Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Μεταπτυχιακο Προγραμμα "Επιστημη Δεδομενων & Μηχανικη Μαθηση"

Εκμάθηση αναπαραστάσεων σε πολυσχεσιακούς γράφους με έμφαση σε θορυβώδεις γράφους

Μεταπτυχιακη Διπλωματικη Εργασια

του

ΔΗΜΗΤΡΙΟΥ Ν. ΣΥΡΡΑΦΟΥ

Επιβλέπων: Γιώργος Στάμου
Καθηγητής Ε.Μ.Π.

Εργαστηριο Τεχνητης Νοημοσυνης και Συστηματων Μαθησης (AILS)

Αθήνα, 28 Μαρτίου 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Μεταπτυχιακό Πρόγραμμα "Επιστήμη Δεδομένων & Μηχανική Μάθηση"
Εργαστήριο Τεχνητής Νοημοσύνης και Συστημάτων Μάθησης (AILS)

Εκμάθηση αναπαραστάσεων σε πολυσχεσιακούς γράφους με έμφαση σε θορυβώδεις γράφους

Μεταπτυχιακη Διπλωματικη Εργασια

του

ΔΗΜΗΤΡΙΟΥ Ν. ΣΥΡΡΑΦΟΥ

Επιβλέπων: Γιώργος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28η Μαρτίου 2023

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γιώργος Στάμου	Στέφανος Κόλλιας	Αθανάσιος Βουλόδημος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Μεταπτυχιακό Πρόγραμμα "Επιστήμη Δεδομένων & Μηχανική Μάθηση"
Εργαστήριο Τεχνητής Νοημοσύνης και Συστημάτων Μάθησης (AILS)

Περίληψη

Οι γράφοι είναι μια διαδεδομένη δομή δεδομένων για τη μοντελοποίηση πολύπλοκων συστημάτων του πραγματικού κόσμου. Συγκεκριμένα, τα περισσότερα συστήματα του πραγματικού κόσμου είναι ετερογενή, που σημαίνει ότι περιέχουν κόμβους και ακμές διαφορετικών τύπων. Για να χρησιμοποιηθούν μοντέλα μηχανικής σε τέτοιους γράφους χρειάζονται εκφραστικές αναπαραστάσεις γράφων που αποτυπώνουν τις δομικές τους πληροφορίες. Γιάυτό το λόγο, η εκμάθηση αναπαράστασης γράφου έχει τραβήξει την προσοχή τελευταία λόγω της επιτυχίας των νευρωνικών δικτύων γράφων (Graph Neural Networks) σε διάφορες απαιτητικές εργασίες μηχανικής μάθησης σε δεδομένα με τη μορφή γράφου. Επιπλέον, οι πολυσχεσιακοί γράφοι γνώσης που χρησιμοποιούνται περιέχουν συχνά θορυβώδεις πληροφορίες, είτε με τη μορφή ακμών που συνδέουν μη σχετικούς κόμβους είτε με σχέσεις χαμηλής πληροφορίας. Για να αντιμετωπίσουμε αυτό το πρόβλημα, προτείνουμε ένα μοντέλο για την ταξινόμηση κόμβων σε ετερογενείς γράφους που αξιοποιεί αναπαραστάσεις που προέρχονται από ένα συνελικτικό νευρωνικό δίκτυο γράφων (Graph Convolutional Neural Network) για να αναγνωρίσει τις τοπικές εξαρτήσεις μεταξύ κόμβων και να τις συνδυάσει με πληροφορίες σε επίπεδο σχέσης που προέρχονται από μια μέθοδο ενσωμάτωσης πολυσχεσιακών γράφων (Multi-relational graph embedding). Η μέθοδος, επονομαζόμενη Split Relation Graph Convolutional Networks (SRGCN), μοντελοποιεί ξεχωριστά τις αναπαραστάσεις του κάθε κόμβου με βάση τα διαφορετικά είδη ακμών στο γράφο, δίνοντας έτσι έμφαση στης σημαντικές σχέσεις. Συγκρίναμε την προτεινόμενη μεθοδολογία με άλλα ευρέως χρησιμοποιούμενα νευρωνικά δίκτυα γράφων σε μια πληθώρα πειραμάτων, δίνοντας έμφαση στη δυνατότητα αυτών να αξιοποιούν χρήσιμες σχέσεις στο γράφο και να αντιμετωπίζουν το θόρυβο που μπορεί να υπάρχει. Για τον σκοπό αυτό πραγματοποιήσαμε πολλά πειράματα σε γράφους που περιέχουν διάφορες μορφές θορύβου, παρατηρώντας τη συμπεριφορά του μοντέλου μας και των υπόλοιπων νευρωνικών. Το προτεινόμενο μοντέλο επιτυγχάνει βελτίωση της απόδοσης σε σύγκριση με τα GNN γενικής χρήσης, καθώς και ανθεκτικότητα έναντι του θορύβου σε διάφορα σενάρια.

Λέξεις Κλειδιά

Δεδομένα σε Μορφή Γράφου, Εκμάθηση Αναπαράστασης Γράφων, Νευρωνικά Δίκτυα Γράφων, Ετερογενείς Γράφοι, Ενσωματώσεις Γράφων Γνώσης, Θορυβώδεις Γράφοι

Abstract

Graphs are a ubiquitous data structure for modelling real-world complex systems. Specifically, most of them are heterogeneous, meaning they contain nodes of different types, and multi-relational, exhibiting multiple types of edges. In order to use machine learning models to tackle different downstream tasks on the graphs, we need expressive graph representations that capture structural information for these graphs, in order. Thus, graph representation learning has drawn significant attention recently due to the success of Graph Neural Networks in various challenging machine learning tasks on graph-structured data. Additionally, real-world graphs often contain noisy information in the form of edges connecting non-relevant nodes or low-informational relations. To address this issue, we propose a model for heterogeneous graph node classification that leverages neighbourhood aggregation representations derived from Graph Convolutional Network (GCN) to capture local dependencies between nodes and combine this information with relation-level information derived from a graph embedding learning method. Our methodology, named Split Relation Graph Convolutional Networks (SRGCN), models the representations of each node separately according to the different types of relations in the graph, enabling it to focus on important relations. We compared our model with other commonly used GNNs on a plethora of experiments, focusing on their ability to leverage informative edges, especially when noise was introduced on the graphs. To this end, we conducted many experiments on graphs under different noise scenarios, gaining insights on the behaviour of different models. Ultimately, the introduced model achieved better performance compared to the strong baseline GNNs and exhibited robustness against noise in all scenarios.

Keywords

Graph-structured Data, Graph Representation Learning, Graph Neural Networks, Heterogeneous Graphs, Knowledge Graph Embeddings, Noisy Graphs

Στη μνήμη του θείου μου, Ανδρέα

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Γιώργο Στάμου, για την ευκαιρία που μου έδωσε αρχικά να φοιτήσω στο ΔΠΜΣ αλλά και να ασχοληθώ με το συγκεκριμένο θέμα, καθώς και για τη συμβολή του στην ακαδημαϊκή μου πορεία μέχρι στιγμής.

Επίσης θα ήθελα να ευχαριστήσω τους Κωνσταντίνο Μπουγιατιώτη και Δημήτρη Κελέση, διδακτορικούς ερευνητές στο ΕΚΕΦΕ Δημόκριτος, για το έμπρακτο ενδιαφέρον τους, τη διάθεση τους να βοηθήσουν, το χρόνο τους και την υπομονή τους όλο αυτό το διάστημα που συνεργαστήκαμε στα πλαίσια της εργασίας και να τους ευχηθώ καλή επιτυχία στην εκπόνηση της διδακτορικής τους διατριβής. Έμαθα πολλά από αυτούς και η συνεργασία μας ήταν άκρως εποικοδομητική και ταυτόχρονα ευχάριστη.

Εκτός αυτών, θα ήθελα να πω ένα ευχαριστώ στη Φεναρέτη, τον Δημήτρη, την Ευμορφία και τον Βασίλη που συνέβαλαν έμπρακτα στην προσπάθεια μου αυτή, έδειξαν κατανόηση της δυσκολίας του εγχειρήματός μου και μου παρείχαν κάθε δυνατή διευκόλυνση σε όλη τη διάρκεια της εκπόνησης αυτής της εργασίας.

Επίσης να ευχαριστήσω τους φίλους και συμφοιτητές Βασίλη και Μελκόν για τη βοήθεια τους και τη συνεργασία μας γενικότερα κατά τη φοίτησή μας στο ΔΠΜΣ.

Στο σημείο αυτο θα πρωτοτυπήσω, απευθύνοντας ένα μεγάλο μα κάθε άλλο παρά συμβατικό ευχαριστώ στους γονείς μου και στον αδερφό μου, για την υλική, οικονομική αλλά κυριώς συναισθηματική στήριξη άνευ όρων και καθημερινά. Χωρίς τη βοήθειά τους, η ολοκλήρωση των σπουδών μου στο ΔΠΜΣ δε θα ήταν εφικτή.

Τέλος, ευχαριστώ ειλικρινά όλους εκείνους, που με το δικό τους τρόπο συνέβαλαν ανιδιοτελώς να ανταπεξέλθω στις δυσκολίες που παρουσιάστηκαν και να ολοκληρώσω επιτυχώς αυτή την προσπάθεια.

Contents

Prologue

This thesis has been conducted in collaboration with the Artificial Intelligence & Learning Systems (AILS) laboratory of National Technical University of Athens (NTUA) and the National Centre For Scientific Research (NCSR) Demokritos.

For the experiments and result visualizations presented in this thesis the Python programming language has been used. The Python deep learning framework used for the construction and training of the models presented is PyTorch . The neural network layers for geometric deep learning used in the models, as well as the datasets used for training and evaluation are implemented in the PyTorch Geometric library. The library used for the training of the knowledge graph embeddings is PyKeen .

For the training of the deep learning models as well as the we used an Nvidia GPU, namely the Nvidia GeForce GTX 1660.

The code for the construction, training and evaluation of the models as well as the logs containing the different results presented in this thesis are open-source and can be found on Github[1].

[1] https://github.com/dsyrrafos/thesis-gnn

Κεφάλαιο 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Εισαγωγή

1.1.1 Κίνητρο

Η εκμάθηση αναπαραστάσεων γράφων (Graph representation learning) είναι ένα αναδυόμενο πεδίο λόγω του εύρους των επιστημονικών πεδίων και των εφαρμογών που χρησιμοποιούνται οι γράφοι σαν τρόπος αναπαράστασης δεδομένων. Οι γράφοι συμβολίζονται συνήθως ως συνδυασμός ενός συνόλου κόμβων και ενός πίνακα γειτνίασης που υποδηλώνει την αλληλεπίδραση μεταξύ των κόμβων του γράφου. Αν και αυτή η αναπαράσταση είναι χρήσιμη για να κατανοήσουμε τη δομή του γράφου, οι παραδοσιακοί αλγόριθμοι μηχανικής μάθησης δεν μπορούν να χρησιμοποιήσουν αυτήν την αναπαράσταση, καθώς δεν μπορούν να εφαρμοστούν σε μη ευκλείδεια δεδομένα, όπως οι γράφοι.

Πολλές προσεγγίσεις έχουν υιοθετηθεί στη μάθηση αναπαράστασης κόμβων. Τα Νευρωνικά Δίκτυα Γράφων (Graph Neural Networks), εν συντομία GNN είναι τα πιο δημοφιλή λόγω της υψηλής απόδοσης που έχουν επιτύχει σε διάφορες εργασίες εκμάθησης αναπαράστασης γράφων. Για γράφους γνώσης, πολλές μέθοδοι ενσωμάτωσης κόμβων και σχέσεων (Node and relation embeddings) έχουν αναπτυχθεί μέσω εργασιών ανακατασκευής γράφων πολλαπλών σχέσεων (Knowledge graph reconstruction).

Ο θόρυβος στους γράφους είναι ένα πολύ κοινό εμπόδιο στην εξαγωγή χρήσιμων αναπαραστάσεων γράφων. Επιπλέον, η πλειοψηφία των γράφων γνώσης του πραγματικού κόσμου κατασκευάζονται μέσω της εξόρυξης γνώσης από τον Ιστό. Οι αυτοματοποιημένες μέθοδοι κατασκευής γράφων γνώσης είναι συχνά ατελείς και αυτό οδηγεί σε θορυβώδη γράφους και χαμηλές πληροφορίες ή ακόμα και θορυβώδεις σχέσεις που επηρεάζουν αρνητικά την ποιότητα του παραγόμενου γράφου γνώσης. Τα GNN, αν και παρουσιάζουν μεγάλες εκφραστικές δυνατότητες, είναι ευάλωτα στο θόρυβο. Έτσι, τα μοντέλα βαθιάς μάθησης που μπορούν να εξαγάγουν ισχυρές αναπαραστάσεις έναντι του εγγενούς θορύβου αυτών των γράφων έχουν σημαντική αξία.

1.1.2 Συμβολή

Σε αυτή τη διπλωματική εργασία, προτείνουμε τη χρήση ενσωματώσεων σχέσεων (relation embeddings) που εξάγονται από ένα μοντέλο κωδικοποιητή-αποκωδικοποιητή εκπαιδευμένο στην εργασία ολοκλήρωσης γράφου (Knowledge graph completion) σε συνδυασμό με αναπαραστάσεις κόμβων που προέρχονται από ένα GNN για την επίτευξη πιο εκφραστικών αναπαραστάσεων των κόμβων του γράφου. Τα πειράματά μας υποδηλώνουν ότι μπορούμε να επιτύχουμε βελτιωμένη απόδοση σε μια εργασία ταξινόμησης κόμβων αξιοποιώντας τόσο τις πληροφορίες δομής γειτονιάς που εξάγονται από ένα GNN όσο και τις δομικές πληροφορίες που προέρχονται από ένα μοντέλο ενσωμάτωσης γράφου γνώσης. Επιπλέον, η προτεινόμενη αρχιτεκτονική μας στοχεύει να παρέχει ισχυρές αναπαραστάσεις θορυβωδών δομών γράφων. Τα αποτελέσματα του πειραματισμού μας ενθαρρύνουν τη χρήση ενσωματώσεων γράφων γνώσης έναντι one-hot κωδικοποίησης για την αναπαράσταση των σχέσεων, καθώς η εκφραστικότητα των τελευταίων δεν επαρκεί για να διαχωριστούν οι χρήσιμες από τις θορυβώδεις σχέσεις. Επιπλέον, τα ευρήματά μας δείχνουν ότι το προτεινόμενο μοντέλο έχει αμελητέα απώλεια απόδοσης σε γράφους που επηρεάζονται από σχέσεις εντελώς θορυβώδεις. Τέλος, η πτώση απόδοσης περιορίζεται όταν ο θόρυβος επηρεάζει περιορισμένο αριθμό σχέσεων.

1.2 Θεωρητικό υπόβαθρο και σχετικές εργασίες

1.2.1 Γράφοι σαν μέθοδος αναπαράστασης δεδομένων

Οι γράφοι είναι μια διαδεδομένη δομή δεδομένων για τη μοντελοποίηση πολύπλοκων συστημάτων του πραγματικού κόσμου. Χρησιμοποιούνται ευρέως ως μέθοδος αναπαράστασης σε διάφορες εφαρμογές. Στην επιστήμη υπολογιστών παίζουν πολύ σημαντικό ρόλο και έχουν αποτελέσει αντικείμενο μελέτης επί πολλά χρόνια. Το κύριο πλεονέκτημα των γράφων είναι το γεγονός ότι αυτή η συγκεκριμένη δομή μπορεί να μοντελοποιήσει αντικείμενα του πραγματικού κόσμου μαζί με τις αλληλεπιδράσεις μεταξύ τους. Για το λόγο αυτό οι γράφοι επιλέγονται ως μέθοδος αναπαράστασης σε μια πληθώρα εφαρμογών, όπως τα κοινωνικά δίκτυα, συστήματα συστάσεων, μοριακά γραφήματα, δίκτυα αλληλεπίδρασης πρωτεϊνών ή φαρμάκων κλπ [14]. Ως εκ τούτου έχει εκδηλωθεί επιστημονικό ενδιαφέρον γύρω από την εφαρμογή αλγορίθμων μηχανικής και βαθιάς μάθησης σε δεδομένα σε μορφή γράφων, με τα ΓΝΝ να αποτελούν το δημείο αναφοράς [17].

Πολλά από τα σύνολα δεδομένων του πραγματικού κόσμου δεν μπορούν να αναπαρασταθούν με απλούς γράφους που περιέχουν κόμβους και ακμές ενός μόνο τύπου. Γράφοι που περιέχουν κόμβους και ακμές διαφορετικών τύπων ονομάζονται ετερογενείς [38]. Μια ειδική περίπτωση ετερογενών γράφων είναι οι γράφοι γνώσης, οι οποίοι χρησιμοποιούν την δομή του γράφου για την αναπαράσταση της ανθρώπινης γνώσης, αναπαριστώντας γεγονότα με τη μορφή οντοτήτων, σχέσεων και σημασιολογική περιγραφή αυτών (ετικέτες, χαρακτηριστικά και τύποι οντοτήτων) [18].

1.2.2 Εκμάθηση αναπαράστασης γράφων

Ο σκοπός της εκμάθησης αναπαράστασης γραφήματος είναι η εξαγωγή αναπαραστάσεων χαμηλών διαστάσεων από τους κόμβους ενός γράφου. Αυτό έχει μεγάλη σημασία γιατί η ποιότητα και η εκφραστικότητα των διανυσμάτων αναπαράστασης του γράφου είναι ο πιο σημαντικός παράγοντας για την απόδοση των μοντέλων μηχανικής μάθησης σε διάφορες εργασίες [24].

Αυτό που διαφέρει στην περίπτωση των γράφων είναι το γεγονός ότι οι συνήθεις κατηγορίες προβλημάτων επιβλεπόμενης και μη επιβλεπόμενης μάθησης δεν είναι τόσο συχνές λόγω των ιδιαίτερων χαρακτηριστικών των γράφων ως αναπαράσταση συνόλων δεδομένων [57]. Επιπλέον, η ύπαρξη ακμών ακτός από κόμβους προσφέρονται για εξειδικευμένες εργασίες μηχανικής μάθησης, όπως η πρόβλεψη ακμών [29]. Γενικά υπάρχουν τρία πλαίσια για εφαρμογές μηχανικής μάθησης σε γράφους: σε επίπεδο κόμβου (ταξινόμηση κόμβων), σε επίπεδο ακμής (ταξινόμηση και πρόβλεψη ακμών) και σε επίπεδο γράφου (ταξινόμηση γράφου) [46].

Η ταξινόμηση κόμβων είναι ίσως η πιο κοινή εργασία σε δεδομένα με τη μορφή γράφων. Δεδομένου ενός γράφου $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ και ετικέτες σε ένα υποσύνολο κόμβων $V_{train} \in \mathcal{V}$, ο στόχος στην ταξινόμηση κόμβων είναι η πρόβλεψη των ετικετών των μη επισημασμένων κόμβων.

Το πρόβλημα της ταξινόμησης κόμβων είναι ένα πρόβλημα επιβλεπόμενης μάθησης. Στην επιβλεπόμενη μάθηση, ο αλγόριθμος μάθησης παρατηρεί ένα επισημασμένο σύνολο δεδομένων (σύνολο εκπαίδευσης) που αποτελείται από ζεύγη (χαρακτηριστικό, ετικέτα) και συμβολίζονται με $\{(x_1, y_1), ..., (x_n, y_n)\}$. Ο στόχος είναι να προβλέψουμε την ετικέτα y για κάθε νέα μη επισημασμένη είσοδο x. Ωστόσο, οι ετικέτες y είναι συχνά δύσκολο, ακριβό και αργό να αποκτηθούν. Αυτό ακριβώς αντιμετωπίζει η ημιεπιβλεπόμενη μάθηση, κατά την οποία λαμβάνοντας υπόψη ένα σχετικά μικρό σύνολο δεδομένων με ετικέτα $\{(x, y)\}$ και ένα μεγάλο σύνολο δεδομένων χωρίς ετικέτα $\{x\}$, μπορεί κανείς να μάθει από και τα δύο για ταξινόμηση.

1.2.3 Νευρωνικά δίκτυα γράφων

Η επικρατούσα αρχιτεκτονική βαθιάς μάθησης για δεδομένα σε μορφή γράφων είναι τα Νευρωνικά Δίκτυα Γράφων (*Graph Neural Networks*) [35]. Ο στόχος των GNNs είναι να μάθει μια αναπαράσταση $h_u \in \mathbb{R}$ για κάθε κόμβο u του γράφου, που εμπεριέχει πληροφορίες από τη γειτονιά του κάθε κόμβου. Για να μάθει αυτές τις αναπαραστάσεις, το GNN χρησιμοποιεί ένα γενικό πλαίσιο, που ονομάζεται πλαίσιο μεταφοράς νευρωνικών μηνυμάτων (*neural message passing framework*) [11]. Στην αρχή, κάθε κόμβος u αρχικοποιείται με μια κάποια αναπαράσταση h_u. Σε κάθε επανάληψη, ενημερώνουμε την κατάσταση του κόμβου, με βάση την προηγούμενη κατάστασή του και τις αναπαραστάσεις των γειτόνων του. Η διαδικασία ανανέωσης των αναπαραστάσεων των κόμβων γίνεται μέσω μίας διαδικασίας ανταλλαγής και συνάθροισης μηνυμάτων μεταξύ των γειτονικών κόμβων, η οποία μπορεί να περιγραφεί με τις παρακάτω πράξεις:

$$m_u^{k+1} = AGGREGATE(h_u^k, \forall u \in N(u)) \tag{1.1}$$

$$h_u^{k+1} = UPDATE(h_u^k, m_u^{k+1}) \tag{1.2}$$

όπου $N(u)$ είναι η γειτονιά του κόμβου u , $AGGREGATE$ και $UPDATE$ είναι διαφορήσιμες συναρτήσεις, που συνήθως προσεγγίζονται από νευρωνικά δίκτυα [16] , m_u^{k+1} είναι το συναθροισμένο μήνυμα από τους γείτονες στην επανάληψη $k+1$ και h_u^{k+1} είναι η αναπαράσταση του κόμβου u στο $k+1$ επανάληψη. Η συνάρτηση $AGGREGATE$ θα πρέπει να είναι αμετάβλητη μετάθεση. Εκτός από το μήνυμα και τη συνάθροιση, ένα επίπεδο GNN εφαρμόζει επιπλέον μη γραμμικότητα στην ενεργοποίηση του επιπέδου για να κάνει δυνατή την προσέγγιση των μη γραμμικών συναρτήσεων στο δίκτυο. Οι πιο κοινές συναρτήσεις ενεργοποίησης είναι η σιγμοειδής συνάρτηση και η συνάρτηση $ReLU$. Το σχήμα 3.5 είναι μια απεικόνιση της προαναφερθείσας διαδικασίας.

Συνελικτικό δίκτυο γράφων GCN

Το Συνελικτικό Δίκτυο Γράφων (*Graph Convolutional Network*) [26] είναι το πιο ευρέως χρησιμοποιούμενο GNN. Ακολουθεί την προσέγγιση μετάδοσης νευρωνικών μηνυμάτων. Κάθε συνελικτικό επίπεδο l αποτελείται από έναν πίνακα βάρους $\mathbf{W}^{(l)}$. Για κάθε κόμβο u, παίρνει ως είσοδο την αναπαράσταση του προηγούμενου επιπέδου (δηλαδή, $\mathbf{h}_u^{(l-1)}$), τον πολλαπλασιάζει με τον πίνακα βάρους $\mathbf{W}^{(l)}$ για να δημιουργήσετε τη νέα αναπαράσταση $\mathbf{h}_u^{(l)}$. Αυτό αποτελεί το μήνυμα κάθε γειτονικού κόμβου του u στον συγκεκριμένο κόμβο. Στη συνέχεια, τα μηνύματα συγκεντρώνονται και εφαρμόζεται επίσης μια μη γραμμική συνάρτηση ενεργοποίησης.

