ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗΝ
ΑΝΑΛΥΤΙΚΗ ΤΩΝ ΕΠΙΧΕΙΡΗΣΕΩΝ ΚΑΙ ΕΠΙΣΤΗΜΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Διπλωματική Εργασία

ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΓΡΑΦΩΝ ΣΤΗΝ ΕΠΙΧΕΙΡΗΜΑΤΙΚΗ ΑΝΑΛΥΤΙΚΗ

της

ΚΙΛΙΚΙΔΟΥ ΜΑΡΙΑ του ΓΕΩΡΓΙΟΥ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ

ΚΑΛΑΜΠΟΚΗΣ ΕΥΑΓΓΕΛΟΣ,

ΚΑΘΗΓΗΤΗΣ ΤΜΗΜΑΤΟΣ

ΟΡΓΑΝΩΣΗΣ & ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

Μάρτιος 2023

THANKS TO

Before presenting the results of this thesis, I feel obliged to thank all those who helped me in conducting and completing. First and foremost, I would like to express my heartfelt gratitude to my supervisor, Professor Kalampoki Evangelo. His guidance and advice were critical in completing this thesis. Last but not least, I'd like to thank my family and friends for their patience and unwavering faith in me.

ΕΥΧΑΡΙΣΤΙΕΣ

Πριν από την παρουσίαση των αποτελεσμάτων της παρούσας διπλωματικής εργασίας, αισθάνομαι την υποχρέωση να ευχαριστήσω όλους όσους με βοήθησαν στη διεξαγωγή και την ολοκλήρωση της. Πρώτα απ' όλα, θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη στον επιβλέποντα καθηγητή μου, κ. Καλαμπόκη Ευάγγελο. Η καθοδήγηση και οι συμβουλές του ήταν καθοριστικές για την ολοκλήρωση της παρούσας διπλωματικής εργασίας. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για την ακλόνητη πίστη τους στο πρόσωπό μου.

ABSTRACT

The aim of this thesis is to study existing solutions to the problem of link prediction in Graph Neural Networks (GNN), focusing on the Wikidata dataset. Source code analysis of one of the solutions is also presented. The data covers all items of human entities contained in Wikidata. These were taken from the Open Graph Benchmark (OGB) data repository and retrieved using the ogb_wikikg2 library. The prediction code used, leverages 2.500.604 entities and 17.137.181 edges between them. The analyzed algorithm is written in Python.

The research questions of the paper focus on the following: (1) How feasible is it to retrieve a subset of the ogb_wikikg2 dataset using query. (2) Is it feasible to run the algorithm on a basic computer? (3) What are the necessary adjustments to run the algorithm.

In the first part of this thesis, we present the OGB repository, the GNN, Wikidata and the description of the methodology followed. This is followed by the creation of a suitable query, the result of which simulates the data used in the model. This is followed by an analysis of the steps of preprocessing the data and predicting using the Mean Reciprocal Rank (MRR) metric through a variety of GNN methods. We present the prediction results of the existing model, taken repository of OGB. Finally, we present the tuning to the algorithm parameters. Tuning was applied to find the optimal MRR value, using parameters that correspond to our computational power.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική επιδιώκει τη μελέτη των υπαρχουσών λύσεων στο πρόβλημα της πρόβλεψης των ακμών στα Νευρωνικά Δίκτυα Γράφων (GNN), εστιάζοντας στο σύνολο δεδομένων της Wikidata. Παρουσιάζει ακόμη την ανάλυση του πηγαίου κώδικα σε μια εκ των λύσεων. Τα δεδομένα περιέχουν όλα τα item των ανθρώπινων οντοτήτων που περιέχονταν στη Wikidata. Αυτά προήλθαν από το αποθετήριο δεδομένων Open Graph Benchmark (OGB) και ανακτήθηκαν μέσω της βιβλιοθήκη ogb_wikikg2. Ο κώδικας πρόβλεψης που χρησιμοποιήθηκε, αξιοποιεί 2.500.604 οντότητες και 17.137.181 σε αριθμό ακμές μεταξύ αυτών. Ο αλγόριθμος που αναλύθηκε είναι σε γλώσσα Python.

Τα ερευνητικά ερωτήματα της εργασίας επικεντρώνονται στα εξής: (1) Πόσο εφικτή είναι η ανάκτηση ενός μέρους του συνόλου δεδομένων ogb_wikikg2, με τη χρήση query. (2) Είναι εφικτό να λειτουργήσει ο αλγόριθμος σε έναν απλό υπολογιστή. 3) Ποιες είναι οι απαραίτητες ρυθμίσεις για να τρέξει ο αλγόριθμος.

Στο πρώτο μέρος της εργασίας μας παρουσιάζεται το αποθετήριο OGB, τα GNN, η Wikidata και η περιγραφή της μεθοδολογίας που ακολουθήθηκε. Έπειτα ακολουθεί η δημιουργία κατάλληλου query, το αποτέλεσμα του οποίου προσομοιάζει τα δεδομένα που χρησιμοποιούνται στο μοντέλο. Ακολουθεί η ανάλυση των βημάτων προεπεξεργασίας των δεδομένων και η πρόβλεψη μέσω της μετρικής Mean Reciprocal Rank (MRR), μέσω ποικίλων μεθόδων του GNN. Παρουσιάζουμε τα αποτελέσματα της πρόβλεψης του ήδη υπάρχοντος μοντέλου, που πάρθηκε από το αποθετήριο του OGB. Τέλος, παραθέτουμε τα αποτελέσματα του tuning στις παραμέτρους του αλγορίθμου. Το tuning εφαρμόστηκε για να βρεθεί η βέλτιστη τιμή της μετρικής, με τη χρήση παραμέτρων που ανταποκρίνεται στη δική μας υπολογιστική ισχύ.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΈΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1 Εισαγωγή

Πληθώρα καθημερινών ζητημάτων απαιτούν την επεξεργασία των δεδομένων με γράφο, οι οποίοι παρέχουν χρήσιμες πληροφορίες σχετικά με τα στοιχεία και τις σχέσεις τους. Προβλήματα όπως η εκμάθηση μοριακών αποτυπωμάτων, η πρόβλεψη πρωτεϊνικών διεπαφών και η ταξινόμηση ασθενειών, δημιουργούν την ανάγκη κατασκευής ενός μοντέλου εκμάθησης. Τα Νευρωνικά Δίκτυα Γράφων (GNN) έχουν λάβει αυξημένη προσοχή στη Μηχανική Μάθηση (Machine Learning ML) και την τεχνητή νοημοσύνη, λόγω των ελκυστικών ιδιοτήτων τους για τη μάθηση δομημένων δεδομένων με γράφους. Τα GNN είναι νευρωνικά μοντέλα που αποτυπώνουν την μετάδοση μηνυμάτων μεταξύ των στοιχείων.

Παρόλο που τα GNN έχουν υψηλές επιδόσεις και είναι ευρέως διαδεδομένα, δεν παραβλέπετε το μειονέκτημα που μοιράζονται με άλλα μοντέλα βαθιάς μάθησης, δηλαδή η αδυναμία εξήγησης και κατανόησης τους από τον άνθρωπο. Χωρίς τη δυνατότητα κατανόησης αλλά και επαλήθευσης των εσωτερικών μηχανισμών λειτουργίας, καθίσταται δύσκολη η εφαρμογή τους σε ζητήματα που αφορούν τη δικαιοσύνη, την ιδιωτικότητα και την ασφάλεια. Αυτό εγείρει την ανάγκη της ανάπτυξης τεχνικών ερμηνείας για τα GNN.

Ένα από τα προβλήματα που μπορούμε να αναγάγουμε στα GNN, είναι η αναπαράσταση, η επίλυση, και η πρόβλεψη σε δεδομένα της Wikidata. Η Wikidata είναι ένας γράφος γνώσης, στο οποίο περιέχονται οντότητες του πραγματικού κόσμου και οι σχέσεις μεταξύ αυτών.

1.1 Σκοπός της Διπλωματικής

Σκοπός της παρούσας διπλωματικής είναι η μελέτη του προβλήματος της πρόβλεψης των ακμών στα δεδομένα της ogb-wikikg2, τα οποία είναι διαθέσιμα στο OGB. Συγκεκριμένα,

1. Μελετώνται οι υπάρχουσες λύσεις και αναλύεται ο πηγαίος κώδικας της λύσης που επιλέχθηκε.

2. Παρουσιάζεται η δυνατότητα ανάκτησης ενός μέρους του συνόλου δεδομένων της ogb-wikikg2, με την ανάπτυξη ενός Wikidata query.

3. Παρουσιάζεται το tuning στις παραμέτρους του αλγορίθμου, ο οποίος χρησιμοποιεί το συνδυασμό των μεθόδων Triple Relation Encoding (TriplRE) & StarGraph [1]. Το tuning εφαρμόστηκε για να βρεθεί η βέλτιστη τιμή της μετρικής MRR, με τη χρήση παραμέτρων που ανταποκρίνεται στη δική μας υπολογιστική ισχύ.

1.2 Δομή της Διπλωματικής

Τα κεφάλαια που δομείται η εργασία, έχουν ως εξής:

• Στο Κεφάλαιο 2 “Open Graph Benchmark”, παρουσιάζουμε την δομή του αποθετηρίου, εξηγούμε το περιεχόμενο κάθε συνόλου δεδομένων και κάνουμε μια εισαγωγή στις μεθόδους που χρησιμοποιούνται.

• Στο Κεφάλαιο 3 “Νευρωνικά Δίκτυα Γράφων”, παρουσιάζουμε τα GNN σαν έννοια και δομή, επισημαίνοντας την χρησιμότητα και τα μειονεκτήματα τους.

• Στο Κεφάλαιο 4 “Wikidata”, κάνουμε αναφορά στις ιδιότητες της Wikidata, την χρησιμότητα της και εισαγωγή στα δεδομένα που περιέχει. Τέλος παραθέτουμε κάποιες από τις προϋποθέσεις που πρέπει να τηρούν τα δεδομένα ως προς τα χαρακτηριστικά τους.

• Στο Κεφάλαιο 5 “Μεθοδολογία ”, παρουσιάζουμε την πορεία που ακολουθήθηκε στην διπλωματική εργασία.

• Στο Κεφάλαιο 6 “Wikidata Query ”, παρουσιάζουμε την έννοια του query, δίνουμε ένα ερώτημα και το επιλύουμε. Δημιουργούμε ένα query που προσομοιώνει τα δεδομένα του ogb_wikikg2.

• Στο Κεφάλαιο 7 “Ανάλυση Κώδικα”, κάνουμε ανάλυση του κώδικα που αφορά τη φόρτωση, τη παραμετροποίηση και τη πρόβλεψη κάνοντας χρήση της μετρικής MRR.

• Στο Κεφάλαιο 8 “ Συμπεράσματα”, αναφέρουμε το τρόπο επιλογής τη μεθόδου ανάλυσης και παρουσιάζουμε τις τιμές και τα αποτελέσματα από το tuning στη μέθοδο TripleRE & StarGraph.

• Στο Παράρτημα Ι παρουσιάζονται η βιβλιογραφία και οι δικτυακοί τόποι που αναφέρονται στην εργασία.

• Στο Παράρτημα ΙΙ παρουσιάζεται τα ακρωνύμια των ξενικών όρων, τα οποία χρησιμοποιούνται σε αυτή την εργασία για την διευκόλυνση του αναγνώστη.

ΚΕΦΑΛΑΙΟ 2: OPEN GRAPH BENCHMARK

Παρουσιάζεται το αποθετήριο δεδομένων Open Graph Benchmark. Περιγράφονται περιληπτικά τα πιο σημαντικά και ευρύτερα χαρακτηριστικά κατηγοριοποίησης των γράφων. Όπως είναι φυσικό, πραγματοποιείται μια εκτεταμένη περιγραφή των δεδομένων που συναντώνται στο OGB. Απαραίτητη προϋπόθεση επεξεργασίας των δεδομένων είναι η επιλογή και χρήση του κατάλληλου μοντέλου πρόβλεψης. Επομένως, στο ίδιο κεφάλαιο περιγράφονται οι σημαντικότερες τεχνικές πρόβλεψης με χρήση γράφων.

2 Open Graph Benchmark

2.1 Εισαγωγή στο OGB

Το OGB παρέχει μια ποικιλία εφαρμογών του ρεαλιστικού κόσμου που καλύπτουν καίριους τομείς, έχοντας ως στόχο την αναπαραγωγή γνώσης και την εξόρυξη δεδομένων, μέσω της Μηχανικής Μάθησης Γράφων (graph ML). Τα σύνολα δεδομένων που διαθέτει το OGB, εκτείνονται από τα κοινωνικά μέχρι και τα μοριακά δίκτυα. Από τα πειράματα, τους διαχωρισμούς και τις προβλέψεις των δεδομένων, προκύπτουν πολλά μελλοντικά ερευνητικά ερωτήματα προς διερεύνηση [2]. Τα δεδομένα του OGB, ενημερώνονται τακτικά και προστίθενται νέα, από τα μέλη της κοινότητας του. Όλα αυτά τα αρχεία, οι αξιολογήσεις και ο κώδικας είναι διαθέσιμα στην σελίδα https://ogb.stanford.edu.

2.2 Χαρακτηριστικά των γράφων

Η διερεύνηση και η αντιμετώπιση των προβλημάτων στα σύνολο δεδομένων που προαναφέρθηκαν, επιτυγχάνεται με την χρήση των γράφων. Οι γράφοι αποτελούν εναλλακτικό και πρωτοποριακό τρόπο διερεύνησης και αντιμετώπισης, έναντι άλλων μη αποτελεσματικών μεθόδων. Μέσω αυτών εκφράζονται φυσικά οι σχέσεις μεταξύ των οντοτήτων, δημιουργούνται νέες συνδέσεις και εξάγονται συμπεράσματα για τα δεδομένα. Με την εφαρμογή των γράφων, ο χρήστης έχει την δυνατότητα να εφαρμόσει μοντέλα ML, με τα οποία επιδιώκει να κάνει προβλέψεις ή να ανακαλύπτει νέα μοτίβα σχέσεων μεταξύ των δεδομένων. Τα μοντέλα ML συνήθως λαμβάνουν τα δεδομένα με την μορφή συστοιχιών, ενώ οι γράφοι τα διακρίνουν σε δυο τύπους πληροφοριών: κόμβοι, ακμές. Για παράδειγμα, στο γράφο ενός κοινωνικού δικτύου, οι κόμβοι μπορεί να αντιπροσωπεύουν τους χρήστες και τα χαρακτηριστικά τους (π.χ. όνομα, φύλο, ηλικία, πόλη), ενώ οι ακμές τις σχέσεις μεταξύ των χρηστών.

Το μέγεθος των συνόλων δεδομένων καθορίζεται από το πλήθος των κόμβων που διαθέτουν. Αρκετά από τα δεδομένα που συναντώνται στο OGB είναι πολύ μικρότερα σε σχέση με αυτά που προκύπτουν καθημερινά. Ως εκ τούτου, ο τρόπο αντιμετώπισης των δεδομένων, τόσο ως προς την επιλογή πρωτοκόλλου, όσο τον διαχωρισμό τους και

την επιλογή της μεθόδου αξιολόγησης που χρησιμοποιείται, είναι απόρροια του μεγέθους τους. Λόγω του τρόπου αντιμετώπισης, κρίθηκε η ανάγκη διαχωρισμού των δεδομένων, ανάλογα με τα παρακάτω χαρακτηριστικά.

- Κλίμακα μεγέθους

  Τα δεδομένα ανάλογα με τον αριθμό των κόμβων ή των ακμών διακρίνονται σε μικρά, μεσαία και μεγάλα. Αρκετά από τα δεδομένα που συναντώνται στο OGB είναι πολύ μικρότερα (λιγότερους από 250.000 κόμβους) , σε σύγκριση με τα πραγματικά δεδομένα. Η κλιμάκωση των δεδομένων είναι: μικρά έως 1 εκατομμύριο κόμβους ή λιγότερες από 10 εκατομμύρια ακμές, μεσαία έως 100 εκατομμύρια κόμβους ή 1 δισεκατομμύριο ακμές, ενώ τα υπόλοιπα ανήκουν στην μεγαλύτερη κλίμακα.
- Τομέας δεδομένων

  Η ύπαρξη μεγάλης θεματολογίας, δημιούργησε την ανάγκη διαχωρισμού των δεδομένων σε τομείς όπως: δεδομένα με βάση το χρόνο, το είδος, την δομή.
- Κατηγορίες εργασιών

  Διαχωρίζονται ανάλογα με την ποικιλομορφία των δεδομένων και τις απαιτήσεις της πρόβλεψης (δηλαδή πρόβλεψη σε επίπεδο κόμβων, ακμών είτε και ολόκληρου του γράφου).

2.3 Σύνολα Δεδομένων

Τα δεδομένα που παρουσιάζονται και γίνονται σημείο αναφοράς πολλών χρηστών του OGB, ποικίλουν ως προς τη δομή αλλά και την θεματολογία. Αυτή τη στιγμή, η OGB περιλαμβάνει 15 διαφορετικά σύνολα δεδομένων, με το μεγαλύτερο πλήθος αυτών να είναι κατασκευασμένα από το OGB. Παρακάτω σκιαγραφούνται μερικά από τα σύνολα δεδομένων που συναντώνται και αναφέρονται συγκεκριμένες ιδιότητες των γράφων τους.

ogbn-products: Περιλαμβάνει δεδομένα που προέρχονται από τον ιστότοπο της Amazon και αφορά προϊόντα που αγοράζονται συνδυαστικά. Τα δεδομένα είναι μη

κατευθυνόμενα, χωρίς βάρη[1] και οι κόμβοι αντιπροσωπεύουν τα προϊόντα, ενώ οι ακμές συνδέουν τα προϊόντα που ο καταναλωτής αγοράζει συνδυαστικά.

ogbn-proteins: Ένας μη κατευθυνόμενος γράφος με βάρη, όπου οι κόμβοι του αντιπροσωπεύουν τις πρωτεΐνες και οι ακμές τις συσχετίσεις μεταξύ αυτών. Επίσης γνωρίζουμε ότι η πρωτεΐνη προέρχεται από 8 είδη, κατά συνέπεια κάθε ακμή διαθέτει 8-διάστατα χαρακτηριστικά.

ogbn-arxiv: Το arxiv είναι ένα αποθετήριο ανοικτής πρόσβασης ηλεκτρονικών άρθρων, που έχουν εγκριθεί για δημοσίευση έπειτα από εποπτεία, αλλά χωρίς να έχει προηγηθεί αξιολόγηση από επιτροπή. Αποτελείται από επιστημονικές εργασίες στους τομείς των Θετικών Επιστημών, στις οποίες υπάρχει πρόσβαση μέσω διαδικτύου. Τα δεδομένα αντιπροσωπεύουν το δίκτυο παραπομπών μεταξύ των εργασιών αυτών. Κάθε κόμβος είναι ένα έγγραφο και οι ακμές υποδηλώνουν τη σύνδεση μεταξύ δυο εγγράφων. Η εφαρμογή των μεθόδων στα δεδομένα περιλαμβάνει επίσης, την εύρεση του μέσου όρου συχνότητας των λέξεων στο τίτλο και στην περίληψη.

ogbn-papers100m: Δεδομένα που περιλαμβάνουν πάνω από 100 εκατομμύρια εργασίες. Η κατασκευή του γράφου γίνεται με τον ίδιο τρόπο με τα δεδομένα arxiv.

ogbn-mag: Τα δεδομένα περιλαμβάνουν τέσσερις οντότητες: έγγραφα, συγγραφείς, ιδρύματα και πεδία μελέτης. Οι οντότητες απαρτίζουν τους κόμβους, ενώ οι ακμές απεικονίζουν τις σχέσεις μεταξύ αυτών. Οι ακμές μπορεί να συνδέουν τα παρακάτω ζευγάρια οντοτήτων: έγγραφα με συγγραφέα, συγγραφέας με ιδρύματα, έγγραφα με πεδία μελέτης, έγγραφο με έγγραφο.

ogbl-ppa: Ένα μη κατευθυνόμενο σύνολο δεδομένων χωρίς βάρη, όπου οι κόμβοι αντλούν στοιχεία από 58 διαφορετικά είδη πρωτεϊνών και οι ακμές υποδεικνύουν τις συσχετίσεις τους. Κάθε πρωτεΐνη προέρχεται από 58 είδη πρωτεϊνών, κατά συνέπεια κάθε ακμή διαθέτει 58- διάστατα χαρακτηριστικά.

ogbl-collab: Ένα μη κατευθυνόμενο δίκτυο συνεργασίας μεταξύ συγγραφέων. Οι κόμβοι αναπαριστούν τους συγγραφείς ενώ οι ακμές τη συνεργασία μεταξύ αυτών. Κάθε κόμβος διαθέτει χαρακτηριστικά από το μέσο όρο συχνότητας των λέξεων των εργασιών. Δυο

[1] Η παράμετρος σε ένα νευρωνικό δίκτυο που μετασχηματίζει τα δεδομένα εισόδου στα κρυφά επίπεδα του δικτύου.

σημαντικές πληροφορίες είναι το έτος και ο αριθμός των εργασιών που δημοσιεύονται κάθε έτος. Εξαιτίας των οποίων, στο γράφο μπορεί να υπάρχουν πολλαπλές γραμμές μεταξύ δυο κόμβων.

ogbl-ddi: Είναι μια συλλογή δεδομένων με φάρμακα και τις αλληλεπιδράσεις τους, μη κατευθυνόμενη και χωρίς βάρη. Κάθε κόμβος αντιστοιχίζεται με ένα εγκεκριμένο ή πειραματικό φάρμακο, ενώ οι ακμές επισημαίνουν τις αλληλεπιδράσεις μεταξύ αυτών. Μια ακόμη ερμηνεία των ακμών σχετίζεται με το αποτέλεσμα της λήψης δύο φαρμάκων, σχετικά με το εάν αυτά δρουν ανεξάρτητα το ένα από το άλλο.

ogbl-citation2: Δίκτυο ενός υποσυνόλου εγγράφων, παρόμοιο με το ogbn-arxiv.

ogbl-wikikg2: Το wikikg2 είναι ένας Γράφος Γνώσης (KG) που εξάγεται από τη Wikidata. Περιέχει ένα σύνολο τριπλετών (κεφάλι, σχέση, ουρά), που αποτυπώνει τους διάφορους τύπους σχέσεων μεταξύ των οντοτήτων στον κόσμο. Ανακτά όλες τις σχέσεις στα Wikidata και φιλτράρει σπάνιες οντότητες. Το wikikg2 περιέχει πάνω από 2.500.000 οντότητες και 500 τύπους σχέσεων. Σκοπός του KG είναι, να ανακαλύπτει και να εμφανίζει δημόσια γνωστές, τεκμηριωμένες πληροφορίες, όταν αυτό θεωρείται χρήσιμο.

ogbl-biokg: Είναι ένας Γράφος Γνώσης, με δεδομένα από ένα αποθετήριο βιοϊατρικών δεδομένων. Περιέχει 5 τύπους οντοτήτων: ασθένειες, πρωτεΐνες, φάρμακα, παρενέργειες και πρωτεϊνικές λειτουργίες. Οι ακμές που συνδέουν ίδιου τύπου οντότητες, είναι αμφίδρομες.

ogbg-molhiv, ogbg-molpcba: Δύο σύνολα δεδομένων πρόβλεψης μοριακών ιδιοτήτων διαφορετικών μεγεθών: ogbg-molhiv (μικρό) και ogbg-molpcba (μεσαίο). Τα χαρακτηριστικά κόμβου εισόδου είναι 9-διάστατα και περιέχουν τον ατομικό αριθμό, καθώς και επιπρόσθετα χαρακτηριστικά ατόμου (atom), όπως το τυπικό φορτίο και αν το άτομο βρίσκεται στον δακτύλιο. Τα αντίστοιχα χαρακτηριστικά των άκρων εισόδου είναι τρισδιάστατα περιέχουν το τύπο δεσμού, τη στερεοχημεία δεσμού καθώς και ένα πρόσθετο χαρακτηριστικό που υποδεικνύει αν ο δεσμός είναι συζευγμένος.

ogbg-code2: Μια συλλογή από Abstract Syntactic Trees (AST) που ανακτήθηκαν από περίπου 450 χιλιάδες μεθόδους της Python. Το ogbg-code2 επιτρέπει να συλλάβουμε τον πηγαίο κώδικα με την υποκείμενη δομή του γράφου, πέρα από την αναπαράσταση της συμβολικής ακολουθίας.

