



Stage d'été – Semaine 2

QGNN-TimeCausality

Avancement du projet



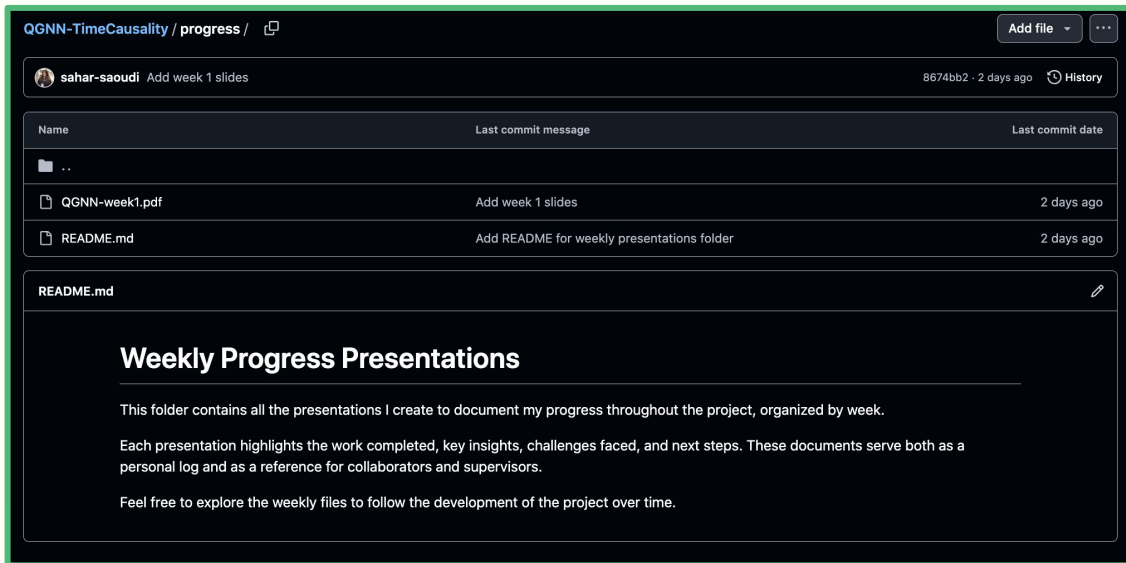
01

mise à jour des outils

Ajout de mes présentations sur mon Repository Github

Je ajouté les presentation sur mon avancement du projet dans un dossier 'progress' sur mon repo.

 [Lien](#)



QGNN-TimeCausality / progress /

sahar-saoudi Add week 1 slides 8674bb2 · 2 days ago History

Name	Last commit message	Last commit date
..		
QGNN-week1.pdf	Add week 1 slides	2 days ago
README.md	Add README for weekly presentations folder	2 days ago

README.md

Weekly Progress Presentations

This folder contains all the presentations I create to document my progress throughout the project, organized by week.

Each presentation highlights the work completed, key insights, challenges faced, and next steps. These documents serve both as a personal log and as a reference for collaborators and supervisors.

Feel free to explore the weekly files to follow the development of the project over time.