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right) \tag{1.3}$$

Δίκτυο GraphSAGE

Τα περισσότερα GNNs δεν γενικεύονται σε μη ορατά δεδομένα, καθώς δημιουργούν τις ενσωματώσεις βελτιστοποιώντας τις προβλέψεις σε κόμβους σε ένα σταθερό γράφημα. Το *GraphSAGE* [13] αποτελεί ένα γενικό πλαίσιο για την ενσωμάτωση επαγωγικών κόμβων. Σε αντίθεση με την προσέγγιση ενσωμάτωσης του GCN, το GraphSAGE δεν συγκεντρώνει απευθείας τα μηνύματα από τη γειτονιά του κόμβου v, αλλά αρχικά αντλεί μια αναπαράσταση της γειτονιάς συγκεντρώνοντας τις αναπαραστάσεις των γειτόνων και στη συνέχεια συνενώνει την αναπαράσταση του κόμβου v από το προηγούμενο επίπεδο $\mathbf{h}_u^{(l-1)}$ με την προαναφερθείσα αναπαράσταση γειτονιάς (Σχήμα 3.7).

$$\mathbf{h}_v^{(l)} \leftarrow \sigma\left(\mathbf{W}^{(l)} \cdot CONCAT(\mathbf{h}_v^{(l-1)}, AGG(\{\mathbf{h}_u^{l-1}, \forall u \in N(v)\}))\right) \tag{1.4}$$

Δίκτυο προσοχής γράφων (GAT)

Στα παραπάνω νευρωνικά δίκτυα γράφων, η συμβολή κάθε κόμβου στο σχήμα συνάθροισης υπαγορεύεται από τον βαθμό του. Και στο GCN και στο GraphSAGE, υπάρχει ένας παράγοντας στάθμισης $\alpha_{vu} = \frac{1}{|N(v)|}$ του μηνύματος του κόμβου u προς τον κόμβο v . Αυτό σημαίνει ότι όλοι οι γείτονες $u \in N(v)$ είναι εξίσου σημαντικοί για τον κόμβο v. Τα δίκτυα προσοχής γραφήματος (*Graph Attention Networks*) [41] αντιμετωπίζουν αυτόν τον

περιορισμό εφαρμόζοντας διαφορετικά βάρη εστιάζοντας στα σημαντικά μέρη των δεδομένων εισόδου (Σχήμα 3.8) και εξαφανίζοντας τα υπόλοιπα μη πληροφοριακά μέρη. Με άλλα λόγια, στην περίπτωση του GAT, δεν είναι όλοι οι γείτονες του κόμβου εξίσου σημαντικοί. Η σημασία του κάθε γείτονα εξαρτάται αποτελεί εκπαιδεύσιμη παράμετρο και υπολογίζεται κατά την εκπαίδευση.

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right) \tag{1.5}$$

Επέκταση GNNs σε πολυσχεσιακούς γράφους

Το πλαίσιο των GNNs μπορεί να επεκταθεί για να αντιμετωπίσει ετερογενή γραφήματα. Στο GCN, το βάρος $\mathbf{W}^{(l)}$ μοιράζεται σε όλες τις ακμές στο επίπεδο l. Για τον χειρισμό πολλαπλών σχέσεων, η αρχιτεκτονική σχεσιασκού GCN (*Relational GCN*) εισάγει ξεχωριστούς πίνακες βάρους για κάθε τύπο σχέσης [36] και μόνο οι ακμές του ίδιου τύπου σχέσης r σχετίζονται με το ίδιο βάρος προβολής $\mathbf{W}_r^{(l)}$ (Εικόνα 3.9). Στη συνέχεια, ακολουθώντας την προσέγγιση μετάδοσης νευρωνικών μηνυμάτων γίνεται η ανανέωση της αναπαράστασης κάθε κόμβου για κάθε σχέση του πολυσχεσιακού γράφου.

$$h_v^{(l+1)} = \sigma\left(W_0^{(l)} h_v^{(l)} + \sum_{r \in R} \sum_{u \in N(v)_r} \frac{1}{c_{v,r}} W_r^{(l)} h_u^{(l)}\right) \tag{1.6}$$

Το πρόβλημα που προκύπτει κατά την άμεση εφαρμογή της παραπάνω εξίσωσης είναι η ταχεία αύξηση του αριθμού των παραμέτρων, ειδικά με γράφους με μεγάλο πλήθος σχέσεων. Αυτό οδηγεί σε υψηλό υπολογιστικό κόστος και κίνδυνο υπερπροσαρμογής (overfitting). Για να αποφευχθεί αυτό προτείνεται η χρήση της αποσύνθεσης βάσης κατά τη διάρκεια της εκπαίδευσης [36].

1.2.4 Ενσωματώσεις γράφων γνώσης

Τα γραφήματα γνώσης, ως μια μορφή δομημένης ανθρώπινης γνώσης έχουν τραβήξει μεγάλη ερευνητική προσοχή τα τελευταία χρόνια. Ως αποτέλεσμα, η εκμάθηση αναπαραστάσεων σε γράφους γνώσης είναι ένα αναδυόμενο πεδίο που έχει παράγει σημαντικά αποτελέσματα. Η εκμάθηση αναπαραστάσεων γράφου γνώσης είναι η διαδικασία προβολής γράφων γνώσης σε έναν συνεχή διανυσματικό χώρο χαμηλών διαστάσεων [48] (Εικόνα 3.10). Οι μέθοδοι ενσωμάτωσης που χρησιμοποιούνται σε αυτή τη μελέτη ακολουθούν την προσέγγιση κωδικοποιητή - αποκωδικοποιητή για την εκπαίδευση των ενσωματώσεων και ανήκουν στην κατηγορία των ρηχών ενσωματώσεων (*shallow graph embedding*) [14].

Μοντέλο κωδικοποιητή - αποκωδικοποιητή

Στο πλαίσιο κωδικοποιητή - αποκωδικοποιητή η διαδικασία δημιουργίας ενσωματώσεων για τους κόμβους και τις ακμές του γραφήματος χωρίζεται σε δύο λειτουργίες. Πρώτον, ένας κωδικοποιητής αντιστοιχίζει κάθε κόμβο του γραφήματος εισόδου σε ένα διάνυσμα χαμηλής

διάστασης. Μετά από αυτό, ένας αποκωδικοποιητής χρησιμοποιεί το κωδικοποιημένο διάνυσμα για να ανακατασκευάσει πληροφορίες σχετικά με τη γειτονιά κάθε κόμβου στο αρχικό γράφημα [14].

Οι αποκωδικοποιητές χωρίζονται στις παρακάτω κατηγορίες:

- **Μεταφραστικοί**: Αντιπροσωπεύουν σχέσεις ως μεταφράσεις στον χώρο ενσωμάτωσης. Σε αυτήν τη μεθοδολογία ενσωμάτωσης, κάθε σχέση αναπαρίσταται χρησιμοποιώντας μια ενσωμάτωση διαστάσεων d. Η πιθανότητα ακμής είναι ανάλογη με την απόσταση μεταξύ της ενσωμάτωσης του κόμβου της κεφαλής και του κόμβου της ουράς, αφού μεταφραστούν οι κόμβοι κεφαλής και ουράς σύμφωνα με το διανυσματικό χώρο της συγκεκριμένης σχέσης. Σχετικά μοντέλα: TransE, TransR, TransH
- **Πολυ-γραμμικοί εσωτερικού γινομένου**: Στη μεθοδολογία αυτή ο αποκωδικοποιητής εφαρμόζει την πράξη του εσωτερικού γινομένου στα διανύσματα που δημιούργησε ο κωδικοποιητής (DistMult). Βασικός περιορισμός της προσέγγισης αυτής είναι η αδυναμία διαχείρισης αντι-συμμετρικών σχέσεων. Αυτό αντιμετωπίζεται με την χρησιμοποίηση μιγαδικών ενσωματώσεων των στοιχείων του γράφου (ComplEx, RotatE)

Τα προαναφερθέντα πολυσχεσιακά μοντέλα αντιμετωπίζουν το έργο της ενσωμάτωσης γράφων γνώσης χρησιμοποιώντας διαφορετικές συναρτήσεις βαθμολογίας στο τμήμα του αποκωδικοποιητή τους. Η επιλογή της συνάρτησης βαθμολογίας και ο διανυσματικός χώρος των δημιουργούμενων ενσωματώσεων είναι οι κύριοι παράγοντες που καθορίζουν την ικανότητά τους να αναπαριστούν συγκεκριμένους τύπους σχέσεων. Ως αποτέλεσμα, αυτοί οι αποκωδικοποιητές μπορούν να χαρακτηριστούν από την ικανότητά τους να αναπαριστούν τέτοιους τύπους σχέσεων (Πίνακας 3.1).

1.2.5 Θόρυβος σε γράφους

Τα ελλιπή, ανούσια, παραμορφωμένα ή κατεστραμμένα δεδομένα σε σύνολα δεδομένων είναι γνωστά ως θόρυβος. Ο θόρυβος μπορεί να έχει σημαντικό αντίκτυπο στη συνολική απόδοση ενός μοντέλου μηχανικής μάθησης [15]. Ο θόρυβος παρατηρείται συνήθως σε διάφορα δομημένα ή αδόμητα δεδομένα και τα δεδομένα που αναπαρίστανται ως γράφοι δεν αποτελούν εξαίρεση.

Ο θόρυβος στους γράφους μπορεί να αποδοθεί σε διάφορους παράγοντες. Επιπλέον, η μορφή με την οποία μπορεί να παρατηρηθεί ο θόρυβος στα γραφήματα μπορεί να ποικίλλει. Μπορεί να είναι το αποτέλεσμα μιας επιδιωκόμενης προσπάθειας αλλαγής της δομής του δικτύου (π.χ. επιθέσεις αντιπάλου, bots σε κοινωνικά δίκτυα) ή μπορεί να προέρχεται από ατέλειες αυτοματοποιημένων μεθόδων κατασκευής γραφημάτων [30] (π.χ. αυτόματα κατασκευασμένα γραφήματα γνώσης). Οι δύο κύριες μορφές θορύβου είναι: i) Ελλιπείς ή περιττές ακμές μεταξύ κόμβων (κόμβοι διαφορετικής κατανομής χαρακτηριστικών, κλάσεις, κοινότητες κ.λπ.). ii) Δηλητηριασμένοι κόμβοι (κόμβοι με διαταραγμένα χαρακτηριστικά ή ετικέτες) προσαρτημένοι σε κανονικούς κόμβους που μπορούν να βλάψουν τις μεθόδους συνάθροισης γειτονιάς για εξαγωγή αναπαραστάσεων [49].

Παρά το γεγονός ότι τα νευρωνικά δίκτυα γραφικών έχουν επιδείξει μεγάλη επιτυχία στη μοντελοποίηση δεδομένων με τη μορφή γράφων, έχουν δείξει αρκετά αξιοσημείωτη ευπάθεια σε διάφορους τύπους θορύβου. Οι ακμές θορύβου, καθώς και οι δηλητηριασμένοι κόμβοι μπορεί να οδηγήσουν σε σημαντική υποβάθμιση της απόδοσης του GNN. Αυτό οφείλεται κυρίως στη φύση του πλαισίου μετάδοσης νευρωνικών μηνυμάτων που χρησιμοποιούν αυτά τα νευρωνικά δίκτυα [7]. Οι ακμές θορύβου συνήθως συνδέουν κόμβους διαφορετικών κλάσεων ή με διαφορετικά χαρακτηριστικά, επομένως η συγκέντρωση πληροφοριών γειτονικών κόμβων διαδίδει σφάλματα, αναμιγνύει χρήσιμες πληροφορίες με θόρυβο και τελικά οδηγεί σε κακές αναπαραστάσεις των κόμβων του γράφου [19]. Προκειμένου να αμυνθεί το νευρωνικό δίκτυο έναντι αυτών των ειδών θορύβου, έχουν προταθεί διαφορετικά αντίμετρα, τα οποία επικεντρώνονται στον χειρισμό του θορύβου τόσο κατά τη διάρκεια της εκπαίδευσης όσο και της αξιολόγησης.

1.3 Μεθοδολογία

1.3.1 Σενάρια θορύβου

Θα ασχοληθούμε με την ύπαρξη θορύβου σε γράφους που εμφανίζεται με τη μορφή λανθασμένων ακμών και θα ταξινομήσουμε την ύπαρξη θορύβου στις ακόλουθες κατηγορίες:

- **Θορυβώδεις σχέσεις**: Ορισμένοι γράφοι μπορεί να περιέχουν σχέσεις με ακμές που συνδέουν αποκλειστικά μη σχετικούς κόμβους. Αυτή η περίπτωση είναι πιο συνηθισμένη σε αυτόματα κατασκευασμένους γράφους γνώσης, εάν η μέθοδος εξαγωγής σχέσεων είναι λανθασμένη ή μια σχέση δεν παρέχει χρήσιμες πληροφορίες σχετικά με την ομοιότητα των κόμβων από προεπιλογή.

- **Ακμές θορύβου σε χρήσιμες σχέσεις**: Σε αυτή τη ρύθμιση, εστιάζουμε στον αντίκτυπο των θορυβωδών ακμών που ανήκουν σε μια σχέση που παρέχει πληροφορία. Αυτό μπορεί να είναι αποτέλεσμα συγκεκριμένων επιθέσεων στη δομή του γράφου ή λόγω ατελειών μιας αυτοματοποιημένης μεθόδου κατασκευής γράφων που δημιουργεί περιττές ακμές. Αυτές οι ακμές βλάπτουν το σχήμα συνάθροισης γειτόνων, επειδή οι μη σχετικοί γείτονες συμβάλλουν στην τελική αναπαράσταση κάθε κόμβου.

1.3.2 Μοντέλα αναφοράς

Προκειμένου να αξιολογήσουμε την απόδοση της προτεινόμενης αρχιτεκτονικής μας, επιλέξαμε δύο αρχιτεκτονικές GNN, οι οποίες είναι από τα πιο συχνά χρησιμοποιούμενα μοντέλα στη σχετική βιβλιογραφία για διάφορες εργασίες εκμάθησης αναπαράστασης γραφημάτων. Η επιλογή αυτών των δύο μοντέλων, εκτός από την παροχή ενός μέτρου σύγκρισης για την αρχιτεκτονική μας, σχετίζεται με το γεγονός ότι αυτές οι δύο αρχιτεκτονικές χρησιμοποιούν δύο διαφορετικούς μηχανισμούς που θα μπορούσαν ενδεχομένως να παρέχουν ανθεκτικότητα έναντι του θορύβου. Τα μοντέλα αυτά είναι:

- **GCN / GraphSAGE**: Τα δύο αυτά μοντέλα διατηρούν διαφορετικό πίνακα βαρών για κάθε σχέση. Ο σχεδιασμός αυτός θα μπορούσε να αγνοήσει σχέσεις χαμηλής πληροφορίας ή ακόμα και θορύβου προσαρμόζοντας τα αντίστοιχα βάρη κάθε σχέσης.

- **GAT**: Το GAT διατηρεί επίσης διαφορετικό πίνακα βαρών για κάθε σχέση και επιπλέον έχει και ένα βάρος "προσοχής" για κάθε ακμή. θεωρητικά παρέχει πρόσθετη ασφάλεια έναντι ακμών θορύβου, καθώς το μοντέλο διαισθητικά μπορεί να αποδώσει ελάχιστα βάρη προσοχής σε αυτές τις ακμές.

1.3.3 Προτεινόμενη αρχιτεκτονική

Περιγραφή αρχιτεκτονικής (SRGCN)

Το μοντέλο SRGCN αποτελείται από τρία κύρια στοιχεία και η διαδικασία μοντελοποίησης απεικονίζεται στο σχήμα 4.1. Το πρώτο είναι ένα συνελικτικό δίκτυο γραφημάτων. Το μοντέλο περιέχει ένα GCN για κάθε μία από τις σχέσεις του γραφήματος εισόδου. Κάθε GCN χρησιμοποιεί ως είσοδο τον αρχικό γράφο, αλλά με έναν μόνο τύπο σχέσης, και εξάγεται μια αναπαράσταση για κάθε κόμβο. Αυτή η ενσωμάτωση αντιπροσωπεύει τον κόμβο για αυτήν τη συγκεκριμένη σχέση. Το δεύτερο στοιχείο είναι ένα ρηχό μοντέλο ενσωμάτωσης γραφήματος γνώσης. Το μοντέλο ενσωμάτωσης γράφου δημιουργεί μια αναπαράσταση (δηλαδή μια ενσωμάτωση) για κάθε σχέση του αρχικού γράφου. Στη συνέχεια, για κάθε κόμβο, η αναπαράστασή του ανά σχέση (από το GCN) και η αντίστοιχη αναπαράσταση σχέσης (από το μοντέλο ενσωμάτωσης γραφήματος) συνδέονται, δημιουργώντας μια μοναδική αναπαράσταση για κάθε διαφορετική σχέση για τον κόμβο. Τέλος, αυτές οι αναπαραστάσεις συναθροίζονται και τροφοδοτούνται ως είσοδος σε ένα νευρωνικό δίκτυο εμπρόσθιας τροφοδοσίας για την ταξινόμηση. Στην περίπτωσή μας, επιλέξαμε τη συνένωση ως συνάρτηση συνάθροισης. Η ιδέα είναι να μην αναμειγνύονται οι πληροφορίες για διαφορετικές σχέσεις σε αυτό το στάδιο. Έτσι, το τελικό MLP πάνω από αυτές τις συγκεντρωτικές ενσωματώσεις θα μπορεί να χρησιμοποιήσει τις διακριτές αναπαραστάσεις που δημιουργούνται από κάθε σχέση, για να εκτελέσει την ταξινόμηση κόμβων. Τέλος, η διαδικασία ενσωμάτωσης γράφου που χρησιμοποιείται είναι RotatE λόγω της υπεροχής της απόδοσής του.

Κίνητρο της μεθοδολογίας

Η κύρια ιδέα της προτεινόμενης αρχιτεκτονικής είναι να αξιοποιήσει τόσο την ικανότητα της αρχιτεκτονικής GNN να καταγράφει εξαρτήσεις σε επίπεδο γειτονιάς όσο και τις πληροφορίες σε επίπεδο σχέσης για κάθε σχέση ενός πολυσχεσιακού γράφου. Το SRGCN βασίζεται στο GNN τμήμα για την εξαγωγή αναπαραστάσεων κόμβων. Τα GNN, μέσω του πλαισίου μεταφοράς νευρωνικών μηνυμάτων, μπορούν να διαδώσουν πληροφορίες σχετικά με τους γειτονικούς κόμβους του κόμβου-στόχου. Παρόλα αυτά, οι τυπικές αρχιτεκτονικές GNN είναι ευάλωτες στον δομικό θόρυβο παρά τη μεγάλη τους ισχύ στη μοντελοποίηση γράφων [19]. Το κίνητρό μας ήταν να εμπλουτίσουμε την τελική αναπαράσταση εισάγοντας γενικές πληροφορίες για ολόκληρο το γράφο και για κάθε σχέση ειδικότερα. Αυτό συμβαίνει μέσω ενός μοντέλου

ενσωμάτωσης γράφου γνώσης. Ο συνδυασμός των ενσωματώσεων κόμβων που προέρχονται από το σχήμα συνάθροισης γειτονιάς της αρχιτεκτονικής GCN με τις ενσωματώσεις σχέσεων ενός μοντέλου ενσωμάτωσης γράφων μπορεί να οδηγήσει σε μια εκφραστική αναπαράσταση που διακρίνει τις χρήσιμες σχέσεις από τις χαμηλής πληροφορίες ή θορυβώδεις σχέσεις. Για να επαληθεύσουμε την τελευταία υπόθεση, πραγματοποιήσαμε πειράματα για να αξιολογήσουμε τη χρήση μιας μεθόδου ενσωμάτωσης γράφου γνώσης, όπως το RotatE, συγκρίνοντας τα αποτελέσματα με μια "πιο ρηχή", όπως μια one-hot αναπαράσταση σχέσων . Τέλος, θέλουμε να περιορίσουμε την εξάρτηση του μοντέλου μας από τα αρχικά χαρακτηριστικά κόμβου. Στην περίπτωση πολύ θορυβωδών γράφων, τα αρχικά διανύσματα χαρακτηριστικών τείνουν να παίζουν πιο σημαντικό ρόλο, καθώς η αξία των δομικών πληροφοριών μειώνεται από το θόρυβο. Επιπλέον, τα χαρακτηριστικά κόμβου χρησιμοποιούνται για την αναγνώριση των ακμών θορύβου συγκρίνοντας την ομοιότητα χαρακτηριστικών του συνδεδεμένου κόμβου [7]. Στην προσέγγισή μας, προσπαθούμε να αξιοποιήσουμε τη δομή του γράφου όσο το δυνατόν περισσότερο και να βελτιώσουμε την απόδοση εντοπίζοντας τις σχέσεις που επηρεάζονται από το θόρυβο.

Πολυπλοκότητα μοντέλου

Για μια διεξοδική σύγκριση των μοντέλων, είναι λογικό να λαμβάνεται υπόψη και η πολυπλοκότητά τους. Το SRGCN και το GCN περιλαμβάνουν έναν πίνακα βαρών $\mathbf{W}_r \in \mathbb{R}^{F \times H}$, όπου F είναι ο αριθμός των χαρακτηριστικών εισόδου και H είναι το κρυφό μέγεθος ενσωμάτωσης . Και υπάρχει ένα ξεχωριστό W_r για κάθε σχέση r και σε κάθε επίπεδο (l). Λαμβάνοντας υπόψη ένα μόνο επίπεδο και για τα δύο μοντέλα, η πολυπλοκότητα της μνήμης θα είναι τότε $\mathcal{O}_{GNN} = \mathcal{O}(RFH)$, όπου R είναι ο αριθμός των σχέσεων και F, H όπως και πριν. Έτσι, όσον αφορά το αρχικό τμήμα αναπαράστασης κόμβου που βασίζεται σε GNN, και τα δύο μοντέλα έχουν τον ίδιο αριθμό παραμέτρων.

Για το μοντέλο SRGCN συνδυάζουμε επίσης αυτές τις ενσωματώσεις που βασίζονται σε GNN με την ενσωμάτωση παράγωγης σχέσης, για κάθε σχέση. Επομένως, πρέπει να προσθέσουμε το γενικό κόστος προσαρμογής του αντίστοιχου μοντέλου ενσωμάτωσης γράφου. Η επιλογή μας ήταν το RotatE που έχει υπολογιστική πολυπλοκότητα $\mathcal{O}_{GE} = \mathcal{O}(2VH_{GE} + 2RH_{GE})$ [52], όπου το V είναι τον αριθμό των κόμβων στο γράφημα και H_{GE} το επιλεγμένο μέγεθος ενσωμάτωσης για το μοντέλο ενσωμάτωσης γράφου. Έχοντας τις ενσωματώσεις σχέσεων από το μοντέλο ενσωμάτωσης γραφήματος, μπορούμε να συνδέσουμε κάθε ενσωμάτωση που βασίζεται σε GNN για συγκεκριμένη σχέση με την αντίστοιχη ενσωμάτωση σχέσεων. Κάθε ένα από αυτά έχει μέγεθος $H_{GNN} + H_{GE}$. Το μοντέλο GCN δεν έχει πρόσθετη πολυπλοκότητα σε αυτό το βήμα.

