inférence appliquées aux embeddings passe également par des données en entrée de meilleure qualité et, par conséquent, d'une optimisation de l'agrégation temporelle.

En effet, le partitionnement des graphes temporels revêt une importance primordiale. Par exemple, dans le cas où la répartition des interactions dans le temps est hétérogène, un simple découpage en fenêtres temporelles de durées égales pourrait conduire à des instantanés disparates en densité d'information et à une instabilité importante. D'autre part, le choix du nombre de pas de temps est un aspect primordial : quand ce nombre est trop faible, l'information temporelle en entrée est en partie perdue étant donné que chaque instantané consiste en un graphe statique où l'ordonnancement des interactions n'existe pas. À l'inverse, un nombre excessif de pas de temps conduirait à des embeddings instables et à une complexité de traitement élevée.

Compte tenu de ces différentes considérations, le découpage d'une séquence d'interactions entre nœuds en pas de temps est un axe de recherche à part entière. Il existe une multitude de travaux traitant de ce sujet. Certaines approches [107, 108] tentent de déterminer les effets d'une agrégation sur les caractéristiques du graphe temporel. D'autres travaux [109, 110] proposent des méthodes d'agrégation optimales selon différents critères (tels que la perte d'information temporelle ou l'instabilité des séquences de graphes statiques à l'issue de l'agrégation). Enfin, quelques approches [111] proposent des métriques d'évaluation d'un partitionnement temporel donné, vis-à-vis de la stabilité ou de la fiabilité du graphe temporel agrégé.

5.2.2 Disparition et apparition des nœuds d'un graphe temporel

Un autre point important que nous n'avons pas abordé dans les travaux menés est celui de l'évolution du nombre de nœuds sur les différents pas de temps. En effet, des nœuds peuvent apparaître ou disparaître pendant le déroulement de l'historique des interactions d'un graphe temporel. Tel est par exemple le cas du jeu de données AMiner où la majorité des co-auteurs (i.e. les nœuds) n'ont pas d'interactions au cours des premiers pas de temps.

Dans les différentes méthodes d'embedding que nous avons présentées (GeoEmb, TN2V / TsTN2V, les autoencodeurs temporalisés), cet aspect est géré de manière tacite : les nœuds sont considérés comme présents tout au long de l'historique du graphe temporel quand bien même ils peuvent ne pas avoir d'interactions à certains pas de temps. Ainsi, par effet de lissage temporel induit par le processus de construction des représentations, les nœuds ont des embeddings à tous les pas de temps.

À notre connaissance, les méthodes d'embedding de l'état de l'art n'adressent pas particulièrement cet aspect. Par ailleurs, il serait intéressant d'inclure des mécanismes de gestion de l'absence de nœuds à certains pas de temps. Par exemple, nous pourrions forcer l'embedding d'un nœud absent à un pas de temps à *s'éloigner* relativement aux autres nœuds dans l'espace latent : l'idée serait de refléter le fait qu'une paire de nœuds est proche quand les interactions qui les lient sont fortes, mais également régulières.

Note finale

Dans ce dernier chapitre, nous avons abordé plusieurs aspects liés de près ou de loin à la problématique de l'embedding de graphes temporels et à leurs applications. La série des différentes pistes de recherche évoquées est évidemment non exhaustive. En effet,

comme mentionné dans l'introduction, les travaux menés s'inscrivent dans un sujet en effervescence et en passe de devenir mature. Beaucoup d'avancées ont été réalisées et le plus intéressant reste à venir : l'embedding d'hypergraphes temporels, pour ne citer qu'un exemple.