2.4 Είδη Προβλέψεων

Το αποθετήριο OGB διαθέτει τα δεδομένα του με τη μορφή κόμβων και ακμών στους χρήστες. Σκοπός του είναι η πρόβλεψη νέων πιθανών σχέσεων και συνδέσμων μεταξύ των κόμβων. Ειδικότερα, οι κατηγορίες πρόβλεψης graph ML που εφαρμόζονται στα σύνολο δεδομένων διακρίνονται σε τρεις: κόμβων, ακμών και ολόκληρων γράφων. Το είδος πρόβλεψης που θα επιλεγεί από τους χρήστες εξαρτάται από το τύπο, τη χρησιμότητα των αποτελεσμάτων και τα συμπεράσματα που επιθυμούν να εξάγουν από τα δεδομένα. Για την επίτευξη της πρόβλεψης χρησιμοποιούνται οι ήδη υπάρχοντες κόμβοι, ακμές και γράφοι.

- Πρόβλεψη κόμβων
  Αποσκοπεί στην ταξινόμηση των κόμβων του γράφου. Συγκεκριμένα τα μοντέλα ταξινόμησης κόμβων χρησιμοποιούνται για την πρόβλεψη των τάξεων των unlabeled κόμβων. Κατά τη διάρκεια της εκπαίδευσης, η ιδιότητα που αντιπροσωπεύει την κλάση του κόμβου αναφέρεται ως η ιδιότητα στόχος. Στα δεδομένα τα οποία εφαρμόστηκε αυτό το είδος πρόβλεψης είναι τα ogbn-products, ogbn-proteins, ogbn-arxiv, ogbn-papers100M και ogbn-mag.
- Πρόβλεψη ακμών
  Τα δεδομένα που εφαρμόστηκε η πρόβλεψη αυτή είναι τα ogbl-ppa, ogbl-collab, ogbl-ddi, ogbl-citation2, ogbl-wikikg2 και ogbl-biokg. Η πρόβλεψη ακμών είναι μια διεργασία που αποσκοπεί στην εύρεση σχέσεων μεταξύ των ζευγών των κόμβων σε ένα γράφο. Κατά τη διάρκεια της εκπαίδευσης του μοντέλου link prediction, τα ζεύγη κόμβων επισημαίνονται ως γειτονικά ή μη γειτονικά.
- Πρόβλεψη ολόκληρων γράφων
  Συναντώνται στα δεδομένα: ogbg-molhiv, ogbg-molpcba, ogbg-ppa, ogbg-code2. Οι προβλέψεις γράφων βρίσκουν σημαντικές εφαρμογές στις φυσικές επιστήμες, όπως στη πρόβλεψη μοριακών ιδιοτήτων.

2.5 Μέθοδοι

Η διερεύνηση και επίλυση των προβλημάτων στα σύνολα δεδομένων επιτυγχάνεται με την χρήση μεθόδων πρόβλεψης, όπως προαναφέρθηκε. Σε κάθε σύνολο δεδομένων δοκιμάζονται περισσότερες από μια μέθοδοι, με σκοπό να βρεθεί η πλέον

αποτελεσματικότερη ως προς την μετρική MRR αλλά και τη πολυπλοκότητα. Οι συνήθεις μέθοδοι που συναντώνται στο OGB παρουσιάζονται παρακάτω. Έχουν καταχωρηθεί αλφαβητικά, επισημαίνοντας αυτές που δίνουν βέλτιστη απόδοση στη πρόβλεψη.

CLUSTER- GRAPH CONVOLUTIONAL NETWORK (GCN)

Μια μέθοδος για κλιμακούμενη εκπαίδευση νευρωνικών δικτύων γράφων μεγάλου βάθους. Χωρίζει τους γράφους σε έναν σταθερό αριθμό mini-batch και ενημερώνει τις παραμέτρους.

DEEPERGCN

Το DeeperGCN είναι ένας τύπος GCN, που χρησιμοποιεί πολλαπλά στρώματα, για να πετύχει μια βαθύτερη και πιο ισχυρή αναπαράσταση των δεδομένων.

GRAPHSAGE

Το GraphSAGE είναι ικανό να προβλέψει την ενσωμάτωση ενός νέου κόμβου, χωρίς να απαιτείται διαδικασία επανεκπαίδευσης. Για να επιτευχθεί αυτό, το GraphSAGE μαθαίνει συναρτήσεις aggregator που επιτρέπουν την ενσωμάτωση ενός νέου κόμβου, δεδομένο των χαρακτηριστικών και των γειτονικών κόμβων του.

GIANT

Το GIANT κάνει χρήση του Extreme Multi-label Classification (XMC). Το XMC είναι σημαντικό για τη τελειοποίηση του μοντέλου, και βασίζεται σε πληροφορίες γράφων μεγάλων συνόλων δεδομένων.

GRAPH ISOMORPHISM NETWORK (GIN)

Έχει την μέγιστη ικανότητα αναπαράστασης και γενίκευσης διαφόρων δομών δεδομένων [3].

GRAPH ATTENTION MULTI-LAYER PERCEPTRON (GAMLP)

Εξετάζει τη διάδοση των χαρακτηριστικών. Ο περιορισμός του GAMLP είναι ότι απαιτούνται όλα τα χαρακτηριστικά στην εκπαίδευση του μοντέλου, γεγονός που οδηγεί σε υψηλό κόστος μνήμης.

GRAPHORMER

Βασίζεται στην Transformer αρχιτεκτονική. Η Transformer αρχιτεκτονική χρησιμοποιεί μετασχηματιστές, οι οποίοι έχουν σχεδιαστεί για να χειρίζονται διαδοχικά δεδομένα εισόδου, όπως η φυσική γλώσσα, για εργασίες όπως η μετάφραση και η σύνοψη κειμένου. Ωστόσο, οι μετασχηματιστές δεν επεξεργάζονται απαραίτητα τα δεδομένα με τη σειρά [4].

MULTI-LAYER PERCEPTRON (MLP)

Είναι ένα πολυστοιβαδικό δίκτυο με διάδοση προς τα εμπρός. Χαρακτηρίζεται έτσι, γιατί η διάδοση του σήματος γίνεται μόνο από τη μία στοιβάδα στην επόμενη και όχι αντίστροφα. Ένα MLP χαρακτηρίζεται από πολλά στρώματα κόμβων εισόδου, που συνδέονται μεταξύ των επιπέδων εισόδου και εξόδου. Το MLP χρησιμοποιεί backpropogation για την εκπαίδευση του δικτύου.

PAIRWISE LEARNING FOR NEURAL LINK PREDICTION (PLNLP)

Αντιμετωπίζει την πρόβλεψη συνδέσμων ως ζεύγη, μαθαίνει να ταξινομεί το πρόβλημα και αποτελείται από τέσσερα κύρια στοιχεία, τα οποία είναι, η κωδικοποίηση, ο προγνωστικός σύνδεσμος, ο αρνητικός δειγματολήπτης και η αντικειμενική συνάρτηση [5].

SAGN

Το SAGN χρησιμοποιεί έναν μηχανισμό αυτοπροσοχής για την ανακατασκευή των δεδομένων εισόδου στο δίκτυο και συνδυάζει πληροφορίες από διαφορετικούς κόμβους και συνδέσεις του γράφου.

2.5.1 Αποτελεσματικότεροι μέθοδοι

Η ύπαρξη πινάκων με τα αποτελέσματα των δοκιμών για κάθε σύνολο δεδομένων του OGB, συνεισφέρει στη σύγκριση των μεθόδων με βάση τη μετρική MRR. Συγκεντρωτικά, για κάθε τύπο δεδομένων, οι υψηλότερες επιδόσεις επιτυγχάνονται με τη χρήση των παρακάτω μεθόδων: GIANT, SAGN, MCR, GAMLP, PLNLP, GRAPHSAGE, DEEPERGCN και GRAPHORMER. Αρκετές είναι οι φορές που γίνεται συνδυασμός στη χρήση των μεθόδων, για την επίτευξη τη βέλτιστης πρόβλεψης.

Η κατηγορία πρόβλεψης κόμβων χρησιμοποιεί κατά κόρον τη μέθοδο GIANT. Εφαρμόζοντας ταυτόχρονα και κάποια από τις μεθόδους SAGN, GAMLP και GRAPHSAGE επιτυγχάνεται το καλύτερο αποτέλεσμα. Παίρνοντας ως παράδειγμα τα δεδομένα ogbn-products, παρατηρείται ότι, η μεμονωμένη χρήση των μεθόδων SAGN, GAMLP και GRAPHSAGE υστερεί από την βέλτιστη απόδοση κατά τουλάχιστον 0,3%.

Από την εφαρμογή της PLNLP προκύπτει, ότι μάθηση κατά ζεύγη μεγιστοποιεί την απόδοση των νευρωνικών μοντέλων γράφων στο πρόβλημα πρόβλεψης συνδέσμων. Αυτό αποδεικνύεται από την εφαρμογή της μεθόδου σε πλήθος δεδομένων, αλλά και από τη σύγκριση της με μεθόδους όπως Classification, SAGE, SEAL.

Η GRAPHSAGE είναι σε θέση να διατηρεί ισχυρή προγνωστική ακρίβεια, ενώ παράλληλα είναι σημαντική και η βελτίωση του χρόνου εκτέλεσης της. Η απόδοση της μεθόδου, όπως παρατηρείται στα δεδομένα μας, εξαρτάται από την τιμή της υπερ-παραμέτρου K. Παρατηρούμε ότι γίνεται χρήση της σε δυο από τους τύπους προβλέψεων, δηλαδή ως προς του κόμβους και ακμές [6].

Η μέθοδος GRAPHORMER κάνει χρήση γράφων FLAG[2] για την αύξηση του μεγέθους των δεδομένων. Σκοπός της είναι ο μετριασμός του προβλήματος της υπερβολικής προσαρμογής στα σύνολα δεδομένων. Εμπειρικά διαπιστώνετε, ότι οι διαφορετικές επιλογές υπερ-παραμέτρων της FLAG (δηλ., μέγεθος βήματος, αριθμός βημάτων, μέγιστη διατάραξη) επηρεάζουν σε μεγάλο βαθμό την απόδοση του Graphormer. Τέλος, η εφαρμογή της μεθόδου συναντάται μόνο σε προβλήματα πρόβλεψης με χρήση ολόκληρων γράφων [7].

Η μέθοδος DEEPERGCN εφαρμόζετε συχνά με τη χρήση FLAG, ώστε να δείξει την αποτελεσματικότητά της στα GNN, σε μεγάλο βάθος. Παρά την καλή απόδοση του μοντέλου, για να προχωρήσουν τα GCN σε βάθος απαιτούν περισσότερους πόρους μνήμης Graphics Processing Unit (GPU) και καταναλώνουν περισσότερο χρόνο [8].

[2] Ένας τύπος γραφήματος που κωδικοποιεί σημαντικές πληροφορίες σχετικά με τις σχέσεις μεταξύ των κόμβων, πέρα από τις συνδέσεις ανά ζεύγη.

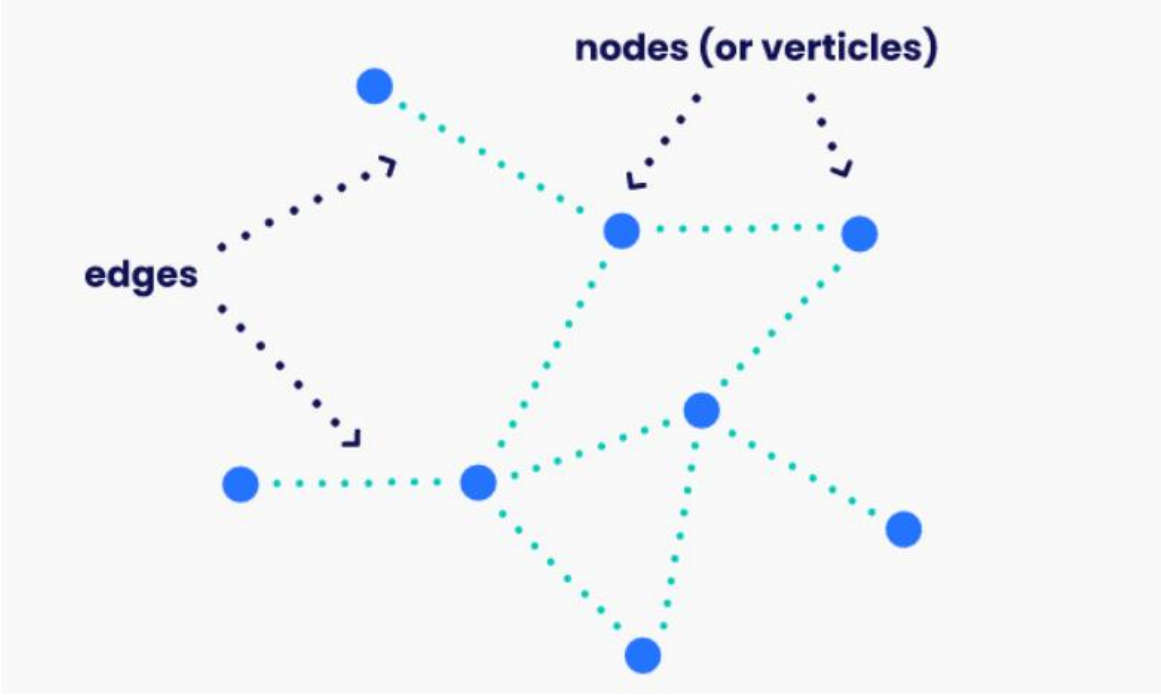
ΚΕΦΑΛΑΙΟ 3 : ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΓΡΑΦΩΝ

Στο παρόν κεφάλαιο παραθέτετε η βασική δομή των Νευρωνικών Δικτύων Γράφων. Γίνεται μια αναλυτική παρουσίαση των πλεονεκτημάτων-μειονεκτημάτων χρήσης γράφων και αναφέρονται περιγραφικά κάποιες από τις καθημερινές εφαρμογές τους. Περιγράφονται επίσης οι διαφορετικές αρχιτεκτονικές GNN, όπου η αποτελεσματικότητα της κάθε μιας είναι ανάλογη των παραμέτρων.

3 Νευρωνικά Δίκτυα Γράφων

3.1 Εισαγωγή στους γράφους

Το GNN είναι ένα είδος δομής δεδομένων, δηλαδή μια συγκεκριμένη προσέγγιση στην επεξεργασία, στην αποθήκευση και στην ερμηνεία δεδομένων. Σε αντίθεση με ένα τυπικό γράφημα με άξονα x και y, το GNN αναφέρεται σε ένα γράφο. Δηλαδή σε μια αναπαράσταση δεδομένων που σχηματίζει κόμβους (nodes) και ακμές (edges).



Εικόνα 1, Απεικόνιση ενός γράφου[9]

Τα GNN είναι μια κατηγορία αρχιτεκτονικών νευρωνικών δικτύων που έχουν σχεδιαστεί για να λειτουργούν σε δεδομένα με δομή γράφου. Τα συμπεράσματα που παράγει, είναι αποτέλεσμα των δυο κύριων μερών: κόμβους και ακμές. Μέσω της ανάλυσης των κόμβων και ακμών, τα νευρωνικά ερμηνεύουν τα δεδομένα και στη συνέχεια επιλύουν τα προβλήματα και πραγματοποιούν προβλέψεις.

Η εφαρμογή των GNN για την λήψη αποφάσεων μπορεί να εφαρμοστεί σε πολλούς τομείς της κοινωνίας, όπως η βιομηχανία. Κάποιες από τις λειτουργίες που αναλαμβάνουν, είναι ο εντοπισμός μοτίβων (μεταξύ ακμών και κόμβων) αλλά και ανωμαλιών. Χρησιμοποιούνται ακόμη για τη μοντελοποίηση πολύπλοκων σχέσεων

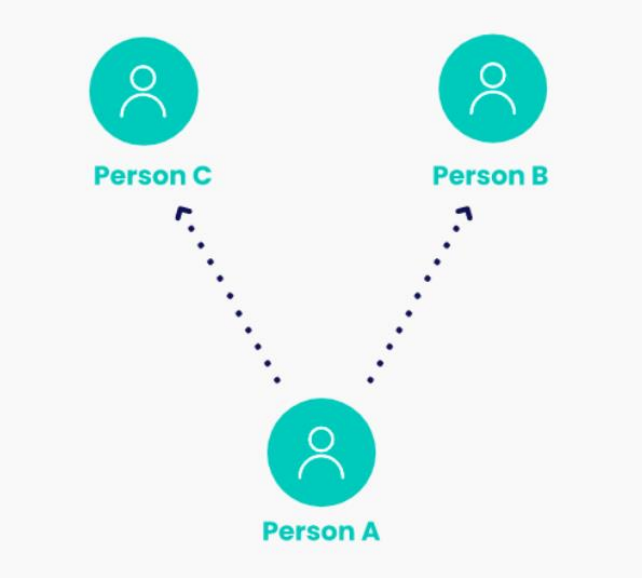
μεταξύ αντικειμένων σε ένα γράφο, όπως οι σχέσεις μεταξύ ανθρώπων σε ένα κοινωνικό δίκτυο ή των ατόμων σε ένα μόριο.

3.2 Αναπαράσταση των GNN

Όπως έχει προαναφερθεί, τα GNN λειτουργούν χρησιμοποιώντας γράφους ως είσοδο και στη συνέχεια εξάγουν δεδομένα ή σχηματίζουν συμπεράσματα και προβλέψεις με βάση αυτά. Οι γράφοι που συναντώνται και χρησιμοποιούνται από τα GNN είναι διαχωρισμένοι ανάλογα με τις συνδέσεις τους, στους κατευθυνόμενους ή μη κατευθυνόμενους. Παρακάτω γίνεται η χρήση των μέσων κοινωνικής δικτύωσης ως παράδειγμα για την διαφοροποίηση των κατευθυνόμενων και μη κατευθυνόμενων γράφων [9].

- Κατευθυνόμενοι γράφοι

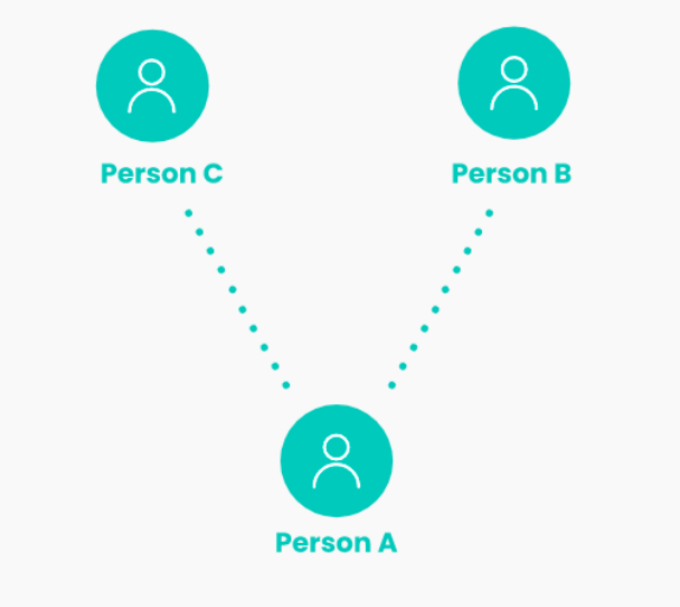
  Για το παράδειγμα μας, υιοθετούμε την πλατφόρμα του Facebook και υποθέτουμε την ύπαρξη τεσσάρων ατόμων: της Μαρία, του Νίκο, του Κώστα, της Τζένης. Υποθέτουμε ότι η Μαρία αναζητεί τα προφίλ των 3 φίλων της στο Facebook, ενώ ο Νίκος, Κώστας, Τζένη δεν κάνουν καμία αναζήτηση. Δεδομένου ότι η σχέση αυτή είναι μονόδρομη, δηλαδή ένας κόμβος (πρόσωπο) συνδέεται με έναν ή περισσότερους κόμβους (πρόσωπα), σχηματίζεται ένας ''Κατευθυνόμενος γράφο''.



Εικόνα 2, Απεικόνιση ενός κατευθυνόμενου γράφου [9]

- Μη κατευθυνόμενοι γράφοι

  Κάνοντας χρήση των ίδιων χαρακτήρων, ας υποθέσουμε ότι η Μαρία και ο Νίκος αναζητούν και οι δύο το προφίλ των φίλων τους καθώς ακόμη η Μαρία και ο Κώστας αναζητούν επίσης το προφίλ των άλλων φίλων τους. Δημιουργείται έτσι μια αμφίδρομη σχέση μεταξύ του πρώτου και του δεύτερου ζεύγους στο γράφο, χωρίς την παρουσία βελών στον γράφο. Σε μια ανάλογη περίπτωση όπως η παραπάνω, η αμοιβαία σύνδεση του ατόμου Β και του ατόμου Γ με Α, μπορεί να οδηγήσει τον ιστότοπο στην υπόθεση σχέσης μεταξύ του Β και του Γ ατόμου. Ο παρακάτω γράφος αναπαριστά μια τέτοια δυναμική, δηλαδή τον σχηματισμό ενός ''Μη Κατευθυνόμενου γράφου''.



Εικόνα 3, Απεικόνιση ενός μη κατευθυνόμενου γράφου [9]

3.3 Αναγκαιότητα των GNN

Στον σύγχρονο κόσμο οι γράφοι αποτελούν το πιο φυσικό και εύχρηστο τρόπο παρουσίασης της καθημερινότητας. Παραδείγματα γράφων αποτελούν, οι σχεσιακές βάσεις που χρησιμοποιούνται στις περισσότερες εταιρείες, τα κοινωνικά δίκτυα (Facebook,Instagram, LinkedIn) και οι γράφοι παραπομπών (Wikipedia). Ακόμα, και οι εικόνες θα μπορούσαν να θεωρηθούν ως γράφοι, λόγω της δομής πλέγματος. Τα GNN έχουν βασικά χαρακτηριστικά που τα καθιστούν κατάλληλα για την εργασία με δεδομένα γράφων του πραγματικού κόσμου [10]:

- Μπορούν να χειριστούν εισόδους μεταβλητού μεγέθους: Είναι σημαντικό επειδή οι γράφοι του πραγματικού κόσμου μπορεί να είναι πολύπλοκοι. Τα GNN έχουν την ιδιότητα να είναι σε ένα μεγάλο βαθμό αυτάρκης, δηλαδή καθορίζουν τα σημεία των δεδομένων τους.
- Μπορούν να ενσωματώσουν χαρακτηριστικά κόμβων και ακμών: Χρήσιμο για εργασίες όπως η ταξινόμηση κόμβων ή η πρόβλεψη συνδέσμων.
- Μπορούν να μάθουν από την τοπική δομή: Τα καθιστά κατάλληλα για εργασίες που περιλαμβάνουν συλλογισμούς, σχετικά με τις σχέσεις μεταξύ διαφορετικών κόμβων.

Παρά τα πλεονεκτήματα αυτά, τα GNN παρουσιάζουν επίσης ορισμένους περιορισμούς [11]:

- Μπορεί να είναι υπολογιστικά δαπανηρά: Η εκπαίδευση των GNN μπορεί να απαιτεί μεγάλο αριθμό παραμέτρων και να είναι απαιτητική υπολογιστικά, ιδίως για μεγάλους γράφους.
- Μπορεί να είναι δύσκολο να ερμηνευθούν: Αυτό μπορεί να καταστήσει δύσκολη την ερμηνεία των αποτελεσμάτων που παράγουν.
- Μπορεί να δυσκολευτούν με εξαρτήσεις μεγάλης εμβέλειας: Τα GNN ενδέχεται να δυσκολεύονται στη σύλληψη των εξαρτήσεων μεγάλης εμβέλειας σε γράφους, ιδίως εάν ο γράφος έχει μεγάλη διάμετρο[3].