Τέλος, έχουμε την πολυπλοκότητα μνήμης του τελικού επιπέδου MLP. Η συνάρτηση συνάθροισης του GCN είναι το άθροισμα. Έτσι, η διάσταση εισόδου του πυκνού επιπέδου ταξινόμησης είναι H_{GNN} και η πολυπλοκότητα του MLP είναι $\mathcal{O}(H_{GNN})$. Το μοντέλο SRGCN χρησιμοποιεί τη συνάρτηση συνένωσης ως τελεστή συνάθροισης. Έτσι, η διάσταση εισόδου του πυκνού επιπέδου ταξινόμησης είναι $R(H_{GNN} + H_{GE})$. Συνολικά, η σύγκρισή

τους ως προς τις παραμέτρους φαίνεται στον Πίνακα 4.1.

1.4 Πειράματα και αποτελέσματα

1.4.1 Μετρική αξιολόγησης

Τα μοντέλα εκπαιδεύονται στην εργασία ταξινόμησης κόμβων και αξιολογούνται σύμφωνα με το *F1-score* που επιτυγχάνουν στην πρόβλεψη της κλάσης των μη επισημασμένων κόμβων. Το F1-score υπολογίζεται χρησιμοποιώντας τον ακόλουθο τύπο:

$$\mathrm{F1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{1.7}$$

1.4.2 Σύνολα δεδομένων

Υιοθετούμε δύο ευρέως χρησιμοποιούμενα σύνολα δεδομένων ετερογενών γράφων από διαφορετικούς τομείς για να αξιολογήσουμε την απόδοση των προτεινόμενων και μοντέλων αναφοράς. Μια επισκόπηση της δομής κάθε συνόλου δεδομένων γραφήματος φαίνεται στο Σχήμα 5.1. Μια περίληψη των οντοτήτων και των σχέσεων των συνόλων δεδομένων γραφημάτων παρουσιάζεται στον Πίνακα 5.1. Τα σύνολα αυτά είναι τα IMDb και DBLP. Πρόκειται για ετερογενείς γράφους κινηματογραφικών ταινιών και βιβλιογραφικών αναφορών.

1.4.3 Προβλεπτική ικανότητα προτεινόμενου μοντέλου

Από τα αποτελέσματα (Πίνακες 5.2 και 5.3) βλέπουμε ότι το προτεινόμενο μοντέλο υπερέχει σε απόδοση σε σχέση με τα μοντέλα αναφοράς έστω και σε ένα μικρό βαθμό. Στην περίπτωση του συνόλου δεδομένων IMDB βλέπουμε ότι το GAT είναι σημαντικά χειρότερο από το GraphSAGE, κάτι που δεν συνέβει στην περίπτωση του DBLP. Το MLP που δεν λαμβάνει υπόψη τη δομή του γράφου καθόλου υστερεί σημαντικά σε σχέση με τα εξεταζόμενα GNN. Από τους χρόνους εκπαίδευσης βλέπουμε ότι το προτεινόμενο μοντέλο καθώς και το GAT χρειάζονται περίπου το διπλάσιο χρόνο από το GraphSAGE, παρόλα αυτά οι χρόνοι εκπαίδευσης είναι σχετικά συγκρίσιμοι παρά το γεγονός ότι το μοντέλο μας έχει μεγαλύτερη χωρική πολυπλοκότητα λόγω των παραγόμενων ενσωματώσεων σχέσων του γράφου. Από τα παραπάνω θεωρούμε ότι το προτεινόμενο μοντέλο είναι ανταγωνιστικό και μπορεί να χρησιμοποιηθεί σαν μοντέλο γενικού σκοπού ανεξάρτητα από την παρουσία θορύβου στο σύνολο δεδομένων.

1.4.4 Ανθεκτικότητα στο θόρυβο

Εκτός από την ικανότητα του SRGCN να αποδίδει καλύτερα στα σύνολα δεδομένων αναφοράς, θέλαμε επίσης να διερευνήσουμε πώς αυτή η αρχιτεκτονική θα μπορούσε να παρέχει σταθερές αναπαραστάσεις σε γραφήματα με μεγάλες ποσότητες ακμών θορύβου. Η διαίσθησή μας είναι ότι η ενσωμάτωση πληροφοριών για ολόκληρο το σύνολο των ακμών σε μια συγκεκριμένη σχέση μπορεί να βοηθήσει το συνολικό μοντέλο να αναγνωρίσει τις θορυβώδεις

σχέσεις και να επικεντρωθεί στις αναπαραστάσεις κόμβων που προέρχονται από τις χρήσιμες σχέσεις. Για να επαληθεύσουμε την υπόθεσή μας, πραγματοποιήσαμε διάφορα πειράματα σε διαφορετικά σενάρια θορύβου. Για να επιταχύνουμε τον πειραματισμό και να προσαρμοστούμε στις ανάγκες κάθε πειράματος, χωρίζουμε την πειραματική διαδικασία σε δύο γύρους.

Πειράματα σε συνθετικά δεδομένα

Στον πρώτο γύρο πειραματισμού σχετικά με την επίδραση των άκρων θορύβου, πραγματοποιήθηκαν πειράματα σε συνθετικά δεδομένα. Τα συνθετικά δεδομένα μπορούν να μιμηθούν δεδομένα λειτουργίας ή παραγωγής και να βοηθήσουν στην εκπαίδευση μοντέλων μηχανικής και βαθιάς εκμάθησης [32]. Ο σκοπός αυτής της πειραματικής εγκατάστασης είναι η διεξαγωγή πειραμάτων με σκοπό την απόκτηση γνώσεων σχετικά με τις επιπτώσεις του θορύβου, τη συμπεριφορά των αρχιτεκτονικών GNN και την επιτάχυνση του πειραματισμού, διατηρώντας παράλληλα τον έλεγχο επί του συνόλου δεδομένων λόγω του μικρού του μεγέθους.

Το συνθετικό σύνολο δεδομένων αποτελείται από 50 κόμβους, όλοι του ίδιου τύπου. Αυτοί οι κόμβοι ανήκουν σε 2 κλάσεις, με κάθε κλάση να έχει ίσο αριθμό κόμβων (25). Ο γράφος περιέχει ένα σύνολο 16 επισημασμένων κόμβων, ισόποσα κατανεμημένους στις 2 κλάσεις (δηλαδή 8 η καθεμία). Οι υπόλοιποι 34 μη επισημασμένοι κόμβοι αποτελούν το δοκιμαστικό σύνολο και χωρίζονται εξίσου στις 2 κλάσεις (δηλαδή 17 η καθεμία). Οι ακμές μεταξύ των κόμβων δεν είναι σταθερές και διαφοροποιούνται ανάλογα με τις ανάγκες κάθε πειράματος. Η αναπαράσταση κάθε κόμβου αρχικοποιείται τυχαία σύμφωνα με μια ομοιόμορφη κατανομή. Η επιλεγμένη διάσταση των αρχικών διανυσμάτων είναι 32.

Τα σενάρια θορύβου που εξετάσαμε είναι τα ακόλουθα:

- **Εξ' ολοκλήρου θορυβώδεις σχέσεις**:Ο στόχος αυτού του πειράματος είναι να εξετάσει τη συμπεριφορά των GNN όταν εφαρμόζεται σε γραφήματα που περιέχουν θόρυβο με τη μορφή μιας διακριτής σχέσης θορύβου. Διαισθητικά, η προσθήκη θορύβου στο γράφημα θα πρέπει να προκαλέσει πτώση της απόδοσης του μοντέλου GNN, εκτός από το GNN που έχει τη δυνατότητα να αγνοήσει τη θορυβώδη σχέση και να λάβει υπόψη μόνο τους γείτονες σύμφωνα με την πληροφοριακή σχέση κατά τη διαδικασία εκμάθησης των ενσωματώσεων. Για να αξιολογήσουμε την απόδοση των μοντέλων ορίσαμε ως "χρήσιμη" σχέση μία σχέση που οι ακμές της συνδέουν κόμβους της ίδιας κλάσης, και σαν σχέση θορύβου μία σχέση που οι ακμές της συνδέουν κόμβους διαφορετικής κλάσης.

  Ξεκινήσαμε έχοντας μια "χρήσιμη" και μία σχέση θορύβου με ίδιο πλήθος ακμών. Κάθε κόμβος είχε ίδιο πλήθος ακμών για καθεμία από τις σχέσεις. Παρατηρήσαμε ότι τα μοντέλα αναφοράς μπορούσαν να ταξινομήσουν σωστά όλους τους μη επισημασμένους κόμβους. Συνεχίσαμε αυξάνοντας το πλήθος των ακμών της θορυβώδους σχέσης. Πάλι η πτώση απόδοσης των μοντέλων αναφοράς ήταν αμελητέα. Στο δεύτερο πείραμα, αυξήσαμε το πλήθος των ακμών θορύβου, αυτή τη φορά όμως προσθέτοντας νέες σχέσεις θορύβου, διατηρώντας πάλι το πλήθος των ακμών κάθε σχέσης σε κάθε κόμβο ίσο. Στο Σχήμα 5.3 βλέπουμε το F1-score για διαφορετικές τιμές του βαθμού κόμβου και πλήθους

σχέσεων θορύβου. Ο βαθμός κόμβου είναι ίδιος για όλους τους κόμβους. Παρατηρούμε ότι από ένα συγκεκριμένο πλήθος σχέσεων θορύβου τα μοντέλα αναφοράς άρχισαν να παρουσιάζουν σημαντική πτώση στην ακρίβεια. Αντιθέτως, το SRGCN συνέχισε να ταξινομεί σωστά όλους τους μη επισημασμένους κόμβους σε όλες τις περιπτώσεις. Το GAT παρουσίασε μια μικρή υπεροχή σε απόδοση σε σχέση με το απλό GCN.

Όσον αφορά το πλήθος των συνελικτικών επιπέδων, παρατηρήσαμε ότι αυξάνοντας το πλήθος ως ένα σημείο οδήγησε σε βελτίωση της ακρίβειας. Το GCN πετυχαίνει τη βέλτιστη ακρίβεια με μεγαλύτερο αριθμό απο συνελικτικά επίπεδα σε σχέση με το GAT (Σχήμα 5.2).

Μία ακόμα ενδιαφέρουσα παρατήρηση είχε να κάνει με το πλήθος των ακμών της κάθε σχέσης. Στο σχήμα 5.4 βλέπουμε την ακρίβεια πρόβλεψης για 3 διαφορετικές τιμές του βαθμού κάθε κόμβου. Κάθε κόμβος έχεις το ίδιο βαθμό και σε κάθε σχέση. Η αρχική εικασία μας ήταν ότι όσο το η σχέση θορυβώδεις προς "χρήσιμες" ακμές αυξάνεται, θα μειώνεται η ακρίβεια των μοντέλων. Παρόλα αυτά, αυξάνοντας το πλήθος των ακμών, αλλά ταυτόχρονα διατηρώντας τη σχέση αυτή σταθερή, παρατηρήσαμε ότι η πτώση της ακρίβειας περιορίζεται. Συμπεραίνουμε ότι η απόλυτη αύξηση του σήματος πληροφορίας από τη "χρήσιμη" σχέση βοηθάει τα GNN να πετύχουν μεγαλύτερη ακρίβεια, ακόμα και αν ο αυξάνεται παράλληλα και ο θόρυβος.

- **Ακμές θορύβου σε χρήσιμες σχέσεις**: Το δεύτερο σενάριο θορύβου που εξετάσαμε είναι αυτό στο οποίο οι "χρήσιμες" σχέσεις περιέχουν επιπρόσθετες ακμές θορύβου, δηλαδή ακμές που συνδέουν κόμβους διαφορετικής κλάσης. Αρχικά εξετάσαμε την επίδραση του θορύβου αυτής της μορφής όταν οι ακμές θορύβου συνδέουν μόνο επισημασμένους κόμβους, ενώ οι μη-επισημασμένοι κόμβοι συνδεονται μονο με "χρήσιμες" ακμές, δηλαδή με κόμβους της ίδιας κλάσης. Από τα αποτελέσματα (Πίνακας 5.5) προκύπτει ότι παρατηρείται μικρή πτώση της ακρίβειας πρόβλεψης μόνο για μεγάλο πλήθος ακμών θορύβου. Επίσης αυξάνοντας το πλήθος των συνελικτικών επιπέδων βελτιώνει την ακρίβεια πρόβλεψης μέχρι ένα συγκεκριμένο πλήθος. Τα GCN και GAT έχουν παρόμοια αποτελέσματα. Στη συνέχεια προσθέτουμε ακμές θορύβου και σε μη-επισημασμένους κόμβους (Πίνακας 5.6) . Σε αυτή την περίπτωση παρατηρούμε πολύ μεγαλύτερη πτώση της ακρίβειας πρόβλεψης σε σχέση με το προηγούμενο σενάριο. Επίσης, και σε αυτό το σενάριο, το απόλυτο πλήθος "χρήσιμων" ακμών, όσο μεγαλύτερο είναι, τόσο μικρότερη είναι η πτώση της ακρίβειας. Για μεγαλύτερο πλήθος ακμών θορύβου τα μοντέλα με μεγαλύτερο πλήθος συνελικτικών επιπέδων έχουν καλύτερη απόδοση.

Παράλληλα, ο πειραματισμός μας στα συνθετικά δεδομένα ήταν μια καλή ευκαιρία να αξιολογήσουμε και τα διάφορα μοντέλα για τις μεθόδους ενσωματώσεων γράφων. Παρουσιάζουμε τα αποτελέσματα για 2 συνδυασμούς πλήθους ακμών - πλήθους σχέσεων θορύβου από τα οποία προκύπτει ότι τα RotatE και TransH είναι τα μόνα για τα οποία πετυχαίνουμε απόλυτη ακρίβεια στις προβλέψεις (Πίνακας 5.4) . Για να αξιολογήσουμε τη χρήση των ενσωματώσεων

σχέσεων συνολικά πραγματοποιήσαμε πειράματα και και με μία one-hot κωδικοποίηση των σχέσεων. Παρατηρήσαμε ότι τα αποτελέσματα σε αυτή την περίπτωση ήταν πολύ παρόμοια με αυτά του απλού GCN.

Πειράματα σε αληθινά δεδομένα

Με βάση τις παρατηρήσεις μας στα πειράματα στα συνθετικά δεδομένα, πραγματοποιήσαμε ένα δεύτερο γύρο πειραμάτων, αυτή τη φορά σε πραγματικά δεδομένα (IMDb και DBLP) για να επαληθεύσουμε τη συμπεριφορά που παρατηρήσαμε σε ένα πιο ρεαλιστικό σενάριο.

- **Εξ' ολοκλήρου θορυβώδεις σχέσεις**: Στο σενάριο όπου έχουμε θόρυβο με τη μορφή διακρτών σχέσεων, για να προσομοιώσουμε την περίπτωση όπου υπάρχουν αμιγώς θορυβώδεις σχέσεις, δημιουργήσαμε σχέσεις ανάλογες με τις ήδη υπάρχουσες που όμως συνδέουν μη σχετικές οντότητες. Συγκρίνουμε τα αποτελέσματα με 2 μοντέλα διαφορετικών κρυφών διαστάσεων στα συνελικτικά επίπεδα: 128 καθώς ήταν η τιμή με τα καλύτερα αποτελέσματα αλλά και 64 γιατί είναι το μέγεθος των συνελικτικών επιπέδων που χρησιμοποιήσαμε στο SRGCN. Πραγματοποιήσαμε δύο πειράματα: στο πρώτο προσθέσαμε θόρυβο υπό τη μορφή πολλαπλών σχέσεων θορύβου (Πίνακες 5.7 και 5.9), ενώ στο δεύτερο είχαμε μόνο μία σχέση θορύβου για κάθε πραγματική σχέση και αυξήσαμε το πλήθος των ακμών της σχέσης θορύβου (Πίνακες 5.8 και 5.10). Παρατηρώντας τα αποτελέσματα, βλέπουμε ότι το SRGCN έχει ελάχιστη απώλεια απόδοσης παρά την ύπαρξη τέτοιων σχέσεων, τη στιγμή που τα μοντέλα αναφοράς επηρεάζονται εμφανώς αρνητικά από την ύπαρξη των σχέσεων θορύβου. Στο σενάριο όπου ο θόρυβος υπάρχει σε πολλαπλές σχέσεις, η πτώση της ακρίβειας πρόβλεψης ήταν μεγαλύτερη, κάτι που συμβαδίζει με τις παρατηρήσεις μας στα πειράματα των συνθετικών δεδομένων. Τέλος, όπως βλέπουμε η ακρίβεια που πετυχαν τα μοντέλα αναφοράς σε αυτό το σενάριο ήταν παρόμοια.

- **Ακμές θορύβου σε χρήσιμες σχέσεις**: Στο δεύτερο σενάριο προσθέτουμε ακμές θορύβου στις πραγματικές σχέσεις. Αρχικά προσθέσαμε θόρυβο σε κάθε μία σχέση ξεχωριστά και στη συνέχεια σε περισσότερες από μία ταυτόχρονα. Στους Πίνακες 5.11 και 5.12 βλέπουμε ότι η ύπαρξη θορύβου σε συγκεκριμένες σχέσεις οδηγεί σε μεγαλύτερη πτώση του f1, παρά το γεγονός ότι το πλήθος ακμών θορύβου που προσθέσαμε δεν ήταν το μεγαλύτερο. Συγκεκριμένα όταν προσθέσαμε ακμές θορύβου μεταξύ των Director και Movie στο IMDb και Author και Paper στο DBLP παρατηρήσαμε τη μεγαλύτερη πτώση στο F1-score παρόλο που το πλήθος των ακμών θορύβου δεν ήταν το μέγιστο. Επιπρόσθετα, παρατηρούμε ότι το SRGCN εχει σημαντικά καλύτερα αποτελέσματα από τα μοντέλα αναφοράς οταν μολύνουμε με θόρυβο μόνο μία σχέση. Στις περιπτώσεις που προσθέσαμε θόρυβο σε όλες τις σχέσεις, το κέρδος της μεθοδολογίας μας ήταν σαφώς μικρότερο. Αυτό διαισθητικά είναι λογικό, καθώς η μεθοδολογία μας βασίζεται στο να μεγιστοποιεί την πληροφορία από τις "καθαρές" σχέσεις για να πετυχαίνει καλύτερα αποτελέσματα και σε αυτή την περίπτωση δεν υπήρχαν τέτοιες.

Πραγματοποιήσαμε επίσης πειράματα για να δούμε την επίδοση του SRGCN για διάφορα ποσοστά θορύβου στην σχέση με τη μεγαλύτερη σημασία (Σχήμα 5.5). Βλέπουμε ότι το SRGCN υπερισχύει έναντι των μοντέλων αναφοράς σε όλες τις διαφορετικές τιμές αναλογίας θόρυβος προς πληροφορία. Επιπρόσθετα, στην περίπτωση του DBLP βλέπουμε ότι το MLP που δεν λαμβάνει υπόψη τη δομή του γράφου και άρα δεν επηρεάζεται από το θόρυβο καθόλου αρχίζει να έχει καλύτερα αποτελέσματα σε σχέση με τα baselines όταν οι ακμές θορύβου είναι 50% των "χρήσιμων" ακμών, ενώ το SRGCN καταφέρνει να διατηρεί καλύτερο F1-score από το MLP μέχρι για πλήθος ακμών θορύβου 75% των "χρήσιμων" ακμών.

Τέλος, πραγματοποιήσαμε πειράματα για διαφορετικό πλήθος συνελικτικών επιπέδων για να εξετάσουμε την επίδρασή του σε σύνολα δεδομένων με πολύ θόρυβο (Σχήμα 5.6). Η ενδιαφέρουσα πληροφορία που αποκομίσαμε είναι ότι σε αντίθεση με το σενάριο οπου είχαμε αμιγώς θορυβώδεις σχέσεις, εδώ τα καλύτερα αποτελέσματα προέρχονται από πιο "ρηχά" GNNs. Η εξήγησή μας γι αυτό είναι ότι καθώς εδώ "χρήσιμες" ακμές και ακμές θορύβου συνυπάρχουν στην ίδια σχέση, αυξάνοντας το πλήθος των επιπέδων μεγαλώνουμε τη γειτονιά που επηρεάζει την τελική αναπαράσταση κάθε κόμβου και συμπεριλαμβάνουμε περισσότερο θόρυβο.

1.5 Συμπεράσματα

1.5.1 Σύνοψη

Σε αυτή τη διατριβή, προτείναμε μια σύνθετη αρχιτεκτονική βασισμένη σε συνελικτικά νευρωνικά δίκτυα γράφων και ενσωματώσεις γράφων, για να επιλύσουμε ένα πρόβλημα ταξινόμησης κόμβων σε πολυσχεσιακά δίκτυα, εστιάζοντας σε γράφους με θόρυβο. Τα ευρήματά μας οδηγούν στο συμπέρασμα ότι η ενσωμάτωση πληροφοριών σε επίπεδο σχέσης σε ένα μοντέλο που βασίζεται στο GNN μπορεί να ενισχύσει την εκφραστικότητα των αναπαραστάσεων και να βελτιώσει την απόδοση ολόκληρου του μοντέλου.

Επιπλέον, τα πειράματά μας σε γράφους με διαφορετικά σενάρια θορύβου παρείχαν χρήσιμες πληροφορίες σχετικά με τη συμπεριφορά των ευρέως χρησιμοποιούμενων GNN και την ανθεκτικότητά τους έναντι πολλών ακμών θορύβου. Εντοπίσαμε σενάρια θορύβου που βλάπτουν την απόδοση αυτών των μοντέλων, όπως ο τυχαίος θόρυβος σε συγκεκριμένες ή πολλαπλές σχέσεις.

Η κύρια συμβολή μας είναι η προτεινόμενη αρχιτεκτονική (SRGCN), η οποία πέτυχε καλύτερη απόδοση στο πρόβλημα ταξινόμησης κόμβων σε σύνολα δεδομένων αναφοράς. Επιπλέον, ξεπέρασε σημαντικά τα βασικά μοντέλα σε όλα τα σενάρια θορύβου. Συγκεκριμένα, η αρχιτεκτονική μας δεν υπέστη σχεδόν καμία πτώση απόδοσης όταν προσθέσαμε σχέσεις που περιείχαν μόνο θόρυβο και υπερίσχυε όταν εισάγαμε θόρυβο μολύνοντας μια συγκεκριμένη σχέση. Έτσι, αποδείξαμε εμπειρικά την κύρια διαίσθηση πίσω από το μοντέλο μας, ότι οι επιπλέον πληροφορίες που εισήχθησαν από τις ενσωματώσεις σχέσεων του SRGN βοήθησαν το μοντέλο μας να αναγνωρίσει χρήσιμες σχέσεις από τις θορυβώδεις.