Annexe A

GeoEmb : Matrice d'Initialisation

L'objectif est de démontrer qu'une matrice d'initialisation de GeoEmb satisfaisant une équidistance entre les embeddings de N nœuds, avec une distance entre embeddings égale à 1, peut être exprimée en dimension $N-1$ comme suit :

$$M_{t=0}^{N} = \begin{pmatrix} 0 & 0 & 0 & 0 & ... & 0 & 0 \\ \alpha_1 & 0 & 0 & 0 & ... & 0 & 0 \\ \frac{\alpha_1}{2} & \alpha_2 & 0 & 0 & ... & 0 & 0 \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \alpha_3 & 0 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \frac{\alpha_3}{4} & \frac{\alpha_4}{5} & ... & \alpha_{N-2} & 0 \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \frac{\alpha_3}{4} & \frac{\alpha_4}{5} & ... & \frac{\alpha_{N-2}}{N-1} & \alpha_{N-1} \end{pmatrix} \text{ avec } \alpha_i = \sqrt{\frac{i+1}{2i}} \tag{A.1}$$

Nous procédons par étapes. D'abord, soit une matrice $N \times (N-1)$ satisfaisant une équidistance unitaire entre ses lignes et dont la partie triangulaire supérieure est nulle :

$$M_{t=0}^{N} = \begin{pmatrix} 0 & 0 & 0 & ... & 0 & 0 \\ X_1^2 & 0 & 0 & ... & 0 & 0 \\ X_1^3 & X_2^3 & 0 & ... & 0 & 0 \\ X_1^4 & X_2^4 & X_3^4 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ X_1^{N-1} & X_2^{N-1} & X_3^{N-1} & ... & X_{N-2}^{N-1} & 0 \\ X_1^N & X_2^N & X_3^N & ... & X_{N-2}^N & X_{N-1}^N \end{pmatrix} \tag{A.2}$$

Nous remarquons certaines propriétés que cette matrice doit posséder. Pour i et $j > i$ quelconques :

$$X_{i-1}^{j} = \frac{X_{i-1}^{i}}{i} \tag{A.3}$$

Cette déduction provient du fait que, dans la matrice $M^N_{t=0}$, la projection du point p_j (i.e. $j^{\text{ème}}$ ligne de la matrice $M^N_{t=0}$) sur le sous-espace contenant les i premiers points se trouve être le barycentre de ces i points : c'est ce qui garantit l'équidistance de p_j par rapport aux i premiers points. Cela est illustré dans la matrice A.4. Dans un souci de lisibilité, nous notons $\alpha_i = X_i^{i+1}$.

$$M^N_{t=0} = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ \alpha_1 & 0 & 0 & \dots & 0 & 0 \\ \frac{\alpha_1}{2} & \alpha_2 & 0 & \dots & 0 & 0 \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \alpha_3 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \frac{\alpha_3}{4} & \dots & \alpha_{N-2} & 0 \\ \frac{\alpha_1}{2} & \frac{\alpha_2}{3} & \frac{\alpha_3}{4} & \dots & \frac{\alpha_{N-2}}{N-1} & \alpha_{N-1} \end{pmatrix} \tag{A.4}$$

Cela étant établi, il s'agit de trouver les valeurs des α_i. Pour ce faire, prenons le cas de 2 points p_{i-1} et p_i de la matrice $M^N_{t=0}$ (i.e. 2 lignes consécutives de la matrice) telles que exprimées dans la matrice A.4. Étant donné leur équidistance à chacun des autres points, et en particulier au tout premier (i.e. l'origine du référentiel), nous pouvons déduire que :

$$\begin{aligned} & \left\|\overrightarrow{OP_{i-1}}\right\| - \left\|\overrightarrow{OP_i}\right\| = 0 \\ \Longrightarrow \quad & \alpha_i^2 + \frac{\alpha_{i-1}^2}{i} - \alpha_{i-1}^2 = 0 \\ \Longrightarrow \quad & \alpha_i^2 = \frac{i^2-1}{i^2}\alpha_{i-1}^2 \\ \Longrightarrow \quad & \alpha_i^2 = \frac{(i+1)\cdot\cancel{(i-1)}}{i\cdot\cancel{i}} \times \frac{\cancel{i}\cdot\cancel{(i-2)}}{\cancel{(i-1)}\cdot\cancel{(i-1)}} \times \dots \times \frac{\cancel{(2+1)}\cdot(2-1)}{\cancel{2}\cdot 2} \times \alpha_1^2 \\ \Longrightarrow \quad & \alpha_i = \sqrt{\frac{i+1}{2i}} \end{aligned} \tag{A.5}$$