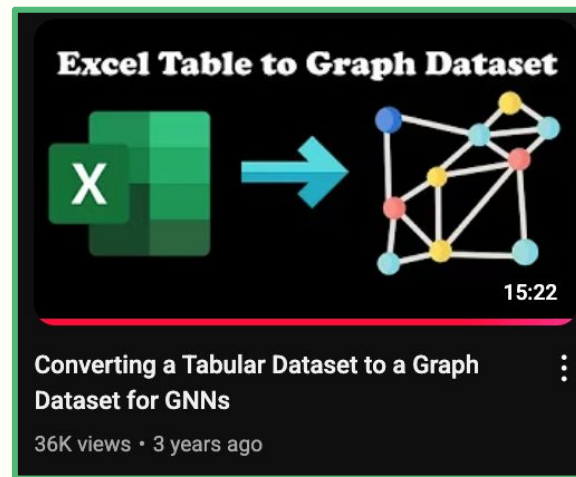


02

Ressources consultées et principaux apprentissages

Tutoriel sur la conversion de dataset tabulaire à un graph pour les GNNs

- convertir un dataset tabulaire CSV en graphe, exploitable par des modèles de GNN
- Le processus repose sur l'identification de nœuds, arêtes, attributs, et labels à partir de données tabulaires.
- Utilisation de pandas pour charger et explorer le dataset
- Sélectionner les colonnes pertinentes comme features
- extraire les labels associés à chaque nœud. Il peut s'agir de classifications, régressions ou même tâches au niveau des arêtes ou du graphe.
- Définir les relations entre les nœuds (les arêtes) selon un critère logique (exemple : Partage d'un attribut commun, Proximité spatiale ou temporelle, Interactions observées)
- Optionnel : ajouter des poids ou attributs aux arêtes.
- Utilisation des librairies comme PyTorch Geometric ou DGL
- Le plus difficile est souvent la construction des arêtes, car elle dépend fortement du contexte métier.
- Les outils comme pandas, NumPy, networkx (pour visualiser) et PyTorch Geometric sont essentiels dans le processus.



```
tabular_to_graph.ipynb
```

```
[ ] # Sort to define the order of nodes
sorted_df = first_df.sort_values(by='int_player_id')
# Select node features
node_features = sorted_df[['str_positions', 'int_long_passing', 'int_ball_control', 'int_dribbling']]
# Convert non-numeric columns
pd.set_option('mode.chained_assignment', None)
positions = node_features['str_positions'].str.split(", ", expand=True)
node_features['first_position'] = positions[0]
# One-hot encoding
node_features = pd.concat([node_features, pd.get_dummies(node_features['first_position'])], axis=1, join='inner')
node_features.drop(['str_positions', 'first_position'], axis=1, inplace=True)
node_features.head()
```

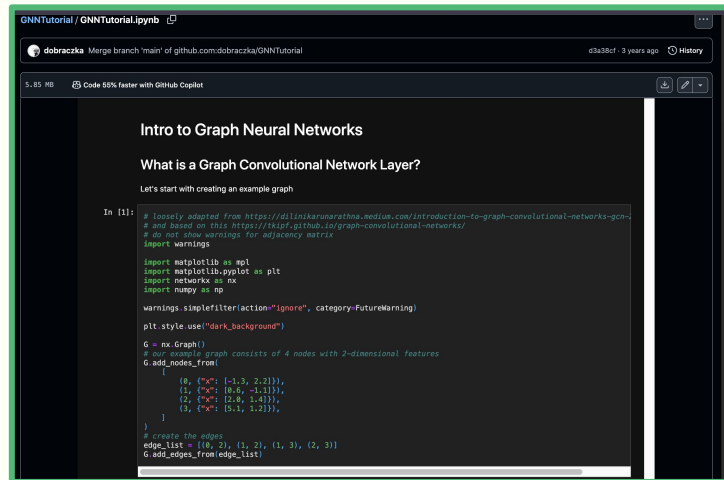
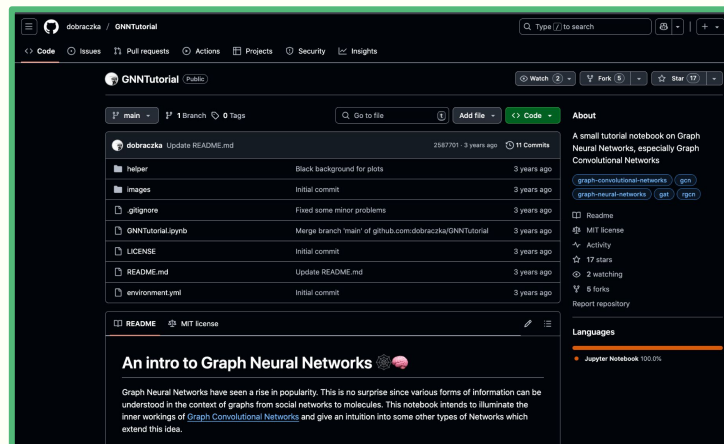
	int_long_passing	int_ball_control	int_dribbling	CAM	CB	CDM	CF	CM	CK	LB	LM	LWB	RS	RW	RWB	ST
0	91	96	96	0	0	0	0	0	0	0	0	0	0	0	0	0
33	77	92	88	0	0	0	0	0	0	0	0	0	0	0	0	1
57	40	30	12	0	0	0	0	0	0	1	0	0	0	0	0	0
121	81	95	95	0	0	0	0	0	0	0	1	0	0	0	0	0
89	93	92	88	1	0	0	0	0	0	0	0	0	0	0	0	0

That's already our node feature matrix. The number of nodes and the ordering is implicitly defined by it's shape. Each row corresponds to one node in our final graph.