1.5.2 Μελλοντικές προεκτάσεις

Αυτή η διατριβή προσφέρει μερικές πιθανές επεκτάσεις για περαιτέρω πειραματισμό. Όσον αφορά την αρχιτεκτονική, ο πειραματισμός στα GNN που χρησιμοποιούνται στην αρχιτεκτονική SRGCN, εκτός από το GCN και το GraphSAGE, παρουσιάζει ιδιαίτερο ενδιαφέρον, καθώς θα μπορούσε ενδεχομένως να οδηγήσει σε σημαντική βελτίωση της απόδοσης. Αποφασίσαμε επίσης να χρησιμοποιήσουμε τη συνένωση των ενσωματώσεων που προέρχονται από τα GNN και τους αποκωδικοποιητές ως συνάρτηση συνάθροισης. Περαιτέρω πειραματισμός στην επιλογή συνάρτησης συνάθροισης θα μπορούσε να προσφέρει χρήσιμες πληροφορίες ή ακόμα και βελτίωση της απόδοσης. Εκτός από αυτό, θα ήταν ενδιαφέρον να αξιολογήσουμε την απόδοση του μοντέλου μας σε περισσότερα σενάρια θορύβου. Για παράδειγμα, στο μέλλον, θα μπορούσαμε να αξιολογήσουμε την ευρωστία των διαφορετικών μοντέλων σε καταστάσεις όπου ο θόρυβος στοχεύει συγκεκριμένους κόμβους. Τέλος, δεδομένου ότι το μοντέλο μας έδειξε ικανοποιητικά αποτελέσματα σε αυτά τα σενάρια θορύβου, θεωρούμε σκόπιμη τη σύγκριση του μοντέλου μας με άλλες μεθοδολογίες που έχουν σχεδιαστεί ειδικά για τη βελτίωση της απόδοσης σε γράφους με θόρυβο.

Chapter 2

Introduction

2.1 Motivation

Graph representation learning is an emerging field due to the wide range of scientific fields and applications that graphs are used in. Modelling physics systems, learning molecular fingerprints, protein structures or drug interactions require models that can utilize graph-structure information to extract entities representations [55]. Graphs are usually represented as a combination of a set of attributed nodes and an adjacency matrix denoting the interactions between the nodes of the graph. Although this representation is useful for us to understand the structure of the graph in some sense, traditional machine learning models can not utilize this representation since they are not applicable in non-euclidean data, like graphs [14].

Following the deep learning trend, the adaptation of deep learning models like Convolutional Neural Networks (CNNs) on non-euclidean data lead to the development of geometric deep learning methods and Graph Neural networks (GNNs). GNNs are the most popular graph representation models, due to their superior performance in various graph representation learning tasks [46]. They accomplish that by passing and aggregating information between neighbouring nodes, eventually capturing neighbourhood-level information in node representations. Apart from deep learning, other methods that capture structural information of a graph beyond the neighbourhood level have drawn scientific interest. For example, graph embedding methods aim to embed components of graphs including entities and relations into continuous vector spaces that encode and preserve the inherent structure of the whole graph [43].

However, noise in graphs is a very common obstacle in deriving useful graph representations [9]. For instance, the majority of the real-world graphs used are constructed through knowledge mining from the web. Automated knowledge graph construction methods are often imperfect, and this results in noisy graphs and low-information or even noisy relations that can affect the quality of the constructed graph negatively [30]. GNNs are vulnerable to noise due to the propagation of noisy messages between the node neighbourhoods [7]. Therefore, deep learning models that can extract robust representations against

the inherent noise of these graphs are of significant value.

Moreover, these real-world multi-relational graphs can contain large number of relations. Some relations can be more informational than others, meaning the interaction they describe is more fundamental for the represented system or the downstream application we want to model. The existence of noise in these relations can lead to significant degradation of the quality of the generated representations. Thus, preserving the structural properties of informative relations is of high priority to mitigate the noise impact on the graph representation [27].

2.2 Approach and contribution

In this thesis, we aim to address the aforementioned GNN limitations but also to evaluate the incorporation of structural information about each relation of a multi-relational graph in the graph nodes' representation. We propose the utilization of relation embeddings extracted by a knowledge graph embedding model (such as RotatE), combined with node embeddings derived by a GNN, to achieve more expressive representations of the nodes of the graph. The proposed methodology, named Split Relation Graph Convolutional Network (SRGCN for brevity), can achieve improved performance in the node classification task by leveraging both information about the neighbourhood structure extracted by a GNN and relation-level structural information derived from the knowledge graph embedding model. In addition to that, our proposed architecture provides robust representations in noisy graph. Our experimentation results encourage the use of knowledge graph embeddings over just shallow lookup embeddings, since the expressiveness of the latter is not sufficient to discriminate informative from noisy relations. Furthermore, our findings show that our proposed model has neglectable performance loss in graphs affected by totally noisy relations, when in comparison widely-used GNN models greatly drop their performance. Finally, our experimentation indicates that the performance loss of the proposed architecture is mitigated significantly when noise affects a limited number of relations, compared to the baseline models.

In general, the contributions of this thesis can be summarized into the following points:

- The introduction of a hybrid GNN and graph embedding model, that is able to capitalize on informative relations to produce expressive graph representations. These representations, according to our experiments, are able to preserve useful relation information and reduce the negative impact of noise edges significantly better than our baselines GNN models.
- Extensive comparison on synthetic and real-world datasets, under different noise scenarios of the introduced model versus commonly used models.
- Empiric results that prove the usefulness of the proposed architecture and plenty of insights regarding the behaviour of the examined GNNs under specific types of noise.

Apart from the above contributions, the code and experimental results are available to everyone on GitHub for results' replication and further experimentation purposes.

2.3 Thesis structure

In Chapter 3 we provide the theoretical background of our work. We introduce the basic concepts and notation about graphs, graph representation learning, graph neural networks and knowledge graph embedding methods. We also give a brief overview of research on noisy graph representation learning.

In Chapter 4 we present the proposed architecture, delve into details of its features and the intuition behind it.

In Chapter 5 we present our experimental setup, the datasets used, the results of our experimentation and the insights we derived regarding our proposed model, as well as other baseline GNNs.

In Chapter 6 we conclude this thesis by presenting a summary of the results and providing possible future research ideas.

Chapter 3

Theoretical Background and Related Work

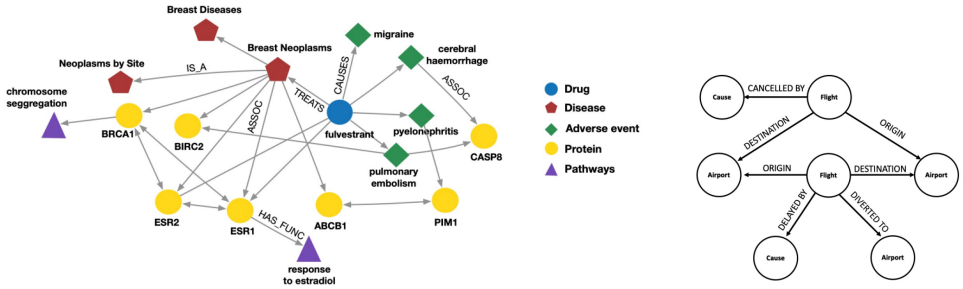
3.1 Introduction to graphs

3.1.1 Graph-structured data

Graphs are a ubiquitous data structure for modeling complex systems of the real world. They are widely used as a representation method in various scientific fields. Especially, in computer science, they play a very important role and have been studied for many years. The main advantage of graphs is their ability to model objects of the real world (i.e., nodes of the graph) alongside the interactions between pairs of these objects (i.e., edges of the graph). Therefore, graphs are being selected as a representation method in a variety of applications, such as social networks, recommendation systems, molecular graphs, protein or drug interaction networks etc. [14]. Lately, applying machine and deep learning algorithms on graph-structured data has been evolved into an emerging field of research mainly due to the promising results such models (i.e., GNNs) have achieved on various challenging tasks [17]. Before analyzing how these models can handle graph structured data, it makes sense to give some formal definitions on this special data structure.

3.1.2 Graph basics and notation

Formally, a graph $\mathcal{G} = (V, E)$ is comprised of a set of nodes V and a set of edges E. An edge $e \in E$ is associated with two (distinct in most cases) nodes $u, v \in V$, and denoted as $e = (u, v)$. The most common way to represent the edges of a graph is through its *adjacency matrix* $A \in \mathbb{R}^{|V|\times|V|}$. To form the adjacency matrix, we arbitrary enumerate graph nodes and match their numbers to the corresponding rows and columns of the matrix, then for each pair (u, v) we assign $\mathbf{A}[u, v] = 1$ if the pair is in E, otherwise $\mathbf{A}[u, v] = 0$, indicating that no edge exists between these nodes. In weighted graphs, the value of the weight is assigned to the corresponding cells instead of 1. If the graph is undirected, meaning that node order in the every edge does not contain any information



(a) Biomedical Knowledge Graph (b) Event Graph

Figure 3.1: Examples of heterogeneous real world graphs.

about their interaction, then the adjacency matrix is symmetrical.

3.1.3 Multi-relational, heterogeneous and knowledge graphs

Many of the real world applications (datasets) cannot be re represented by simple graphs containing nodes and edges of a single type. Some graphs contain entities that, except for the label differ also in terms of the type.

Apart from weights, graph edges may also have different types too. Such graphs are called *multi-relational*. An important subset of multi-relational graphs are *heterogeneous graphs*. In heterogeneous graphs [38], both edges and nodes are associated with multiple types. A heterogeneous graph $G = (V, E)$ consists of a node set V and an edge set E . A heterogeneous graph is also associated with a node type mapping function $\phi : V \rightarrow A$ and an edge type mapping function $\psi : E \rightarrow R$. A and R denote the sets of predefined node and edges types. We represent this kind of graphs with an adjacency tensor $A \in \mathbb{R}^{|V|\times|R|\times|V|}$, where R is the set of relations. Examples of heterogeneous graphs are shown in Figure 3.1.

Knowledge graphs

Knowledge graphs are an example of heterogeneous graphs. They use a graph-structured model to represent human knowledge, by capturing facts in the form of entities, relationships and semantic descriptions (labels, attributes and types of entities) [18]. Knowledge graphs have drawn attention mainly due to the inclusion of such models in industrial-scale projects like Google's Knowledge Graph and Amazon's Product Graph [23]. Their importance mainly lies on the fact that they can be used for reasoning and rule extraction, fundamental concepts in the context of relational data mining [37].

3.2 Graph representation learning

Graph representation learning has been an emerging research area recently. The purpose of graph representation learning is to extract low-dimensional representations through

the combination of node features (i.e., vectors containing information for every graph node) with graph topology. This is of a large significance because the quality and expressiveness of the graph representations are the most important factor that affects the performance of machine learning models in various tasks. Many techniques have been proposed for generating effective graph representation vectors, which generally fall into two categories: traditional graph embedding methods like random walk and matrix factorization based methods [47] or graph and graph neural networks based methods, which will be analyzed in the following sections [24].

3.2.1 Machine learning on graphs

As mentioned before, a variety of systems of the real world can be represented as graphs. Machine learning algorithms have demonstrated their success on tackling diverse problems associated with structured data, and graph structured data are not an exception. What differs in the case of graphs is the fact that the usual categories of supervised and unsupervised learning are not so common due to the special characteristics of graph datasets. For example, labeled data are often more difficult to be collected in systems described as graphs [57]. Specifically in graph structured data, the existence of edges alongside nodes offered the possibility for graph specific machine learning tasks such as link prediction [29]. Generally, there are three tasks for machine learning applications on graphs, according to the output of the machine learning model: node-level (node classification), edge-level (edge classification and link prediction) and graph-level (graph classification) tasks [46].

In this point it is useful to give some formal definitions and examples of graph machine learning frameworks that will be the main part of the subject of this study.

Semi-supervised learning for node classification

In supervised learning, the learning system observes a labeled set (lablel meaning an indication of the class of each node) of datapoints (training set) consisting of (feature, label) pairs, denoted by $\{(x_1, y_1), ..., (x_n, y_n)\}$. The goal is to predict the label y for any new input feature x. However, labels y are often hard, expensive and slow to obtain, because it may require custom annotation by humans. On the other hand, unlabeled data x is usually available in large quantity and costs little to collect compared to labeled data. Traditional supervised classification methods cannot use unlabeled data to train classifiers.

Before explaining the semi-supervised setting, it is useful to provide a detailed definition of the node classification task. Node classification is probably the most common task on graph-structured data. Given a graph $G = (V, E)$ and labels on a subset of nodes $V_{train} \in V$, the goal in node classification is to predict the labels of the rest of the nodes that we are unaware of their class (unlabeled nodes). Examples of node classification tasks are classifying papers or authors in a citation network [1] or finding malicious accounts or bots on social networks [22]. Essentially, we assume relations between the nodes, in

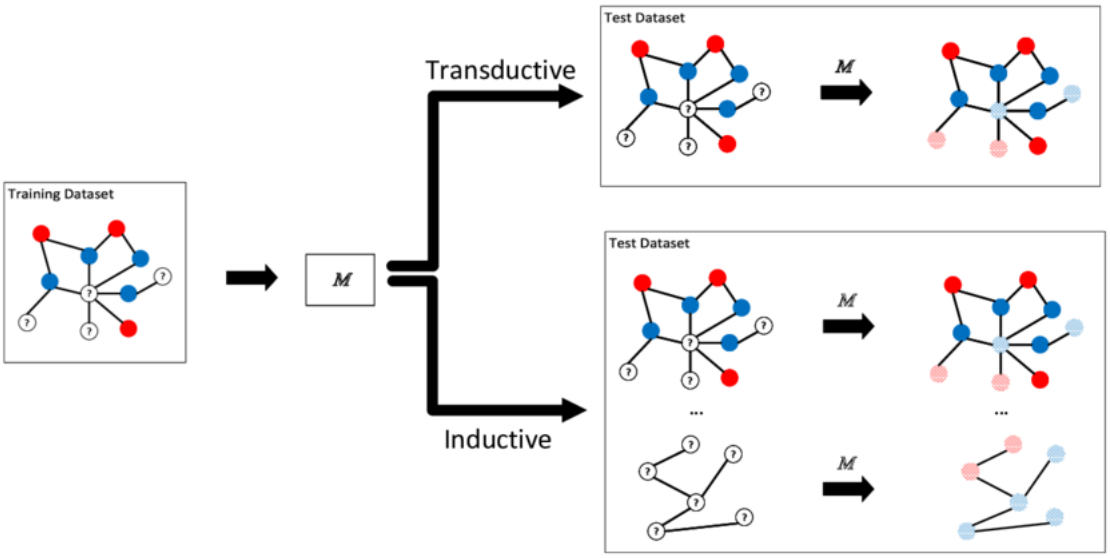


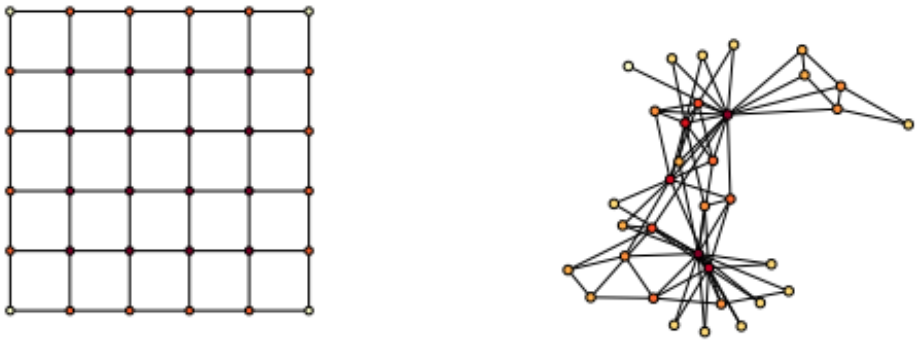
Figure 3.2: A Comparison between transductive and inductive setting

order to find the missing labels. These relations can exploit homophily [31], which states that nodes have high probability to share characteristics with their neighbors. Another concept is structural equivalence [33], which is the the tendency of nodes that have similar neighborhoods structures, to share similar labels.

The question semi-supervised learning addresses is: given a relatively small labeled dataset $\{(x, y)\}$ and a large unlabeled dataset $\{x\}$, can one devise ways to learn from both for classification? The name "semi-supervised learning" comes from the fact that the data used is between supervised and unsupervised learning. Semi-supervised learning promises higher accuracies with less annotating effort. It is therefore of great theoretical and practical interest, especially in graph-structured data where labeled data is hard to find and most datasets contain a small fraction of labeled nodes [57]. There are two settings for semi-supervised learning in graphs [6] (Figure 3.2):

- **Transductive setting**: In the test phase, the model tries to predict the labels of the given unlabeled nodes. In transductive learning, the model has observed all the data beforehand, both labeled and unlabeled nodes. Even though the class of the unlabeled nodes is unknown, learning algorithms can make use of patterns and additional information present in this data during the learning process.

- **Inductive setting**: In the test phase, new unlabeled nodes from the same distribution are provided to the model to infer. This setting is similar to traditional supervised learning, where unlabeled data are not accessed by the model at all during the training procedure.



(a) Grid (Euclidean) (b) Arbitrary graph (Non-Euclidean)

Figure 3.3: An illustration of Euclidean vs non-Euclidean graphs.

3.2.2 Graph neural networks

As a natural consequence of the rapid rise of deep learning models in various fields lately, a lot of research attention has been drawn on deep learning applications on graphs as well. Learning representations for complex structured data is a challenging task. In the deep learning era, many successful models have been introduced for certain kinds of structured data. More specifically, sequential data, such as text and videos, can be modelled via recurrent neural networks which can capture sequential dependencies. In addition to that, convolutional neural networks have achieved state-of-the-art performance in extracting image features due to their ability to recognise spatial patterns.

Plenty of machine learning tasks are dealing with data represented as graphs. However, CNNs can only operate on regular Euclidean data. These structures can be regarded as instances of graphs, as show in Figure 3.3. Therefore, it makes sense to find a way to generalize the already successful CNNs on graphs. Extending CNNs from Euclidean domain to non-Euclidean is an emerging research area. To this day, graph neural networks have shown their great power in working with graph structured data for various applications [3].

Permutation invariant property

In order to deal with graph data, the most common methodology is to represent the graph with its adjacency matrix and use the feature vectors of the nodes and the adjacency matrix as the input to a feed forward neural network (Figure 3.4). The problem that arises by following this approach, is that feed forward neural network produces different outputs for different node orderings (i.e., for different adjacency matrices). But permutations of the adjacency matrix, represent the same graph. Therefore, it is essential to make sure that the neural network architecture is permutation invariant [46].

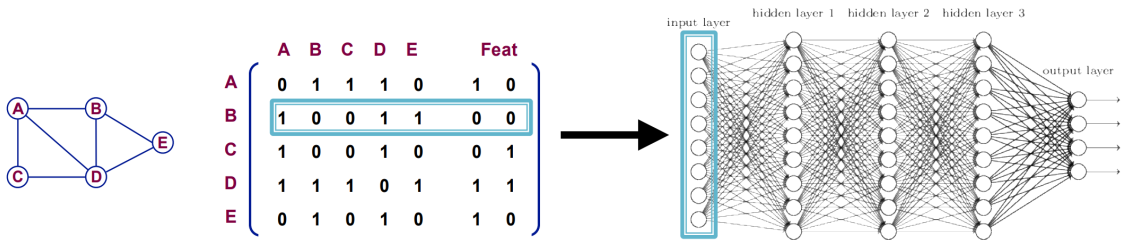


Figure 3.4: An illustration of why naive MLP approach fails for graphs

Neural message passing framework

The current most commonly used architecture when considering graph-structured data is the *Graph Neural Network* (GNN) [35]. The goal of GNN is to learn a representation $h_u \in \mathbb{R}$ for every node u in the graph, that combines its feature vector with the feature vectors of its immediate neighbors. The representation h_u of every node u can be used in order to produce the label of the node u for a node classification task. To learn these representations, GNN uses a general framework, called *neural message passing* [11], in which every node exchanges messages with its neighbors. In start, every node u is represented with an initial feature vector h_u. In every iteration, we update the state of the node, based on its previous state and the representations of its neighbors. In mathematical notation, we have the following equations:

$$m_u^{k+1} = AGGREGATE(h_u^k, \forall u \in N(u)) \tag{3.1}$$

$$h_u^{k+1} = UPDATE(h_u^k, m_u^{k+1}) \tag{3.2}$$

where $N(u)$ is the neighborhood of node u , $AGGREGATE$ and $UPDATE$ are differentiable functions, usually approximated by neural networks [16] , m_u^{k+1} is the aggregated message from the neighbors in $k+1$ iteration and h_u^{k+1} is the representation of the node u in $k+1$ iteration. The $AGGREGATE$ function should be permutation invariant, as it was mentioned before. Apart from the message and aggregation, a GNN layer additionally applies non-linearity to the activation of the layer to extract the final representation to add expressiveness and making it possible for the network to approximate nonlinear functions. Most common activation functions are the *sigmoid* function ($\sigma(\cdot)$) and the *rectified linear unit* function ($ReLU(\cdot)$). Figure 3.5 is a visualization of the aforementioned procedure.

3.2.3 Classical GNN layers

CNN vs GNN

Before we dive into the state-of-the-art GNN-based architectures, it would be useful to analyze the relation between CNNs and GNNs, since the neural networks we are going to describe in the next sections are convolution-based.

CNNs are feature extractors that have achieved state-of-the-art results in tasks related to structured data with grid-like topology such as images. But, as mentioned before, grid-

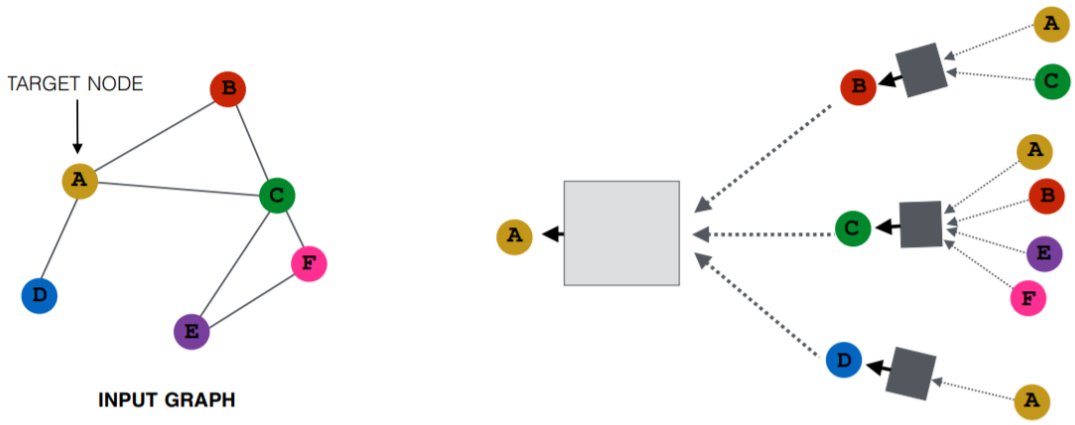
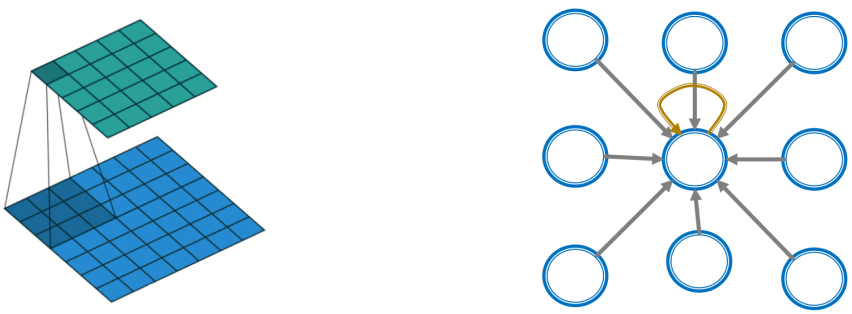


Figure 3.5: An illustration of message passing (arrows) and aggregation (grey boxes) in message passing framework



(a) Image (b) Graph

Figure 3.6: An illustration of how an image can be represented as graph.

like structures can be represented as graphs. In the case of a single channel image $N(v)$ represents the 8 neighbour pixels of pixel v (Figure 3.6).

CNN can be seen as a special GNN with fixed neighbor size and ordering. In the case of CNNs, the size of the filter is pre-defined. The advantage of GNN is the fact that it processes arbitrary graphs with different degrees for each node. In addition to that, CNNs are not permutation equivariant, like GNN. That means that switching the order of pixels leads to different outputs of the network [55].