Annexe B

Formulation de l'Erreur en Rotation

Étant donné une matrice orthogonale R de taille $d \times d$, nous cherchons à prouver que :

$$\|R - I\|_F = \sqrt{2\,d - 2\,Tr(R)} \tag{B.1}$$

où I est la matrice identité de dimension d, $Tr(R)$ est la trace de la matrice R et $\|R - I\|_F$ représente la norme Frobenius de la matrice $R - I$. D'abord, soit R une matrice orthogonale :

$$R = \begin{pmatrix} a_1^1 & a_2^1 & a_3^1 & ... & a_d^1 \\ a_1^2 & a_2^2 & a_3^2 & ... & a_d^2 \\ a_1^3 & a_2^3 & a_3^3 & ... & a_d^3 \\ ... & ... & ... & ... & ... \\ a_1^d & a_2^d & a_3^d & ... & a_d^d \end{pmatrix} \tag{B.2}$$

La démonstration découle directement du développement du terme $\|R - I\|_F$.

$$\begin{aligned} \|R - I\|_F &= \sqrt{\sum_{i=1}^{d} \left(\left(a_i^i - 1\right)^2 + \sum_{j=1,\, j\neq i}^{d} \left(a_j^i\right)^2 \right)} \\ &= \sqrt{\sum_{i=1}^{d} \left(1 - 2a_i^i + \sum_{j=1}^{d} \left(a_j^i\right)^2 \right)} \\ &= \sqrt{\sum_{i=1}^{d} \left(2 - 2a_i^i\right)} \quad \text{vu que } R \text{ est orthogonale} \\ &= \sqrt{2\,d - 2\,Tr(R)} \end{aligned} \tag{B.3}$$

Annexe C

Simulation d'une Matrice de Rotation

Étant donné un facteur de rotation r, nous souhaitons construire une matrice de rotation/réflexion R synthétique de taille $d \times d$. Pour ce faire, nous tirons profit de la forme canonique d'une matrice orthogonale (tableau 4.1) ainsi que de la formulation de l'erreur en rotation ξ_{rot} qui en découle (équation 4.8).

Afin de simplifier la procédure de simulation, nous faisons le choix de créer une matrice de rotation/réflexion R dont l'erreur en rotation ξ_{rot} correspond exactement au facteur de rotation r.

À noter que des valeurs extrêmes de r peuvent conditionner la présence ou l'absence d'une réflexion dans la matrice orthogonale simulée : par exemple, dans les cas où r est très faible, voire nul, la matrice simulée en sortie ne peut contenir de réflexion car cela entraînerait d'emblée une erreur en rotation importante. À l'inverse, un r élevé impose une réflexion dans le cas d'une dimension d'embedding impaire et l'interdit pour une dimension paire. Enfin, quand r a des valeurs intermédiaires, le choix de la présence ou de l'absence d'une réflexion dans la matrice en sortie R est arbitré par une variable aléatoire.

La procédure de simulation est explicitée dans l'algorithme ci-dessous où la fonction *matCan* fait référence à la construction d'une matrice canonique (i.e. matrice diagonale par blocs) sur la base de ses angles de rotation élémentaires ainsi que d'éventuels compléments, à savoir des 1 et -1 (cf. tableau 4.1).