Variables Terminal

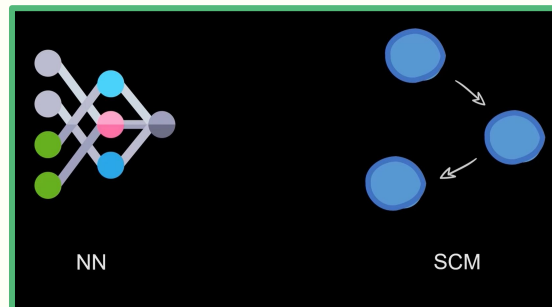
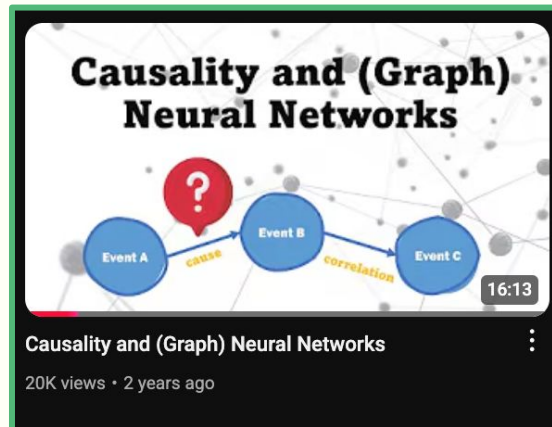
GNNs sur un ensemble de données de molécules associées au VIH/SIDA

- Implémente une classe MoleculeDataset personnalisée pour PyTorch Geometric.
- Lit des fichiers .csv contenant des données moléculaires.
- Génère des fichiers PyTorch pour chaque molécule sous forme de graphe.
- Utilise rdkit pour manipuler les structures chimiques.
- Prépare des graphes à partir de SMILES (représentations moléculaires).
- Prend en compte un mode test pour séparer les données d'évaluation.
- Comment structurer un dataset moléculaire pour un modèle GNN.
- Comment transformer des données chimiques tabulaires en graphes utilisables par PyG.
- Comment séparer données brutes (raw) et données traitées (processed).



Video sur la Causalité et les GNNs

- Explication de la causalité (relationships where intervention in one variable [cause] contributes to change in another variable [effect]) et de comment bien comprendre si deux variables ont une relation causale.
- Explication des SCMs (*Structural Causal Models*), qui sont des graphes contenant les variables et connectant leurs relations causales à travers des arêtes.
- Causal inference vs causal discovery.
- *Causal inference* : construire un graphe basé sur des hypothèses causales, puis le tester pour valider les relations causales.
- *Causal discovery* : apprendre la structure du SCM à partir des données.
- Une librairie pertinente pour effectuer de l'apprentissage de structure causale est bnlearn.
- Les SCMs sont des graphes utilisés pour tester la causalité.
- DoWhy : librairie qui peut aider à faire de la causal inference en Python
- Les GNNs peuvent apprendre sur des structures causales en exploitant les relations entre variables comme arêtes.
- Intégrer la causalité dans les GNNs peut améliorer la robustesse et la généralisation des modèles.
- On peut entraîner un GNN à prédire des effets causaux ou des interventions (ex. : que se passe-t-il si on change une variable)



Relating Graph Neural Networks to Structural Causal Models

Matej Zečević
Computer Science Department, TU Darmstadt
MATEJ.ZECJEVIC@CS.TU-DARMSTADT.DE

Devendra Singh Dhani
Computer Science Department, TU Darmstadt
DEVENDRA.DHANI@CS.TU-DARMSTADT.DE