Graph convolutional network

Graph Convolutional Network is the most widely used GNN layer in the literature. It was proposed by Kipf and Welling [26] and it has achieved state-of-the-art performance in a variety of tasks associated with graph-structured data including natural language processing, physics, chemistry, biology, material science and social network analysis [54]. It follows the neural message passing approach. Every convolutional layer l consists of a

weight matrix $\mathbf{W}^{(l)}$. For each node u, it takes as input the representation of the previous layer (i.e, $\mathbf{h}_u^{(l-1)}$), multiplies it with the weight matrix $\mathbf{W}^{(l)}$ to produce the new representation $\mathbf{h}_u^{(l)}$. This constitutes the message of each neighbour node of u to this particular node. The messages then are being aggregated and a non-linear activation function is applied as well to add expressiveness. The following formula describes in mathematical notation the operations taking place in a GCN layer.

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right) \tag{3.3}$$

The term $\frac{1}{|N(v)|}$ in the message equation is used to normalize the message of the layer by the node degree of each node. In addition to that, in GCN graph is assumed to have self-loops that are included in the summation. The message and aggregation operations can be formulated as:

$$\mathbf{m}_v^{(l)} = \frac{1}{|N(v)|}\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)} \tag{3.4}$$

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{m}_u^{(l)}\right) \tag{3.5}$$

GraphSAGE

The majority of approaches in the literature that generate node embeddings are inherently transductive. Most of them do not generalize to unseen data, since they create the embeddings by optimizing predictions over nodes in a fixed graph. Hamilton et al. [13] proposed a general framework, called *GraphSAGE* (SAmple and aggreGatE), for inductive node embedding. Unlike the embedding approach of GCN, the GraphSAGE is not directly aggregating the messages from the neighbourhood of node v, but first derives a representation of the neighbourhood by aggregating the representations of the neighbours and then concatenates the hidden representation of node v from the previous layer $\mathbf{h}_u^{(l-1)}$ with the aforementioned neighbourhood representation. This leads to a neural message passing approach with a two-stage aggregation, as formulated below, that can learn the topological structure of each node's neighbourhood and generalize on unseen nodes. An overview of the GraphSAGE architecture can be seen in Figure 3.7

First the message for each node u of the neighbourhood $N(v)$ of node v is computed

$$\mathbf{m}_u^{(l)} \leftarrow \{\mathbf{h}_u^{l-1}, \forall u \in N(v)\} \tag{3.6}$$

- **Stage 1**: Aggregate from node neighbours

$$\mathbf{h}_{N(v)}^{(l)} \leftarrow AGG(\mathbf{m}_u^{(l)}) \tag{3.7}$$

- **Stage 2**: Further aggregate over node itself

$$\mathbf{h}_v^{(l)} \leftarrow \sigma\left(\mathbf{W}^{(l)} \cdot CONCAT(\mathbf{h}_u^{(l-1)}, \mathbf{h}_{N(v)}^{(l)})\right) \tag{3.8}$$

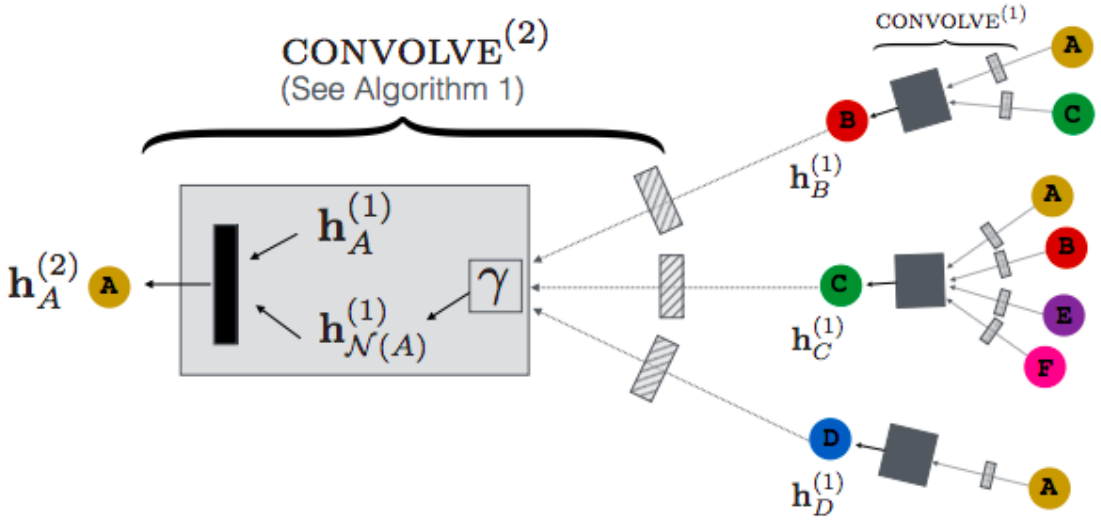


Figure 3.7: An illustration of the GraphSAGE architecture

Altogether the representation update procedure for each node v in a GraphSAGE layer can be expressed mathematically by the following formula:

$$\mathbf{h}_v^{(l)} \leftarrow \sigma\left(\mathbf{W}^{(l)} \cdot CONCAT(\mathbf{h}_v^{(l-1)}, AGG(\{\mathbf{h}_u^{l-1}, \forall u \in N(v)\}))\right) \tag{3.9}$$

In addition to that, a GraphSAGE layer applies ℓ_2 normalization to $\mathbf{h}_v^{(l)}$. The normalization step in GraphSAGE performs the following operation:

$$\mathbf{h}_v^{(l)} \leftarrow \frac{\mathbf{h}_v^{(l)}}{||h_v^{(l)}||_2} \quad \forall v \in V \tag{3.10}$$

Without the ℓ_2 normalization, the embeddings vectors have different scales (ℓ_2 - norm) for vectors. After ℓ_2 normalization, all vectors have the same ℓ_2 - norm. In some cases (not always), normalization of embedding results in performance improvement.

Graph attention network

Graph Attention Networks are another neural network architecture that operates on graph-structured data and follows the neural message passing framework. It was proposed by Veličković et al. [41] to address the shortcomings of prior methods based on graph convolutions.

In the graph neural networks we described above, the contribution of each node to the aggregation scheme is dictated by its degree. In both GCN and GraphSAGE, there is a weighting factor (importance) $\alpha_{vu} = \frac{1}{|N(v)|}$ of node u's message to node v . This weighting factor is defined explicitly based on structural properties of the graph (node degree). This means that all neighbours $u \in N(v)$ are equally important to node v.

Graph Attention Networks face this limitation by applying different weights focusing on the important parts of the input data and fading out the rest non-informative parts.

In other words, in GAT's case, not all node's neighbours are equally important. The intuition behind GAT is that the neural network should devote more computing power on that small but important part of the data. Which part of the data is more important depends on the context and is learned through training. This mechanism is illustrated in Figure 3.8.

For the computation of the embedding $\mathbf{h}_v^{(l)}$ of node v, GAT introduces a trainable attention weight a_{vu} that determines how much node v attends the message of each neighbour $u \in N(v)$. The computation of the weights α_{vu} is taking place through an attention mechanism α. This mechanism computes the attention coefficients e_{vu} across pairs of nodes u, v based on their messages as shown below:

$$e_{vu} = \alpha(\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}, \mathbf{W}^{(l)}\mathbf{h}_v^{(l-1)}) \tag{3.11}$$

The term e_{vu} denotes the importance of node u's message to node v. This attention score e_{vu} is then normalized into the final attention weight α_{vu} by applying the *softmax* function so that the individual attention weights are summing up to 1.

$$\alpha_{vu} = \frac{exp(e_{vu})}{\sum_{k \in N(v)} exp(e_{vk})} \tag{3.12}$$

Incorporating these weights into the standard GCN embedding update process leads to an embedding update formula for the GAT that looks like this:

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu}\mathbf{W}^{(l)}\mathbf{h}_u^{(l-1)}\right) \tag{3.13}$$

Attention mechanisms are being widely adopted in many state-of-the-art architectures due to their computational and storage efficiency and the fact that they perform extremely well in a large variety of tasks. In the case of graphs, it is also worth mentioning that they are independent of graph size (fixed number of parameters) and also readily applicable to inductive problems, as it is a shared edge-wise mechanism and therefore does not depend on the global graph structure [53].

3.2.4 Extension of GNNs in multi-relational graphs

In the previous section, we presented an overview of the neural message passing framework and how it can handle graph-structured data to extract expressive representations of the graph. In this section we will describe how this framework can be extended to deal with heterogeneous graphs (i.e., graphs with different types of nodes and relations). To provide mathematical notation and equations, we will use GCN as instance to demonstrate the function of a Relational-GNN, but the following process can be implemented with any convolutional GNN identically. Recall that in GCN, the hidden representation for each node v at $(l+1)^{th}$ layer is computed by

$$h_v^{l+1} = \sigma\left(\sum_{u \in N(v)} \frac{1}{c_v} W^{(l)} h_u^{(l)}\right) \tag{3.14}$$

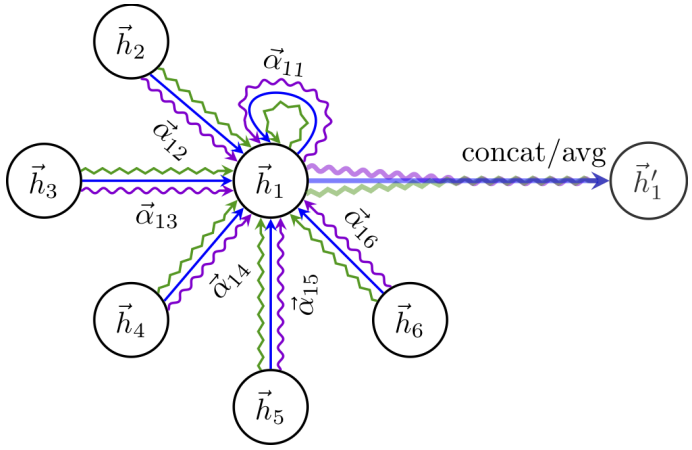
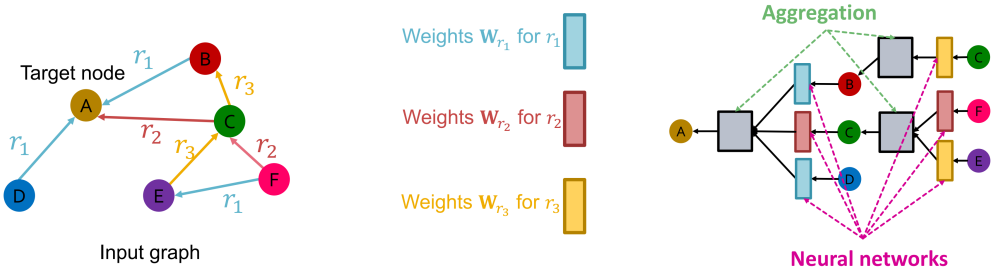


Figure 3.8: An illustration of a GAT layer with multi-head attention



(a) Input graph and relation weights

(b) Relational-GNN

Figure 3.9: An illustration of Relational-GNN on heterogeneous graphs following the neural message passing framework

where c_v is the normalization constant. In GCN, weight $\mathbf{W}^{(l)}$ is shared by all edges in layer l. To handle multiple relations, the Relational-GCN architecture introduces individual weight matrices for each relation type [36] and only edges of the same relation type r are associated with the same projection weight $\mathbf{W}_r^{(l)}$ (Figure 3.9). Then, following the neural message passing approach

- Computes outgoing message using node representation and weight matrix associated with the edge type (message)
  - Each neighbour of a given relation type

$$\mathbf{m}_{u,r}^{(l)} = \frac{1}{c_{v,r}} \mathbf{W}_r^{(l)} \mathbf{h}^{(l)} \tag{3.15}$$

  - Self-loop

$$\mathbf{m}_v^{(l)} = \mathbf{W}_0^{(l)} \mathbf{h}_v^{(l)} \tag{3.16}$$

- Aggregates incoming messages and self-loop message, apply activation function and generate new node representations (aggregation)

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum(\{\mathbf{m}_{u,r}^{(l)}, u \in N(v)\} \cup \{\mathbf{m}_v^{(l)}\})\right) \tag{3.17}$$

So the hidden representation of nodes in $l^{(th)}$ layer in R-GCN can be formulated as the following equation:

$$h_v^{(l+1)} = \sigma\left(W_0^{(l)} h_v^{(l)} + \sum_{r \in R} \sum_{u \in N(v)_r} \frac{1}{c_{v,r}} W_r^{(l)} h_u^{(l)}\right) \tag{3.18}$$

where $N(v)_r$ denotes the set of neighbour indices of node v under relation $r \in \mathcal{R}$ and $c_{v,r}$ is a normalization constant. In node classification, Schlichtkrull et al. [36] propose $c_{v,r} = |N(v)_r|$.

The problem that arises when applying the above equation directly is the rapid growth of the number of parameters, especially with high multi-relational data. This leads to high computational cost and overfitting risk. To prevent that, Schlichtkrull et al. [36] propose the use of basis decomposition during training.

$$\mathbf{W}_r^{(l)} = \sum_{b=1}^{B} \alpha_{rb}^{(l)} \mathbf{V}_b^{(l)} \tag{3.19}$$

Therefore, the weight $\mathbf{W}_r^{(l)}$ is a linear combination of basis transformation $V_b^{(l)}$ with coefficients $\alpha_{rb}^{(l)}$. The number of bases B is much smaller than the number of relations in the knowledge base.

3.2.5 Knowledge graph embeddings

As we mentioned in Section 2.1.3, knowledge graphs, as a form of structured human knowledge have drawn great research attention over the past several years. As a result, representation learning on knowledge graphs is an emerging field having produced significant results. Knowledge graph representation learning is the process of embedding knowledge graphs including both entities and relations into a continuous low-dimensional vector space [48] (Figure 3.10). The embedding methods utilized in this study follow the encoder - decoder approach for the training of the embeddings and belong to the shallow embeddings category, in the sense that the encoder that maps nodes to embeddings is simply an embedding lookup [14].

Edges in knowledge graphs are represented as triples (u, τ, v) which is interpreted as "head u has relation τ with the tail v". The goal of representation learning in knowledge graphs is, given a true triple (u, τ, v), to generate an embedding vector for (u, τ) that should be close (in the embedding space) to the embedding of v.

Most of the methods used for knowledge graph representation learning were originally designed for the task of knowledge graph completion. Therefore, it would be useful to give a short definition of this particular task.

Knowledge graph completion

In knowledge graph completion, the input graph is a multi-relational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the edges are defined as tuples of the form $e = (u, \tau, v)$ indicating the existence of a

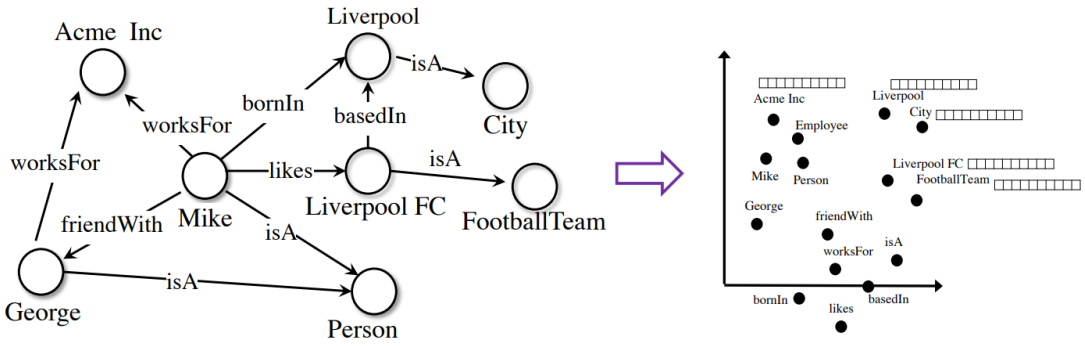


Figure 3.10: An illustration of the Knowledge Graph Embedding task

particular relation $t \in \mathcal{T}$ holding between these nodes. In general, the goal of knowledge graph completion is to detect missing nodes or edges in the graph, i.e nodes or edges that should exist in real-world setting but are omitted in the graph representation [14].

Encoder - decoder framework

In the *encoder - decoder* framework the process of generating embeddings for the nodes and edges of the graph is divided into two operations. First, an *encoder* model maps each node of the input graph into a low-dimensional vector. After that, a *decoder* model utilizes the encoded vector to reconstruct information about each node's neighbourhood in the original graph [14].

The encoder model is a function that maps nodes $v \in \mathcal{V}$ to vectors $\mathbf{z}_v \in \mathbb{R}^d$. This can be formulated as:

$$\text{ENC} : \mathcal{V} \rightarrow \mathbb{R}^d \tag{3.20}$$

$$\text{ENC}(v) = \mathbf{Z}[v] \tag{3.21}$$

where $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V} \times d|}$ is the matrix that consists of the embedding vectors for all nodes of the graph.

The decoder attempts to reconstruct certain graph statistics utilizing the node embeddings produced by the encoder. This statistics can vary, for example the decoder might try to figure out u's neighbours from vector $\mathbf{z}_u$. The corresponding signature for the above is:

$$\text{DEC} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+ \tag{3.22}$$

The goal of this encoder - decoder model is to optimize the reconstruction of the specified graph statistic. This is achieved by minimizing a reconstruction loss. This loss is computed by comparing the decoder output with a graph-based similarity measure between nodes (i.e., $\mathbf{S}[u, v]$). The corresponding mathematical expression for this process is the following:

$$\text{DEC}(\text{ENC}(u), \text{ENC}(v)) = \text{DEC}(\mathbf{z}_u, \mathbf{z}_v) \approx \mathbf{S}[u, v] \tag{3.23}$$

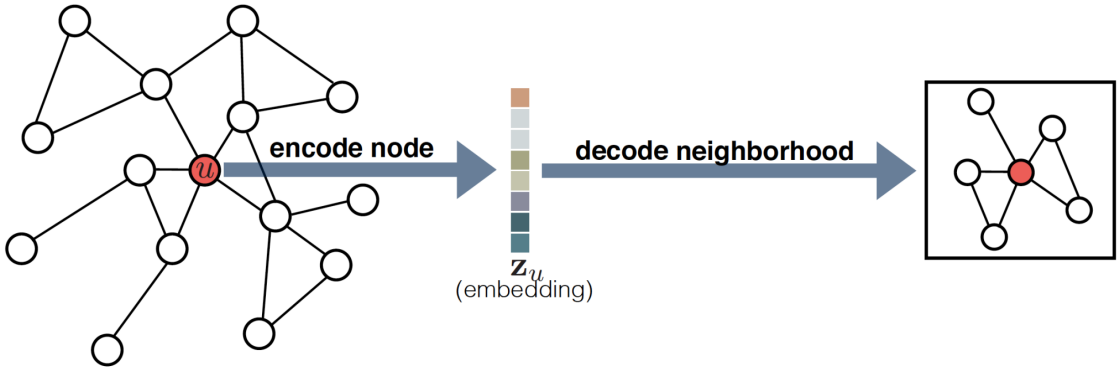


Figure 3.11: An illustration of the Encoder - Decoder framework

The aforementioned reconstruction loss is computed over a subset of nodes of the original graph (i.e, training set $\mathcal{D}$) by pairwise comparisons

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell(\text{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u, v]) \tag{3.24}$$

where $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function that computes the difference between the output of the decoder model (i.e., $\text{DEC}(\mathbf{z}_u, \mathbf{z}_v)$) and the true similarity values $\mathbf{S}[u, v]$. In node classification tasks, the loss function ℓ is the *cross-entropy* loss and it is minimized through stochastic gradient descent.

Extension to multi-relational graphs

The encoder - decoder framework that we described in the previous section can be further extended to handle multi-relational graphs as well, since the training of the embeddings of a multi-relational graph can also be treated as a reconstruction task. Attempting to solve the knowledge graph completion task, the encoder - decoder model is optimizing node embeddings $\mathbf{z}_u$ and $\mathbf{z}_v$ of two nodes u and v to reconstruct the relationship between these nodes.

In knowledge graphs, where multiple types of nodes and relations exist, the decoder part of the model is extended to handle the multi-relational setting. To achieve that, the decoder is given as input the pair of node embeddings created by the encoder together with a relation type. The signature describing this mapping is the following:

$$\mathbb{R}^d \times \mathcal{R} \times \mathbb{R}^d \to \mathbb{R}^+ \tag{3.25}$$

where $\mathcal{R}$ is referring to the relation type. The value that the decoder outputs corresponds to the likelihood of the existence of the edge (u, τ, v). The model is literally defined by the decoder operation, the similarity measure and the loss function.

Translational decoders

The first category of decoders that are relevant to this study are *translational* decoders, since they represent relations as translations in the embedding space. The first model is the *TransE* model. For a triple (u, τ, v) of the knowledge graph, the decoder of this particular model is defined as

$$\text{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) = -||\mathbf{z}_u - \mathbf{r}_\tau + \mathbf{z}_v|| \tag{3.26}$$

In this embedding methodology, each relation is represented using a d-dimensional embedding. The likelihood of an edge is proportional to the distance between the embedding of the head node and the tail node, after translating the head and tail nodes according to the relation-specific space [2]. Two fundamental limitations of this approach is the fact that TransE cannot model symmetric relations nor 1-to-N relations. These limitations are being addressed by *TransR* model, another model of the translation class. In order to overcome these limitations, the decoder of TransR models entities as vectors in the entity space $\mathbb{R}^d$ and each relation as vector in the relation space $\mathbf{r} \in \mathbb{R}^k$ with $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ as the projection matrix of this particular relation. In other words, TransR is projecting embdeddings from entity space $\mathbb{R}^d$ to relation space $\mathbb{R}^k$. The scoring function of the decoder remains the same as in TransE case, but now the embeddings $\mathbf{z}_{\perp u}$ and $\mathbf{z}_{\perp v}$ are referring to the projections in the relation space [28].

$$\text{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) = -||\mathbf{z}_{\perp u} - \mathbf{r}_\tau + \mathbf{z}_{\perp v}|| \tag{3.27}$$

There are also translation-based embedding models that are defined by more complex decoders. One of these models is the *TransH* model, another extension of the basic TransE model. In the case of the TransH model, the decoder is defined as

$$\text{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) = -||(\mathbf{z}_u - \mathbf{w}_r^\top \mathbf{z}_u \mathbf{w}_r) + \mathbf{r}_\tau - (\mathbf{z}_v - \mathbf{w}_r^\top \mathbf{z}_v \mathbf{w}_r)|| \tag{3.28}$$

In this particular approach, the entity embeddings are being projected onto a trainable relation-specific hyperplane, which is defined by the vector $\mathbf{w}_r$, before applying the translation [45].