3.4 Εφαρμογές των Νευρωνικών Δικτύων Γράφων

Ενδεικτικά αναφέρονται πέντε εφαρμογές των GNN. Δυνητικά μπορούν να κατασκευαστούν απεριόριστου πλήθους εφαρμογών που θα επιλύουν προβλήματα του πραγματικού κόσμου [12].

1. Πρόληψη της απάτης

Με αυτόν τον τρόπο, τα GNN μπορούν να χρησιμοποιηθούν για να σχηματίσουν προβλέψεις, σχετικά με το ποια δραστηριότητα μπορεί να πραγματοποιήσει στο μέλλον

[3] Η μέγιστη απόσταση μεταξύ δύο κόμβων στο γράφο.

η εν λόγω κακόβουλη οντότητα. Χρησιμοποιώντας επομένως τα πρότυπα αυτά, γίνεται ο εντοπισμός των ύποπτων χρηστών [13].

2. Διαδικτυακή ασφάλεια

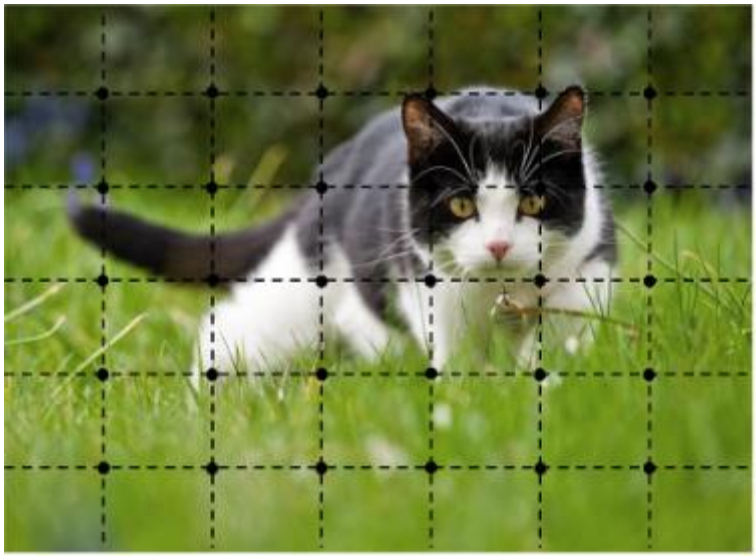
Απότοκος της εφαρμογής των GNN στην πρόληψη της απάτης (π.χ. στα μέσα κοινωνικής δικτύωσης) είναι η διαδικτυακή ασφάλεια. Τα GNN είναι ζωτικής σημασίας για την ενίσχυση της διαδικτυακής ασφάλειας σε όλα τα μέσα του διαδικτύου. Η εφαρμογή των γράφων στα διαδικτυακά δίκτυα, δίνει την δυνατότητα στους ανθρώπους να κατανοήσουν την δραστηριότητα των χρηστών στα δίκτυα [14].

3. Ιατρική

Αναλύουν δεδομένα ασθενών, όπως η ηλικία και το φύλο τους, και εντοπίζουν τάσεις στις διαγνώσεις και στις θεραπείες τους. Ακόμη με ανάλυση των φαρμάκων (ως προς συστατικά) μελετάτε η αποτελεσματικότητα αλλά και τυχόν παρενέργειες σε συνδυασμό φαρμάκων [15].

4. Υπολογιστική Όραση

Η υπολογιστική όραση (computer vision) είναι ένα σύστημα τεχνητής νοημοσύνης, που επιτρέπει στους υπολογιστές να σαρώνουν εικόνες, να τις κατηγοριοποιούν και να σχηματίζουν συμπεράσματα με βάση την προκύπτουσα ταξινόμηση της εικόνας [16].



Εικόνα 4, Σχηματισμός γράφου με βάση τα χαρακτηριστικά εικόνας [12]

3.5 Αρχιτεκτονικές των GNN

Στην ενότητα αυτή παρουσιάζεται ένα αρκετά μεγάλο και αντιπροσωπευτικό κομμάτι των αρχιτεκτονικών, που έχουν σχεδιαστεί ειδικά για να εφαρμόζονται σε δεδομένα με δομή γράφου. Υπάρχουν πολλές διαφορετικές αρχιτεκτονικές που έχουν προταθεί για την υλοποίηση των GNN [17], όπου η αποτελεσματικότητα της κάθε μιας καθίσταται από παραμέτρους όπως:

- Τύπος γράφου (αραιός ή πυκνός γράφος)
- Μέγεθος του γράφου
- Τύπος χαρακτηριστικών των κόμβων ή των ακμών (συνεχή ή κατηγορικά χαρακτηριστικά)
- Εργασία πρόβλεψης (εργασίες κατηγοριοποίησης ή παλινδρόμησης)
- Διαθέσιμοι πόροι (ανάγκες μνήμης, κ.λπ..)

Αναλυτικότερα κάποιες από τις αρχιτεκτονικές των GNN:

- GCNs

  Τα GCNs είναι ένας τύπος GNN, που χρησιμοποιεί συνελικτικά στρώματα για την επεξεργασία των δεδομένων γράφου. Βασίζονται στην ιδέα της χρήσης συνελικτικών φίλτρων για την εξαγωγή χαρακτηριστικών από τα δεδομένα γράφου, παρόμοια με το τρόπο που χρησιμοποιούνται τα Convolutional Neural Network (CNN) για την εξαγωγή χαρακτηριστικών από δεδομένα εικόνας [18], [19].
- Graph Attention Network (GAT)

  Τα GAT είναι ένας τύπος GNN που χρησιμοποιεί μηχανισμούς προσοχής για να σταθμίζει τη σημασία των διαφόρων κόμβων στο γράφο. Βασίζονται στην ιδέα της χρήσης ενός μηχανισμού αυτοπροσοχής, ώστε να επιτρέπει πιο εξελιγμένη συνάθροιση πληροφοριών και καλύτερο χειρισμό δεδομένων με δομή γράφου [20], [21].
- General regression neural network (GRNN)

  Τα GRNN είναι ένας τύπος GNN που χρησιμοποιούν επαναλαμβανόμενα στρώματα για την επεξεργασία των δεδομένων του γράφου με την πάροδο του χρόνου. Βασίζονται στην ιδέα της χρήσης των Recurrent neural network (RNN)

για την επεξεργασία διαδοχικών δεδομένων, με τη δομή του γράφου να χρησιμεύει ως ακολουθία [22]–[24] .

- Generative Graph Model

  Τα Generative Graph Model είναι ένας τύπος GNN που χρησιμοποιείται για τη δημιουργία νέων γράφων, παρόμοιων με ένα δεδομένο σύνολο γράφων εκπαίδευσης. Βασίζονται στην ιδέα της χρήσης ενός νευρωνικού δικτύου για την εκμάθηση της υποκείμενης δομής και των μοτίβων των γράφων εκπαίδευσης και, στη συνέχεια, τη δημιουργία νέων γράφων που είναι συνεπείς σε αυτή τη δομή.

Εν κατακλείδι, για να επιλεγεί η καλύτερη αρχιτεκτονική, ανάλογα με τις ανάγκες των δεδομένων αλλά και του χρήστη, θα πρέπει να γίνει σύγκριση των επιδόσεων και των αποτελεσμάτων του εκάστοτε προβλήματος.

ΚΕΦΑΛΑΙΟ 4: WIKIDATA

Προκειμένου να επιτευχθεί όσο το δυνατό καλύτερος σχολιασμός της Wikidata, γίνεται ιστορική αναδρομή στα προβλήματα που επέλυσε και στη συνεισφορά της. Περιγράφεται διεξοδικά το σύνολο δεδομένων της, το οποίο αποσκοπεί στην κατανόηση της γραφικής αναπαράστασης των δεδομένων. Επιπλέον, περιγράφονται οι προϋποθέσεις χρήσης και οι απαιτήσεις για τη ποιότητα των δεδομένων.

4 Wikidata

4.1 Ορισμός της Wikidata

Η Wikidata είναι μια ελεύθερη, συνεργατική, πολύγλωσση και δευτερογενής βάση δεδομένων [25]. Πρόκειται για ένα από τα πολυάριθμα έργα που βασίζονται σε wiki, τα οποία φιλοξενεί και συντηρεί το Ίδρυμα Wikimedia, το οποίο περιέχει το γνωστό ιστότοπο Wikipedia. Ένας από τους στόχους της Wikidata είναι η συλλογή δομημένων δεδομένων για την υποστήριξη της Wikipedia, της Wikimedia Commons και άλλων εγχειρημάτων του Wikimedia. Παρακάτω αναλύονται οι όροι από τον ορισμό της Wikidata.

- **Ελεύθερα**

  Τα δεδομένα της Wikidata κυκλοφορούν με την άδεια Creative Commons Public Domain Dedication 1.0 και μπορούν να επαναχρησιμοποιηθούν σε διάφορα σενάρια. Με την δημοσίευση τους είναι ελεύθερα για χρήση, αντιγραφή και επεξεργασία, χωρίς να απαιτείται άδεια.

- **Συνεργατικά**

  Περιγράφει την δυνατότητα εισαγωγής δεδομένων από τους συντάκτες της Wikidata, οι οποίοι αναλαμβάνουν τους κανόνες που αφορούν την παραγωγή και τη διαχείριση περιεχομένου, εισάγουν και συντηρούν τα δεδομένα της. Επιπλέον η εισαγωγή δεδομένων γίνεται μέσω bots, χωρίς να είναι ακόμη σαφές σε ποιο βαθμό αυτά συμβάλλουν στην ποιότητα τους.

- **Πολύγλωσσα**

  Τα δεδομένα εισάγονται σε οποιαδήποτε γλώσσα και είναι άμεσα διαθέσιμα σε όλες τις υπόλοιπες γλώσσες, καθιστώντας παράλληλα εφικτή την επεξεργασία τους. Οι γλώσσες οι οποίες είναι διαθέσιμες είναι περισσότερες από 200.

- **Δευτερεύουσα βάση δεδομένων**

  Η Wikidata έχει τη δυνατότητα, να καταγράφει τις πηγές από τις οποίες προήλθαν τα δεδομένα και να τις συνδέει με άλλες βάσεις δεδομένων. Αυτό αντανακλά την ποικιλομορφία της διαθέσιμης γνώσης και ενισχύει την επαληθευσιμότητα.

- **Συλλογή δομημένων δεδομένων**

  Επιτρέπει την επαναχρησιμοποίηση των δεδομένων και δίνει την δυνατότητα επεξεργασίας και κατανόησης τους από τους υπολογιστές.

- **Υποστήριξη για τα Wikimedia**

  Wikimedia είναι ο μητρικός οργανισμός των Wikimedia Commons, Wikipedia, κ.λπ... Η Wikidata βοηθά τη Wikipedia με την συντήρηση των πληροφοριών και συνδέσμων με άλλες γλώσσες. Μειώνοντας έτσι το φόρτο εργασίας για τους συντάκτες, ενισχύοντας ταυτόχρονα και την ποιότητα των δεδομένων.

4.2 Αναδρομή στη Wikidata

Η Wikidata αποτελεί ένα γράφο γνώσης, που ξεκίνησε από το ίδρυμα Wikimedia τον Οκτώβριο του 2012. Η εξέλιξη της από την οπτική του μεγέθους και αποδοχής, έχει ήδη οδηγήσει στην υιοθέτησή της ως πηγής γνώσης, για ποικίλους λόγους.

Ένας γράφος γνώσης αναπαριστά την γνώση αποτυπωμένη σε γράφους, οι οποίοι παριστάνουν και επεξηγούν οντότητες του πραγματικού κόσμου και τις σχέσεις μεταξύ τους. Πλήθος γράφων γνώσης έχουν αναπτυχθεί πριν από τη Wikidata, με σημαντικότερους τη DBpedia και YAGO. Οι γράφοι γνώσεις είναι σημαντικοί για τη δημιουργία νέων, προσαρμοσμένων πληροφοριών μέσω των δεδομένων που παρέχουν.

4.3 Το πρώτο έργο της Wikidata

Η πρώτη “αποστολή” για τη Wikidata ήταν να συγχρονίσει τις 287 γλωσσικές εκδόσεις της Wikipedia [26]. Για να είναι η Wikidata πολύγλωσση, θα πρέπει για παράδειγμα, το αντικείμενο που αντιπροσωπεύει τη Αθήνα να είναι το ίδιο και διαθέσιμο σε όλες τις γλώσσες. Ευτυχώς, η Wikipedia περιλαμβάνει ήδη έναν συνδεδεμένο μηχανισμό, τους συνδέσμους, που εμφανίζονται στην αριστερά πλευρά κάθε άρθρου και ενώνουν άρθρα σε ποικίλες γλώσσες. Αυτοί οι σύνδεσμοι προήλθαν από καταχωρήσεις κειμένου που επεξεργάζονται οι χρήστες στο κάτω μέρος του άρθρου. Παράδειγμα αποτελούν τα άρθρα που αφορούν την Αθήνα, που συνδέονται με το ίδιο άρθρο αλλά σε διαφορετική γλώσσα.

Η αποθήκευση και η διαχείριση των συνδέσμων σε μια κοινή τοποθεσία ήταν το πρώτο έργο της Wikidata [26]. Για κάθε σελίδα της Wikipedia κατασκευάστηκε μια σελίδα στη Wikidata, για τη διαχείριση των συνδέσμων όλων των σχετικών άρθρων της Wikipedia, σε όλες τις γλώσσες. Η κοινότητα της Wikidata δημιούργησε bots για τη μετακίνηση συνδέσμων από τη Wikipedia στη Wikidata, με στόχο την αφαίρεση των συνδέσμων της Wikipedia. Ενώ η Wikidata έχει την ιδιότητα να συνδέει μόνο σελίδες που αφορούν το ίδιο θέμα, υπάρχει η δυνατότητα να προσθέτει προσαρμοσμένους συνδέσμους σε ένα άρθρο, χωρίς να είναι αμφίδρομοι. Η προσθήκη γλωσσικών συνδέσμων, κατέστησε τη Wikidata ως ένα τεράστιο σύνολο αρχικών στοιχείων, που στηρίζονται σε πραγματικές σελίδες της Wikipedia.

4.4 Διαχείριση της Wikidata

Υπεύθυνοι για την υπηρεσία παροχής δεδομένων είναι η κοινότητα της Wikidata. Ειδικότερα αναλαμβάνει τη προσθήκη, την επεξεργασία και συντήρηση όλων των δεδομένων που αποτελούν τη δομή της. Η πλειοψηφία των εργασιών που αναφέρθηκαν παραπάνω είναι απλές και απαιτούν ελάχιστες ικανότητες ή γνώσεις. Ωστόσο, υπάρχουν διεργασίες πιο απαιτητικές που έχουν τη δυνατότητα, δυνητικά, να επηρεάσουν ένα μεγαλύτερο μέρος των γράφων. Με αυτό το τρόπο γίνεται η διάκριση των εργασιών σε ελαφριές και βαριές. Παράδειγμα μιας ελαφριάς εργασίας, είναι η έλλειψη μιας πληροφορίας σχετικά με το τόπο γέννησης του Σαίξπηρ. Στη περίπτωση αυτή, οι χρήστες μπορούν απλώς να καταχωρίσουν έναν ισχυρισμό χρησιμοποιώντας την ιδιότητα του τόπου γέννησης και συσχετίζοντάς τη με το στοιχείο Stratford-upon-Avon. Οι ενέργειες της εισαγωγής και τροποποίησης ισχυρισμών και ετικετών απαιτεί λιγοστές εξειδικευμένες γνώσεις και είναι σε μεγάλο βαθμό ανεξάρτητες από άλλες. Οι εργασίες που προαναφέρθηκαν αφορούν τις εργασίες με ελαφρά συνεισφορά. Αντίθετα, η προσθήκη δηλώσεων που χρησιμοποιούν τις ιδιότητες instance of ή subclass of, ανήκουν στην “βαριά” εργασία. Απαιτείται από τα άτομα της κοινότητας να είναι αρκετά εξοικειωμένα με τις έννοιες της Μηχανικής, ώστε να κατανοήσουν τη διαφορά μεταξύ των δύο ιδιοτήτων και να τις χρησιμοποιούν ανάλογα. Επομένως, μια απαιτητική εργασία μπορεί να αποσυντεθεί σε μικρότερες εργασίες, οι οποίες είναι εξαρτώμενες η μία από την άλλη και περιλαμβάνουν τόσο ελαφρές όσο και βαριές εργασίες.

Οι συντάκτες της πληροφορίας στα Wikidata όπως προαναφέρθηκε, είναι άνθρωποι αλλά και bots. Τα bots που δημιουργούνται από τους χρήστες, για να έχουν τη δυνατότητα επεξεργασίας των δεδομένων και εκτέλεσης κάποιας λειτουργίας, απαιτούν έγκριση από τη κοινότητα. Παρόλα αυτά, αρκετά είναι αυτά που λειτουργούν χωρίς επίσημη έγκριση. Η λειτουργία του bot δεν είναι αυτόνομη, αλλά χειρίζεται από έναν χρήστη, ο οποίος είναι υπεύθυνος για τη συντήρηση και ασφαλή του χρήση ως προς το γράφο. Τα καταγεγραμμένα bots σε λειτουργία είναι 450 σε αριθμό και αυτά είναι υπεύθυνα για το 95% των επεξεργασιών στη Wikidata. Η τακτική της Wikidata υποχρεώνει τα bots να υποβάλλουν αναφορά γα κάθε πρόταση που αναλαμβάνουν και να λειτουργούν χωρίς να παραβιάζουν οποιονδήποτε κανόνα της Wikidata.

Θίγοντας το ζήτημα του πεδίου εφαρμογής των bots, διαπιστώνεται ότι είναι σύνηθες να εξειδικεύονται σε ένα μόνο θέμα-κατηγορία. Για παράδειγμα, κάποια είναι υπεύθυνα για να ασχολούνται με τον τομέα της βιολογίας, άλλα με χώρες, και άλλα αντιστοιχίζουν πληροφορίες της Wikidata με τα αντίστοιχα τους σε άλλους γράφους γνώσης, όπως η DBpedia. Ακόμη κάποιες από τις λειτουργίες για τις οποίες είναι υπεύθυνα είναι ο έλεγχος παραβιάσεων, η μετακίνηση σελίδων και η διόρθωση λαθών στο όνομα των χρηστών. Επιπρόσθετα, ανέλαβαν την μετακίνηση γλωσσικών συνδέσμων από τη Wikipedia στη Wikidata, ώστε να επιτευχθεί η δημιουργία διαγλωσσικών συνδέσμων με σκοπό τη σύνδεση των ποικίλων γλωσσικών εκδόσεων. Εν κατακλείδι, οι λειτουργίες επιτυγχάνονται είτε από τους χρήστες είτε από bots, χωρίς να υπάρχει σαφής προτίμηση διάκρισης.

4.5 Πώς λειτουργούν τα Wikidata

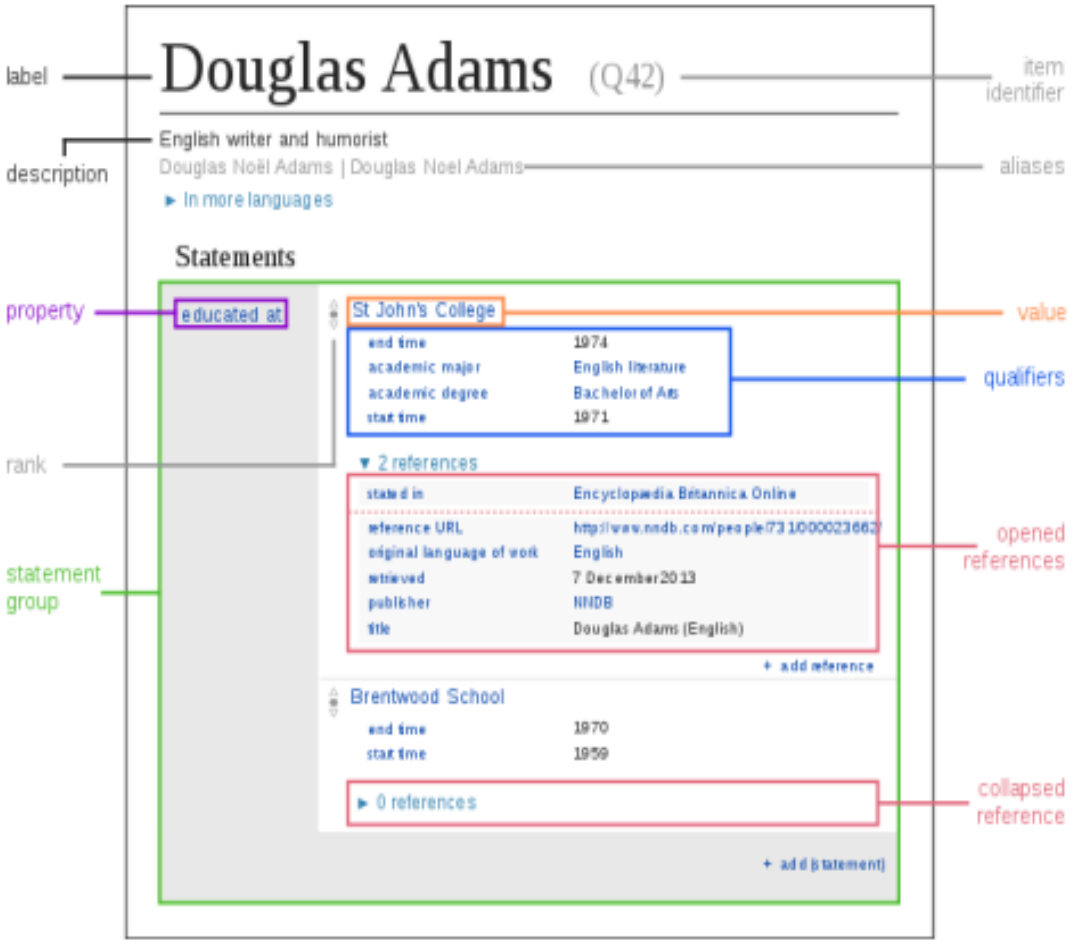
4.5.1 Δομή Δεδομένων

Η δημιουργία μιας δομής δεδομένων απαιτεί εκτενή σχεδιασμό. Η οργάνωση της Wikidata είναι οργανωμένη σε σελίδες, όπως συμβαίνει και με τα δεδομένα. Κάθε θέμα για το οποίο υπάρχουν δομημένα δεδομένα ονομάζεται οντότητα. Κάθε οντότητα αποτελεί μια σελίδα, όπου συγκεντρώνονται όλα τα δεδομένα σχετικά με αυτή. Κάθε σελίδα διαθέτει ένα μοναδικό αναγνωριστικό, που αρχίζει πάντα με το γράμμα «Q» ακολουθούμενο από κάποιους αριθμούς (π.χ. Q17151 η σελίδα που αναφέρεται στην πόλη Θεσσαλονίκη). Αν και χρήσιμο για τις μηχανές και για την αναπαράσταση της

γνώσης σε πολλές διαφορετικές γλώσσες, αυτό το αναγνωριστικό δεν είναι πολύ φιλικό και κατανοητό προς τον άνθρωπο.

Το σύστημα διακρίνει μέχρι στιγμής δύο τύπους οντοτήτων, τα items και τα property. Τα item αντιπροσωπεύουν άτομα και κλάσεις, ενώ τα properties της Wikidata προσομοιάζουν τις ιδιότητες των Resource Description Framework (RDF). Αυτή τη στιγμή η Wikidata περιλαμβάνει περισσότερα από 90 εκατομμύρια items. Τα items μπορούν να αντιπροσωπεύουν αφηρημένες έννοιες όπως το ύψος, καθώς και αντικείμενα του πραγματικού κόσμου όπως μια καρέκλα, ένα βουνό.