Algorithme 2: Simulation d'une matrice de rotation/réflexion

Données en entrée: $(d,\ r)$
$Tr = d \cdot \left(1 - 2r^2\right)$;
λ aléatoire uniforme dans $[-1,\ 1]$;
si $\lambda > \frac{|Tr|}{d-2}$ **alors**
  si *d pair* **alors**
    $\left(\omega_1,\ \omega_2,\ \ldots,\omega_{(d-2)/2}\right)$ aléatoires satisfaisant :
      $\sum \omega_i = Tr/2$;
      $\forall i \in [\![1,\ (d-2)/2]\!] \quad : \quad |\omega_i| \leq 1$;
    $\forall i \in [\![1,\ (d-2)/2]\!] \quad : \quad \theta_i = \arccos(\omega_i)$;
    $R_{can} = matCan\left(angles = \{\theta_1,\ \theta_2,\ \ldots,\theta_{(d-2)/2}\},\ complement = \{1,\ -1\}\right)$;
  sinon
    $\left(\omega_1,\ \omega_2,\ \ldots,\omega_{(d-1)/2}\right)$ aléatoires satisfaisant :
      $\sum \omega_i = (Tr+1)/2$;
      $\forall i \in [\![1,\ (d-1)/2]\!] \quad : \quad |\omega_i| \leq 1$;
    $\forall i \in [\![1,\ (d-1)/2]\!] \quad : \quad \theta_i = \arccos(\omega_i)$;
    $R_{can} = matCan\left(angles = \{\theta_1,\ \theta_2,\ \ldots,\theta_{(d-1)/2}\},\ complement = \{-1\}\right)$;
  fin
sinon
  si *d pair* **alors**
    $\left(\omega_1,\ \omega_2,\ \ldots,\omega_{d/2}\right)$ aléatoires satisfaisant :
      $\sum \omega_i = Tr/2$;
      $\forall i \in [\![1,\ d/2]\!] \quad : \quad |\omega_i| \leq 1$;
    $\forall i \in [\![1,\ d/2]\!] \quad : \quad \theta_i = \arccos(\omega_i)$;
    $R_{can} = matCan\left(angles = \{\theta_1,\ \theta_2,\ \ldots,\theta_{d/2}\},\ complement = \{\}\right)$;
  sinon
    $\left(\omega_1,\ \omega_2,\ \ldots,\omega_{(d-1)/2}\right)$ aléatoires satisfaisant :
      $\sum \omega_i = (Tr-1)/2$;
      $\forall i \in [\![1,\ (d-1)/2]\!] \quad : \quad |\omega_i| \leq 1$;
    $\forall i \in [\![1,\ (d-1)/2]\!] \quad : \quad \theta_i = \arccos(\omega_i)$;
    $R_{can} = matCan\left(angles = \{\theta_1,\ \theta_2,\ \ldots,\theta_{(d-1)/2}\},\ complement = \{1\}\right)$;
  fin
fin
Q matrice $d \times d$ orthogonale aléatoire;
$R = Q^T\ R_{can}\ Q$;
retourner R

Annexe D

Borne Maximale de l'Erreur de Stabilité

Comme nous pouvons l'observer dans les figures 4.8, 4.9 et 4.10, la valeur de l'erreur de stabilité ξ_{st} que nous pouvons obtenir sur la base d'embeddings synthétiques ne dépasse pas 0.8. En réalité, la limite supérieure de ξ_{st} est de 1. Cette limite est pratiquement impossible à atteindre avec des embeddings aléatoires. En effet, ξ_{st} est à 1 uniquement dans des conditions très particulières.

Ci-dessous des exemples de paires d'embeddings (correspondant à 2 pas de temps consécutifs) présentant une erreur de stabilité ξ_{st} égale à 1. À noter que chacune des paires d'embeddings ci-dessous est alignée (i.e. présentant des embeddings centrés et correctement orientés) et normalisée ; autrement dit, leurs erreurs en translation, en rotation/réflexion et en échelle sont nulles.

Embeddings en t_0	**Embeddings en t_1**
$\begin{bmatrix} 0 & 0 \\ -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & -1 \\ 0 & 0 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & 0 & 0 & 0 \\ \alpha & \beta & \alpha & \beta \\ 0 & 0 & 0 & 0 \\ \beta & \alpha & \beta & \alpha \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -\beta & -\alpha & -\beta & -\alpha \\ 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 \\ -\alpha & -\beta & -\alpha & -\beta \\ 0 & 0 & 0 & 0 \end{bmatrix}$

TABLEAU D.1: Exemples de paires d'embeddings présentant une erreur de stabilité ξ_{st} égale à 1. α et β représentent respectivement $cos\left(\frac{2\pi}{3}\right) + sin\left(\frac{2\pi}{3}\right)$ et $cos\left(\frac{2\pi}{3}\right) - sin\left(\frac{2\pi}{3}\right)$.