Multi-linear dot-product decoders

Another approach to learn multi-relational graph embeddings is by applying the dot-product operation to the embeddings of the graph nodes and relations. An example of such a model is the *DistMult* [50]. The decoder in this case performs the following operation

$$\text{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) =< \mathbf{z}_u, \mathbf{r}_\tau, \mathbf{z}_v > \tag{3.29}$$

The main limitation of this approach is its inability to handle anti-symmetric relations. Therefore, *ComplEx* extends the previous dot-product-based approach handling such relations [40]. ComplEx's decoder is defined as

$$\text{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) = \mathfrak{Re}(< \mathbf{z}_u, \mathbf{r}_\tau, \bar{\mathbf{z}}_v >) \tag{3.30}$$

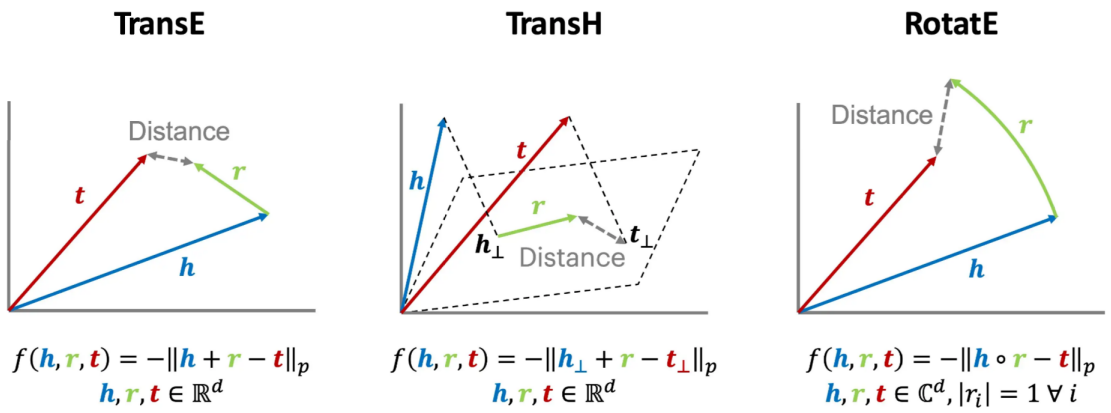


Figure 3.12: An illustration of the scoring functions of TransE, TransH and RotatE

where $\mathbf{z}_u, \mathbf{r}_\tau, \mathbf{z}_v \in \mathbb{C}^d$ are complex-valued embeddings and $\mathfrak{Re}$ denotes the real part of the complex vector. This approach can represent anti-symmetric relations effectively leveraging the complex conjugate $\bar{\mathbf{z}}_v$ of the embedding of the tail node. Another approach that utilizes the rotation operation on the complex plane is the *RotatE* [39] model, its decoder can be defined as:

$$\text{DEC}(\mathbf{z}_u, \tau, \mathbf{z}_v) = -||\mathbf{z}_u \circ \mathbf{r}_\tau - \mathbf{z}_v|| \tag{3.31}$$

where $\circ$ denotes the Hadamard product and the embeddings $\mathbf{z}$ are complex values as well.

Representational abilities

The aforementioned multi-relational models address the task of knowledge graph embedding learning utilizing different score functions in their decoder part. The selection of the score function and the vector space of the generated embeddings are the main factors that define their ability to represent specific types of relations. As a result, these decoders can be characterized by their ability to represent such relation types.

- **Symmetry**: A relation is symmetric when for each edge τ of this relation between two nodes u, v holds:

$$(u, \tau, v) \in \mathcal{E} \leftrightarrow (v, \tau, u) \in \mathcal{E} \tag{3.32}$$

- **Anti-Symmetry**: A relation is anti-symmetric when for each edge τ of this relation between two nodes u, v holds:

$$(u, \tau, v) \in \mathcal{E} \leftrightarrow (v, \tau, u) \notin \mathcal{E} \tag{3.33}$$

- **Inversion**: The characteristic of inversion suggests that the existence of an edge τ_1 of a certain relation between two nodes u, v implies the existence of an edge τ_2 of another relation between the same nodes

$$(u, \tau_1, v) \in \mathcal{E} \leftrightarrow (v, \tau_2, u) \in \mathcal{E} \tag{3.34}$$

- **Compositionality**: The attribute of compositionality indicates the ability of the decoder to handle the composition of two relation representations

$$(u, \tau_1, y) \in \mathcal{E} \wedge (y, \tau_2, v) \in \mathcal{E} \rightarrow (u, \tau_3, v) \in \mathcal{E} \tag{3.35}$$

- **1-to-N**: A relation is a 1-to-N relation if (u, τ, v_1) and (u, τ, v_2) can exist in the same relation.

Most of the decoders we described above can handle relations that are characterized by most of these traits, but not all of them. Table 3.1 summarizes the representational abilities of the decoders that are relevant to this study .

Model	Symmetry	Anti-Symmetry	Inversion	Composition	1-to-N
TransE	✗	✓	✓	✓	✗
TransR	✓	✓	✓	✓	✓
TransH	✓	✓	✓	✓	✓
DistMult	✓	✗	✗	✗	✓
ComplEx	✓	✓	✓	✗	✓
RotatE	✓	✓	✓	✓	✓

Table 3.1: A summary of the representational abilities of decoders

3.3 Noise in graphs

Incomplete, meaningless, distorted or corrupted data in datasets is known as noise. Noise can have a significant impact on the overall performance of a machine learning model [15]. Noise is commonly observed in various structured or unstructured data and graph-represented data are no exception. In the next sections we are going to give a brief overview of the forms of noise in graph-structured data as well as how graph machine learning models can be equipped with a layer of defense against these kinds of noise.

3.3.1 Forms of noise in graph-structured data

Noise in graphs can be attributed to various factors. In addition to that, the form in which noise can be observed in graphs can vary. It can be the result of an intended attempt to change the network's structure (e.g. adversarial attacks, bots on social networks) or it can emanate from imperfections of automated graph construction methods (e.g. automatic constructed knowledge graphs). The two main forms of noise are: i) Missing or redundant links between nodes (nodes of different feature distribution, classes, communities etc.). ii) Poisoned nodes (nodes with perturbed features or labels) attached to normal nodes that can harm the neighbour aggregation methods for representation extraction. Subsequently,

we are going to present a brief overview of the main factors leading to noise in graph structured data.

Automatic constructed knowledge graphs

Knowledge graphs are a very valuable data structure for many applications. In many cases, such graphs have to be constructed based on both structured and unstructured data. Recently, extracting graph representations from various non-graph-structured data sources has drawn significant attention due to the expressiveness graph representations can incorporate. By these means, the most informative entities and relations of the domain under investigation are being extracted and form a graph-structured representation. In Figure 3.13 an illustration of an automated knowledge graph extraction is presented [51].

On the other hand, automatically constructed knowledge graphs often contain noisy facts, since information extraction methods are imperfect [30]. Facts derived from automated knowledge graphs methods are often inaccurate, when extractors try to maximize rule coverage. This leads to relations containing large number of misleading links between entities or even be non-informative entirely [49].

Adversarial attacks on graphs

A fundamental case of artificial noise in graphs are adversarial attacks. Adversarial attacks aim to fool a neural network to output a wrong prediction with slight and unnoticeable perturbations to the input of the network, either the feature matrix of the nodes or the adjacency matrix of the graph [5] [20] . An example of adversarial attack on graph data is illustrated in Figure 3.14, where a specific node is targeted by the addition of a single edge. Many real-world graphs are naturally low-rank and sparse as the entities usually tend to form communities and would only be connected with a small number of neighbors [56]. Adversarial attacks on GCNs tend to add adversarial edges that link nodes of different communities as this is more efficient to reduce node classification performance of GNNs. Introducing links connecting nodes of different communities in a sparse graph can significantly increase the rank of the adjacency matrix, thus damaging the low rank and sparsity properties of graphs [20].

3.3.2 Noise-robust models

Despite the fact that Graph Neural Networks have exhibited great success in modeling graph-structured data, they have shown quite notable vulnerability to various types of noise. Noisy and adversarial edges, as well as poisoned or limited labeled nodes can lead to significant degradation of the GNN performance. This is mainly due to the nature of the message passing framework that these neural networks utilize [7]. Noise edges usually connect nodes of different classes or with different attributes, thus aggregating information of neighbouring nodes propagates errors, mixes useful information with noise and eventually leads to poor representations of the nodes of the graph [19].

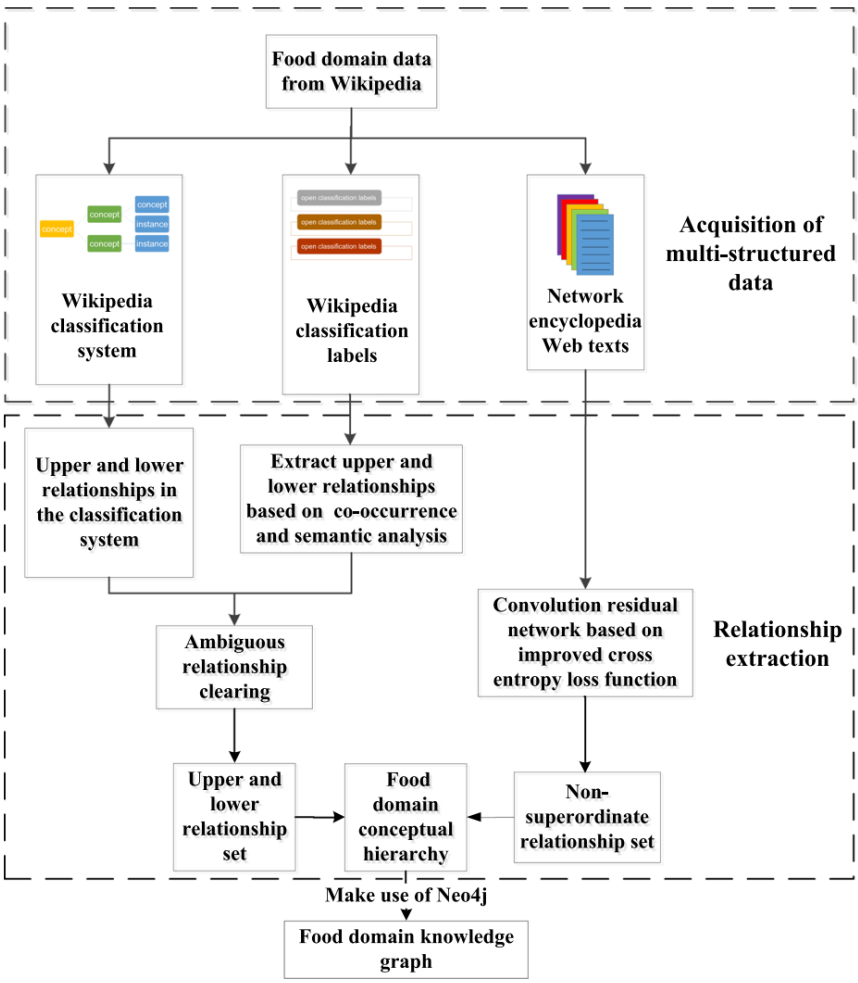


Figure 3.13: An example of automatic knowledge graph extraction - The roadmap of food domain knowledge graph from Wikipedia

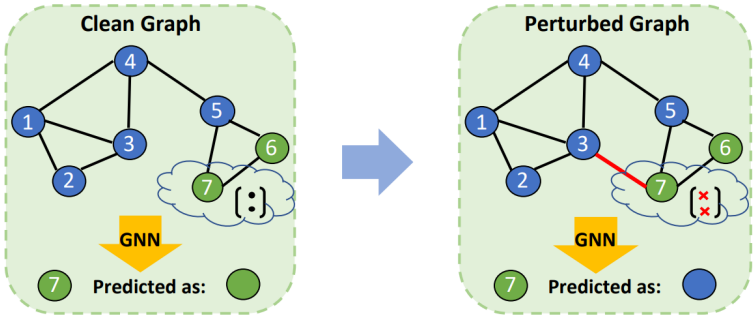


Figure 3.14: An example of adversarial attack on graph data

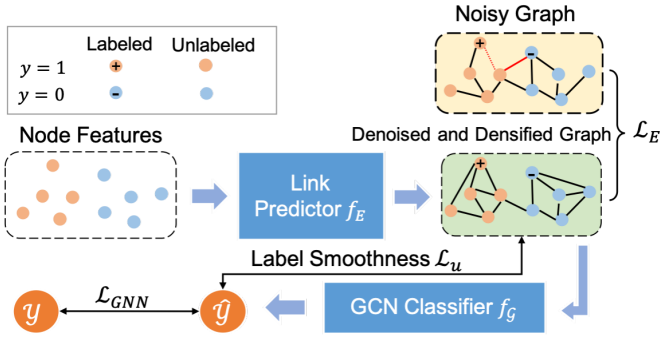


Figure 3.15: An illustration of Dai et al. link predictor - GCN approach for noisy graph representation learning

In order to defend the graph neural network against these kinds of noise, different countermeasures have been proposed, which are focusing on handling noise both during training and test time. Dai et al. [7] train an MLP on the task of link prediction. The link predictor learns to assign weights to every edge according to the feature similarity of the nodes connected by that particular edge. The link predictor is able to down-weight or eliminate noisy edges of the original graph. A side profit of this process is that it can also predict missing links, which is particularly useful in sparse labeled graphs. Then trains a GNN on the denoised and densified graph on a node classification task. The downside of this approach is that it overrates node similarity in terms of the node attributes and does not rely so much on the actual edges. It also depends on the expressiveness of the initial node representation. An illustration of this approach is presented in Figure 3.15.

Wang et al. [42] attempt to recognise noise by training two generator models, a graph generator and a noise generator, in an unsupervised setting. The graph generator's task is to identify normal structures. It captures useful graph prior knowledge and utilize it to generate normal graph structures. This prior heuristic knowledge comes in form of several graph-related statistics such as homophily, community, hierarchical structures and power-law degree distributions. The noise generator captures and generates arbitrary noises. It generates noisy graphs according to specific distributions that are distinguishable from the normal edges distribution. Eventually, both generators are optimized jointly through maximum likelihood estimation, to derive a noiseless adjacency matrix for the original graph. The proposed framework is illustrated in Figure 3.16.

Hafidi et al. [12] propose Bayesian node classification for node classifications tasks on noisy graphs. According to their approach, training a simplified graph-based Bayesian classifier on the intrinsic features of each node and those of its first-order neighbours, showing that it can ignore 1-hop noise neighbours.

Kang et al. [21] make the assumption that noisy graph matrices can be decomposed into two matrices, a clean and a noisy one. The decomposition is based on the fact that clean data form a low-rank matrix while the erroneous data form a sparse matrix. The

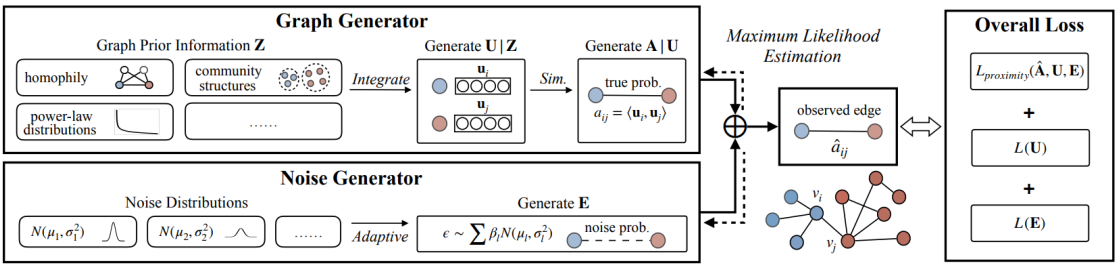


Figure 3.16: An illustration of Wang et al. generative approach to eliminate noise from noisy graphs

final graph is derived from the clean matrix using adaptive neighbours graph construction approach. The final and the clean graph are optimized simultaneously leveraging the alternating simulation approach.

Jin et al. [19] focused on robustness against adversarial attacks. They propose a model which tries to eliminate the crafted adversarial structure by iteratively reconstructing the clean graph by preserving graph characteristics such as low-rank, sparsity and feature smoothness.

Chapter 4

Methodology

In this chapter, we are going to describe a novel GNN-based architecture aiming to achieve expressive representation learning on heterogeneous graphs that shows robustness against potential presence of noise edges. In addition to that, we are going to explain the intuition behind this model and the particular settings in which it could be applicable.

4.1 Problem statement

The process of representation learning has been conducted in a supervised learning setting on a node classification task, as described in Section 2.2.1 . We define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes, and $\mathcal{E}$ is the set of edges. The edges of the graph are characterized by a type $r \in \mathcal{R}$, where $\mathcal{R}$ is the set of relation types of the graph $\mathcal{G}$. Every node v of the graph is associated with a feature vector$\mathbf{x}_v \in \mathbb{R}^M$. We can stack all the feature vectors in one tensor $\mathbf{X} \in \mathbb{R}^{M \times N}$ for the graph in total. Moreover, we can model all the edges present in the graph, through the adjacency tensor $\mathbf{A} \in \{0, 1\}^{N \times \mathcal{R} \times N}$. If $A[v_i, r, v_j] = 1$, then node v_i is connected to node v_j with an edge of type r.

For each node V of graph $\mathcal{G}$, our goal is to calculate the conditional output distribution of this node as follows:

$$p\left(y_v | \mathbf{X}, \mathbf{A}, \mathbf{\Theta}\right) \tag{4.1}$$

where y_v denotes the class of node v, $\mathbf{X}$ the attributes of the nodes of the graph and $\mathbf{\Theta}$ is a set of trainable parameters [44]. Our goal is to find a function $f(X, A, \Theta)$ that models the aforementioned probability density function. We formulate this task as an optimization problem that can be formulated by the following expression:

$$\min_{\Theta} \mathcal{L}(\mathbf{y}, f(\mathbf{X}, \mathbf{A}, \mathbf{\Theta})) \tag{4.2}$$

where $\mathcal{L}$ is called the objective function and could be any loss function that is applicable to the node classification task. In our case, the model is trained in to minimize a categorical cross-entropy loss given by the following formula:

$$CE(\mathbf{y}, f(\mathbf{x})) = -\sum_{i=1}^{C} \left(y_i \log f(x)_i\right) \tag{4.3}$$

where y_i and $f(x)_i$ are the actual and predicted values of the i-th class. The total loss over all training nodes is given by the following equation:

$$\mathcal{L} = \sum_{(\mathbf{x},\mathbf{y})\in\mathcal{T}} CE(\mathbf{y}, f(\mathbf{x})) \tag{4.4}$$

where $\mathcal{T}$ denotes the training set of nodes (i.e, all pairs of data and labels $(\mathbf{x}, \mathbf{y})$). The output of the model $f(\mathbf{x})$ is generated after the application of a softmax function to squash the output $g(x)$ in the range (0,1) and all the resulting outputs to add up to 1, in order to be consistent with the probability density function properties. The final output of the model will be:

$$f(\mathbf{x})_i = \frac{e^{g(x)_i}}{\sum_{j=1}^{C} e^{g(x)_j}} \tag{4.5}$$

where C is the number of classes and $g(x)_i$ denotes the i-th coordinate of the vector output of function $g(x)$.

We want to calculate the learnable parameters $\mathbf{\Theta}$ of the model in order to minimize the prediction loss in a setting where the input graph is a multi-relational heterogeneous graph.

4.2 Settings of noise

In this section, we are going to briefly introduce the settings in which we are going to infuse structural noise to the original graph. We are going to break down the experiments that have been conducted in three defined settings, that have been chosen in such a manner that cover the main aspects in which noise can be observed in a graph, whether it is related to an adversarial attack or the graph is noisy by its nature (e.g. derived from an automated graph representation process that created non-informational edges or relations), as described in Section 2.3.1. We are going to classify the existence of noise in graphs in the following categories.

Noisy relations

In this setting, the focus is on low-informational or even noisy relations. Some graphs can contain relations with edges connecting non-relevant nodes exclusively. This case is more common on automatically constructed knowledge graphs, if the relation extraction method is highly erroneous or a relation does not provide useful information about the nodes similarity by default. Ideally, we would eliminate this kind of relations, but training multiple models for different combinations of relations can be extremely expensive, even impossible for knowledge graphs with large number of relations, due to the exponential increase of the possible combinations. A robust model against this type of noise presence should minimize the effect of the noisy relations on the generated node embeddings.

Noise edges in informational relations

In this setting, we are focusing on the impact of noisy edges that belong to a relation that is informative in general. This can be the result of specific attacks to the graph structure, aiming to connect non-relevant nodes, or due to imperfections of an automated graph construction method that creates edges that should not exist. These edges harm the neighbour aggregation scheme, because non-relevant neighbours contribute to the update of the target node's representation. A robust model should be able to minimize the misleading information propagation of the noise related edges while effectively propagating useful information provided by the correct links.

4.3 Baseline models

In order to evaluate the performance of our proposed architecture, we selected two GNN architectures, which are among the most commonly-used models in the related literature for various graph representation learning tasks. The selection of these two models, except for providing a strong baseline performance for our architecture to overcome, is related to the fact that these two architectures are utilizing two different mechanisms that could potentially provide robustness against the various forms of noise presented in the previous section. Therefore, the choice of these GNN layers in particular aims to the investigation of their behaviour and potentially the evaluation of these mechanisms against intensive noise in general.

GCN - GraphSAGE

According to the internal architecture of the GCN (as described in detail in Section 2.2.3), in a heterogeneous input graph setting, each layer of this particular network architecture is equipped with a weight matrix $\mathbf{W}_r$ for each individual relation r . This weight matrix $\mathbf{W}_r$ is shared among edges of the same relation type $r \in R$. As a reminder, the representation update in a relational GCN layer is performed according to the following scheme:

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{u\in N(v)}\frac{1}{c_{v,r}}\mathbf{W}_r^{(l)}\mathbf{h}_u^{(l)} + \mathbf{W}_0^{(l)}h_v^{(l)}\right) \tag{4.6}$$

As we can see from the above expression, the representation of node v in the $(l+1)$-th layer is derived from an aggregation (sum) over the embeddings of the previous layer multiplied by the weights of the current layer for each relation. Although the RGCN keeps different weight matrices for each relation, this aggregation scheme could be vulnerable to noise relations, meaning that it may not have the ability to ignore such low-information or even noise relations. As far as edge addition on removal is concerned, RGCN is not providing a safety layer against such kind of attacks.

GraphSAGE is a general purpose GNN as well. The difference between GraphSAGE and GCN is the fact that the network aggregates messages of the neighbours in the first

place, the concatenates these representations with the target node representation of the previous layer. As far as its robustness against noise is concerned, it does not differ greatly compared to the GCN.

GAT

The second baseline model for the experimental procedure is the Graph Attention Network. Compared to the Graph Convolutional Network we mentioned above, GAT is used extensively in the related literature regarding noise robust graph neural networks [19] [7]. As mentioned in Section 2.2.3, the embedding update expression for a GAT layer is the following:

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right) \tag{4.7}$$

Respectively to the extension of GCN in heterogeneous graph-structured data, the GAT weight update in a heterogeneous setting looks like this:

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{r \in \mathcal{R}} \sum_{u \in N(v)_r} \alpha_{vu_r} \mathbf{W}_r^{(l)} \mathbf{h}_u^{(l-1)}\right) \tag{4.8}$$

From the above expression, the key difference of this neural network architecture and the vanilla RGCN is the existence of the attention weights α_{vu}. These weights are applied to each node of the neighbourhood of the target node. As far as noise is concerned, this particular network provides additional safety against attacks in the form of adversarial edge additions, since the model intuitively can assign little attention weights to these noisy edges. In addition to that, the GAT network also provides separate weight matrices for each relation type and applies an aggregation function to derive the final representation, identically to the R-GCN, which potentially can minimize the effect of low-information relations.

4.4 Proposed architecture

In this section, we are going to provide a detailed description of the proposed GNN-based architecture for the formulated problem, but also the intuition behind the proposed model named Split Relation Graph Convolutional Network (SRGCN for brevity).