Σχεδόν κάθε άρθρο της Wikipedia, σε οποιαδήποτε γλώσσα και αν είναι αποτυπωμένο, αντιπροσωπεύει ένα item. Αντίστοιχα κάθε item αντιστοιχεί σε μια σελίδα, στην οποία έχουν πρόσβαση οι χρήστες.[27]



Εικόνα 5, Περιγραφή στοιχείων σε ένα item [28]

4.5.1.1 Items (αντικείμενα)

Κάθε σελίδα που αντιπροσωπεύει ένα item αποτελείται από ένα label (ετικέτα), ένα description (περιγραφή), καθώς και ένα πλήθος aliases (ψευδώνυμα). Στη Wikidata τα αντικείμενα χρησιμοποιούνται για να αντιπροσωπεύουν οτιδήποτε γνωστό στον άνθρωπο, συμπεριλαμβανομένων θεμάτων, εννοιών και αντικειμένων. Για παράδειγμα, ο Douglas Adams, το "Πρωτάθλημα", η "Ελλάδα", ο "Φιλόσοφος Σωκράτης" και ο "σκύλος", είναι όλα αντικείμενα στη Wikidata.

Ένα κύριο χαρακτηριστικό των item είναι η μοναδικότητα, δηλαδή κάθε αντικείμενο αντιπροσωπεύεται από ένα και μοναδικό item. Επιπλέον χαρακτηριστικό είναι η συνδεσιμότητα των item μεταξύ τους. Δηλαδή οι σελίδες επιτρέπουν τη σύνδεση μεταξύ αυτών, ώστε όλα τα δεδομένα στη Wikidata να μπορούν να διασυνδεθούν. Ένας σύνδεσμος προς μια άλλη σελίδα συνήθως προστίθεται ως property. Εάν δεν υπάρχει κάποιο κατάλληλο property, δίνεται η δυνατότητα προσθήκης του από το χρήστη.

4.5.1.2 Properties (Ιδιότητες)

Οι ιδιότητες αναπαριστούν τις σχέσεις των αντικειμένων, όπως η ημερομηνία γέννησης. Αυτές οι σχέσεις μοιάζουν με μια τριπλέτα RDF, της μορφής αντικείμενο-ιδιότητα-τιμή, όπου η τιμή μπορεί να είναι είτε ένα στοιχείο είτε μια λέξη. Οι ιδιότητες, όπως και τα items, περιγράφονται σε σελίδες και χρησιμοποιούν αναγνωριστικά που αρχίζουν με "P" (π.χ. P19 η ιδιότητα που αναφέρεται στο τόπο γέννησης). Κάθε property ορίζει το τύπο δεδομένων των τιμών που δέχεται ως statement.

Τα properties είναι παρόμοια με τα items, για δυο λόγους. Όπως τα items, κάθε property έχει μια ετικέτα, μια περιγραφή, και ένα ψευδώνυμο (ή ψευδώνυμα) που μπορούν να προστεθούν σε πολλές γλώσσες. Τα properties περιέχουν επίσης statements, που βοηθούν στη πληρέστερη περιγραφή τους, συμπεριλαμβανομένων των περιορισμών σχετικά με το τρόπο χρήσης του property.

Ωστόσο η ειδοποιός διαφορά είναι ότι τα properties δεν διαθέτουν κάποια ενότητα στις σελίδες τους, ώστε να επιτυγχάνεται η σύνδεση ιστοτόπων με άλλα έργα της Wikimedia. Καθώς ακόμη δεν διαθέτουν και εξωτερικά αναγνωριστικά.

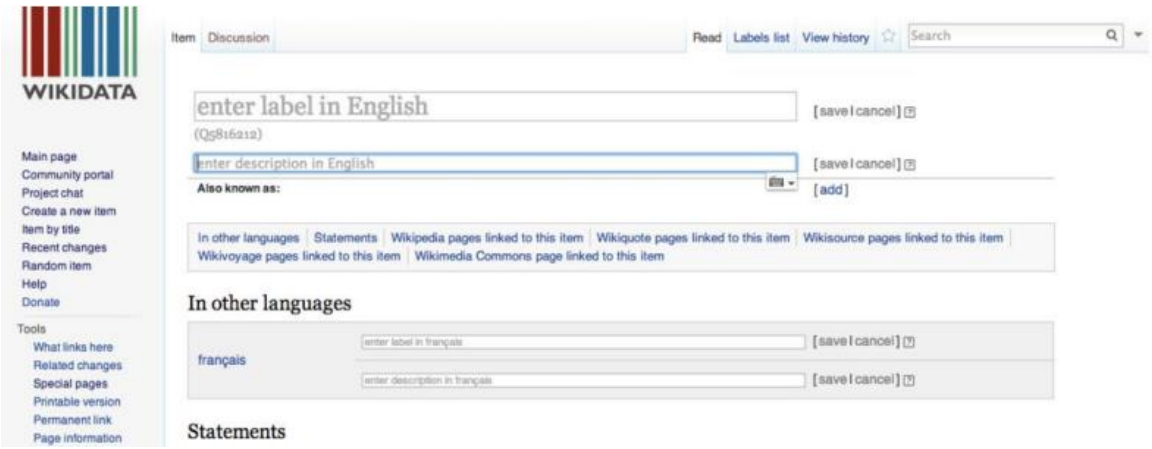
Εικόνα 6, Παράδειγμα μιας τριπλέτας [29]

4.5.1.3 Label (Ετικέτες)

Η ετικέτα προσομοιάζει το τίτλο μιας σελίδας που περιγράφει το αντικείμενο και πρέπει να είναι όσο το δυνατόν πιο σύντομη, χωρίς να είναι απαραίτητα μοναδική. Ωστόσο, κανένα στοιχείο δεν μπορεί να έχει την ίδια ετικέτα και την ίδια περιγραφή. Το κάθε item διαθέτει πλήθος labels σε κάθε γλώσσα, χωρίς να είναι απαραίτητο αυτά να συσχετίζονται μεταξύ τους.

4.5.1.4 Description (Περιγραφή)

Οι ετικέτες, οι περιγραφές και τα ψευδώνυμα ονομάζονται όροι και χρησιμοποιούνται για την αναζήτηση και εμφάνιση αντικειμένων. Η περιγραφή ενός άρθρου της Wikidata, είναι μια σύντομη φράση που προορίζεται να διευκρινίσει άρθρα με τις ίδιες ή παρόμοιες ετικέτες. Δεν απαιτείται η πρωτοτυπία της περιγραφής. Δηλαδή αν και δύο προϊόντα δεν μπορούν να έχουν την ίδια ετικέτα και περιγραφή, πολλά στοιχεία έχουν την ίδια περιγραφή.



Εικόνα 7, Υπόδειξη του σημείου περιγραφής [27]

4.5.1.5 Statement (Δήλωση αντικειμένου)

Στη Wikidata μια έννοια ή ένα αντικείμενο αντιπροσωπεύεται από ένα item. Ένα statement είναι ο τρόπος με τον οποίο οι πληροφορίες για ένα item καταγράφονται στη Wikidata. Αυτό συμβαίνει συνδυάζοντας ένα property με τουλάχιστον μία τιμή. Για παράδειγμα, «τοποθεσία -> Ελλάδα». Σε ένα statement, το property περιγράφει την τιμή δεδομένων και μπορεί να θεωρηθεί ως μια κατηγορία δεδομένων.

Στο παράδειγμα μας με τον Douglas Adams, στο property εκπαίδευση (P69), προστίθεται ως τιμή το κολέγιο St John College (Q691283). Τόσο το κολλέγιο όσο και η ιδιότητα εκπαίδευση έχουν τις δικές τους σελίδες στη Wikidata, επιτρέποντας έτσι το item Douglas Adams να συνδέεται με αυτά.

Στα statements ακόμη μπορούν να προστεθούν οι qualifiers, οι οποίοι παρέχουν πρόσθετες πληροφορίες για την οντότητα. Σε κάθε statement ακόμη μπορεί να συναντάται πάνω από ένα reference, τα οποία έχουν σκοπό να υποστηρίζουν το qualifier.

4.5.1.6 Αναφορά (Reference) στα statement

Οι αναφορές μπορεί να είναι παραπομπές ή αναφορές σε ιστοτόπους που μπορεί να αντιπροσωπεύουν ένα στοιχείο. Το reference δηλαδή έχει τη δυνατότητα να προσδιορίζει τη πηγή από την οποία προήλθε η πληροφορία. Σημαντικός είναι ο περιορισμός της Wikidata για τη μη επαναχρησιμοποίηση των αναφορών. Δηλαδή το σύστημα

αποθηκεύει τις αναφορές, εκ των οποίων ένα μεγάλο ποσοστό τυγχάνει να είναι ίδιο σε πολλές περιπτώσεις. Αναφορικά, τον Απρίλιο του 2013, υπάρχουν 23.225.184 αναφορές σε δηλώσεις της Wikidata, αλλά μόνο οι 124.068 είναι διαφορετικές. Η σύνταξη με την οποία παρουσιάζονται τα references είναι ζεύγη ιδιοτήτων-τιμών.

Εικόνα 8, Επεξήγηση της τοποθεσίας των Qualifier, Reference [27]

4.5.1.7 Ειδικές τιμές none, some

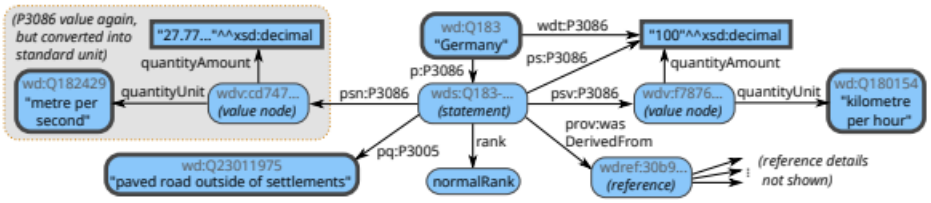
Υπάρχουν φορές που συναντώνται property των οποίων η τιμή δεν είναι διαθέσιμη ή είναι άγνωστη. Οι ελλείπουσες τιμές εξακολουθούν να παρέχουν σημαντικές πληροφορίες, ανάλογα με το property, και πρέπει να καταγράφονται στη Wikidata. Για παράδειγμα, θα μπορούσαμε να πούμε ότι η Ελισάβετ Α της Αγγλίας (Q7207) δεν είχε καμία τιμή για την property του παιδιού (P40), το οποίο είναι πολύ διαφορετικό από το να μην καταγράφει καθόλου. Θα μπορούσαμε επίσης να πούμε ότι ο William Shakespeare (Q692) έχει μια άγνωστη τιμή για την ημερομηνία γέννησης του(P569).

Για την επίλυση του προβλήματος της ύπαρξης κενού ή άγνωστης τιμής, στη Wikidata συναντάται η χρήση των τιμών none και some. Το None χρησιμοποιείται για να υποδείξει ότι η συγκεκριμένη ιδιότητα δεν έχει καμία τιμή. Το Some αντίστοιχα, χρησιμοποιείται όταν γνωρίζουμε ότι μια ιδιότητα έχει μια τιμή, αλλά δε γνωρίζουμε ή δε μπορούμε να δώσουμε περισσότερες λεπτομέρειες. Αυτό είναι παρόμοιο με τη χρήση των κενών κόμβων στο RDF. Οι δύο αυτές τιμές μπορούν να χρησιμοποιηθούν σε όλα τα μέρη όπου επιτρέπονται οι κανονικές τιμές ιδιοτήτων.

4.5.2 Encoding Wikidata in RDF

Οι γράφοι γνώσης της Wikidata χρησιμοποιούν αρχεία μορφής JSON για την αποθήκευση και επεξεργασία τους. Η δομή αυτή του γράφου προσομοιάζει τη μορφή του RDF, δηλαδή τις συνδέσεις των στοιχείων με τα properties [30]. Το σημείο ως προς το οποίο διαφέρουν τα Wikidata από το RDF, είναι το μεγάλο πλήθος γνώσεων-πληροφορίας που μπορεί να μεταφέρει ο γράφος KG έναντι του RDF.

Για την αποτύπωση των δεδομένων της Wikidata με την μορφή RDF, έχει αναπτυχθεί ειδικό λεξιλόγιο. Ένα τέτοιο παράδειγμα απεικονίζεται στη παρακάτω εικόνα, το οποίο είναι αντιπροσωπευτικό λόγω της τοποθέτησης των βοηθητικών κόμβων και labels για λόγους υποστήριξης.



Εικόνα 9, Ένας σύνθετος γράφος της Wikidata [30]

4.5.3 Χαρακτηριστικά των δεδομένων

Διαφορετικοί τύποι δεδομένων είναι σήμερα διαθέσιμοι, συμπεριλαμβανομένων των μεγάλων δεδομένων, των πειραματικών δεδομένων, των ανοικτών δεδομένων και των μεταδεδομένων. Παρά το γεγονός ότι όλα είναι δεδομένα, κάθε κατηγορία έχει διαφορετικό νόημα και λειτουργία. Ωστόσο όλα μοιράζονται τον κοινό στόχο της καλύτερης εξήγησης και κατανόησης του περιβάλλοντος στο οποίο ζούμε. Με μια πιο γενική έννοια, μπορεί κανείς να πει ότι τα δεδομένα μπορούν να χρησιμοποιηθούν για τη δημιουργία πληροφοριών, δεδομένου ότι είναι αλληλένδετα μεταξύ τους. Οι τιμές των δεδομένων μπορεί να είναι είτε ποσοτικές είτε αριθμητικές, όπως μια μέτρηση ή ένα ποσό. Μπορεί επίσης να είναι ποιοτικές, όπως μια σύγκριση ή μια περιγραφή. Το ακόλουθο παράδειγμα μπορεί να συμβάλει στην κατανόηση όλων όσων ειπώθηκαν παραπάνω. Μια τιμή δεδομένων, είναι το υψόμετρο του Βόρα που είναι 2.524 μέτρα. Παρόλο που αναγνωρίζεται ότι το 2.524 αναφέρεται στις υψηλότερες κορυφές της

Ελλάδας, ο ίδιος ο αριθμός δεν έχει καμία σημασία. Για να είναι χρήσιμη η πληροφορία πρέπει να διευκρινιστεί η μονάδα μέτρησης και τα ύψη τυχόν γειτονικών βουνών. Εν κατακλείδι, προέχει η οργάνωση και σύνδεση των δεδομένων, για να επιτευχθεί η εκμάθηση και η εξαγωγή πληροφορίας.

4.5.3.1 Πού βρίσκονται τα δεδομένα

Τα δεδομένα είναι παντού γύρω μας. Υπάρχουν πολλοί διαφορετικοί τύποι πηγών δεδομένων, όπως αυτές που παρέχουν οικονομικά, βιολογικά ή κοινωνικά δεδομένα. Όλα αυτά είναι πιθανές πηγές, αλλά δεδομένα θεωρούνται μόνο όσα είναι καταγεγραμμένα και οργανωμένα. Η απουσία μιας υποκείμενης δομής, καθιστά τα δεδομένα χωρίς νόημα και άχρηστα. Με τη χρήση του όρου "δομημένα δεδομένα", εννοούμε δεδομένα που έχουν ταξινομηθεί [31].

4.5.3.2 Ποιότητα δεδομένων

Τα τελευταίο διάστημα έχει αυξηθεί η έρευνα γύρω από τη ποιότητα των δεδομένων στη Wikidata. Οι ασυνέπειες στη ταξινόμηση και οργάνωση της Wikidata είναι στο στόχαστρο, εφόσον βρέθηκε πληθώρα λαθών που προέρχονται από τη κακή χρήση των item. Ο τρόπος με τον οποίον ελέγχονται τα δεδομένα για τη ποιότητα τους, εξαρτάται είτε από την εργασία που πρόκειται να εκτελεστεί είτε από τις ανάγκες των καταναλωτών [31].

Παρακάτω παρουσιάζονται κάποιες μετρικές μέτρησης της ποιότητας, που χρησιμοποιούνται στα πλαίσια αξιολόγησης.

- Ακρίβεια
  Ο βαθμός στον οποίο τα δεδομένα γίνονται αποδεκτά ως αληθή και απαλλαγμένα από σφάλματα. Βάση ισχυρισμών για να μπορούν τα δεδομένα να είναι ακριβή, πρέπει οι τιμές τους να αντιστοιχούν σε μια κατάσταση στον πραγματικό κόσμο, δηλαδή να υπάρχουν ανεξάρτητα από τον παρατηρητή.
- Αξιοπιστία

Ο βαθμός στον οποίο ο χρήστης θεωρεί τα δεδομένα αληθινά. Εξαρτάται τόσο από την φερεγγυότητα των δημιουργών των δεδομένων, όσο και από την αξιολόγηση των χρηστών. Στην διάσταση της αξιοπιστίας περιλαμβάνονται ακόμη η φήμη, η αντικειμενικότητα και η επαληθευσιμότητα. Συνεπώς για να είναι αξιόπιστα τα δεδομένα πρέπει η πηγή τους και το περιεχόμενό τους να είναι αξιόλογα (φήμη), αμερόληπτα, αδέσμευτα από προκαταλήψεις (αντικειμενικά), και η ορθότητά τους να είναι εύκολο να ελεγχθεί (επαληθεύσιμη).

- Συνέπεια

  Ένα συνεπές σύνολο δεδομένων είναι απαλλαγμένο από τις παραβιάσεις των κανόνων που ορίζονται σε ένα σύνολο items. Με βάση μια άλλη οπτική, ένα σύνολο δεδομένων είναι συνεπές εάν είναι ελεύθερο από αντικρουόμενες πληροφορίες.

- Συνάφεια

  Αφορά το πόσο ωφέλιμα και ουσιώδη είναι τα δεδομένα για την εργασία του εκάστοτε χρήστη.

- Πληρότητα

  Ο βαθμός στον οποίο τα δεδομένα είναι επαρκούς εύρους, βάθους και πεδίου εφαρμογής για την εκάστοτε εργασία.

- Επικαιρότητα

  Αξιολογεί πόσο ενήμερα είναι τα δεδομένα. Ο διαφορετικός χρόνος ενημέρωσης των πηγών δεδομένων, έχει ως αποτέλεσμα τα δεδομένα να χάσουν την επικαιρότητα.

- Ευκολία κατανόησης

  Αφορά το πόσο σαφή και κατανοητά είναι τα δεδομένα.

- Προσβασιμότητα

  Ο βαθμός στον οποίο τα δεδομένα είναι διαθέσιμα ή εύκολα και άμεσα ανακτήσιμα. Οι πηγές δεδομένων στο διαδίκτυο πρέπει να είναι διαθέσιμες να παράγουν πληροφορίες προσαρμοσμένες στο χρήστη.

- Διασύνδεση

  Αναφέρεται στο πόσο εφικτό είναι οι οντότητες που αντιπροσωπεύουν την ίδια έννοια να συνδέονται μεταξύ τους.

- Άδεια χρήσης

Η εξαγωγή συμπερασμάτων και πληροφοριών επιτυγχάνεται μέσω των συνδέσμων των συνόλων δεδομένων στον ιστό. Ορισμένες πηγές δεδομένων δεν είναι εφικτό να επαναχρησιμοποιηθούν. Συνεπώς, είναι σημαντικό να παρέχονται στους χρήστες σύνολα δεδομένων με άδεια χρήσης, η οποία συγκαταλέγει την δυνατότητα επαναχρησιμοποίησης και διαμοιρασμού.

ΚΕΦΑΛΑΙΟ 5: ΜΕΘΟΔΟΛΟΓΙΑ

Σκοπός του κεφαλαίου είναι ανάπτυξη της μεθοδολογίας που ακολουθήθηκε στη παρούσα διπλωματική εργασία.

5 Μεθοδολογία

5.1 Εισαγωγή στη Μεθοδολογία

Σε αυτό το κεφάλαιο περιγράφεται λεπτομερώς η μεθοδολογία που ακολουθήθηκε. Αρχικά παρουσιάστηκε ο τρόπος δημιουργίας ενός μέρους του συνόλου δεδομένων ogb-wikikg2, με τη χρήση Wikidata query. Επιλέχθηκε και αναλύθηκε ενδεικτικά ένα σύνολο 10 ιδιοτήτων, που χρησιμοποιήθηκε για την δημιουργία ενός μέρους των τριπλετών του συνόλου δεδομένων. Μετά την ολοκλήρωση του query, ακολουθεί η ανάλυση του κώδικα. Αναλύεται το αρχείο dataloader.py, δηλαδή αναλύεται ο τρόπος που έγινε η φόρτωση και επεξεργασία των δεδομένων του ogb-wikikg2. Σε αυτό το αρχείο δημιουργούνται τα σύνολα δεδομένων test, train και valid. Έπειτα ακολουθεί το αρχείο transfile.py, που παρουσιάζει τη δημιουργία βοηθητικών παραμέτρων που χρησιμοποιούνται στη πρόβλεψη. Στη συνέχεια ακολουθεί η ανάλυση του αρχείου model.py, το οποίο παρουσιάζει τις διαθέσιμες μεθόδους, και τις παραμέτρους που χρησιμοποιούνται στη πρόβλεψη. Σε αυτό επιτυγχάνεται η εκπαίδευση του μοντέλου, χρησιμοποιώντας ένα υποσύνολο των τελικών δεδομένων. Τελικό στάδιο του κώδικα model.py, πριν την ολοκλήρωση, είναι η αξιολόγηση της μεθόδου που επιλέχθηκε. Ακολουθούν τα αποτελέσματα της μεθόδου που επιλέχθηκε. Παρουσιάζονται τα αποτελέσματα των πειραμάτων του tuning στις παραμέτρους του μοντέλου.

5.2 Περιγραφή της Μεθοδολογίας

Αρχικά κατανοήθηκαν τα δεδομένα της Wikidata, τα οποία προσφέρονται από την ιστοσελίδα https://www.wikidata.org/wiki/Wikidata:Introduction/el. Στη συνέχεια έγινε η επιλογή της μεθόδου που θα χρησιμοποιηθεί προς ανάλυση, καθώς στο OGB παρουσιάζεται πλήθος εφαρμοσμένων μεθόδων στα δεδομένα ogb-wikikg2. Την επιλογή της μεθόδου, ακολούθησε η μελέτη του κώδικα και του paper που αναφέρονταν στην μέθοδο αυτή. Η μέθοδος που επιλέχθηκε ήταν συνδυασμός των μεθόδων StarGraph & TripleRE. Το paper και ο κώδικας που χρησιμοποιήθηκαν βρίσκονται στο αποθετήριο του OGB [32].

Η μέθοδος TripleRE είναι μια μέθοδος για την ενσωμάτωση σχεσιακών πληροφοριών στα GNN. Η μέθοδος χρησιμοποιείται συχνά στους γράφους γνώσης, όπου ο στόχος είναι η πρόβλεψη των σχέσεων μεταξύ οντοτήτων στο γράφο. Τα δεδομένα που λαμβάνει αναπαρίσταται ως μια τριπλέτα (h, r, t), όπου h και t είναι οι οντότητες κεφαλής και ουράς που συνδέονται με την ακμή και r είναι ο τύπος της σχέσης [33], [34]. Η μέθοδος μαθαίνει να κωδικοποιεί αυτές τις τριπλέτες ως latent διανύσματα, τα οποία μπορούν να χρησιμοποιηθούν για την πρόβλεψη πιθανής σχέσης μεταξύ οντοτήτων.

Η StarGraph είναι μια μέθοδος που αποθηκεύει γράφους οντοτήτων, που περιέχουν ελλιπείς πληροφορίες ως προς τη σύνδεση των κόμβων. Περιλαμβάνει δύο στάδια, τη δημιουργία και την κωδικοποίηση ενός υπογράφου για κάθε κόμβο [33], [34]. Οι ληφθέντες γράφοι οντοτήτων, μαζί με τα relation embeddings, χρησιμοποιούνται για τον υπολογισμό των βαθμολογιών[4] των τριπλετών.

[4] Η πιθανότητα η τριπλέτα να είναι αληθής

ΚΕΦΑΛΑΙΟ 6: WIKIDATA QUERY

Σκοπός του κεφαλαίου είναι η δημιουργία ενός query, του οποίου τα αποτελέσματα- τριπλέτες, θα προσομοιώνουν ένα μέρος των δεδομένων του ogb_wikikg2. Ακόμη γίνεται μια αναλυτική παρουσίαση του sparql query που χρησιμοποιούνται στα RDF. Περιγράφονται οι ενότητες που αποτελούν το query και παρουσιάζεται αντίστοιχο παράδειγμα.