Bibliographie

[1] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[2] M BELKIN. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Nuearl Processing Systems (NIPS14), 2002*, pages 585–591, 2002.

[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 891–900, 2015.

[4] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114, 2016.

[5] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[6] 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.

[7] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[9] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

[12] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Continuous-time dynamic network embeddings. In *Companion Proceedings of the The Web Conference 2018*, pages 969–976, 2018.

[13] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. Embedding temporal network via neighborhood formation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2857–2866, 2018.

[14] Sedigheh Mahdavi, Shima Khoshraftar, and Aijun An. dynnode2vec: Scalable dynamic network embedding. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3762–3765. IEEE, 2018.

[15] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[16] Chengbin Hou, Han Zhang, Shan He, and Ke Tang. Glodyne: Global topology preserving dynamic network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[17] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[18] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the eleventh acm international conference on web search and data mining*, pages 673–681, 2018.

[19] Mounir Haddad, Cécile Bothorel, Philippe Lenca, and Dominique Bedart. Temporalnode2vec: temporal node embedding in temporal networks. In *International Conference on Complex Networks and Their Applications*, pages 891–902. Springer, 2019.

[20] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[21] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242, 2010.

[22] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[23] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[24] Siquan Yu, Jiaxin Liu, Zhi Han, Yong Li, Yandong Tang, and Chengdong Wu. Representation learning based on autoencoder and deep adaptive clustering for image clustering. *Mathematical Problems in Engineering*, 2021, 2021.

[25] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6964–6974, 2021.

[26] Sankeerth Rao Karingula, Nandini Ramanan, Rasool Tahsambi, Mehrnaz Amjadi, Deokwoo Jung, Ricky Si, Charanraj Thimmisetty, and Claudionor Nunes Coelho Jr. Boosted embeddings for time series forecasting. *arXiv preprint arXiv:2104.04781*, 2021.

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

[28] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

[29] Sandra Mitrović, Bart Baesens, Wilfried Lemahieu, and Jochen De Weerdt. Churn prediction using dynamic rfm-augmented node2vec. In *International Workshop on Personal Analytics and Privacy*, pages 122–138. Springer, 2017.

[30] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee, 2015.

[31] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

[32] Jingxin Liu, Chang Xu, Chang Yin, Weiqiang Wu, and You Song. K-core based temporal graph convolutional network for dynamic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[33] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 257–266, 2019.

[34] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. Sign: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198*, 2020.

[35] Ahmed Fathy and Kan Li. Temporalgat: Attention-based dynamic graph representation learning. *Advances in Knowledge Discovery and Data Mining*, 12084: 413, 2020.

[36] Jianan Zhong, Hongjun Qiu, and Benyun Shi. Dynamics-preserving graph embedding for community mining and network immunization. *Information*, 11(5):250, 2020.

[37] Sujit Rokka Chhetri and Mohammad Abdullah Al Faruque. Dynamic graph embedding. In *Data-Driven Modeling of Cyber-Physical Systems using Side-Channel Analysis*, pages 209–229. Springer, 2020.

[38] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhao Li, and Can Wang. Learning temporal interaction graph embedding via coupled memory networks. In *Proceedings of The Web Conference 2020*, pages 3049–3055, 2020.

[39] Karl Pearson. The problem of the random walk. *Nature*, 72(1867):342, 1905.

[40] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[41] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48. ACM, 2013.

[42] Xiaofei Xu, Ke Deng, Fei Hu, and Li Li. An improved historical embedding without alignment. *arXiv preprint arXiv:1910.08692*, 2019.

[43] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[44] Daniele Granata and Vincenzo Carnevale. Accurate estimation of the intrinsic dimension using graph distances: Unraveling the geometric complexity of datasets. *Scientific reports*, 6:31377, 2016.

[45] Mounir Haddad, Cécile Bothorel, Philippe Lenca, and Dominique Bedart. Task-specific Temporal Node Embedding. In *French Regional Conference on Complex Systems*, Dijon, France, May 2021. URL https://hal.archives-ouvertes.fr/hal-03274101.