4.4.1 SRGCN architecture

The SRGCN model consists of three main components, and the modelling procedure is illustrated in Fig. 4.1. The first one is a graph convolutional network. To be more precise, the model contains a GCN for each of the relations of the input graph. Each GCN utilizes as input the original graph but with only one type of relation present, and a representation is extracted for each node. This embedding represents the node for

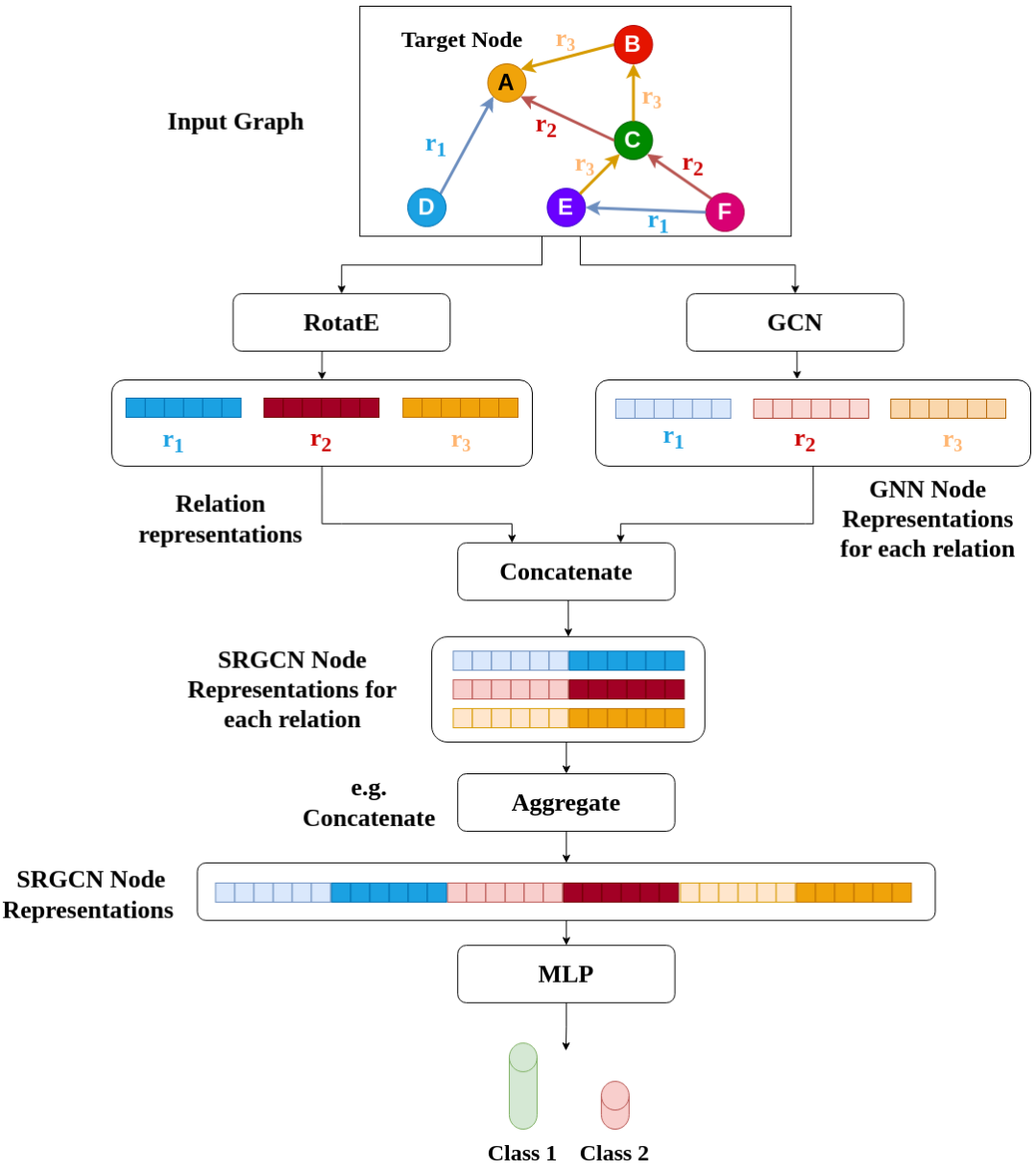


Figure 4.1: An illustration of the SRGCN Architecture

this specific relation. The second component is a shallow knowledge graph embedding model. The graph embedding model generates a representation (i.e. an embedding) for each relation of the original graph. Then, for each node, its representation per relation (from the GCN) and the corresponding relation representation (from the graph embedding model) are concatenated, creating a singular representation per different relation for the node. Finally, these representations are aggregated and are fed as input to a feed-forward neural network for the classification. The aggregation scheme of the model could be represented as follows:

$$AGGREGATE(\mathbf{h}^{L}_{u(r)}, \mathbf{h}^{GE}_{(r)}) \quad \forall u \in \mathcal{V} \tag{4.9}$$

where $AGGREGATE$ denotes an aggregation procedure (i.e. sum, mean, concatenation etc.), $\mathbf{h}^{L}_{u(r)}$ denotes the node representation for node $u \in \mathcal{V}$ of the final GNN layer L for the relation r and $\mathbf{h}^{GE}_{(r)}$ denotes the relation embedding extracted from the graph embedding model for the relation r in the relation set $\mathcal{R}$.

In our case, we opted for the concatenation operation as an aggregation function. The idea is to not blend the information for different relations at this stage, as would be the case through summation or taking the mean. Thus, the final MLP on top of these aggregated embeddings will be able to utilize the distinct representations generated by each relation, to perform the node classification class. Moreover, the graph embedding procedure used is RotatE due to its performance superiority, as also shown in the next chapter.

In-between the layers of both the GCN and the MLP components of the architecture there are non-linear activations (i.e., $ReLU$ function) to add non-linearity to the model, batch normalization (since it has proven to lead to better performance in various cases [34]) and dropout layers to prevent overfitting. The activation function of the final layer of the feed-forward neural network is a $softmax$ function.

4.4.2 Intuition behind SRGCN

The main concept of the proposed architecture is to leverage both the ability of the GNN architecture to capture neighbourhood-level dependencies and the graph-level information about each relation provided by a knowledge graph embedding model. SRGCN is relying on the GNN part to extract node representations. GNNs, via the neural message passing framework, can propagate information about the neighbour nodes of the target node. According to the related literature [19], standard GCN architectures are vulnerable to structural noise despite their great power in modelling graph structured data. Our motivation was to enrich the final representation by injecting general information about the whole graph and each relation in particular. This is happening via a knowledge graph embedding model, as presented in Section 2.2.5. Combining the node embeddings derived from the neighbourhood aggregation scheme of the GCN architecture with the relation embeddings of a graph embedding model can lead to an expressive representation that discriminates useful relations from low-informational or noisy ones, as well as adding more

useful information to the final representation. To verify the latter assumption, we conducted experiments to evaluate the utilization of a knowledge graph embedding method, such as RotatE, by comparing the results with a "more shallow" one, like a relation lookup embedding method. The results of the experiments on noisy graphs verify our assumption that relation lookup embeddings do not provide the necessary expressiveness to improve the model performance, in contrast to a relation representation extracted by a knowledge graph embedding method. Finally, we want to limit our model's reliance on initial node features. In the case of very noisy graphs, initial feature vectors tend to play a more important role, since structural information's value is reduced by noise. Additionally, node attributes are being used to identify noise edges by comparing the connected node feature similarity [7]. In our approach, we try to leverage the graph structure as much as possible and improve the performance by identifying relations affected by noise.

4.4.3 Model complexity

For a thorough comparison of the models, it makes sense to take their complexity into account as well. SRGCN and GCN include a learnable weight matrix $\mathbf{W}_r \in \mathbb{R}^{F \times H}$, where F is the number of input features and H is the hidden embedding size. And there is a distinct W_r for each relation r and in each layer (l). Considering a single layer for both models, the memory complexity will then be $\mathcal{O}_{GNN} = \mathcal{O}(RFH)$, where R is the number of relations and F, H as before. Thus, regarding the initial GNN-based node representation part, both models have the same number of parameters.

Then, for the SRGCN model we also concatenate these GNN-based embeddings with the derived relation embedding, for each relation. Thus, we must add the overhead of fitting the corresponding graph embedding model. Our selection was RotatE which has a computational complexity [52] of $\mathcal{O}_{GE} = \mathcal{O}(2VH_{GE} + 2RH_{GE})$, where V is the number of nodes in the graph and H_{GE} the selected embedding size for the graph embedding model. Having the relation embeddings from the graph embedding model, we can concatenate each relation-specific GNN-based embedding with the corresponding relation embedding. Each one of them has a size of $H_{GNN} + H_{GE}$. The GCN model has no added complexity in this step.

Model	**GNN Params**	**GE Params**	**MLP Params**
GCN	RFH_{GNN}	-	H_{GNN}
SRGCN	RFH_{GNN}	$2(VH_{GE} + RH_{GE})$	$R(H_{GNN} + H_{GE})$

Table 4.1: Number of Parameters for SRGCN versus GCN.

Finally, we have the memory complexity of the final MLP layer. The GCN aggregation function uses the *sum* operation, meaning that the final node representation is the sum of the distinct relation-based ones. Thus, the input dimension of the classification dense layer is H_{GNN} and the complexity of the MLP is $\mathcal{O}(H_{GNN})$. The SRGCN model, uses the

concatenation function as an aggregation operator for the distinct relation-based node-represenations. Thus, the input dimension of the classification dense layer is $R(H_{GNN} + H_{GE})$. Overall, their comparison in terms of parameters is shown in Table 4.1.

Chapter 5

Experiments and Results

In this chapter we are going to describe the experimental process we followed, give an overview of the datasets used for training and evaluation of the models and comment on the results and the insights derived. We will first introduce the evaluation metric used, then we will briefly describe the benchmark datasets and then present the predictive performance o the different models used. Specifically, we will evaluate the model both on the original datasets as and under two different noise scenarios. We will also conduct experiments on noisy synthetic graphs, to audit specific properties of mainstream models.

5.1 Evaluation metric

The models are being trained on the node classification task and evaluated according to the *F1-score* they achieve in the prediction of the unlabelled nodes' class. The F1-score is computed using the following formula:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.1}$$

where *Precision* and *Recall* of the model's predictions are computed as follows:

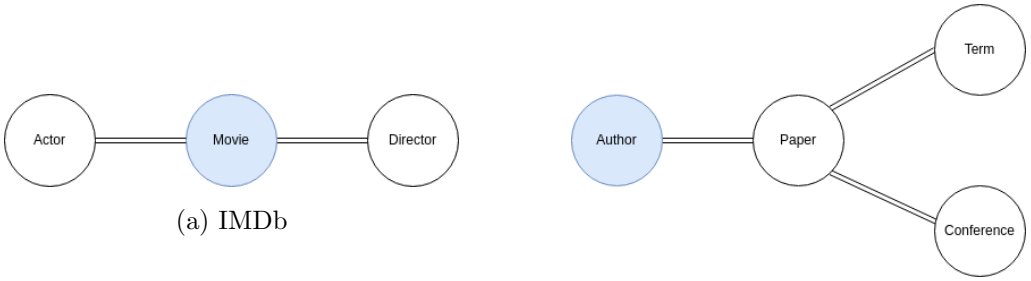
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.3}$$

where the *TP*, *FP*, *TN* and *FN* denote the true and false, positive and negative predictions in the test set respectively. To minimize the effect of randomly initialized weights of the neural networks and provide more robust results, every experiment result reported in this chapter is the average F1-score of 5 executions with different random seeds.

5.2 Benchmark datasets

We adopt two widely used heterogeneous graph datasets from different domains to evaluate the performance of the proposed and baseline models. An overview of the struc-



(a) IMDb

(b) DBLP

Figure 5.1: An Illustration of entities (classification entities in blue) and relations of the benchmark datasets

ture of each graph dataset is shown in Figure 5.1. A summary of the entities and relations of the graph datasets is presented in Table 5.1.

IMDb dataset

IMDb is an online database about movies and television programs, including information such as cast, production crew, and plot summaries. The dataset is a heterogeneous graph containing three types of entities - movies (4,278 nodes), actors (5,257 nodes), and directors (2,081 nodes). There are labels for the movie nodes. The movies are divided into three classes (action, comedy, drama) according to their genre. Additionally, it contains 2 types of relations - (actor-to-movie) and (director-to-movie). These relations are symmetric. Movie features correspond to elements of a bag-of-words representation of its plot keywords. Apart from the above, we also have feature vectors for the actor and director entities. For semi-supervised learning models, the movie nodes are divided into training, validation, and testing sets of 400 (9.35%), 400 (9.35%), and 3,478 (81.30%) nodes, respectively [10].

DBLP dataset

DBLP is a computer science bibliography website. This dataset is a heterogeneous graph containing four types of entities - authors (4,057 nodes), papers (14,328 nodes), terms (7,723 nodes), and conferences (20 nodes). The entities which we have to classify are the authors. The authors are divided into four research areas (database, data mining, artificial intelligence, information retrieval). In addition to that, the dataset contains 3 types of relations - (author-to-paper), (term-to-paper) and (conference-to-paper), all of them symmetric. Each author is described by a bag-of-words feature vector, consisting of their paper keywords. Additionally, the dataset provides feature vectors for the paper and term entities, but not for conference ones. We use a one-hot encoding scheme to represent the conferences. For semi-supervised learning models, the author nodes are divided into training, validation, and testing sets of 400 (9.86%), 400 (9.86%), and 3,257 (80.28%) nodes, respectively [10].

Dataset	Nodes	Edges
IMDb	# movie (M): 4278 # director (D): 2081 # actor (A): 5257	# M-D: 4278 # M-A: 12828
DBLP	# author (A): 4057 # term (T): 14328 # paper (P): 7723 # conference (C): 20	# A-P: 19645 # P-T: 85810 # P-C: 14328

Table 5.1: Summary statistics for benchmark datasets

5.3 Performance on benchmark datasets

Firstly, we compare the proposed methodology on the benchmark datasets versus some strong baseline GNNs. The baseline GNNs are the multi-relational variations of the GraphSAGE and GAT networks, referred from now on as *RGraphSAGE* and *RGAT* respectively. We opted for GraphSage layers instead of GCN ones, due to PyTorchGeometric [8] implementation's limitations[1] of the GCN in heterogeneous graphs with multiple node types. The proposed model also utilizes the GraphSAGE layer as convolutional layer for the same reason.

We trained the models for 100 epochs, keeping track of the best performing model and using that for evaluation. We used the best performing optimizer (*Adam* [25]) and the corresponding learning rates and weight decay, as found in the related literature [10]. In addition to that, hyperparameter tuning was conducted on the RGraphSAGE and RGAT model, in order to avoid adding bias to our proposed model results. The primary focus of the hyperparameter tuning was the number of convolutional and dense layers of the models, and in the case of RGAT also the number of attention heads. Only one dense hidden layer was used in all 3 models. For our model, we used the RotatE graph embeddings, which were trained for 50 epochs.

Model	F1-score	NN	RotatE	Total
MLP	0.487 ± 0.002	3.9	-	3.9
RGraphSage	0.576 ± 0.002	10.6	-	10.6
RGAT	0.525 ± 0.003	22.7	-	22.7
SRGCN	**0.584** ± 0.003	13.8	9	22.8

Table 5.2: F1-score and training time (s) for the IMDb dataset.

[1]Related Github Issue: https://github.com/pyg-team/pytorch_geometric/issues/4271

Model	F1-score	NN	RotatE	Total
MLP	0.744 ± 0.001	3.8	-	3.8
RGraphSage	0.912 ± 0.001	24.8	-	24.8
RGAT	0.916 ± 0.002	43.2	-	43.2
SRGCN	**0.924** ± 0.002	31.3	13	44.3

Table 5.3: F1-score and training time (s) for the DBLP dataset.

In Tables 5.2 and 5.3 we present the F1-scores measured for the results (average f1-score and a standard deviation) of the different models examined as well as the training times (in seconds) for both datasets. We provide the training time of the end-to-end node classifier (NN) and the RotatE model separately, as well as the total training time. From the above results we can conclude that:

- Our model achieves better performance than the baseline models, even by a slight margin. This encourages its use even as a general purpose model for node classification.
- The training time in total is comparable to the baselines even though the model's complexity is higher, mainly due to the knowledge graph embeddings training.

5.4 Robustness against noise

Apart from the ability of the SRGCN to perform better on the benchmark datasets, we also wanted to investigate how this architecture could provide solid representations in graphs with large amounts of noise edges. Our intuition is that incorporating information about the whole set of edges in a particular relation can help the overall model identify noisy relations and focus on the node representations derived from the useful relations. To verify our assumption, we conducted various experiments on different settings of noise. To speed up experimentation and to adapt to the needs of each experiment, we split the experimental process in two rounds, as described in the next sections.

5.4.1 Experiments on synthetic data

In the first round of experimentation on the impact of noise edges, experiments were conducted on synthetic data. The synthetic data can mimic operational or production data and help train machine and deep learning models [32]. The purpose of this experimental setup is to conduct experiments in order to gain insights on the effects of noise, the behaviour of the state-of-the-art GNN architectures and speed up experimentation, while maintaining control over the dataset due to its small size.

Synthetic graph overview

Our synthetic graph dataset consists of 50 nodes, all the same type. These nodes belong to 2 classes, each class having equal number of nodes (25 each). The graph contains a set of 16 labelled nodes, equally distributed into 2 classes (i.e. 8 each). The rest 34 unlabelled nodes constitute the test set, and they are equally split into the 2 classes as well (i.e. 17 each). The edges between the nodes and their corresponding relation type are not fixed and vary according to each experiment's needs. The representation of each node is initialized randomly according to a uniform distribution.The selected dimension of the initial vectors is 32.

Noise as separate relations

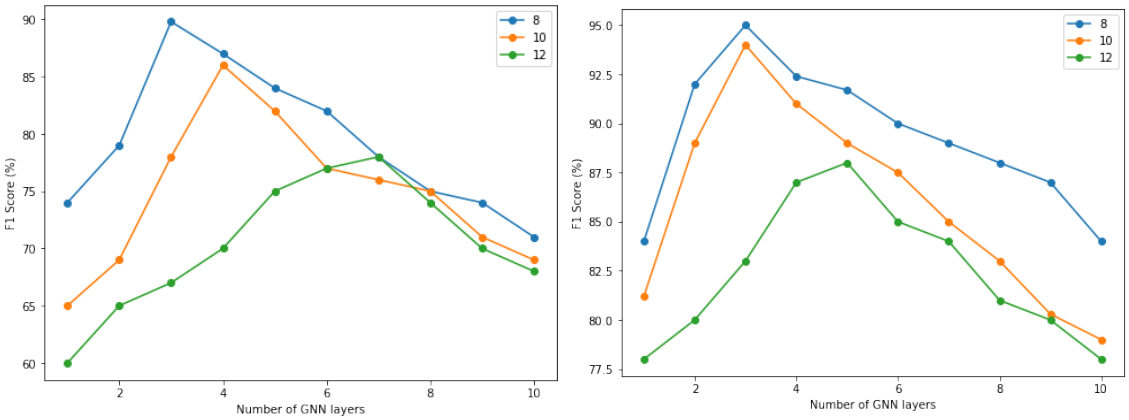
This experiment's goal is to examine the behaviour of GNNs when applied to graphs containing noise in the form of a distinct noise relation. That is when a relation is noisy by design. Adding noise to the graph intuitively should cause the performance of the GNN model to drop, except for the GNN having the ability to ignore the noisy relation and take into account only the neighbours according to the informational relation during the embeddings' learning process.

To evaluate the models' performance, we modified the synthetic graph as follows: First, we set an "informative" relation. Edges of this relation type connect nodes of the same class. Then, we created multiple "noise" relations. Edges of these relation types connect nodes of the opposite class. Initially, every labelled node was connected with 2 labelled nodes of the same class through the informative edge type and with 2 nodes of the opposite class for each noise relation. Similarly, every unlabelled node was connected with 2 labelled nodes of the same class for the informative relation, and with 2 labelled nodes of the opposite class for each of the noise relations. This way, every relation, informative or noise, has the same number of edges. With the aforementioned noisy graph structure, we examine the extent of noise that the GNNs can ignore, while focusing only the informative one.

In order to see the impact of noise, we evaluate the models while increasing the noisy edges. The addition of noise can be done in two ways:

- **Adding noisy edges in an existing uninformative relation**: The experiments in this setting showed that despite adding more noise edges in a single noise relation didn't cause any performance drop for the baseline GNNs. We added noise edges up to 8 times the number of the informative edges and all models maintained perfect accuracy. The GNNs have the ability to discriminate the informational from the noisy relations, when the number of noisy relations is kept low e.g. one noise relation.

- **Adding noise edges through additional noise relations**: In this setting, we progressively add more noise edges, but this time as additional noise relations. We observe that adding more noise relations causes the performance of the baseline



(a) GCN (b) GAT

Figure 5.2: An illustration of the impact of number of graph convolution layers in GNNs performance. Increasing number of layers to a certain point improves performance against noise. After this point further increase of the number of layers causes oversmoothing and performance drop.

models to drop, but our architecture maintains a nearly perfect accuracy despite the large number of noise relations. This behaviour is depicted in Figure 5.3.

It is also clear that increasing the number of convolutional layers improves the model performance until a certain point where oversmoothing (indistinguishable representations of nodes in different classes)[4] kicks in, as shown in Figure 5.2). The GCN model achieves the highest F1-score at higher number of layers compared to GAT, while GAT performs better with fewer layers. Another insight is that, adding more informational edges slows the degradation of the models accuracy, even if the noise-to-info ratio is kept the same. We increased the number of each relation's edges (informative and noise) attached to each node and we observed that more noise relations are needed to achieve the same performance drop. This can be seen in Figure 5.4. That indicates that, in very noisy graphs, a minimum number of "informative" edges is needed for the useful information propagation to be effective.

Graph Embeddings on noisy graphs

We also experimented on the synthetic dataset using different graph embeddings, to gain insights on the best performing embedding model under noise. The results of the experiments also encourage the use of encoder-decoder models for the extraction of the relation representations compared with the one-hot encoding of the relations. Previously, we mentioned that our proposed model managed to achieve almost perfect accuracy in every experiment, but that does not apply for every single decoder used. In Table 5.4 we present the accuracy of our proposed model according to the multi-relational decoder model used for the relation embedding learning. We experimented with two noise scenarios

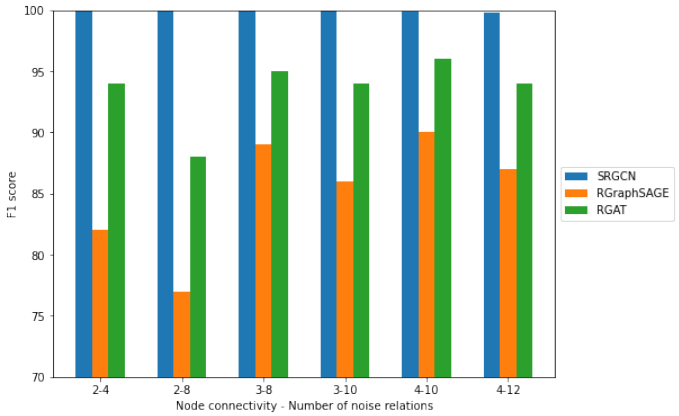
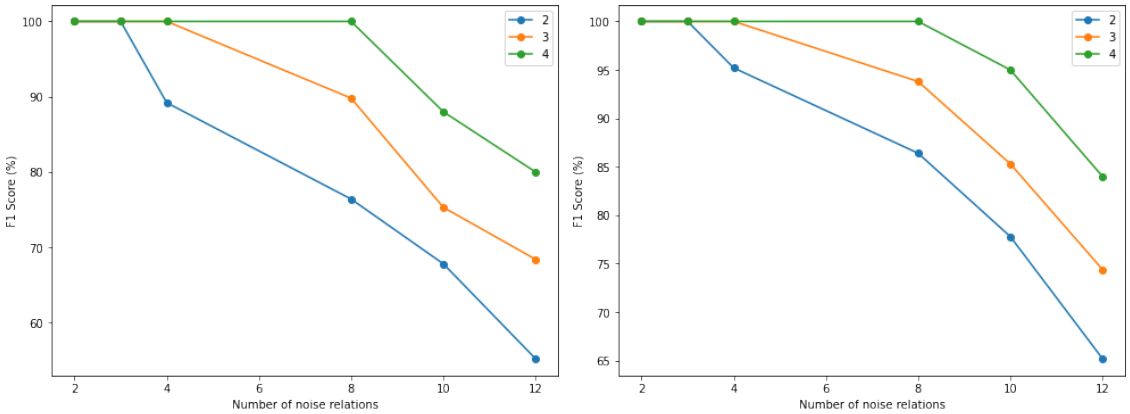


Figure 5.3: F1 scores on synthetic data, with noise introduced as separate relations scenario. The x-axis refers to different noise settings. Specifically, it is the combination of (Number of edges connected to each node for each relation - Number of noise relations).