6 Wikidata Query

6.1 Η έννοια του sparql query

Η SPARQL είναι μια γλώσσα για τη δημιουργία ερωτημάτων, για την αναζήτηση και το χειρισμό δεδομένων RDF. Πρόκειται για μια τυποποιημένη γλώσσα παρόμοια στη λειτουργία της με την SQL, η οποία χρησιμοποιείται για την αναζήτηση ερωτημάτων σε σχεσιακές βάσεις δεδομένων.

Το ερώτημα SPARQL είναι ένας τρόπος ανάκτησης πληροφοριών, οι οποίες καθορίζονται από ένα σύνολο προτύπων. Το query αποτελείται, τις περισσότερες φορές, από τρία μέρη (τα πρότυπα), μέσω των οποίων περιορίζεται το σύνολο δεδομένων που λαμβάνουμε. Ακολουθεί μια περιγραφική ανασκόπηση των προτύπων.

- **SELECT**, καθορίζει τις μεταβλητές του συνόλου δεδομένων
- **WHERE**, ορίζει τους περιορισμούς προσδιορίζοντας τις τιμές των ιδιοτήτων και τις σχέσεις μεταξύ των διαφορετικών οντοτήτων
- **GROUP BY** ή **HAVING**, χρησιμοποιείται προαιρετικά για την ομαδοποίηση ή το φιλτράρισμα των αποτελεσμάτων
- **FILTER**, εφαρμόζει μια συνθήκη φιλτραρίσματος στα αποτελέσματα του ερωτήματος

Κατά την δημιουργία του ερωτήματος και μέσα στην ενότητα **FILTER**, μπορούν να χρησιμοποιηθούν λογικοί τελεστές όπως and, or και not, τελεστές σύγκρισης όπως >=, <=,==, κ.λπ. αλλά και μαθηματικοί τελεστές.

Παρακάτω παρουσιάζεται ένα ενδεικτικό sparql query που ανακτά τα ονόματα των αθλητών, τον τόπο γέννησης και το ύψος τους. Έχοντας ως περιορισμό τη τιμή ύψος να είναι πάνω από 1.85 μέτρα. Ο ιστότοπος που χρησιμοποιείται για την δημιουργία ερωτήματος είναι ο **https://dbpedia.org/sparql**.

```
SELECT ?athlete  ?birthPlace  ?height

WHERE { ?athlete dbo:occupation dbr:Athlete.
        ?athlete  dbo:birthPlace  ?birthPlace .
        ?athlete dbo:height ?height.

 FILTER( ?height > "1.85"^^ xsd:double)
}
```

Η εικόνα 10, παρουσιάζει τα αποτελέσματα που εξάγονται και των οποίων η εμφάνιση καθορίζεται από το Select. Τα αποτελέσματα είναι ενδεικτικά ως προς το πλήθος.

SPARQL | HTML5 table

athlete	birthPlace	height
http://dbpedia.org/resource/Richard_Gomez	http://dbpedia.org/resource/Manila	1.8542
http://dbpedia.org/resource/Richard_Gomez	http://dbpedia.org/resource/Philippines	1.8542
http://dbpedia.org/resource/Dave_Weill	http://dbpedia.org/resource/Berkeley,_California	2.0066
http://dbpedia.org/resource/Dave_Weill	http://dbpedia.org/resource/United_States	2.0066
http://dbpedia.org/resource/Bill_Tancred	http://dbpedia.org/resource/Pakistan	1.9304
http://dbpedia.org/resource/Bill_Tancred	http://dbpedia.org/resource/Balochistan	1.9304
http://dbpedia.org/resource/Bill_Tancred	http://dbpedia.org/resource/Quetta	1.9304
http://dbpedia.org/resource/Fanie_du_Plessis	http://dbpedia.org/resource/South_Africa	1.905
http://dbpedia.org/resource/Fanie_du_Plessis	http://dbpedia.org/resource/Lichtenburg,_North_West	1.905
http://dbpedia.org/resource/Roy_Hollingsworth	http://dbpedia.org/resource/Trinidad	1.905
http://dbpedia.org/resource/Ian_Veneracion	http://dbpedia.org/resource/Manila	1.8542
http://dbpedia.org/resource/Nick_Sweeney	http://dbpedia.org/resource/Dublin	1.9812
http://dbpedia.org/resource/Nick_Sweeney	http://dbpedia.org/resource/Republic_of_Ireland	1.9812
http://dbpedia.org/resource/Sonika_Kaliraman	http://dbpedia.org/resource/India	1.93

Εικόνα 10, Ο πίνακας αποτελεσμάτων του query

6.2 Wikidata query

Στη βάση δεδομένων της Wikidata, το query μπορεί να χρησιμοποιηθεί για την ανάκτηση και το φιλτράρισμα πληροφοριών που είναι αποθηκευμένες σε αυτή. Η ανάκτηση των πληροφοριών επιτυγχάνεται μέσω της υπηρεσίας ερωτημάτων Wikidata Query Service (WDQS). Η WDQS επιτρέπει την δημιουργία ερωτημάτων που απευθύνονται στη βάση δεδομένων της Wikidata, χρησιμοποιώντας τη γλώσσα ερωτημάτων SPARQL [35].

Ακολουθεί ένα ερώτημα σε SPARQL, στο περιβάλλον της WDQS. Το ερώτημα ανακτά όλες τις τριπλέτες από τα Wikidata ,όπου η σχέση (ιδιότητα) είναι μία από τις πέντε που

καθορίζονται στο FILTER (wdt:P31, wdt:P19, wdt:P20, wdt:P106, wdt:P21). Οι μεταβλητές ?entity, ?relation και ?object αντιπροσωπεύουν το υποκείμενο, την ιδιότητα και το αντικείμενο της τριπλέτας, αντίστοιχα.

```
SELECT ?entity ?relation ?object
WHERE
{
?entity ?relation ? object .
FILTER (?relation in (wdt:P31, wdt:P19, wdt:P20, wdt:P106, wdt:P21, wdt:P509, wdt:P611, wdt:P682, wdt:P53, wdt:P1412))
}
```

Το παραπάνω παράδειγμα δημιουργεί ένα μέρος του συνόλου δεδομένων ogb_wikikg2. Ενδεικτικά, κάποιες από τις ιδιότητες των κόμβων του συνόλου δεδομένων ogb_wikikg2 είναι: το instance of (wdt:P31), το μέρος που πέθανε (wdt:P20), το επάγγελμα (wdt:P106), το φύλο του ανθρώπου (wdt:P21), το αίτιο θανάτου (P509), η βιολογική διεργασία (P682), το θρησκευτικό τάγμα (P611), οι γλώσσες στις οποίες είναι γνώστης (P1412), το μέρος που γεννήθηκε (P19) και η οικογένεια-δυναστεία (P53). Ένα δείγμα των τριπλετών που εξάγει το sparql παρουσιάζεται στο πίνακα 1. Για να προσομοιάσουμε τα δεδομένα του ogb-wikikg2, επιλέξαμε να εξάγουμε με το query το ίδιο σε μέγεθος σύνολο τριπλετών (περίπου 2.500.000) από τη Wikidata. Η ειδοποιός διαφορά είναι ότι χρησιμοποιήσαμε μόνο 10 ιδιότητες.

Πίνακας 1, Δείγμα συνόλου δεδομένων από το sparql query

Q114742540	P611	Q432
Q21759443	P682	Q1057
Q2215326	P106	Q116
Q5469452	P21	Q43445
Q51798013	P20	Q21
Q5300482	P509	Q2840
Q2854202	P20	Q29
Q475565	P735	Q923
Q1485261	P735	Q923
Q7026820	P19	Q16
Q1539488	P509	Q9687
Q1364776	P735	Q923
Q46999799	P20	Q16
Q10947916	P53	Q5066
Q21756827	P682	Q1057

Q15955998	P53	Q5066
Q351413	P20	Q21
Q12223054	P53	Q31711
Q2150287	P20	Q29
Q12526025	P20	Q28
Q4768865	P20	Q22
Q7052107	P20	Q25
Q7513009	P21	Q43445
Q62260	P509	Q9687
Q5431086	P53	Q31711
Q5948035	P19	Q15

Μια από τις τριπλέτες της εικόνας αναφέρεται στον Alexsandr Dyachenko (Q475565), του οποίου η ιδιότητα είναι «το όνομα που του δόθηκε (P735)». Ως αντικείμενο λαμβάνει το όνομα Alexsandr (Q923).

ΚΕΦΑΛΑΙΟ 7: ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

Το παρόν κεφάλαιο παραθέτει και σχολιάζει τις σημαντικότερες ενέργειες του κώδικα, που προβλέπει τις σχέσεις μεταξύ των οντοτήτων. Προηγείται η αναλυτική περιγραφή του κώδικα φόρτωσης και παραμετροποίησης των δεδομένων. Αναφέρεται η τοποθεσία των αρχείων και οι εντολές του Torch που συντελούν στην επίτευξη των αλγορίθμων. Τέλος, δίνεται ιδιαίτερη έμφαση στην ανάλυση των μεθόδων πρόβλεψης.

7 Ανάλυση κώδικα

7.1 Πρόβλεψη ακμών στη Wikidata

Η σημαντικότερη πρόκληση της παρούσας διπλωματικής ήταν η ανάλυση του κώδικα [36]. Τα δεδομένα και ο αλγόριθμος πρόβλεψης που ερευνήσαμε, βρίσκονται στην ιστοσελίδα https://ogb.stanford.edu/docs/leader_linkprop/#ogbl-wikikg2. Επιλέξαμε να τα αντλήσουμε από το συγκεκριμένο αποθετήριο, διότι διαθέτει ένα πίνακα σύγκρισης και αξιολόγησης της αποδοτικότητας των μοντέλων μάθησης. Αποτελείται από πολλαπλά σύνολα δεδομένων και παρέχει ένα τυποποιημένο πρωτόκολλο αξιολόγησης των μοντέλων πρόβλεψης συνδέσμων. Οι ερευνητές μπορούν να συγκρίνουν την απόδοση των μοντέλων τους με άλλα μοντέλα τελευταίας τεχνολογίας που προστίθενται.

Ο τρόπος με τον οποίο απεικονίζεται η απόδοση των μοντέλων αλλά και οι πληροφορίες του ερευνητή, φαίνεται στην εικόνα 11.

Rank	Method	Ext. data	Test MRR	Validation MRR	Contact	References	#Params	Hardware	Date
1	**StarGraph + TripleRE + Text**	Yes	0.7305 ± 0.0010	0.7442 ± 0.0006	Liang Yao (Tencent)	Paper, Code	1,927,395,330	Tesla V100 (32GB)	Jan 4, 2023
2	**InterHT+**	No	0.7293 ± 0.0018	0.7391 ± 0.0023	Baoxin Wang (HFL)	Paper, Code	156,332,770	Tesla A100 (80GB)	Dec 23, 2022

Εικόνα 11, Leaderboard των μεθόδων για το ogb_wikikg2

Στα επόμενα υποκεφάλαια θα γίνει ανάλυση του κώδικα πρόβλεψης που συναντάτε σε αρχεία της Python. Τα αρχεία που αναλύονται είναι τα dataloader.py, transformer.py και model.py. Η πρόσβαση σε αυτά δίνεται μέσω της επιλογής code, η οποία παραπέμπει στο project που είναι εναποθετημένο στο GitHub (https://github.com/hzli-ucas/StarGraph). Για να ξεκινήσει η εκτέλεση του προγράμματος, θα πρέπει αρχικά το σύνολο δεδομένων να φορτωθεί μέσω της run_ogb.py, η οποία ενεργοποιεί συναρτήσεις που συναντώνται στα παρακάτω αρχεία.

7.2 Αρχείο dataloader

Στο υποκεφάλαιο αυτό, αναλύουμε κάθε λειτουργία του κώδικα dataloader, ο οποίος είναι υπεύθυνος για την φόρτωση των δεδομένων.

Το module from __future__ που συναντάται στην αρχή του κώδικα, είναι ένας μηχανισμός για το προσδιορισμό και την εισαγωγή χαρακτηριστικών, που δεν αποτελούν μέρος της τρέχουσας έκδοσης της Python. Με την εισαγωγή αυτού του μηχανισμού δίνεται η δυνατότητα χρήσης των χαρακτηριστικών, ακόμα και αν η έκδοση της Python δε τα υποστηρίζει από προεπιλογή. Για παράδειγμα, στη Python 2 η εντολή print χρησιμοποιείται για την εκτύπωση τιμών στην οθόνη, αλλά στη Python 3 είναι μια απλή συνάρτηση. Ο μηχανισμός αυτός δίνει τη δυνατότητα, να χρησιμοποιείται η print ως εντολή εκτύπωσης και στην Python 3.

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import numpy as np
import torch

from torch.utils.data import Dataset
```

Εικόνα 12, Οι βιβλιοθήκες του dataloader.py

Στις πρώτες 3 εντολές που παρατηρείται στην εικόνα 12 εισάγονται τα εξής χαρακτηριστικά:

- **absolute_import**

  Συντελεί στο προσδιορισμό της τοποθεσίας ενός αρχείου, μόνο με τη χρήση του πλήρες ονόματος του αρχείου. Δηλαδή δε γίνεται η αναζήτηση του αρχείου στο προεπιλεγμένο directory της Python. Με αυτό τον τρόπο ο κώδικας γίνεται περισσότερο ευανάγνωστος και λιγότερο επιρρεπής σε προβλήματα που προκαλούνται από αλλαγές στο directory του project μας.

- **division**

  Αλλάζει τη συμπεριφορά του τελεστή διαίρεσης (/), ώστε να επιστρέφει πάντα μια τιμή κινητής υποδιαστολής, ακόμη και όταν οι δύο τελεστές είναι ακέραιοι.

- **print_function**

  Ορίζει τη συμπεριφορά της εντολής print ώστε να είναι συνάρτηση και όχι λέξη-κλειδί.

Στις υπόλοιπες εντολές εισάγονται οι βιβλιοθήκες numpy και torch, οι οποίες χρησιμοποιούνται συνήθως για αριθμητικούς υπολογισμούς και εργασίες ML. Αντίστοιχα η κλάση Dataset, από τη βιβλιοθήκη torch.utils.data, χρησιμοποιείται για την αναπαράσταση ενός συνόλου δεδομένων.

Στη συνέχεια στο κώδικα μας ορίζονται δύο κλάσεις, οι TrainDataset και οι TestDataset. Οι κλάσεις, κατά κύριο λόγο, δημιουργούν ένα σύνολο τριπλετών τις οποίες θα χρησιμοποιήσουν για την εκπαίδευση και δοκιμή του μοντέλου, με σκοπό την επίτευξη καλύτερης πρόβλεψης.

```
class TrainDataset(Dataset):
    def __init__(self, triples, nentity, nrelation, negative_sample_size, mode, count, true_head, true_tail, index_warp=None):
        self.len = len(triples['head'])
        self.triples = triples
        self.nentity = nentity
        self.nrelation = nrelation
        self.negative_sample_size = negative_sample_size
        self.mode = mode
        self.count = count
        self.true_head = true_head
        self.true_tail = true_tail
        self.index_warp = index_warp

    def __len__(self):
        return self.len

    def __getitem__(self, idx):
        head, relation, tail = self.triples['head'][idx], self.triples['relation'][idx], self.triples['tail'][idx]
        positive_sample = [head, relation, tail]

        subsampling_weight = self.count[(head, relation)] + self.count[(tail, -relation - 1)]
        subsampling_weight = torch.sqrt(1 / torch.Tensor([subsampling_weight]))

        negative_sample = torch.randint(0, self.nentity, (self.negative_sample_size,))
        if self.index_warp is not None:
            negative_sample = self.index_warp[negative_sample]
        positive_sample = torch.LongTensor(positive_sample)

        return positive_sample, negative_sample, subsampling_weight, self.mode

    @staticmethod
    def collate_fn(data):
        positive_sample = torch.stack([_[0] for _ in data], dim=0)
        negative_sample = torch.stack([_[1] for _ in data], dim=0)
        subsample_weight = torch.cat([_[2] for _ in data], dim=0)
        mode = data[0][3]
        return positive_sample, negative_sample, subsample_weight, mode
```

Εικόνα 13, Η κλάση TrainDataset στο dataloader.py

Όπως φαίνεται στην εικόνα 13, η κλάση TrainDataset δέχεται τα ορίσματα:

- triples

  Είναι ένα λεξικό με τρία κλειδιά: 'head', 'relation' και 'tail', καθένα από τα οποία αντιστοιχίζεται σε μια λίστα ακεραίων αριθμών. Οι ακέραιοι αντιστοιχούν στους

δείκτες των οντοτήτων κεφαλής, των σχέσεων, και των οντοτήτων ουράς κάποιου γράφου γνώσης.

- Nentity

Ακέραιος αριθμός που αντιπροσωπεύει τον αριθμό των οντοτήτων στο γράφο γνώσης.

- nrelation

Ακέραιος που αντιπροσωπεύει τον αριθμό των σχέσεων στο γράφο γνώσης.

- negative_sample_size

Ο αριθμός των αρνητικών δειγμάτων που πρέπει να δημιουργηθούν για κάθε τριπλέτα στο σύνολο δεδομένων. Σκοπός τους είναι η δημιουργία ενός μεγάλου αριθμού αρνητικών δειγμάτων, που μπορούν να χρησιμοποιηθούν για την εκπαίδευση ενός μοντέλου, ώστε να διακρίνονται οι αληθείς τριπλέτες από τις ψευδείς.

- Mode

Συμβολοσειρά που μπορεί να δέχεται τιμές "head-batch" ή "tail-batch’’. Υποδεικνύει το τρόπο κατασκευής του συνόλου δεδομένων, ώστε να βελτιστοποιεί τη δειγματοληψία αρνητικών παραδειγμάτων για τις οντότητες της κεφαλής ή της ουράς των τριπλετών, αντίστοιχα.

- count

Μια λίστα ακέραιων αριθμών που αντιπροσωπεύει τη συχνότητα κάθε τριπλέτας στο σύνολο δεδομένων.

- index_warp

Μια προαιρετική λίστα ακεραίων αριθμών, όπου η προκαθορισμένη τιμή της είναι None και αφορά την εύρεση τριπλετών.

Η συνάρτηση __getitem__ της κλάσης TrainDataset, δέχεται σαν ορίσματα το αντίκειμενο self και ένα συγκεκριμένο δείκτη. Με τη χρήση αυτών λαμβάνει την κεφαλή, τη σχέση και την ουρά της τριπλέτας και τις αποθηκεύει στις μεταβλητές head, relation και tail, αντίστοιχα. Στη συνέχεια κατασκευάζει ένα θετικό δείγμα, το οποίο είναι μια λίστα με περιεχόμενο τις τιμές των head, relation και tail. Υπολογίζει το βάρος

υποδειγματοληψίας για αυτό το δείγμα, αναζητώντας τον αριθμό των πλειάδων (head, relation), (tail, relation) και τα προσθέτει. Έπειτα το βάρος υποδειγματοληψίας αντικαθίσταται από την τετραγωνική ρίζα του αντιστρόφου του βάρους υποδειγματοληψίας. Δημιουργείται μια δέσμη αρνητικών δειγμάτων, από έναν τανυστή τυχαίων ακεραίων αριθμών μεταξύ 0 και nentity με μέγεθος negative_sample_size. Εάν η λίστα index_warp δεν είναι None, τότε οι δείκτες αντιστοιχίζονται σε δείκτες ενός διαφορετικού συνόλου δεδομένων. Η συνάρτηση επιστρέφει τα αρνητικά δείγματα, το βάρος υποδειγματοληψίας, το τρόπο λειτουργίας του συνόλου δεδομένων, και ένα τανυστή (τροποποιημένη λίστα θετικών δειγμάτων).

```python
class TestDataset(Dataset):
    def __init__(self, triples, args, mode, random_sampling):
        self.len = len(triples['head'])
        self.triples = triples
        self.nentity = args.nentity
        self.nrelation = args.nrelation
        self.mode = mode
        self.random_sampling = random_sampling
        if random_sampling:
            self.neg_size = args.neg_size_eval_train

    def __len__(self):
        return self.len

    def __getitem__(self, idx):
        head, relation, tail = self.triples['head'][idx], self.triples['relation'][idx], self.triples['tail'][idx]
        positive_sample = torch.LongTensor((head, relation, tail))

        if self.mode == 'head-batch':
            if not self.random_sampling:
                negative_sample = torch.cat([torch.LongTensor([head]), torch.from_numpy(self.triples['head_neg'][idx])])
            else:
                negative_sample = torch.cat(
                    [torch.LongTensor([head]), torch.randint(0, self.nentity, size=(self.neg_size,))])
        elif self.mode == 'tail-batch':
            if not self.random_sampling:
                negative_sample = torch.cat([torch.LongTensor([tail]), torch.from_numpy(self.triples['tail_neg'][idx])])
            else:
                negative_sample = torch.cat(
                    [torch.LongTensor([tail]), torch.randint(0, self.nentity, size=(self.neg_size,))])

        return positive_sample, negative_sample, self.mode

    @staticmethod
    def collate_fn(data):
        positive_sample = torch.stack([_[0] for _ in data], dim=0)
        negative_sample = torch.stack([_[1] for _ in data], dim=0)
        mode = data[0][2]

        return positive_sample, negative_sample, mode
```

Εικόνα 14, Η κλάση TestDataset στο dataloader.py

Αντίστοιχα η κλάση TestDataset δέχεται πληθώρα ορισμάτων, όπου μερικά από αυτά ακολουθούν παρακάτω.

- Args

Αντικείμενο που περιέχει ορίσματα τα οποία χρησιμοποιούνται για την εκτέλεση του προγράμματος, μεταξύ των οποίων είναι τα nentity και nrelation.

- random_sampling

  Ένα boolean με τιμές True ή False. Αν αυτό ορίζεται ως True, η δημιουργία αρνητικών δειγμάτων τριπλέτας θα γίνει με τυχαία δειγματοληψία.

Και οι δύο κλάσεις (TestDataset, TrainDataset) περιλαμβάνουν τις συναρτήσεις __len__ και __getitem__ , σκοπός των οποίων είναι η ανάκτηση του μήκους του συνόλου δεδομένων και του στοιχείου σε ένα συγκεκριμένο δείκτη. Σε αυτές τις κλάσεις συναντάται ακόμη η συνάρτηση collate_fn. Γενικότερος σκοπός των συναρτήσεων είναι η συγκέντρωση μιας παρτίδας δειγμάτων σε έναν ενιαίο τανυστή.

Παρακάτω, παραθέτονται αναλυτικότερα η λειτουργία των συναρτήσεων.

Η συνάρτηση __len__ επιστρέφει το μήκος του συνόλου δεδομένων, το οποίο ισούται με το πλήθος των τριπλετών στο λεξικό triples.

Με την υλοποίηση της η μέθοδος __getitem__ της κλάσης TestDataset επιστρέφει μια πλειάδα που περιέχει ένα θετικό δείγμα, ένα αρνητικό δείγμα και το τρόπο λειτουργίας των δεδομένων (mode). Το θετικό δείγμα είναι μια τριπλέτα ακεραίων αριθμών που αντιπροσωπεύουν μια οντότητα κεφαλής, μια σχέση και μια οντότητα ουράς. Το αρνητικό δείγμα είναι ένας τανυστής που περιέχει μια οντότητα κεφαλής ή μια οντότητα ουράς και μια λίστα αρνητικών δειγμάτων. Εάν η λειτουργία είναι 'head-batch', επιστρέφει τη κεφαλή της τριπλέτας και είτε τις προ-υπολογισμένες αρνητικές κεφαλές για την εν λόγω τριπλέτα (εάν το random_sampling είναι False) είτε ένα τυχαίο σύνολο οντοτήτων (εάν το random_sampling είναι True). Εάν η λειτουργία είναι "tail-batch", επιστρέφει την ουρά της τριπλέτας και είτε τις προ-υπολογισμένες αρνητικές ουρές για την εν λόγω τριπλέτα (εάν το random_sampling είναι False) είτε ένα τυχαίο σύνολο οντοτήτων (εάν το random_sampling είναι True).

Η συνάρτηση collate_fn είναι μια συνάρτηση ταξινόμησης, που μπορεί να χρησιμοποιηθεί για τη συγκέντρωση λίστας δειγμάτων. Χρησιμοποιείται συχνά στην κλάση DataLoader για να συγκεντρώσει μια λίστα δειγμάτων από ένα σύνολο δεδομένων, τα οποία περνούν στο μοντέλο για εκπαίδευση ή αξιολόγηση. Η collate_fn δέχεται μια λίστα δειγμάτων και επιστρέφει μια πλειάδα τανυστών που περιέχουν τα θετικά δείγματα, τα αρνητικά δείγματα, τα βάρη των υποδειγμάτων και το τρόπο λειτουργίας του κάθε δείγματος. Τα θετικά δείγματα στοιβάζονται μεταξύ τους

χρησιμοποιώντας την torch.stack, με τη παράμετρο dim να έχει οριστεί σε 0, ώστε ο τανυστής που προκύπτει να έχει τον ίδιο αριθμό γραμμών με τον αριθμό των δειγμάτων στη λίστα εισόδου. Όμοιο μοτίβο ακολουθείται στα αρνητικά δείγματα και στα βάρη των υποδειγμάτων, στα οποία στόχος είναι ο προκύπτων τανυστής να έχει τον ίδιο αριθμό γραμμών με τον αριθμό των δειγμάτων στη λίστα εισόδου.