[46] Karl Pearson. X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 50(302):157–175, 1900.

[47] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[48] Yasi Wang, Hongxun Yao, Sicheng Zhao, and Ying Zheng. Dimensionality reduction strategy based on auto-encoder. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*, pages 1–4, 2015.

[49] Paulo E Rauber, Alexandre X Falcao, Alexandru C Telea, et al. Visualizing time-dependent data using dynamic t-sne. 2016.

[50] Yiru Zhao, Bing Deng, Chen Shen, Yao Liu, Hongtao Lu, and Xian-Sheng Hua. Spatio-temporal autoencoder for video anomaly detection. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1933–1941, 2017.

[51] Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *arXiv preprint arXiv:1511.06309*, 2015.

[52] Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference*, pages 400–404. Springer, 1975.

[53] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[54] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[55] Supriya Pandhre, Himangi Mittal, Manish Gupta, and Vineeth N Balasubramanian. Stwalk: learning trajectory representations in temporal graphs. In *Proceedings of the ACM India joint international conference on data science and management of data*, pages 210–219, 2018.

[56] Koya Sato, Mizuki Oka, Alain Barrat, and Ciro Cattuto. Dyane: dynamics-aware node embedding for temporal networks. *arXiv preprint arXiv:1909.05976*, 2019.

[57] Eugenio Valdano, Luca Ferreri, Chiara Poletto, and Vittoria Colizza. Analytical computation of the epidemic threshold on temporal networks. *Physical Review X*, 5(2):021005, 2015.

[58] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical formulation of multilayer networks. *Physical Review X*, 3(4):041022, 2013.

[59] Mounir Haddad, Cécile Bothorel, Philippe Lenca, and Dominique Bedart. Temporalizing static graph autoencoders to handle temporal networks. In *2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2021.

[60] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[61] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[62] Lin Li and William M Campbell. Matching community structure across online social networks. *arXiv preprint arXiv:1608.01373*, 2016.

[63] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[64] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[65] Furkan Gürsoy, Mounir Haddad, and Cécile Bothorel. Alignment and stability of embeddings: measurement and inference improvement. *arXiv preprint arXiv:2101.07251*, 2021.

[66] Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. Dynamic embeddings for user profiling in twitter. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1764–1773, 2018.

[67] Andrea Palmucci, Hao Liao, Andrea Napoletano, and Andrea Zaccaria. Where is your field going? a machine learning approach to study the relative motion of the domains of physics. *PloS one*, 15(6):e0233997, 2020.

[68] Jin Xu, Yubo Tao, Yuyu Yan, and Hai Lin. Exploring evolution of dynamic networks via diachronic node embeddings. *IEEE transactions on visualization and computer graphics*, 26(7):2387–2402, 2018.

[69] Uriel Singer, Ido Guy, and Kira Radinsky. Node embedding over temporal graphs. *arXiv preprint arXiv:1903.08889*, 2019.

[70] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[71] Chunsheng Fang, Mojtaba Kohram, Xiangxiang Meng, and Anca Ralescu. Graph embedding framework for link prediction and vertex behavior modeling in temporal social networks. In *Proceddings of the SIGKDD Workshop on Social Network Mining and Analysis*, 2011.

[72] Francesco Sanna Passino, Anna S Bertiger, Joshua C Neil, and Nicholas A Heard. Link prediction in dynamic networks using random dot product graphs. *arXiv preprint arXiv:1912.10419*, 2019.

[73] Isuru Udayangani Hewapathirana, Dominic Lee, Elena Moltchanova, and Jeanette McLeod. Change detection in noisy dynamic networks: a spectral embedding approach. *Social Network Analysis and Mining*, 10(1):1–22, 2020.

[74] Sunipa Dev, Safia Hassan, and Jeff M Phillips. Closed form word embedding alignment. *Knowledge and Information Systems*, 63(3):565–588, 2021.

[75] John C Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

[76] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*, 2018.