Petar Veličković
DeepMind, London
PETARV@GOOGLE.COM

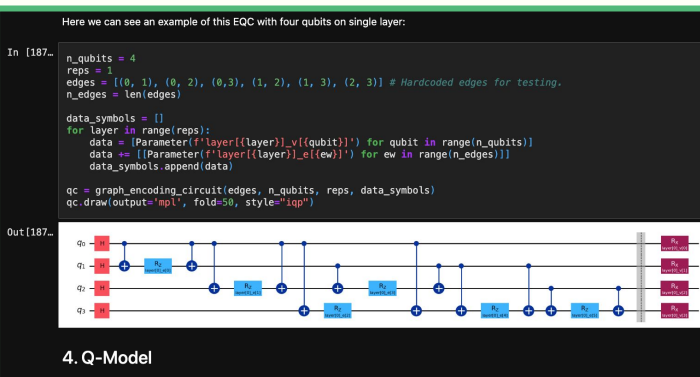
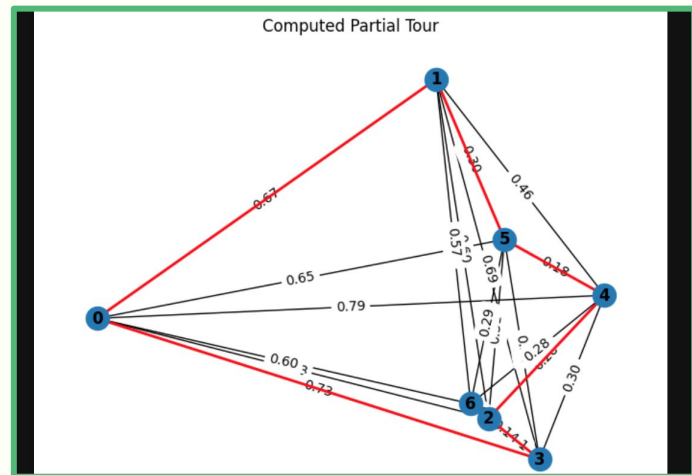
Kristian Kersting
Computer Science Department, Centre for Cognitive Science, TU Darmstadt,
and Hessian Center for AI (hessian.AI)
KERSTING@CS.TU-DARMSTADT.DE

Abstract

Causality can be described in terms of a structural causal model (SCM) that carries information on the variables of interest and their mechanistic relations. For most processes of interest the underlying SCM will only be partially observable, thus causal inference tries leveraging the exposed.

QGNN pour résoudre le problème du voyageur de commerce

- Explorer la résolution du problème du voyageur de commerce (TSP) via l'apprentissage par renforcement quantique.
- Utilisation de circuits quantiques équivariants (EQC) pour exploiter les symétries dans les graphes.
- Intégration d'un modèle d'apprentissage par renforcement pour approximer des solutions optimales au TSP.
- Les EQC assurent une invariance par permutation des nœuds, importante pour la robustesse du modèle.
- Implémentation d'un Q-learning quantique avec Qiskit et TorchConnector.
- Application de variational quantum circuits (VQC), adaptés aux dispositifs quantiques actuels (bruyants).
- Méthode : Génération de graphes aléatoires complets avec des distances euclidiennes entre nœuds / Utilisation de l'algorithme de Christofides comme heuristique de base / Construction d'un modèle de Q-learning guidé par une fonction de récompense encourageant les chemins courts
- Le modèle converge vers des solutions proches de l'optimum en environ 300 itérations : faisabilité d'utiliser des approches quantiques hybrides pour des problèmes d'optimisation combinatoire.





03

Questions & Prochaines étapes

Questions

- Est ce qu'on va implémenter une approche classique pour le projet pour la comparer après a l'approche quantique ?

Prochaines etapes

- Commencer l'implémentation, en particulier le nettoyage des ensembles de données et l'exploration de leur représentation sous forme de graphes, en veillant à respecter les relations causales.
- Poursuivre les lectures en parallèle.
- Réfléchir au type de QGNN le plus adapté au projet ainsi qu'aux différentes méthodes d'encodage des entrées que je vais utiliser
- Continuer les présentations pour communiquer l'avancement du projet



Merci de votre attention!