```
class BidirectionalOneShotIterator(object):
    def __init__(self, dataloader_head, dataloader_tail):
        self.iterator_head = self.one_shot_iterator(dataloader_head)
        self.iterator_tail = self.one_shot_iterator(dataloader_tail)
        self.step = 0

    def __next__(self):
        self.step += 1
        if self.step % 2 == 0:
            data = next(self.iterator_head)
        else:
            data = next(self.iterator_tail)
        return data

    @staticmethod
    def one_shot_iterator(dataloader):
        '''
        Transform a PyTorch Dataloader into python iterator
        '''
        while True:
            for data in dataloader:
                yield data
```

Εικόνα 15, Η κλάση BidirectionalOneShotIterator στο dataloader.py

Η κλάση BidirectionalOneShotIterator της εικόνας 15, λειτουργεί ως επαναλήπτης σε ένα ζεύγος αντικειμένων του DataLoader. Η συνάρτηση __init__ που περιέχεται στην BidirectionalOneShotIterator, είναι μια ειδική μέθοδος, που καλείται όταν δημιουργείται ένα αντικείμενο της κλάσης. Η μέθοδος __next__ επιστρέφει εναλλάξ το επόμενο στοιχείο από τους επαναλήπτες iterator_head και iterator_tail, ανάλογα με την τιμή του μετρητή step. Ενώ η συνάρτηση one_shot_iterator χρησιμοποιεί έναν βρόχο while True και την εντολή yield για να αποδίδει τα στοιχεία του DataLoader ένα προς ένα, μετατρέποντας ουσιαστικά τον DataLoader σε έναν επαναλήπτη.

7.3 Αρχείο transfile

Η εκτέλεση του κώδικα επιτυγχάνεται με την εισαγωγή της βιβλιοθήκης torch, μια δημοφιλή βιβλιοθήκη βαθιάς μάθησης, και τις μεθόδου nn.

Η μέθοδος nn παρέχει έναν αριθμό επιπέδων και συναρτήσεων για τη δημιουργία νευρωνικών δικτύων. Περιλαμβάνει ένα ευρύ φάσμα στρωμάτων όπως συνελικτικά, αναδρομικά και στρώματα προσοχής, καθώς και συναρτήσεις ενεργοποίησης και κανονικοποίησης.

```python
class DropPath(nn.Module):
    """Drop path

    Randomly drop the input (i.e., output zero) with some probability, per sample.
    """

    def __init__(self, dropout_p=0.0):
        super().__init__()
        self.dropout_p = dropout_p

    def forward(self, x):

        if self.dropout_p == 0.0 or not self.training:
            return x

        keep_prob = 1 - self.dropout_p
        shape = (x.shape[0],) + (1,) * (x.ndim - 1)
        random_tensor = keep_prob + torch.rand(shape, dtype=x.dtype, device=x.device)
        random_tensor.floor_()  # 0 / 1

        # Discussion: https://github.com/rwightman/pytorch-image-models/discussions/895
        output = x / keep_prob * random_tensor
        return output
```

Εικόνα 16, Η κλάση DropPath στο transfile.py

Η κλάση DropPath, θέτει την τιμή της μεταβλητής dropout_p ίση με 0.0, το οποίο συνεπάγεται τη μη ύπαρξη τιμών εξόδου από το στρώμα. Οι συναρτήσεις που συναντώνται στη DropPath είναι:

Η μέθοδος forward που υλοποιεί το dropout. Εάν η μεταβλητή self.dropout_p είναι 0 ή το μοντέλο δε βρίσκεται σε κατάσταση εκπαίδευσης, επιστρέφει την είσοδο x αμετάβλητη. Διαφορετικά, δημιουργεί το τανυστή random tensor που παίρνει τυχαίες τιμές (0 ή 1). Τέλος, πολλαπλασιάζει το x/keep_prob με το τυχαίο τανυστή, μηδενίζοντας

ουσιαστικά κάποια από τα στοιχεία του x με πιθανότητα 0.0 (dropout_p=0.0), και επιστρέφει το αποτέλεσμα.

```python
class MLP(nn.Module):
    """MLP layer, usually used in Transformer"""

    def __init__(
        self,
        in_feat: int,
        mlp_ratio: int = 1,
        out_feat: int = 0,
        dropout_p: float = 0.0,
        act_layer: nn.Module = nn.ReLU,
    ):
        super().__init__()

        mid_feat = in_feat * mlp_ratio
        out_feat = out_feat or in_feat

        self.act = act_layer()

        self.linear1 = nn.Linear(in_feat, mid_feat)
        self.drop1 = nn.Dropout(dropout_p)

        self.linear2 = nn.Linear(mid_feat, out_feat)
        self.drop2 = nn.Dropout(dropout_p)

    def forward(self, x):
        x = self.drop1(self.act(self.linear1(x)))
        x = self.drop2(self.linear2(x))
        return x
```

Εικόνα 17, Η κλάση MLP στο transfile.py

Η κλάση MLP δημιουργεί ένα νευρωνικό δίκτυο που αποτελείται από πολλαπλά πλήρως συνδεδεμένα στρώματα. Πιο συγκεκριμένα, αποτελείται από τρεις τύπους στρωμάτων, το στρώμα εισόδου, το εξόδου και το κρυφό στρώμα. Το στρώμα εισόδου λαμβάνει τα προς επεξεργασία δεδομένα.

Η πρώτη συνάρτηση που περιλαμβάνεται στην MLP, είναι η __init__ , με ορίσματα:

- in_feat

  Που ισούται με τον αριθμό των χαρακτηριστικών εισόδου.

- mlp_ratio

  Ένας ακέραιος αριθμός που καθορίζει τον αριθμό των κρυφών μονάδων στο πρώτο γραμμικό στρώμα, με βάση τον αριθμό των χαρακτηριστικών εισόδου. Από προεπιλογή έχει οριστεί σε 1, που σημαίνει ότι ο αριθμός των κρυφών μονάδων είναι ίσος με τον αριθμό των χαρακτηριστικών εισόδου.

- out_feat

  Ο αριθμός των χαρακτηριστικών εξόδου. Από προεπιλογή έχει οριστεί σε 0, που σημαίνει ότι ο αριθμός των χαρακτηριστικών εξόδου είναι ίσος με τον αριθμό των χαρακτηριστικών εισόδου.

- dropout_p

  Ένας πραγματικός αριθμό που συμβολίζει την πιθανότητα του κάθε στοιχείου του τανυστή εισόδου να μηδενιστεί κατά την εκπαίδευση.

- act_layer

  Αντιπροσωπεύει τη συνάρτηση ενεργοποίησης. Από προεπιλογή έχει οριστεί σε nn.ReLU (Rectified Linear Unit).

Αφού γίνει ορισμός των παραμέτρων δημιουργούνται δύο γραμμικά επίπεδα, τα self.linear1 και self.linear2, με μεγέθη in_feat x mid_feat και mid_feat x out_feat, αντίστοιχα. Ο αριθμός των μονάδων στο πρώτο στρώμα καθορίζεται από το mlp_ratio, ενώ ο αριθμός των μονάδων στο δεύτερο στρώμα είτε καθορίζεται από το out_feat, είτε ορίζεται ίσο με το μέγεθος εισόδου από την προεπιλογή. Δημιουργούνται επίσης δύο στρώματα, τα self.drop1 και self.drop2, με την καθορισμένη πιθανότητα απόρριψης dropout_p και μια συνάρτηση ενεργοποίησης που καθορίζεται από το act_layer.

Η συνάρτηση forward συμβάλλει στην αποφυγή της υπερπροσαρμογής, μειώνοντας την εξάρτηση από οποιοδήποτε χαρακτηριστικό. Τα δεδομένα εισόδου x διέρχονται μέσω δύο γραμμικών στρωμάτων. Το πρώτο στρώμα, self.linear1(x), εφαρμόζει ένα γραμμικό μετασχηματισμό στα δεδομένα εισόδου x και περνάει μέσα από μια συνάρτηση ενεργοποίησης, self.act, η οποία εφαρμόζει ένα μη γραμμικό μετασχηματισμό στα δεδομένα. Στη συνέχεια, το αποτέλεσμα αυτής περνάει από το στρώμα απόρριψης self.drop1, το οποίο μηδενίζει τυχαία κάποια από τα στοιχεία του τανυστή εισόδου. Η έξοδος του πρώτου στρώματος περνάει στη συνέχεια στο δεύτερο γραμμικό στρώμα

self.linear2(x), και στη συνέχεια στο δεύτερο στρώμα απόρριψης, self.drop2. Τέλος, η έξοδος είναι το αποτέλεσμα από το δεύτερο στρώμα απόρριψης.

```python
class AttentionLayer(nn.Module):
    """Multi-head scaled self-attension layer"""

    def __init__(self, num_feat: int, num_heads: int = 8, qkv_bias: bool = False, dropout_p: float = 0.0):
        super().__init__()

        assert num_feat % num_heads == 0

        self.num_feat = num_feat
        self.num_heads = num_heads
        self.head_dim = num_feat // num_heads
        self.scale = self.head_dim ** -0.5

        self.qkv = nn.Linear(self.num_feat, self.num_feat * 3, bias=qkv_bias)
        self.attn_drop = nn.Dropout(dropout_p)

        self.proj = nn.Linear(self.num_feat, self.num_feat)
        self.proj_drop = nn.Dropout(dropout_p)

    def forward(self, x):
        B, L, C = x.shape
        assert C == self.num_feat

        qkv = self.qkv(x)  # [B, L, num_feat * 3]
        qkv = qkv.reshape(B, L, 3, self.num_heads, self.head_dim)  # [B, L, 3, num_heads, head_dim]
        qkv = qkv.permute(2, 0, 3, 1, 4)  # [3, B, num_heads, L, head_dim]
        q, k, v = qkv.unbind(0)  # [B, num_heads, L, head_dim] * 3

        attn = q @ k.transpose(-2, -1)  # [B, num_heads, L, L]
        attn = attn * self.scale
        attn = attn.softmax(dim=-1)
        attn = self.attn_drop(attn)

        x = (attn @ v).transpose(1, 2)  # [B, L, num_heads, head_dim]
        x = x.reshape(B, L, self.num_feat)  # [B, L, num_feat]

        x = self.proj(x)
        x = self.proj_drop(x)

        return x
```

Εικόνα 18, Η κλάση AttentionLayer στο transfile.py

Η κλάση AttentionLayer ορίζει ένα στρώμα προσοχής πολλαπλών κεφαλών. Η χρήση της δίνει την δυνατότητα στο μοντέλο να παρακολουθεί ταυτόχρονα διαφορετικές εισόδους.

Όπως παρατηρείται στην εικόνα 18, η συνάρτηση _init__ δέχεται τα: αριθμό των χαρακτηριστικών εισόδου (num_feat), αριθμός των κεφαλών προσοχής (num_heads), ένα boolean με τιμές True ή False, που υποδεικνύει την ύπαρξη ή μη της μεροληψίας (qkv_bias). Στη συνέχεια, δηλώνονται δύο επιπλέον γραμμικά στρώματα, το qkv και το proj.

Η συνάρτηση forward λαμβάνει έναν τανυστή x σαν είσοδο. Οι μεταβλητές B, L, C, (B είναι το μέγεθος της παρτίδας, L είναι το μήκος της ακολουθίας και C είναι ο αριθμός των χαρακτηριστικών), λαμβάνουν τις τιμές της διάστασης του τανυστή x. Έπειτα το x διέρχεται από το γραμμικό επίπεδο qkv, για να παραχθεί ένας τανυστής με σχήμα (B, L, num_feat * 3). Στη συνέχεια, αναδιαμορφώνεται το qkv σε (B, L, 3, num_heads, head_dim) και στο τέλος λαμβάνει σαν τιμή (3, B, num_heads, L, head_dim). Ορίζονται τα βάρη προσοχής, τα οποία χρησιμοποιούνται στην αναδιαμόρφωση του τανυστή, ο οποίος έχει σαν τιμή το (B, L, num_feat). Στις τελευταίες γραμμές της κλάσης, το x περνά από τα proj, proj_drop πριν επιστραφεί από τη συνάρτηση.

```python
class TransformerBlock(nn.Module):
    def __init__(
        self,
        in_feat: int,
        out_feat: int = 0,
        num_heads: int = 8,
        qkv_bias: bool = False,
        mlp_ratio: int = 4,
        dropout_p: float = 0.0,
        droppath_p: float = 0.0,
        act_layer: nn.Module = nn.GELU,
        norm_layer: nn.Module = nn.LayerNorm,
    ):
        super().__init__()

        out_feat = out_feat or in_feat

        self.droppath = DropPath(droppath_p)

        self.norm1 = norm_layer(in_feat)
        self.norm2 = norm_layer(in_feat)

        self.attn = AttentionLayer(num_feat=in_feat, num_heads=num_heads, qkv_bias=qkv_bias, dropout_p=dropout_p)

        self.mlp = MLP(
            in_feat=in_feat, mlp_ratio=mlp_ratio, out_feat=out_feat, dropout_p=dropout_p, act_layer=act_layer
        )

    def forward(self, x):
        x = x + self.droppath(self.attn(self.norm1(x)))
        x = x + self.droppath(self.mlp(self.norm2(x)))
        # x = self.droppath(self.mlp(self.norm2(x)))
        return x
```

Εικόνα 19, Η κλάση TransformerBlock στο transfile.py

Η κλάση Transformer είναι ένας τύπος αρχιτεκτονικής νευρωνικών δικτύων. Αυτή περιέχει τη συνάρτηση __init__ , με ορισμένες από τις παραμέτρους της να αποτελούν τα:

- out_feat

  Ορίζει τον αριθμό των χαρακτηριστικών εξόδου. Οι τιμές που παίρνει είναι out_feat και in_feat. Εάν δεν δίνεται κάποια τιμή στο out_feat, η προκαθορισμένη τιμή είναι το 0.
- droppath_p

  Η πιθανότητα απόρριψης ολόκληρων μονοπατιών κατά τη διάρκεια της εκπαίδευσης. Χρησιμεύει ως τεχνική κανονικοποίησης.
- norm_layer

  Η συνάρτηση κανονικοποίησης που θα χρησιμοποιηθεί στο TransformerBlock.

Ακόμη η κλάση TransformerBlock χρησιμοποιεί την κλάση droppath, δύο στρώματα κανονικοποίησης norm1 και norm2, τη κλάση AttentionLayer και την MLP.

Η συνάρτηση forward της κλάσης TransformerBlock λαμβάνει έναν τανυστή εισόδου x και εφαρμόζει τις ακόλουθες ενέργειες:

1. Περνάει τον τανυστή εισόδου μέσω του στρώματος κανονικοποίησης norm1, του στρώματος προσοχής attn και της μεταβλητής droppath.
2. Προσθέτει την έξοδο της μεταβλητής droppath στον αρχικό τανυστή εισόδου.
3. Περνά το προκύπτοντα τανυστή μέσω του στρώματος κανονικοποίησης norm2, του mlp και της μεταβλητής droppath και,
4. Προσθέτει την έξοδο της μεταβλητής droppath στον αρχικό τανυστή εισόδου.

7.4 Αρχείο model

Το βασικότερο μέρος της ανάλυσης εστιάζει στο αρχείο model.py. Ακολουθεί η αναλυτική παρουσίαση του κώδικα.

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import logging

import numpy as np
from pandas import merge_asof

import torch
import torch.nn as nn
import torch.nn.functional as F
import random

from torch.utils.data import DataLoader
from ogb_wikikg2.dataloader import TestDataset
from ogb_wikikg2.transfile import TransformerBlock
from collections import defaultdict
from typing import Optional
from tqdm import tqdm
import math
from torch.nn import TransformerEncoderLayer, TransformerEncoder, GRU
import pickle

from ogb.linkproppred import Evaluator
```

Εικόνα 20, Οι βιβλιοθήκες που εισάγονται στο model.py

Ο κώδικας αρχικά εισάγει ορισμένες βοηθητικές συναρτήσεις και βιβλιοθήκες της PyTorch. Κάποιες από τις πιο δημοφιλής βιβλιοθήκη ML είναι το nn (μια συνάρτηση για τον ορισμό νευρωνικών δικτύων), το nn.functional (ορίζει τον αριθμό συναρτήσεων νευρωνικών δικτύων) και το DataLoader (ένα βοηθητικό πρόγραμμα για τη παράλληλη φόρτωση δεδομένων). Επίσης φορτώνονται οι κλάσεις TestDataset και TransformerBlock από τα αρχεία dataloader και transfile αντίστοιχα.

```
class KGEModel(nn.Module):
    def __init__(self, model_name, nentity, nrelation, hidden_dim, gamma, evaluator,
                 drop, triples):
        super(KGEModel, self).__init__()
        self.model_name = model_name
        self.nentity = nentity
        self.nrelation = nrelation
        self.hidden_dim = hidden_dim
        self.epsilon = 2.0
        self.u = 0.1
        self.evaluator = evaluator

        self.drop = drop
        self.triples = triples

        if model_name not in ['TransE', 'DistMult', 'ComplEx', 'RotatE', 'AutoSF', 'PairRE', 'TripleRE']:
            raise ValueError('model %s not supported' % model_name)
```

Εικόνα 21, Η κλάση KGEModel στο model.py

Η κλάση KGEModel αφορά ένα μοντέλο Knowledge Graph Embedding (KGE) και αποτελεί μια μεθοδολογία μάθησης στην οποία οι κόμβοι παρουσιάζονται ως διανύσματα, τα οποία χρησιμοποιούνται για διάφορες εφαρμογές ML.

Για την κατασκευή της κλάσης KGEModel και πιο συγκεκριμένα της συνάρτησης __init__, ορίζονται ως παράμετροι οι model_name, nentity, nrelation, hidden_dim, epsilon, evaluator και u. Όλες οι μεταβλητές και παράμετροι χρησιμοποιούνται για τη διαμόρφωση του KGEModel. Ακολουθεί μια περιγραφική ανάλυση των παραμέτρων.

- model_name

  Καθορίζει τη μέθοδο πρόβλεψης που θα επιλεγεί. Οι επιλογές διακρίνονται στις εξής: TransE, DistMult, ComplEx, RotatE, AutoSF, PairRE και TripleRE.
- Nentity

  Ο αριθμός των οντοτήτων στο γράφο γνώσης.
- Nrelation

  Ο αριθμός των σχέσεων στο γράφο γνώσης.
- hidden_dim

  Η διάσταση των οντοτήτων και των σχέσεων μεταξύ των κόμβων.
- epsilon

  Μια τιμή που προσδιορίζει την ακρίβεια των υπολογισμών. Η χρήση της διασφαλίζει ότι οι βαθμολογίες για αληθοφανείς και μη αληθοφανείς τριπλέτες είναι καλά διαχωρισμένες.

- evaluator

  Τιμή που προσδιορίζει την απόδοση της μεθόδου πρόβλεψης στο σύνολο δοκιμών.

- triples

  Ένα σύνολο τριπλετών (υποκείμενο, σχέση, αντικείμενο).

- u

  Η ταχύτητα των αλλαγών στους υπολογισμούς.

```
if model_name == 'RotatE':
    self.entity_dim = hidden_dim * 2
    self.relation_dim = hidden_dim
elif model_name == 'ComplEx':
    self.entity_dim = hidden_dim * 2
    self.relation_dim = hidden_dim * 2
elif model_name == 'PairRE':
    self.entity_dim = hidden_dim
    self.relation_dim = hidden_dim * 2
elif model_name == 'TripleRE':
    self.entity_dim = hidden_dim
    self.relation_dim = hidden_dim * 3
```

Εικόνα 22, Ορισμός διαστάσεων ανάλογα με την μέθοδο στο model.py

Ανάλογα με το μοντέλο που θα επιλεγεί, καθορίζεται ο αριθμός των διαστάσεων των οντοτήτων και ο αριθμό των σχέσεων, δηλαδή ουσιαστικά οι κόμβους και οι ακμές.

```
anchor_file = 'data/ogbl-wikikg2_20000_anchors_20000_paths_d0.4_b0.0_p0.4_r0.2_pykeen_50sp_bfs.pkl'
neighbor_file = 'data/neighbors+relations_with_max_degrees.pkl'

anchor_dim = self.entity_dim
self.sample_anchors = 20

node_dim = 32
self.node_itself = True

self.sample_neighbors = 5

self.attn_layers_num = 1
self.attn_dim = self.entity_dim
self.attn_heads = 8

merge_strategy = 'mean_pooling'
self.mean_pooling = False
self.linear_proj = False
if merge_strategy == 'mean_pooling':
    self.mean_pooling = True
elif merge_strategy == 'linear_proj':
    self.linear_proj = True
else:
    raise TypeError
```

Εικόνα 23, Η θέση των αρχείων και ο ορισμός μεταβλητών στο model.py

Οι μεταβλητές anchor_file και neighbor_file καθορίζουν τις θέσεις των αρχείων anchor και neighbor αντίστοιχα. Η μορφή των αρχείων που παρουσιάζονται στην εικόνα 23 είναι η pkl. Η χρήση των αρχείων αποσκοπεί στην εκπαίδευση και στην αξιολόγηση του μοντέλου KGE.

Ο συγγραφέας ορίζει ορισμένες παραμέτρους, όπως:

- τις διαστάσεις των anchor (συνδεμένοι κόμβοι)
- τον αριθμό των anchor που θα δειγματοληπτηθούν για κάθε οντότητα κατά την εκπαίδευση του μοντέλου και καθορίζει αν θα συμπεριληφθεί ο ίδιος ο κόμβος στη λίστα των γειτονικών κόμβων

Οι παράμετροι attn_layers_num, attn_dim και attn_heads καθορίζουν τον αριθμό των επιπέδων, τη διάσταση των μηχανισμών προσοχής και τον αριθμό των κεφαλών του νευρωνικού που θα χρησιμοποιηθούν στο μοντέλο KGE αντίστοιχα. Αυτές οι παράμετροι χρησιμοποιούνται για τη διαμόρφωση του μηχανισμού προσοχής του μοντέλου.