[77] Chuanchang Chen, Yubo Tao, and Hai Lin. Dynamic network embeddings for network evolution analysis. *arXiv preprint arXiv:1906.09860*, 2019.

[78] TP Peixoto. The netzschleuder network catalogue and repository (2020). *URL https://networks. skewed. de*, 2020.

[79] SocioPatterns. Sociopatterns, 2021. URL http://www.sociopatterns.org/. [Online; accessed 10-January-2021].

[80] Tom AB Snijders, Gerhard G Van de Bunt, and Christian EG Steglich. Introduction to stochastic actor-based models for network dynamics. *Social networks*, 32 (1):44–60, 2010.

[81] Gerhard G Van de Bunt, Marijtje AJ Van Duijn, and Tom AB Snijders. Friendship networks through time: An actor-oriented dynamic statistical network model. *Computational & Mathematical Organization Theory*, 5(2):167–192, 1999.

[82] Philippe Vanhems, Alain Barrat, Ciro Cattuto, Jean-François Pinton, Nagham Khanafer, Corinne Régis, Byeul-a Kim, Brigitte Comte, and Nicolas Voirin. Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. *PloS one*, 8(9):e73970, 2013.

[83] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*, 6(8):e23176, 2011.

[84] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PloS one*, 10(9):e0136497, 2015.

[85] Yahoo! The yahoo webscope program, 2021. URL https://webscope.sandbox.yahoo.com/. [Online; accessed 10-January-2021].

[86] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[87] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015.

[88] Daniel Maturana and Sebastian Scherer. 3d convolutional neural networks for landing zone detection from lidar. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 3471–3478. IEEE, 2015.

[89] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[90] Ying Li, Haokui Zhang, and Qiang Shen. Spectral–spatial classification of hyperspectral imagery with 3d convolutional neural network. *Remote Sensing*, 9(1):67, 2017.

[91] Yechong Huang, Jiahang Xu, Yuncheng Zhou, Tong Tong, Xiahai Zhuang, Alzheimer's Disease Neuroimaging Initiative (ADNI, et al. Diagnosis of alzheimer's disease via multi-modality 3d convolutional neural network. *Frontiers in Neuroscience*, 13:509, 2019.

[92] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[93] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[94] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[95] Pichao Wang, Zhaoyang Li, Yonghong Hou, and Wanqing Li. Action recognition based on joint trajectory maps using convolutional neural networks, 2016.

[96] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. Sign language recognition using convolutional neural networks. In *European Conference on Computer Vision*, pages 572–578. Springer, 2014.

[97] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics, 2016.

[98] Jinshan Pan, Haoran Bai, and Jinhui Tang. Cascaded deep video deblurring using temporal sharpness prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3043–3051, 2020.

[99] Zhihang Zhong, Ye Gao, Yinqiang Zheng, and Bo Zheng. Efficient spatio-temporal recurrent neural network for video deblurring. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020.

[100] Armin Kappeler, Seunghwan Yoo, Qiqin Dai, and Aggelos K Katsaggelos. Video super-resolution with convolutional neural networks. *IEEE transactions on computational imaging*, 2(2):109–122, 2016.

[101] Ruibin Ma, Rui Wang, Stephen Pizer, Julian Rosenman, Sarah K McGill, and Jan-Michael Frahm. Real-time 3d reconstruction of colonoscopic surfaces for determining missing regions. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 573–582. Springer, 2019.

[102] Lihao Gao, Yu Zheng, Yaqiang Wang, Jiangjiang Xia, Xunlai Chen, Bin Li, Ming Luo, and Yuchen Guo. Reconstruction of missing data in weather radar image sequences using deep neuron networks. *Applied Sciences*, 11(4):1491, 2021.

[103] Thanh Thi Nguyen, Cuong M Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep learning for deepfakes creation and detection: A survey. *arXiv preprint arXiv:1909.11573*, 2019.

[104] Kuangqi Zhou, Yanfei Dong, Kaixin Wang, Wee Sun Lee, Bryan Hooi, Huan Xu, and Jiashi Feng. Understanding and resolving performance degradation in graph convolutional networks. *arXiv preprint arXiv:2006.07107*, 2020.