(a) GCN (b) GAT

Figure 5.4: An illustration of the impact of number of informational edges in GNNs performance. Increasing the number of informational edges while keeping the info-to-noise edges ratio constant mitigates the effect of noise.

(8 and 12 noise relations respectively), which are somewhat extreme to accentuate the effect of noise on the embeddings created.

Model	2-8	3-12
Lookup	0.784	0.753
TransR	0.902	0.888
TransH	**1.0**	**1.0**
DistMult	0.847	0.821
RotatE	**1.0**	**1.0**

Table 5.4: F1-scores for different embedding models in SRGCN. The first number in the column name refers to the number of edges of each relation attached to each node the second to the number of noise relations in the graph.

From the results, it is clear that RotatE and TransH are clearly outperforming the rest of the models, which seem to lack the ability to produce expressive relation embeddings that handle noise as well.

Adding noise on the informational relation

Once again, our goal is to evaluate the performance of baseline GNNs on this particular setting, where noise is being added to the network not as a separate relation, but by polluting the informational relation. Adding edges of noise into the already existing informational relation should cause the accuracy of the model in the classification task to drop. Adding more noise should magnify the degradation of the model's performance. Since homogeneous graphs are not in scope for this thesis, we are creating two informative relations. Each node is connected with edges of each informational relations to 2 other nodes of the same class, leading to 8 informational edges in total for each class and for each relation. Noise is being added as edges connecting nodes of the opposite class, exactly as the previous case. The number of edges providing noise is used as a hyperparameter of this experiment.

Initially, the unlabelled nodes are connected only to nodes of the same class, with one edge for each relation. Gradually, we connect the unlabelled nodes with more labelled ones, this time not only of the same class but of the opposite class as well. The rate of noise to total edges attached to each node is treated again as a hyperparameter to measure the impact of the noise edges added to the graph. We conducted experiments to investigate GNNs behaviour in two major settings.

- **Noise on labelled nodes**: Initially, we want to investigate how noise can affect the training of a GNN. We are focusing on the labelled nodes in the first place. Unlabelled nodes are connected to labelled nodes of the same class. The results of the experiments are presented in Table 5.5. As we can see, noise starts to impact the

Noise-to-info rate	GCN	# Layers GCN	GAT	# Layers GAT
no noise	1.0	2	1.0	2
1	1.0	3	1.0	2
2	1.0	3	1.0	2
3	1.0	4	1.0	2
4	1.0	4	1.0	2
6	0.978	5	0.980	3
8	0.876	6	0.882	4

Table 5.5: F1-scores for the unlabeled nodes classification and the respective number of convolutional layers when noise edges are added only between labeled nodes.

models performance only when applied in massive amounts (6 and 8 times more than the informative edges). We deduce that the GNNs can learn to discriminate noisy information during training. Increasing the number of layers, improves the model performance until a certain threshold. Interestingly, even if the unlabelled nodes should be able to extract their true class from the 1-hop neighbours, the 1-layer GNNs do not perform well.

- **Noise on unlabelled nodes**: The aforementioned results stress the fact that when no noise is added between unlabelled nodes, the GNNs do not have significant degradation in their performance. This time we are going to add noisy edges to the unlabelled nodes as well to evaluate the impact of noise in a more general and realistic scenario. We extended the previous setup by connecting unlabelled nodes to nodes of the opposite class to simulate a noisy setting. The results of our experimentation are presented in Table 5.6. It is obvious that noise in this setting really hurts the GNN performance, with way less additional noise. The number of informative edges greatly impacts the model the same way as in the setting where noise was added as separate relations. This can be seen from the great improvement in results in the last row of the table, where the ratio of informative to noisy increased to 2:1. Increasing the number of convolutions to a specific number also improves the performance of the model to a certain degree, as in the previous noise scenario.

5.4.2 Experiments on real-world datasets

Experiments on the synthetic dataset provide useful insights about the behaviour of noise in graph structured data, as well as an indication of the performance of the different models when dealing with noisy graphs. In this part of the experimental process, the main focus is to evaluate the proposed architecture, as well as the two other competing GNNs, but we pay significant attention to gaining further insights on the models' be-

L-L	U-L	GCN	# Layers GCN	GAT	# Layers GAT
2-2	1-1	0.657	6	0.668	4
2-2	2-2	0.697	6	0.693	4
2-2	1-2	0.958	3	0.938	2

Table 5.6: F1-scores for the unlabelled nodes classification and the respective number of convolutional layers. L-L refers to the number of edges from (L)abelled to (L)abelled nodes, and U-L refers to (U)nlabelled to (L)abelled. The first number in these columns denotes the number of noisy edges and the second of informative ones, per node.

haviour leveraging the knowledge derived from the synthetic data experiments. For the sake of fairness, we present the F1-scores for two variants of the RGraphSAGE and RGAT models. In the first one, the node embeddings generated by the graph convolutional layer are 64-dimensional embeddings, as in the SRGCN. In the second one, the node embeddings generated by the graph convolutional layer are 128-dimensional embeddings, which produced the best results overall. For brevity, SAGE and GAT refer to the RGraphSAGE and RGAT models introduced before.

Add noise as separate relations

In this particular setting, noise is integrated to the graph as a separate relation. Specifically, we create relations with random connections between existing entities and we are going to progressively add more of these relations. We create noise relations that are similar to the actual ones, i.e., between the same entity pairs. For instance, in IMDb dataset we added noise relations Movie-noiseTo-Actor and Movie-noiseTo-Director since such relations existed in the initial dataset. We did not create noise relation between Actor and Director entities since there wasn't similar relation in the original dataset. The number of noise edges in each noise relation are set to be the same as the number of edges in the informational relations. In contrast to the synthetic data, noise relation number will be kept in lower levels, since high number of noise relations is quite uncommon in the real-world.

Noise relations	SAGE-64	SAGE-128	GAT-64	GAT-128	SRGCN
no noise	0.575	0.576	0.523	0.525	**0.584**
1	0.537	0.539	0.491	0.494	**0.581**
2	0.516	0.521	0.450	0.457	**0.576**

Table 5.7: F1-scores for the IMDb dataset, with multiple noise relations.

In the case of multiple noise relations, from the results presented in Table 5.7 and

Noise-to-info ratio	SAGE-64	SAGE-128	GAT-64	GAT-128	SRGCN
no noise	0.575	0.576	0.523	0.525	**0.584**
1	0.537	0.539	0.491	0.494	**0.581**
2	0.532	0.535	0.478	0.480	**0.579**

Table 5.8: F1-scores for the IMDb dataset, with a single noise relation.

Noise relations	SAGE-64	SAGE-128	GAT-64	GAT-128	SRGCN
no noise	0.905	0.912	0.91	0.916	**0.924**
1	0.857	0.869	0.858	0.867	**0.904**
2	0.764	0.776	0.791	0.799	**0.890**

Table 5.9: F1-scores for the DBLP dataset, with multiple noise relations.

Noise-to-info ratio	SAGE-64	SAGE-128	GAT-64	GAT-128	SRGCN
no noise	0.905	0.912	0.91	0.916	**0.924**
1	0.857	0.869	0.858	0.867	**0.904**
2	0.824	0.834	0.813	0.821	**0.892**

Table 5.10: F1-scores for the DBLP dataset, with a single noise relation.

Table 5.9 we observe that our model outperforms significantly its competitor models as the number of noisy relations increases. This behaviour is consistent with our observations in the synthetic data. This time, just a few noise relations cause the performance of the other GNN models to drop, in comparison to the synthetic dataset, where the performance drop occurred when more than 5 noise relations where introduced. Another difference with the toy dataset is the fact that increasing the number of layers didn't improve the performance of the models. On the contrary, oversmoothing of the node representations happened after 4-5 convolutional layers for the IMDb dataset, and 6-7 for the DBLP dataset.

As far as the single noise relation case is concerned, we observed similar results (Table 5.8 and Table 5.10) with the multiple noise relations case, as well as with the corresponding results of the synthetic dataset. In comparison to the toy dataset, the performance drop was more significant in the case of the RGraphSAGE and the RGAT models, but not as much as in the previous case. It is obvious that adding the same amount of noisy edges in the form of multiple relations is causing bigger performance drop than in the form of a single relation.

An interesting observation is the poor performance of the GAT-based architecture in the IMDb dataset, which was not expected according to the synthetic data experiments.

The behaviour of the attentional network in the DBLP dataset was more consistent, but still not better than the ones of the RGraphSAGE.

Adding noise to the informational relation

The experiments in this stage are focusing on the existence of noise not as separate relation, but in an existing one. We conducted experiments with different noise distributions among the existing relations in the benchmark datasets. We added noise edges in each relation separately, as well as, in multiple relations at the same time, and compared the performance of the models in each setting. The results of the experiments for the IMDb and DBLP are shown in the Tables 5.11 and 5.12 respectively.

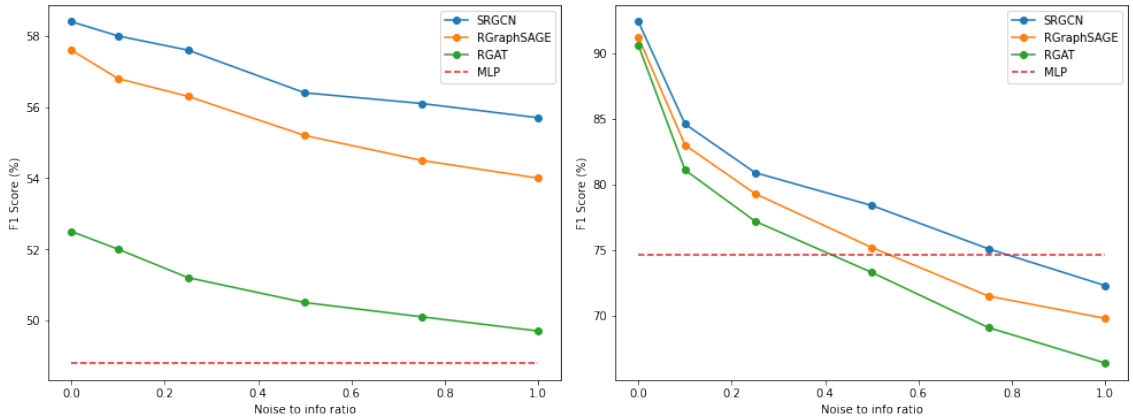
MA-MD	$\|E\|$	**SAGE-64**	**SAGE-128**	**GAT-64**	**GAT-128**	**SRGCN**
no noise	0	0.575	0.576	0.523	0.525	**0.584**
1 - 0	12828	0.551	0.56	0.497	0.499	**0.571**
0 - 1	4278	0.535	0.541	0.494	0.497	**0.552**
.1 - .1	1711	0.543	**0.549**	0.487	0.489	0.546
.5 - .5	8553	0.529	0.532	0.472	0.476	**0.534**

Table 5.11: F1-scores for the IMDb dataset, with noise edges in informational relations. $|E|$ denotes the number of noise edges added to the original graph.

AP-PT-PC	$\|E\|$	**SAGE-64**	**SAGE-128**	**GAT-64**	**GAT-128**	**SRGCN**
no noise	0	0.905	0.912	0.91	0.916	**0.924**
1 - 0 - 0	19645	0.657	0.698	0.656	0.664	**0.722**
0 - 1 - 0	85810	0.903	0.909	0.861	0.871	**0.921**
0 - 0 - 1	14328	0.857	0.875	0.768	0.801	**0.892**
.1 - .1 - .1	11978	0.874	0.886	0.853	0.862	**0.889**
.5 - .5 - .5	59891	0.684	0.704	0.678	0.684	**0.707**

Table 5.12: F1-scores for the DBLP dataset, with noise edges in informational relations. $|E|$ denotes the number of noise edges added to the original graph.

First, some explanations about the notation of the experiments. The first column of each table denotes the type of the relation, as defined in Table 5.1. The numbers in the first column denote the ratios of noisy edges added with respect to the normal edges. The total number of noisy edges added is shown in the second column. We conducted experiments both by adding noise only to one relation at a time and by adding noise to all relations simultaneously.



(a) IMDb (b) DBLP

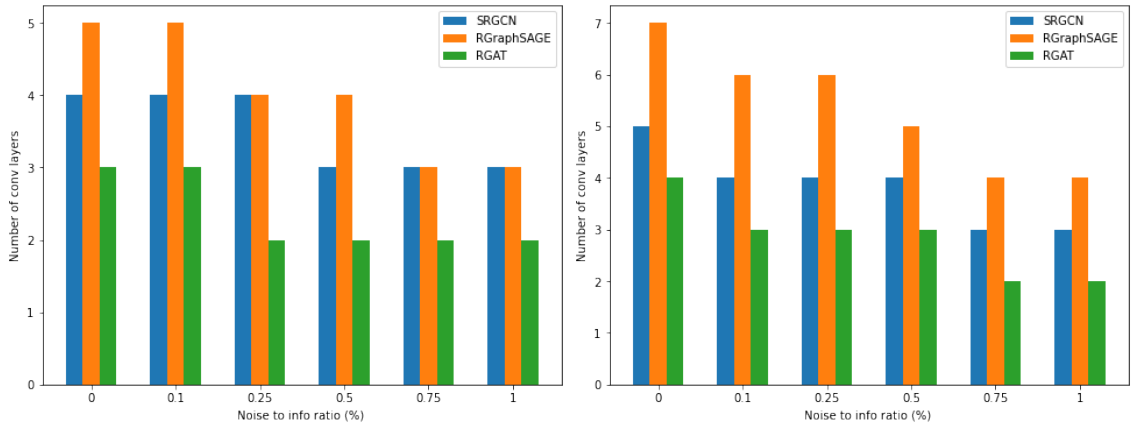
Figure 5.5: Predictive performance of the models according to added noise percentage.

As expected, the presence of noisy edges causes the accuracy of the models' predictions to drop. AM and TP have the largest number of edges in the IMDb and DBLP dataset respectively. Nevertheless, the performance drop in these cases is not the highest. In fact, DM (in IMDb) and AP (in DBLP) are the relations that, when polluted with noise edges, impair the models' performance the most, even though the number of added noisy edges in these experiments is significantly lower. It is obvious that the number of noise edges is not the most important factor for the impact of noise, but it is rather how informative the polluted relation is. The SRGCN performs generally better across all types and levels of noise. Specifically, it is significantly better than the baseline models when only one relation is polluted with noise.

In Figure 5.5 we present the performance of each model for various values of added noise in the most informative relations (Movie-to-Director for IMDb and Author-to-Paper for DBLP). We also compare the results with a structure-agnostic MLP model. We can see that generally the SRGCN model outperforms the baseline models for all noise-to-info ratios. Interestingly, for the DBLP dataset, the baseline models start to have worse performance than the MLP when noisy edges are more than 50% of the informational edges, while SRGCN outperforms the MLP until the 75% noise-to-info ratio point.

Adding noise to multiple relations simultaneously causes performance drop that is proportionate to the importance of each relation. We observed very minor performance gains for the SRGCN compared to its competitors. This is expected, since the advantage of SRGN stems from its ability to incorporate information for each relation as a whole. By polluting every relation, every relation representation is distorted, and therefore this advantage is mitigated.

Another insight from the experiments is the fact that GNNs with smaller number of layers performed better when increasing the proportion of noise. This can be explained if we take into account the neighbour aggregation scheme of the GNNs. By adding more layers, the neighbourhood that contributes to the final representation of each node becomes



(a) Number of layers - IMDb

(b) Number of layers - DBLP

Figure 5.6: Number of convolution layers for the best model, per model category, according to added noise percentage.

bigger. As a result, in noisy-heavy graphs, the graph structure provides little information and the GNNs integrate more noise into their representation. A more detailed illustration of this behaviour is shown in Figure 5.6. We can easily observe the general trend, that fewer layers are better when dealing with more noisy graphs, which is consistent across all models and both datasets. It is interesting that the optimal number of layers for a given noise-to-info ratio is different for each model and dataset, pointing out that both the particularities of the graphs and the expressive power of each model play a significant role on this choice. GAT achieves best performance with the smallest number of layers, which is consistent with out synthetic data experiments. SRGCN, despite utilizing the same graph convolution layer with RGraphSAGE (i.e. GraphSAGE) achieves best performance with fewer layers than RGraphSAGE.

Chapter 6

Conclusions

6.1 Summary

In this thesis, we proposed a complex architecture based on Graph Convolutional Neural Networks and graph embeddings, to tackle the node classification task in multi-relational networks, focusing on graphs with noise. Our findings lead to the conclusion that incorporating relation-level information in a GNN based model can boost the expressiveness of the representations and improve the performance of the whole model.

Moreover, our experiments on graphs with different noise settings provided useful insights about the behaviour of widely used GNNs and their robustness against large amounts of noisy edges. We identified settings of noise that harm the performance of these models, such as arbitrary noise in specific or multiple relations.

Our main contribution is the proposed architecture, SRGCN, which achieved better performance on the node classification task on benchmark datasets. Moreover, it significantly outperformed strong baseline models in all the scenarios where noise was introduced in the graph. Specifically, our architecture suffered almost no performance drop when we added relations that contained only noise and excelled when we introduced noise by polluting a specific relation. Thus, we proved empirically the main intuition behind our model, that the extra information introduced by the global relation embeddings of SRGN helped our model identify useful relations from redundant ones.

6.2 Future work

This thesis offers some potential extensions for further experimentation. As far as the architecture is concerned, experimentation on the GNNs used in the SRGCN architecture, apart from the GCN and GraphSAGE, would be particularly interesting since it could potentially lead to significant performance improvement. We also decided to use the concatenation of the embeddings derived from the GNNs and decoders as an aggregation function. Further experimentation in the aggregation function selection could provide useful insights or even performance improvement. Apart from that, it would be interesting

to evaluate our model's performance on more settings of noise. For example, in the future, we could evaluate the robustness of the different models in situations where noise is targeting specific nodes. This could be helpful to simulate some realistic scenarios where we have adversarial attacks on compromised nodes of the graph. Finally, since our model showed robust results on these noise settings, we consider purposeful to compare our model to other methodologies that are specifically designed to improve performance on noisy graphs.

List of Figures

List of Tables

Bibliography

[1] U. Akujuobi, H. Yufei, Q. Zhang, and X. Zhang. Collaborative graph walk for semi-supervised multi-label node classification. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1–10. IEEE, 2019.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[4] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3438–3445, 2020.

[5] L. Chen, J. Li, J. Peng, T. Xie, Z. Cao, K. Xu, X. He, and Z. Zheng. A survey of adversarial learning on graphs. *arXiv preprint arXiv:2003.05730*, 2020.

[6] G. Ciano, A. Rossi, M. Bianchini, and F. Scarselli. On inductive–transductive learning with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):758–769, 2021.

[7] E. Dai, W. Jin, H. Liu, and S. Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 181–191, 2022.

[8] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[9] J. Fox and S. Rajamanickam. How robust are graph neural networks to structural noise? *arXiv preprint arXiv:1912.10206*, 2019.

[10] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[12] H. Hafidi, M. Ghogho, P. Ciblat, and A. Swami. Bayesian node classification for noisy graphs. In *2021 IEEE Statistical Signal Processing Workshop (SSP)*, pages 246–250. IEEE, 2021.

[13] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] W. L. Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[15] R. Hasan and C. Chu. Noise in datasets: What are the impacts on classification performance?[noise in datasets: What are the impacts on classification performance?]. In *Proceedings of the 11th International Conference on Pattern Recognition Applications and Methods*, 2022.

[16] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[17] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[18] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2021.

[19] W. Jin, Y. Li, H. Xu, Y. Wang, S. Ji, C. Aggarwal, and J. Tang. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

[20] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[21] Z. Kang, H. Pan, S. C. Hoi, and Z. Xu. Robust graph learning from noisy data. *IEEE transactions on cybernetics*, 50(5):1833–1843, 2019.

[22] N. Keerthana, V. Vinod, and S. Sudhakar. A novel method for multi-dimensional cluster to identify the malicious users on online social networks. *Journal of Engineering Science and Technology*, 15(6):4107–4122, 2020.

[23] M. Kejriwal, C. A. Knoblock, and P. Szekely. *Knowledge graphs: Fundamentals, techniques, and applications*. MIT Press, 2021.

[24] S. Khoshraftar and A. An. A survey on graph representation learning methods. *arXiv preprint arXiv:2204.01855*, 2022.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[27] Z. Li, H. Liu, Z. Zhang, T. Liu, and N. N. Xiong. Learning knowledge graph embedding with heterogeneous relation attention networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(8):3961–3973, 2021.

[28] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[29] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications*, 390(6):1150–1170, 2011.

[30] M. Masoud, B. Pereira, J. McCrae, and P. Buitelaar. Automatic construction of knowledge graphs from text and structured data: A preliminary literature review. In *3rd Conference on Language, Data and Knowledge (LDK 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[31] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, pages 415–444, 2001.

[32] U. Meyer and M. Penschuck. Generating synthetic graph data from random network models. In *Algorithms for Big Data: DFG Priority Program 1736*, pages 21–38. Springer, 2023.

[33] L. D. Sailer. Structural equivalence: Meaning and definition, computation and application. *Social networks*, 1(1):73–90, 1978.

[34] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[35] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[36] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

[37] D. Stepanova, M. H. Gad-Elrab, and V. T. Ho. Rule induction and reasoning over knowledge graphs. *Reasoning Web. Learning, Uncertainty, Streaming, and Scalability: 14th International Summer School 2018, Esch-sur-Alzette, Luxembourg, September 22–26, 2018, Tutorial Lectures 14*, pages 142–172, 2018.

[38] Y. Sun and J. Han. Mining heterogeneous information networks: principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012.

[39] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[40] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[41] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[42] J. Wang, Z. Li, Q. Long, W. Zhang, G. Song, and C. Shi. Learning node representations from noisy graph structures. In *2020 IEEE international conference on data mining (ICDM)*, pages 1310–1315. IEEE, 2020.

[43] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[44] Y. Wang, W. Wang, Y. Liang, Y. Cai, J. Liu, and B. Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 207–217, 2020.

[45] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[46] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[47] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2(2):109–127, 2021.

[48] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[49] J. Yan, C. Wang, W. Cheng, M. Gao, and A. Zhou. A retrospective of knowledge graphs. *Frontiers of Computer Science*, 12:55–74, 2018.

[50] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[51] H. Yu, H. Li, D. Mao, and Q. Cai. A relationship extraction method for domain knowledge graph construction. *World Wide Web*, 23:735–753, 2020.

[52] M. Zamini, H. Reza, and M. Rabiei. A review of knowledge graph link prediction using graph neural networks. 2022.

[53] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803, 2019.

[54] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23, 2019.

[55] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[56] K. Zhou, H. Zha, and L. Song. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *Artificial Intelligence and Statistics*, pages 641–649. PMLR, 2013.

[57] X. J. Zhu. Semi-supervised learning literature survey. 2005.