Οι παράμετροι mean_pooling και linear_proj είναι μεταβλητές boolean, που καθορίζουν εάν θα χρησιμοποιηθεί η συγκέντρωση μέσων όρων ή η γραμμική προβολή ως στρατηγική συγχώνευσης, αντίστοιχα.

```
mlp_ratio = 4

# 3 types: node itself, neighbor nodes, anchors
self.type_embedding = nn.Embedding(num_embeddings=3, embedding_dim=self.attn_dim)
nn.init.uniform_(
    tensor=self.type_embedding.weight,  # .weight for Embedding
    a=-self.embedding_range.item(),
    b=self.embedding_range.item()
)
# back to normal relation embs, +1 for the padding relation
self.relation_embedding = nn.Embedding(num_embeddings=nrelation, embedding_dim=self.relation_dim)
nn.init.uniform_(
    tensor=self.relation_embedding.weight,
    a=-self.embedding_range.item(),
    b=self.embedding_range.item()
)

if self.linear_proj:
    self.set_enc = nn.Sequential(
        nn.Linear(self.attn_dim * (self.sample_anchors + self.sample_neighbors + self.node_itself), self.entity_dim * mlp_ratio),
        nn.Dropout(self.drop), nn.ReLU(),
        nn.Linear(self.entity_dim * mlp_ratio, self.entity_dim)
    )
```

Εικόνα 24, Μέρος του κώδικα της κλάσης KGEModel

Η συνάρτηση nn.init.uniform_() κατανέμει ομοιόμορφα τις τιμές της παραμέτρου weight, με εύρος τιμών μεταξύ των a και b.

Αν αληθεύει η συνθήκη self.linear_proj που ελέγχεται στην if, ορίζεται η παράμετρος set_enc, με την χρήση της nn.Sequential, η αποτελείται από τρία στρώματα. Το πρώτο στρώμα έχει μονάδες εισόδου self.attn_dim * (self.sample_anchors + self.sample_neighbors + self.node_itself) και μονάδες εξόδου self.entity_dim * mlp_ratio. Το δεύτερο στρώμα έχει μονάδες εισόδου self.entity_dim * mlp_ratio και μονάδες εξόδου self.entity_dim. Το τρίτο στρώμα έχει μονάδες εισόδου self.entity_dim και μονάδες εξόδου self.entity_dim. Το δεύτερο στρώμα ακολουθείται από ένα στρώμα απόρριψης με ποσοστό απόρριψης self.drop και τη συνάρτηση ενεργοποίησης ReLU. Σκοπός αυτών των στρωμάτων είναι να μειωθεί η διάσταση των embedding και να επιτραπεί στο μοντέλο να εκπαιδευτεί σε πιο ισχυρές και γενικεύσιμες αναπαραστάσεις.

```
type_ids = []

self.node_before_attn = None
if self.node_itself or self.sample_neighbors > 0:
    print("Creating nodes infomation")
    if self.node_itself:
        type_ids.append(0)
        nodes = [[i] for i in range(nentity)]
    else:
        nodes = [[] for i in range(nentity)]
    if self.sample_neighbors > 0:
        type_ids.extend([1] * self.sample_neighbors)
        nb_vocab = pickle.load(open(neighbor_file, 'rb'))
        # convert dict to list
        for i in range(nentity):
            nb_info = nb_vocab.get(i, {'nbs':[], 'rels':[]})
            nodes[i].extend([n for n in nb_info['nbs'][:self.sample_neighbors]])
            if len(nodes[i]) < self.sample_neighbors + self.node_itself:
                nodes[i].extend([nentity for _ in range(self.sample_neighbors + self.node_itself - len(nodes[i]))])
    self.register_buffer('nodes', torch.tensor(nodes, dtype=torch.long))
    del nodes

    self.node_embeddings = nn.Embedding(num_embeddings=nentity+1, embedding_dim=self.attn_dim if node_dim == 0 else node_di
    nn.init.uniform_(
        tensor=self.node_embeddings.weight,  # .weight for Embedding
        a=-self.embedding_range.item(),
        b=self.embedding_range.item()
    )
    self.node_before_attn = None if node_dim == 0 else nn.Linear(node_dim, self.attn_dim)
```

Εικόνα 25, Συνθήκες της κλάσης KGEModel

Στις επόμενες γραμμές της εικόνας 25 , αν η συνθήκη if του elf.node_itself είναι αληθής δημιουργείται μια λίστα κόμβων, όπου κάθε στοιχείο είναι μια λίστα που περιέχει μόνο το δείκτη της αντίστοιχης οντότητας. Σε αντίθετη περίπτωση αν ισχύει η συνθήκη self.sample_neighbors > 0, ανακτά τους γειτονικούς κόμβους και τις αντίστοιχες σχέσεις κάθε οντότητας από το αρχείο neighbor_file. Εν συνεχεία χρησιμοποιώντας το λεξικό nb_vocab προσθέτει τους προαναφερόμενους κόμβους και σχέσεις στο αντίστοιχο στοιχείο της λίστας nodes. Ολοκληρώνοντας, καταχωρεί τους κόμβους και διαγράφει τη λίστα nodes.

```
if self.sample_anchors > 0:
    type_ids.extend([2] * self.sample_anchors)
    print("Creating hashes")
    anchors, _, vocab = pickle.load(open(anchor_file, "rb"))
    anchors.append(-99)
    nanchor = len(anchors)
    token2id = {t:i for i,t in enumerate(anchors)}
    hashes = [
        [token2id[i] for i in vocab[e]['ancs'][:self.sample_anchors]] + [nanchor]*(self.sample_anchors - len(vocab[e]['ancs']))
        for e in range(nentity)
    ]
    self.register_buffer('hashes', torch.tensor(hashes, dtype=torch.long))
    del hashes

    self.anchor_embeddings = nn.Embedding(num_embeddings=nanchor+1, embedding_dim=anchor_dim)
    nn.init.uniform_(
        tensor=self.anchor_embeddings.weight,  # .weight for Embedding
        a=-self.embedding_range.item(),
        b=self.embedding_range.item()
    )

self.register_buffer('type_ids', torch.tensor(type_ids, dtype=torch.long))

self.attn_layers = nn.ModuleList([
    TransformerBlock(in_feat=self.attn_dim, mlp_ratio=mlp_ratio, num_heads=self.attn_heads, dropout_p=self.drop)
    for _ in range(self.attn_layers_num)
])

print('node embedding: {}\tsample anchors: {}\tsample neighbors: {}'.format(self.node_itself, self.sample_anchors, self.sample_neighbors))
print('anchor dim: {}\nentity dim: {}\nrelation dim: {}\nnode dim: {}\nmerge strategy:{}'.format(
    anchor_dim, self.entity_dim, self.relation_dim, node_dim, merge_strategy))
print('attention dim: {}, attention heads: {}\n'.format(self.attn_dim, self.attn_heads))
print('node layers:\n{}\nattention layers:\n{}'.format(
    self.node_before_attn,
    self.attn_layers))

self.evaluator = evaluator
```

Εικόνα 26, Δημιουργία μεταβλητών στην κλάση KGEModel

Η εικόνα 26 ξεκινά με την συνθήκη if self.sample_anchors > 0, η οποία επεκτείνει τη λίστα type_ids με μια νέα λίστα με μήκος self.sample_anchors. Οι μεταβλητές anchors, _ και vocab φορτώνονται από ένα αρχείο με τη χρήση του pickle.load.

Το λεξικό token2id αντιστοιχίζει τα tokens των anchor σε ακέραιους δείκτες. Ενώ το [token2id[i] for i in vocab[e]['ancs'][:self.sample_anchors]] παράγει μια λίστα ακεραίων δεικτών για τα πρώτα σημεία self.sample_anchors της οντότητας e. Η τελική λίστα που ανατίθεται στη μεταβλητή hashes, επαναλαμβάνει όλες τις οντότητες e στο εύρος [0, nentity) και περιέχει anchor για κάθε οντότητα.

Η δημιουργία της λίστας attn_layers επιτυγχάνεται μέσω της μεθόδου nn.ModuleList και της κλάσης TransformerBlock, η λειτουργία της οποίας προαναφέρθηκε στο κεφάλαιο 7.3.

```
def encode_by_index(self, entities: torch.LongTensor) -> torch.FloatTensor:
    assert len(entities.shape) == 1
    embs_seq = None # [entities.shape, sequence_length, feature_dimension]
    if self.sample_anchors > 0:
        hashes = self.hashes[entities]
        anc_embs = self.anchor_embeddings(hashes)
        embs_seq = anc_embs #if embs_seq is None else torch.cat([embs_seq, anc_embs], dim=-2)

    if self.node_itself or self.sample_neighbors > 0:
        nodes = self.nodes[entities]
        node_embs = self.node_embeddings(nodes)
        if self.node_before_attn is not None:
            node_embs = self.node_before_attn(node_embs)
        embs_seq = node_embs if embs_seq is None else torch.cat([node_embs, embs_seq], dim=-2)

    if self.attn_layers_num > 0:
        embs_seq = embs_seq + self.type_embedding(self.type_ids)
        for enc_layer in self.attn_layers:
            embs_seq = enc_layer(embs_seq)

    if self.mean_pooling:
        embs_seq = embs_seq.mean(dim=-2)
    elif self.linear_proj:
        embs_seq = self.set_enc(embs_seq.view(*entities.shape, -1))
    else:
        raise NotImplementedError

    return embs_seq
```

Εικόνα 27, Η συνάρτηση encode_by_index στη κλάση KGEModel

Στο παραπάνω κώδικα (εικόνα 27), η συνάρτηση ξεκινά με τον έλεγχο της συνθήκης self.sample_anchors >0. Εφόσον η συνθήκη είναι αληθής, ανακτά τα anchor_embeddings για καθορισμένες οντότητες χρησιμοποιώντας τα hashes και τα εκχωρεί στην μεταβλητή embs_seq. Στην επόμενη συνθήκη αν self.node_itself ή το self.sample_neighbors >0, ανακτώνται τα node_embs για τις καθορισμένες οντότητες κόμβων. Εάν το self.node_before_attn δεν έχει τη τιμή None, το χαρακτηριστικό node_before_attn εφαρμόζεται στις node_embs.

Στη συνέχεια αν το self.attn_layers_num>0,τότε τα self.type_embedding(self.type_ids) προστίθενται με την ακολουθία embs_seq. Κάθε επίπεδο προσοχής (enc_layer) από τη λίστα self.attn_layers εφαρμόζεται στο embs_seq. Η έξοδος κάθε στρώματος προσοχής χρησιμοποιείται στη συνέχεια ως είσοδος για το επόμενο στρώμα, έως ότου εφαρμοστούν όλα τα στρώματα προσοχής.

Στον έλεγχος που αφορά εάν το self.mean_pooling είναι True, επιστρέφεται ο μέσος όρος της ακολουθίας των entity embeddings κατά μήκος της δεύτερης διάστασης. Εάν η self.linear_proj είναι True ορίζεται η μεταβλητή embs_seq, ειδάλλως εμφανίζεται σφάλμα.

```
def forward(self, sample, mode='single'):
    '''
    '''

    if mode == 'single':
        batch_size, negative_sample_size = sample.size(0), 1
        head = self.encode_by_index(sample[:, 0]).unsqueeze(1)
        relation = self.relation_embedding(sample[:, 1]).unsqueeze(1)
        tail = self.encode_by_index(sample[:, 2]).unsqueeze(1)

    elif mode == 'head-batch':
        tail_part, head_part = sample
        batch_size, negative_sample_size = head_part.size(0), head_part.size(1)
        head = self.encode_by_index(head_part.view(-1)).view(batch_size, negative_sample_size, -1)
        relation = self.relation_embedding(tail_part[:, 1]).unsqueeze(1)
        tail = self.encode_by_index(tail_part[:, 2]).unsqueeze(1)

    elif mode == 'tail-batch':
        head_part, tail_part = sample
        batch_size, negative_sample_size = tail_part.size(0), tail_part.size(1)
        head = self.encode_by_index(head_part[:, 0]).unsqueeze(1)
        relation = self.relation_embedding(head_part[:, 1]).unsqueeze(1)
        tail = self.encode_by_index(tail_part.view(-1)).view(batch_size, negative_sample_size, -1)

    else:
        raise ValueError('mode %s not supported' % mode)

    model_func = {
        'TransE': self.TransE,
        'DistMult': self.DistMult,
        'ComplEx': self.ComplEx,
        'RotatE': self.RotatE,
        'AutoSF': self.AutoSF,
        'PairRE': self.PairRE,
        'TripleRE': self.TripleRE,
    }
```

Εικόνα 28, Ο ορισμός του mode, και οι υπάρχουσες μέθοδοι πρόβλεψης

Η συνάρτηση forward ορίζει τις τριπλέτες. Όταν το mode παίρνει την τιμή 'single', το δείγμα είναι ένα σύνολο τριπλετών. Στη λειτουργία 'head-batch' και 'tail-batch', το δείγμα αποτελείται από δύο μέρη, το negative sample και positive sample.

Στη συνέχεια καθορίζεται το λεξικό model_func με τα διαθέσιμα μοντέλα: 'TransE', 'DistMult', 'ComplEx', 'RotatE', 'AutoSF', 'PairRE' και 'TripleRE'.

```
def PairRE(self, head, relation, tail, mode):
    re_head, re_tail = torch.chunk(relation, 2, dim=2)

    head = F.normalize(head, 2, -1)
    tail = F.normalize(tail, 2, -1)

    score = head * re_head - tail * re_tail + head - tail
    score = self.gamma.item() - torch.norm(score, p=1, dim=2)
    return score
```

Εικόνα 29, Η μέθοδος πρόβλεψης PairRE

Στη παραπάνω εικόνα παρουσιάζεται η μέθοδος πρόβλεψης PairRE. Αρχικά ορίζονται οι κόμβοι, οι ακμές και η λειτουργία (head, relation, tail και mode). Αναλυτικά στα πρώτα βήματα, το relation χωρίζεται σε δύο συνιστώσες κατά μήκος της τελευταίας διάστασης, οι οποίες αποθηκεύονται στις μεταβλητές re_head και re_tail. Έπειτα κανονικοποιούνται οι head και tail χρησιμοποιώντας τη μέθοδο normalize. Μια αρχική βαθμολογία υπολογίζεται με τη χρήση προσθαφαιρέσεων και πολλαπλασιασμού ανάμεσά στους τανυστές head, re_head, tail και re_tail. Η τελική βαθμολογία λαμβάνει τη νόρμα του τανυστή της αρχικής βαθμολογίας και αφαιρεί το αποτέλεσμα από τη τιμή που είναι αποθηκευμένη στο τανυστή gamma.

Μια ακόμη μέθοδος προς σχολιασμό είναι η TripleRE (εικόνα 30). Η λειτουργία της TripleRE, προσομοιάζει αυτήν της παραπάνω μεθόδου. Η ειδοποιός διαφορά είναι ότι το relation χωρίζεται σε τρείς συνιστώσες κατά μήκος της τελευταίας διάστασης, οι οποίες αποθηκεύονται στις μεταβλητές re_head, re_mid και re_tail. Η τελική βαθμολογία περιέχει το score κάθε τριπλέτας, το οποίο υποδηλώνει τη πιθανότητα η τριπλέτα να είναι αληθής στο γράφο γνώσης.

```python
def TripleRE(self, head, relation, tail, mode):
    re_head, re_mid, re_tail = torch.chunk(relation, 3, dim=2)

    e_h = torch.ones_like(head)
    e_t = torch.ones_like(tail)

    head = F.normalize(head, 2, -1)
    tail = F.normalize(tail, 2, -1)

    score = head * (self.u * re_head + e_h) - tail * (self.u * re_tail + e_t) + re_mid
    score = self.gamma.item() - torch.norm(score, p=1, dim=2)
    return score
```

Εικόνα 30, Η μέθοδος πρόβλεψης TripleRE

```python
@staticmethod
def train_step(model, optimizer, train_iterator, args):
    '''
    A single train step. Apply back-propation and return the loss
    '''

    model.train()
    optimizer.zero_grad()
    positive_sample, negative_sample, subsampling_weight, mode = next(train_iterator)

    if args.cuda:
        positive_sample = positive_sample.cuda()
        negative_sample = negative_sample.cuda()
        subsampling_weight = subsampling_weight.cuda()

    negative_score = model((positive_sample, negative_sample), mode=mode)
    if args.negative_adversarial_sampling:
        # In self-adversarial sampling, we do not apply back-propagation on the sampling weight
        negative_score = (F.softmax(negative_score * args.adversarial_temperature, dim=1).detach()
                          * F.logsigmoid(-negative_score)).sum(dim=1)
    else:
        negative_score = F.logsigmoid(-negative_score).mean(dim=1)

    positive_score = model(positive_sample)
    positive_score = F.logsigmoid(positive_score).squeeze(dim=1)
    if args.uni_weight:
        positive_sample_loss = - positive_score.mean()
        negative_sample_loss = - negative_score.mean()
    else:
        positive_sample_loss = - (subsampling_weight * positive_score).sum() / subsampling_weight.sum()
        negative_sample_loss = - (subsampling_weight * negative_score).sum() / subsampling_weight.sum()

    loss = (positive_sample_loss + negative_sample_loss) / 2
```

Εικόνα 31, Ορισμός της συνάρτησης απώλειας

Η συνάρτηση train λειτουργεί ως ένα βήμα εκπαίδευσης για το μοντέλο κατά τη διαδικασία εκμάθησης. Σαν είσοδος η συνάρτηση περιλαμβάνει ένα μοντέλο, ένα βελτιστοποιητή, έναν επαναλήπτη train και κάποια ορίσματα. Η συνάρτηση θέτει πρώτα το μοντέλο σε κατάσταση εκπαίδευσης και μηδενίζει τις κλίσεις στο βελτιστοποιητή. Στη συνέχεια λαμβάνει την επόμενη δέσμη δεδομένων, που θα χρησιμοποιηθεί για

εκπαίδευση από τον επαναλήπτη train. Εάν το όρισμα cuda είναι True, γίνεται ενεργοποίηση της GPU και μεταφέρονται τα δεδομένα σε αυτή. Στη συνέχεια υπολογίζονται τα αρνητικά score για τα δεδομένα, χρησιμοποιώντας το προκαθορισμένο μοντέλο. Η αρνητική βαθμολογία επεξεργάζεται περαιτέρω με χρήση της σιγμοειδούς συνάρτησης και του λογαρίθμου, εάν η συνθήκη negative_adversarial_sampling είναι True. Όμοια υπολογίζονται και τα θετικά score. Γίνεται ο υπολογισμός των θετικών και των αρνητικών απωλειών με βάση τη θετική και την αρνητική βαθμολογία, λαμβάνοντας υπόψη τη συνθήκη uni_weight. Η τελική απώλεια είναι στη συνέχεια ο μέσος όρος των θετικών και αρνητικών απωλειών.

```
if args.regularization != 0.0:
    # Use L3 regularization for ComplEx and DistMult
    regularization = args.regularization * (
            model.entity_embedding.norm(p=3) ** 3 +
            model.relation_embedding.norm(p=3).norm(p=3) ** 3
    )
    loss = loss + regularization
    regularization_log = {'regularization': regularization.item()}
else:
    regularization_log = {}

loss.backward()

optimizer.step()

log = {
    **regularization_log,
    'positive_sample_loss': positive_sample_loss.item(),
    'negative_sample_loss': negative_sample_loss.item(),
    'loss': loss.item()
}

return log
```

Εικόνα 32, Κανονικοποίηση των ενσωματώσεων και η συνάρτηση απώλειας

Εάν το args.regularization δεν είναι ίσο με 0, η μεταβλητή regularization υπολογίζεται ως το άθροισμα των κύβων των κανονικοποιημένων embeddings, πολλαπλασιασμένο με τη παράμετρο κανονικοποίησης args.regularization. Στη συνέχεια το regularization προστίθεται στην απώλεια και δημιουργείται ένα λεξικό regularization_log, που περιέχει

τη τιμή της κανονικοποίησης. Εάν το args.regularization είναι ίσο με 0, δημιουργείται ένα κενό λεξικό.

Τέλος, δημιουργείται ένα αρχείο καταγραφής λεξικού που περιέχει, το regularization_log (που δημιουργήθηκε παραπάνω), τις διάφορες συνιστώσες της απώλειας (όπως οι θετικές και αρνητικές απώλειες δείγματος) και την απώλεια.

```
@staticmethod
def test_step(model, test_triples, args, random_sampling=False):
    '''
    Evaluate the model on test or valid datasets
    '''

    model.eval()

    # Prepare dataloader for evaluation
    test_dataloader_head = DataLoader(
        TestDataset(
            test_triples,
            args,
            'head-batch',
            random_sampling
        ),
        batch_size=args.test_batch_size,
        num_workers=max(1, args.cpu_num // 2),
        collate_fn=TestDataset.collate_fn
    )

    test_dataloader_tail = DataLoader(
        TestDataset(
            test_triples,
            args,
            'tail-batch',
            random_sampling
        ),
        batch_size=args.test_batch_size,
        num_workers=max(1, args.cpu_num // 2),
        collate_fn=TestDataset.collate_fn
    )

    test_dataset_list = [test_dataloader_head, test_dataloader_tail]

    test_logs = defaultdict(list)

    step = 0
    total_steps = sum([len(dataset) for dataset in test_dataset_list])
```

Εικόνα 33, Οι dataloader που θα χρησιμοποιηθούν στην πρόβλεψη

```
with torch.no_grad():
    for test_dataset in test_dataset_list:
        for positive_sample, negative_sample, mode in test_dataset:
            if args.cuda:
                positive_sample = positive_sample.cuda()
                negative_sample = negative_sample.cuda()

            score = model((positive_sample, negative_sample), mode)

            batch_results = model.evaluator.eval({'y_pred_pos': score[:, 0],
                                                  'y_pred_neg': score[:, 1:]})
            for metric in batch_results:
                test_logs[metric].append(batch_results[metric])

            if step % args.test_log_steps == 0:
                logging.info('Evaluating the model... (%d/%d)' % (step, total_steps))

            step += 1

    metrics = {}
    for metric in test_logs:
        metrics[metric] = torch.cat(test_logs[metric]).mean().item()

return metrics
```

Εικόνα 34, Δημιουργία της μετρικής πρόβλεψης

Η συνάρτηση ξεκινά θέτοντας το μοντέλο σε κατάσταση αξιολόγησης με την model.eval(). Στη συνέχεια εισάγει τα δεδομένα στις test_dataloader_head και test_dataloader_tail χρησιμοποιώντας την κλάση TestDataset. Ο test_dataloader_head χρησιμοποιείται για την αξιολόγηση της ικανότητας του μοντέλου να προβλέπει την οντότητα head μιας δεδομένης σχέσης, ενώ ο test_dataloader_tail προβλέπει την οντότητα tail.

Στη λειτουργία αξιολόγησης το μοντέλο υπολογίζει τις βαθμολογίες των θετικών και των αρνητικών δειγμάτων για κάθε τριπλέτα στα σύνολα δεδομένων δοκιμής. Ο evaluator του μοντέλου χρησιμοποιείται για την αξιολόγηση της απόδοσης, χρησιμοποιώντας μετρικές όπως η MRR. Ολοκληρώνοντας, υπολογίζονται οι μέσες τιμές κάθε μετρικής αξιολόγησης για τις προβλέψεις του μοντέλου στο σύνολο δεδομένων δοκιμής, και τις αποθηκεύει στο λεξικό metrics.

ΚΕΦΑΛΑΙΟ 8: ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο παρόν κεφάλαιο περιγράφουμε το τρόπο επιλογής της μεθόδου πρόβλεψης που αναλύθηκε. Επιπρόσθετα απαντώνται τα ερευνητικά ερωτήματα, που τέθηκαν.

8 Συμπεράσματα

8.1 Επιλογή της μεθόδου και αποτελέσματα

Για την επίτευξη της πρόβλεψης ακμών στα δεδομένα του ogb_wikikg2, ερευνήθηκαν ως προς τη μετρική MRR, πλήθος μοντέλων από τους χρήστες. Το MRR είναι μια ευρέως χρησιμοποιούμενη μετρική αξιολόγησης των GNN. Μετράει το μέσο όρο των αμοιβαίων βαθμών της αληθούς απάντησης σε μια ταξινομημένη λίστα προβλέψεων. Με άλλα λόγια, υπολογίζει τον μέσο όρο της αντίστροφης κατάταξης της πρώτης σωστής απάντησης σε μια λίστα προβλέψεων (1/rank) [37].

Στην εικόνα 35, είναι ταξινομημένες με φθίνουσα σειρά οι μέθοδοι .