[105] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

[106] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[107] Gautier Krings, Márton Karsai, Sebastian Bernhardsson, Vincent D Blondel, and Jari Saramäki. Effects of time window size and placement on the structure of an aggregated communication network. *EPJ Data Science*, 1(1):1–16, 2012.

[108] Bruno Ribeiro, Nicola Perra, and Andrea Baronchelli. Quantifying the effect of temporal resolution on time-varying networks. *Scientific reports*, 3(1):1–5, 2013.

[109] Rajmonda Sulo, Tanya Berger-Wolf, and Robert Grossman. Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 127–136, 2010.

[110] Aaron Clauset and Nathan Eagle. Persistence and periodicity in a dynamic proximity network. *arXiv preprint arXiv:1211.7343*, 2012.

[111] Alessandro Chiappori and Rémy Cazabet. Quantitative evaluation of snapshot graphs for the analysis of temporal networks. *arXiv preprint arXiv:2110.13466*, 2021.

DOCTORAT BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Embedding de Graphes Temporels : Temporalisation de Méthodes Statiques et Alignement

Mots clés : Apprentissage de représentation, Embeddings de graphes, Réseaux temporels, Autoencoders de graphes, Classification de nœuds, Reconstruction et prédiction d'arêtes

Résumé : La dernière décennie a vu émerger des travaux couvrant un large spectre des branches du traitement de la donnée ayant pour objectif l'apprentissage de représentation de données (data embedding plus communément) : il s'agit d'un prétraitement qui s'attache à représenter les données sous forme de vecteurs dans un espace latent à faible dimension. Cette technique a été adaptée à plusieurs types de données, entre autres aux graphes issus des réseaux sociaux. Néanmoins, bien que fondamentale, la dimension temporelle des graphes reste sous-explorée par l'état de l'art des méthodes d'embedding. Ce travail de thèse s'est attaché à l'étude du cas de l'embedding de graphes temporels.

Un premier axe a consisté en la conception de méthodes d'embedding temporelles : dans un premier temps, une méthode "exhaustive", puis une autre méthode couplant la notion des marches aléatoires à un mécanisme de lissage temporel entre vecteurs d'embedding consécutifs. Aussi, nous avons conçu une variante "task-specific" utilisant en partie les labels des nœuds en entrée pour améliorer les tâches de classification.

Dans le second volet du travail de thèse, nous avons conçu une approche générique de "temporalisation" d'une catégorie de méthodes d'embedding statiques (les autoencodeurs), améliorant significativement les performances des tâches d'inférence sensibles à la dimension temporelle.

Enfin, le dernier axe a été consacré à l'étude de l'alignement temporel des embeddings, i.e. les potentiels décalages entre les espaces d'embedding des différents pas de temps, et à son impact sur la consistance des représentations latentes apprises.

Title : Temporal Graph Embedding : Static Methods Temporalization and Alignment

Keywords : Representation learning, Graph embeddings, Temporal networks, Graph autoencoders, Node classification, Edge reconstruction and prediction

Abstract : The last decade has seen the emergence of works covering a wide spectrum of data processing branches with the objective of data embedding: this is a pre-processing step that focuses on representing data as vectors in a low-dimensional latent space. This technique has been adapted to several types of data, including graphs from social networks. Nevertheless, although fundamental, the temporal dimension of graphs remains under-explored by the state of the art of embedding methods. This thesis focused on the case of embedding temporal graphs.

A first axis consisted in the design of dynamic embedding methods: an "extensive" method at first, then a method combining the notion of random walks with a temporal smoothing mechanism between consecutive embedding vectors. We have designed a "task-specific" variant using some labels of input nodes to improve the classification tasks.

In the second part of the thesis, we designed a generic approach to "temporalize" a class of static embedding methods (autoencoders), significantly improving the performances of time-sensitive inference tasks.

Finally, the last axis was devoted to the study of the temporal alignment of embeddings, i.e. the potential mismatches between the embedding spaces of different time steps, and its impact on the consistency of the learned latent representations.