StarGraph + TripleRE	No	0.7286 ± 0.0009	0.7335 ± 0.0011	Linhui Feng (360AI)	Paper, Code	93,039,522	Tesla A100 (40GB)	Jan 3, 2023
InterHT+ (256dim)	No	0.7257 ± 0.0018	0.7370 ± 0.0022	Baoxin Wang (HFL)	Paper, Code	148,000,738	Tesla A100 (40GB)	Dec 26, 2022
StarGraph + TripleRE	No	0.7201 ± 0.0011	0.7288 ± 0.0008	Hongzhu Li (360AI)	Paper, Code	86,762,146	Tesla A100(40GB)	May 30, 2022
TranS	No	0.6939 ± 0.0011	0.7058 ± 0.0018	Xuanyu Zhang (DXM AI)	Paper, Code	38,430,804	Tesla V100 (16GB)	Apr 19, 2022
TranS	No	0.6882 ± 0.0019	0.6988 ± 0.0006	Xuanyu Zhang (DXM AI)	Paper, Code	19,215,402	Tesla V100 (16GB)	Apr 28, 2022
TripleRE + NodePiece	No	0.6866 ± 0.0014	0.6955 ± 0.0008	Long Yu (360AI)	Paper, Code	36,421,802	Tesla A100(40GB)	Feb 24, 2022
InterHT	No	0.6779 ± 0.0018	0.6893 ± 0.0015	Baoxin Wang (HFL)	Paper, Code	19,215,402	Tesla V100 (32GB)	Feb 10, 2022
TripleRE + NodePiece	No	0.6582 ± 0.0020	0.6616 ± 0.0018	Long Yu (360AI)	Paper, Code	7,289,002	Tesla A100(40GB)	Dec 25, 2021
ComplEx-RP (50dim)	No	0.6392 ± 0.0045	0.6561 ± 0.0070	Yihong Chen (UCL NLP & FAIR London)	Paper, Code	250,167,400	Tesla V100 (32GB)	Nov 23, 2021
tripleRE	No	0.5794 ± 0.0020	0.6045 ± 0.0024	Long Yu (360AI)	Paper, Code	500,763,337	Tesla P40(24GB)	Dec 17, 2021
NodePiece + AutoSF	No	0.5703 ± 0.0035	0.5806 ± 0.0047	Mikhail Galkin (Mila)	Paper, Code	6,860,602	Tesla V100 (32 GB)	Jul 17, 2021

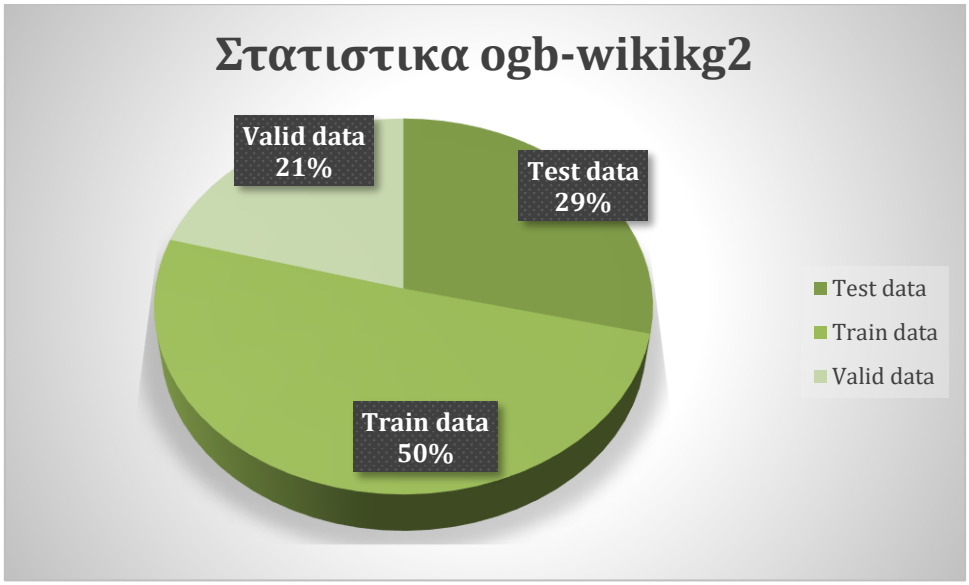
Εικόνα 35, Η κατάταξη των μεθόδων με βάση την μετρική πρόβλεψης MRR

Η μέθοδος που επιλέχθηκε για ανάλυση στη συγκεκριμένη διπλωματική, είναι ο συνδυασμός της Stargraph και της TripleRE [37]. Η επιλογή της μεθόδου, έγινε με βάση την υψηλότερη τιμή της μετρικής MRR. Τα αποτελέσματα που εξάγει η κάθε πρόβλεψη

διακρίνονται στα Test MRR, Valid MRR. Παρακάτω παρουσιάζονται τα αποτελέσματα της Stargraph & TriplRE.

Test MRR 0.7286 ± 0.0009 **Validation MRR** 0.7335 ± 0.0011

Ο συγγραφέας για να επιτύχει υψηλές προβλέψεις χρησιμοποίησε 93.039.522 παραμέτρους, και 2.500.604 οντότητες της Wikidata. Η κατανομή του συνόλου δεδομένων που χρησιμοποιούνται στην εκπαίδευση, στο test και στο validation φαίνεται στην εικόνα 36.



Εικόνα 36, Κατανομή του συνόλου δεδομένων στα train.test και valid

Η υπολογιστική δύναμη που χρησιμοποίησε για να εκτελεστεί ο αλγόριθμος, λόγω του μεγάλου πλήθους δεδομένων και παραμέτρων, ήταν ο NVIDIA Tesla A100 (40GB). Οι παράμετροι και οι τιμές που όρισε ο συγγραφέας του κώδικα ακολουθούν παρακάτω.

python run.py --print_on_screen --cuda --do_train -b 512 -n 64 -adv --model TripleRE -lr 0.00005 --valid_steps 60000 --log_steps 60000 --max_steps 500000 --do_valid --do_test --test_log_steps 60000 --gamma 6.0 -randomSeed 0 -a 2 --drop 0.05 --uni_weight --inverse --val_inverse -u 0.1 --anchor_size 80000 -ancs 20 -d 512 --head_dim 8 --mlp_ratio 2 -path

8.2 Αποτέλεσμα της παραμετροποίησης

Ένας δευτερεύοντας στόχος της διπλωματικής εργασίας ήταν η διενέργεια πειράματος. Το πείραμα αφορά την εκτέλεση του κώδικα, σε υπολογιστικό μέσο που διαθέτει ως χαρακτηριστικό την NVIDIA GeForce 1650 Ti (4GB). Για να επιτευχθεί η εκτέλεση του κώδικα ακολούθησε η μείωση των τιμών των παραμέτρων, λόγω αδυναμίας υπολογιστικής ισχύς. Σκοπός του πειράματος μας, δεν αποτελεί η βελτιστοποίηση της αποτελεσματικότητας του αλγορίθμου, μιας και ο αποτελεσματικότερος αλγόριθμος αναλύθηκε παραπάνω [1].

Στόχος του πειράματος είναι να βρεθούν οι βέλτιστες τιμές των παραμέτρων, ώστε να τρέχει ο αλγόριθμος στο υπάρχον σύστημα. Πριν παρουσιάσουμε τις τιμές των παραμέτρων που διερευνήθηκαν [38]–[40], ας κατανοήσουμε παραμέτρους όπως το batch size και τον ρυθμός μάθησης (learning rate). Η τιμή batch size περιορίζεται από τη μνήμη του μέσου που χρησιμοποιείται, επομένως καθίσταται απαραίτητο η τιμή να προσαρμοστεί στο μέγεθος της μνήμης μας. Σε αντίθεση με την παράμετρο του ρυθμού μάθησης, όπου η τιμή της δεν επηρεάζει τον υπολογιστικό χρόνο. Εάν το learning rate είναι πολύ μικρό, μπορεί να προκύψει υπερπροσαρμογή (overfitting). Τα μεγάλα ποσοστά learning rate βοηθούν στην εξομάλυνση της εκπαίδευσης, αλλά αν το ποσοστό μάθησης είναι πολύ μεγάλο, θα προκύψει υποπροσαρμογή (underfitting).

Αρχικά στο πείραμα προσδιορίστηκε η τιμή της παραμέτρου batch size ίση με 60, καθώς αυτή είναι η παράμετρος που κατά κύριο λόγο επηρεάζει την μνήμη μας. Η τιμή της παραμέτρου θα μείνει σταθερή, καθώς από τις δοκιμές προκύπτει ότι είναι ο μέγιστος αριθμός batch size για τον οποίο ο αλγόριθμος είναι λειτουργικός. Έπειτα γίνεται διερεύνηση των τιμών των υπολοίπων παραμέτρων. Το εύρος των τιμών κατά το tuning των παραμέτρων προσδιορίζονται στο πίνακα 2.

Πίνακας 2, Τιμές των παραμέτρων από το tuning

Μέθοδος TriplRE & StarGraph			
Παράμετρος	**Προκαθορισμένη τιμή παραμέτρου**	**Εύρος τιμών στο Tuning**	**Περιγραφή παραμέτρων**
batch size	512	[2,60]	Αριθμό των παραδειγμάτων εκπαίδευσης που χρησιμοποιούνται σε μια επανάληψη
learning rate	0.00005	[0.0001,0.5]	Ο ρυθμό μάθησης του μοντέλου
valid steps	60000	[100,1800]	Η εκπαίδευση στο validation γίνεται ανά x βήματα
log steps	60000	[100,1800]	Η διαδικασία εκπαίδευσης καταγράφεται κάθε x βήματα
max steps	500000	[1000,18000]	Η εκπαίδευση σταματά έπειτα από x βήματα
test log steps	60000	60000	Ορίζει πόσο συχνά το μοντέλο θα αξιολογεί την απόδοσή του σε ένα σύνολο δοκιμών κατά τη διάρκεια της εκπαίδευσης
gamma	6.0	6.0	Ορίζει την ισχύ ενός συγκεκριμένου τύπου μηχανισμού αλληλεπίδρασης ή προσοχής.
randomseed	0	0	Αρχικοποιεί ένα σύνολο τυχαίων αριθμών
Drop	0.05	0.05	Το ποσοστό απόρριψης ως πιθανότητα
anchor size	80000	[20,10000]	Το μέγεθος του συνόλου αγκυρών
Ancs	20	20	Θέτει τον αριθμό των αγκυρών
head dim	8	8	Η διάσταση των αποτελεσμάτων εξόδου του μηχανισμού προσοχής
mlp ratio	2	2	Η αναλογία του αριθμού των κρυφών μονάδων στα στρώματα προς τη διάσταση των embeddings
N	64	64	Κατανέμει τις εργασίες στα threads της CPU
A	2	2	Αριθμός των κεφαλών προσοχής

Η βέλτιστη τιμή της MRR επιτεύχθηκε με τις παρακάτω τιμές των παραμέτρων.

Πίνακας 3, Οι τιμές των βέλτιστων παραμέτρων

Παράμετρος	**Τιμές των παραμέτρων**
batch size	60
learning rate	0.0001
valid steps	1800
log steps	1800
max steps	18000

test log steps	60000
gamma	6.0
randomseed	0
drop	0.05
anchor size	10000
ancs	20
head dim	8
mlp ratio	2
n	64
a	2

Η τιμή της μετρικής MRR σε όλα τα ζεύγη τιμών των παραμέτρων, δεν ξεπερνούσε το 0.27. Καθώς και η βέλτιστη τιμή που επιτεύχθηκε από τις παραμέτρους (πίνακας 2), είναι:
Test MRR 0.273617 **Validation MRR** 0.281695

Παραθέτεται επίσης, ένα μικρό σύνολο των αποτελεσμάτων της πρόβλεψης και οι τιμές των παραμέτρων με τις οποίες επιτεύχθηκε. Αναφέρονται μόνο οι παράμετροι των οποίων οι τιμές δεν είναι οι προκαθορισμένες.

Πίνακας 4, Τα αποτελέσματα πρόβλεψης από διάφορες τιμές παραμέτρων

Test MRR	**Valid MRR**	**Παράμετροι**
0.000886	0.000833	b=32, valid step=200, test step=200, max step=1000, lr=0.0005
0.110233	0.112087	b=32, valid step=200, test step=200, max step=1000, lr=0.005
0.206182	0.209353	b=32, valid step=400, test step=400, max step=2000, lr=0.0005
0.273617	0.281695	b=60, valid step=8000, test step=8000, max step=18000, lr=0.0001

Η μετρική MRR κυμαίνεται στο διάστημα 0 και 1, όσο πιο υψηλή είναι η τιμή, τόσο καλύτερη είναι η απόδοση του μοντέλου [1].

Αν και στο πείραμα μας χρησιμοποιήθηκε ολόκληρο το σύνολο δεδομένων του ogb-wikikg2, παραθέτεται ένα μικρό δείγμα τριπλετών των δεδομένων που χρησιμοποιήθηκαν (πίνακας 5).

Πίνακας 5, Ένα δείγμα του συνόλου τριπλετών

Q7026367	P800	Q3234372
Q8053832	P27	Q1033
Q17057457	P20	Q2639273
Q2820944	P106	Q11513337
Q6733983	P26	Q6166404
Q14913903	P682	Q14758903
Q2562736	P463	Q1583587
Q3619630	P39	Q30185
Q170260	P166	Q579688
Q1625200	P1435	Q15632117
Q251338	P463	Q2720582
Q7610674	P1412	Q1860
Q3170566	P1412	Q150
Q1730080	P735	Q15731830
Q5533530	P106	Q11513337
Q1886549	P279	Q2994387

8.3 Μελλοντική μελέτη

Η έρευνα στον κλάδο των GNN είναι εξελισσόμενη, και αναπτύσσονται συνεχώς νέες μέθοδοι πρόβλεψης με καλύτερη ακρίβεια αποτελεσμάτων. Οι μελλοντικές μελέτες είναι συνετό να επικεντρωθούν στη χρήση νέων μεθόδων αλλά και στο συνδυασμό αυτών. Συνάμα με τις μεθόδους, θα πρέπει να γίνει ανάλυση των χαρακτηριστικών που βελτιώνουν τη πρόβλεψη του μοντέλου.

Συμπερασματικά, η ακρίβεια την πρόβλεψης είναι εξαρτώμενη από

- την ποιότητα και την ποσότητα των δεδομένων εκπαίδευσης

- την κατάλληλη ρύθμιση των παραμέτρων, όπως ο αριθμός των επιπέδων, ο αριθμός των νευρώνων σε κάθε επίπεδο, ο ρυθμός μάθησης και το batch size

ΠΑΡΑΡΤΗΜΑ Ι: ΠΗΓΕΣ

Πηγές

[1] H. Li, X. Gao, L. Feng, Y. Deng, Y. Yin, and A. Research, “STARGRAPH: KNOWLEDGE REPRESENTATION LEARNING BASED ON INCOMPLETE TWO-HOP SUBGRAPH”, Accessed: Feb. 05, 2023. [Online]. Available: https://github.com/hzli-ucas/StarGraph.

[2] W. Hu *et al.*, “Open Graph Benchmark: Datasets for Machine Learning on Graphs Steering Committee”, Accessed: Feb. 05, 2023. [Online]. Available: https://ogb.stanford.edu.

[3] “IEEE Xplore Full-Text PDF:” https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9189862 (accessed Feb. 05, 2023).

[4] C. Ying *et al.*, “Do Transformers Really Perform Bad for Graph Representation?”, Accessed: Feb. 05, 2023. [Online]. Available: https://github.com/Microsoft/Graphormer.

[5] Z. Wang *et al.*, “Pairwise Learning for Neural Link Prediction”.

[6] “OhMyGraphs: GraphSAGE and inductive representation learning | by Nabila Abraham | Analytics Vidhya | Medium.” https://medium.com/analytics-vidhya/ohmygraphs-graphsage-and-inductive-representation-learning-ea26d2835331 (accessed Feb. 07, 2023).

[7] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs”.

[8] G. Li, C. Xiong, A. Thabet, and B. Ghanem, “DeeperGCN: All You Need to Train Deeper GCNs”, Accessed: Feb. 05, 2023. [Online]. Available: https://www.deepgcns.org

[9] “What Is a GNN? How Do Graph Neural Networks Work? - SEON.” https://seon.io/resources/dictionary/graph-neural-network-gnn/ (accessed Feb. 05, 2023).

[10] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “HOW POWERFUL ARE GRAPH NEURAL NETWORKS?”.

[11] H. Yuan, J. Tang, X. Hu, and S. ji, “XGNN: Towards Model-Level Explanations of Graph Neural Networks,” *Virtual Event*, p. 2020, doi: 10.1145/3394486.3403085.

[12] J. Zhou *et al.*, “Graph neural networks: A review of methods and applications,” *AI Open*, vol. 1, pp. 57–81, Jan. 2020, doi: 10.1016/J.AIOPEN.2021.01.001.

[13] “Build a GNN-based real-time fraud detection solution using Amazon SageMaker, Amazon Neptune, and the Deep Graph Library | AWS Machine Learning Blog.” https://aws.amazon.com/blogs/machine-learning/build-a-gnn-based-real-time-fraud-detection-solution-using-amazon-sagemaker-amazon-neptune-and-the-deep-graph-library/ (accessed Feb. 07, 2023).

[14] V.-A. Nguyen, D. Quoc Nguyen, V. Nguyen, T. Le, Q. Hung Tran, and D. Phung, “ReGVD: Revisiting Graph Neural Networks for Vulnerability Detection,” 2022, doi: 10.1145/3510454.3516865.

[15] A. Vretinaris, C. Lei, V. Efthymiou, X. Qin, and F. Özcan, “Medical Entity Disambiguation Using Graph Neural Networks Entity disambiguation; graph neural network; medical ontology ACM Reference Format,” vol. 10, no. 21, doi: 10.1145/3448016.3457328.

[16] P. Pradhyumna, G. P. Shreya, and Mohana, “Graph Neural Network (GNN) in Image and Video Understanding Using Deep Learning for Computer Vision Applications,” *Proceedings of the 2nd International Conference on Electronics and Sustainable Communication Systems, ICESC 2021*, pp. 1183–1189, Aug. 2021, doi: 10.1109/ICESC51422.2021.9532631.

[17] “Best Graph Neural Network architectures: GCN, GAT, MPNN and more | AI Summer.” https://theaisummer.com/gnn-architectures/ (accessed Feb. 07, 2023).

[18] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, “Graph convolutional networks: a comprehensive review,” *Comput Soc Netw*, vol. 6, no. 1, pp. 1–23, Dec. 2019, doi: 10.1186/S40649-019-0069-Y/FIGURES/1.

[19] T. Danel *et al.*, “Spatial Graph Convolutional Networks,” *Communications in Computer and Information Science*, vol. 1333, pp. 668–675, 2020, doi: 10.1007/978-3-030-63823-8_76/COVER.

[20] “Graph Attention Networks in Python | Towards Data Science.” https://towardsdatascience.com/graph-attention-networks-in-python-975736ac5c0c (accessed Feb. 07, 2023).

[21] P. Veličkovi´veličkovi´c, G. Cucurull, A. Casanova, A. Romero, P. Lì, and Y. Bengio, “GRAPH ATTENTION NETWORKS”.

[22] R. Rooki, “Application of general regression neural network (GRNN) for indirect measuring pressure loss of Herschel–Bulkley drilling fluids in oil drilling,” *Measurement*, vol. 85, pp. 184–191, May 2016, doi: 10.1016/J.MEASUREMENT.2016.02.037.

[23] L. Ding, P. Rangaraju, and A. Poursaee, “Application of generalized regression neural network method for corrosion modeling of steel embedded in soil,” *Soils and Foundations*, vol. 59, no. 2, pp. 474–483, Apr. 2019, doi: 10.1016/J.SANDF.2018.12.016.

[24] D. F. Specht, “A General Regression Neural Network,” *IEEE Trans Neural Netw*, vol. 2, no. 6, pp. 568–576, 1991, doi: 10.1109/72.97934.

[25] “Wikidata:Εισαγωγή - Wikidata.” https://www.wikidata.org/wiki/Wikidata:Introduction/el (accessed Feb. 05, 2023).

[26] D. Vrandě, “Wikidata: A Free Collaborative Knowledge Base”.

[27] “Q16943273 - Wikidata.” https://www.wikidata.org/w/index.php?title=Q16943273&tour=wbitems&data=ok%20ITEM%20PROPERTY%20KLP&data=ok (accessed Feb. 05, 2023).

[28] “Wikidata item - Wikidata.” https://www.wikidata.org/wiki/Q16222597#/media/File:Datamodel_in_Wikidata.svg%20EIKONA%20ME%20TA%20 (accessed Feb. 05, 2023).

[29] Π. Εργασία, “Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης Σχολή Θετικών Επιστημών Τμήμα Πληροφορικής”.

[30] S. Malyshev, M. Krötzsch, L. González, J. Gonsior, and A. Bielefeldt, “Getting the Most out of Wikidata: Semantic Technology Usage in Wikipedia’s Knowledge Graph”, Accessed: Feb. 05, 2023. [Online]. Available: https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format,

[31] A. Piscopo and E. Simperl, “What we talk about when we talk about Wikidata quality: a literature survey,” 2019, doi: 10.1145/3306446.3340822.

[32] “Leaderboards for Link Property Prediction | Open Graph Benchmark.” https://ogb.stanford.edu/docs/leader_linkprop/#ogbl-wikikg2 (accessed Feb. 19, 2023).

[33] H. Li, X. Gao, L. Feng, Y. Deng, Y. Yin, and A. Research, “STARGRAPH: KNOWLEDGE REPRESENTATION LEARNING BASED ON INCOMPLETE TWO-HOP SUBGRAPH”, Accessed: Feb. 19, 2023. [Online]. Available: https://github.com/hzli-ucas/StarGraph.

[34] H. Nilforoshan *et al.*, “Zero-shot causal learning”.

[35] “Wikidata Query Service.” https://query.wikidata.org/ (accessed Feb. 05, 2023).

[36] “PyTorch documentation — PyTorch 1.13 documentation.” https://pytorch.org/docs/stable/index.html (accessed Feb. 05, 2023).

[37] H. Li, X. Gao, L. Feng, Y. Deng, Y. Yin, and A. Research, “STARGRAPH: KNOWLEDGE REPRESENTATION LEARNING BASED ON INCOMPLETE TWO-HOP SUBGRAPH”, Accessed: Feb. 07, 2023. [Online]. Available: https://github.com/hzli-ucas/StarGraph.

[38] P. Probst and B. Bischl, “Tunability: Importance of Hyperparameters of Machine Learning Algorithms,” *Journal of Machine Learning Research*, vol. 20, pp. 1–32, 2019, Accessed: Feb. 11, 2023. [Online]. Available: http://jmlr.org/papers/v20/18-444.html.

[39] “Hyper-parameter Tuning Techniques in Deep Learning | by Javaid Nabi | Towards Data Science.” https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8 (accessed Feb. 11, 2023).

[40] W. Fu, T. Menzies, D. Chen, and A. Agrawal, “Building Better Quality Predictors Using ‘ϵ-Dominance,’” Mar. 2018, Accessed: Feb. 11, 2023. [Online]. Available: http://arxiv.org/abs/1803.04608

ΠΑΡΑΡΤΗΜΑ ΙΙ: ΑΚΡΩΝΥΜΙΑ

Ακρωνύμια

OGB	Open Graph Benchmark
GNN	Graph Neural Network
ML	Machine Learning
KG	Knowledge Graph
AST	Abstract Syntactic Trees
LP	Linear Program
GCN	Graph Convolutional Network
XMC	Extreme Multi-label Classification
MLP	Multilayer Perceptron
GAT	Graph Attention Network
GRNN	Generative regression neural network
KGE	Knowledge Graph Embedding
WDQS	Wikidata Query Service
RDF	Resource Description Framework
MRR	Mean Reciprocal Rank
GIN	Graph Isomorphism Network
GAMLP	Graph Attention Multi-Layer Perception
PLNLP	Pairwise Learning for Neural Link Prediction
GPU	Graphics Processing Unit
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
TriplRE	Triple Relation Encoding