Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Συστηματων Τεχνητης Νοημοσυνης και Μαθησης

Graph Neural Networks for Optimal and Efficient Generation of Textual Counterfactuals

Diploma Thesis

by

Dimitris Lymperopoulos

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2024

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης

Graph Neural Networks for Optimal and Efficient Generation of Textual Counterfactuals

Diploma Thesis

by

Dimitris Lymperopoulos

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 24η Οκτωβρίου, 2024.

.....	.....	.....
Γεώργιος Στάμου	Αθανάσιος Βουλόδημος	Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Ομότιμος Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2024

.....
Δημητρης Λυμπεροπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Οι Εξηγήσεις με Αντιπαράδειγμα παρέχουν αιτιολογία με τη μορφή αλλαγών που πρέπει να γίνουν προκειμένου ένα μοντέλο να λάβει μια διαφορετική απόφαση. Όταν το εν λόγω μοντέλο είναι ένας ταξινομητής μαύρου κουτιού και η είσοδος αποτελείται από δεδομένα κειμένου, πολλα συστήματα εξηγήσεων με αντιπαράδειγμα προσπαθούν να αποκτήσουν γνώσεις σχετικά με την εσωτερική λειτουργία του μοντέλου τροποποιώντας ελαφρώς τις αρχικές περιπτώσεις. Ωστόσο, τα περισσότερα από αυτά είναι υπολογιστικά δαπανηρά λόγω του τεράστιου χώρου εναλλακτικών επιλογών που πρέπει να αναζητήσει κανείς όταν μεταβάλλει ένα κείμενο.

Σε αυτή τη διατριβή, προτείνουμε τη χρήση των πρόσφατα ευδοκιμούντων μοντέλων βαθιάς μάθησης που λειτουργούν ειδικά σε δεδομένα δομημένα με γράφους, τα λεγόμενα Νευρωνικά Δίκτυα Γράφων (ΝΔΓ). Παρουσιάζουμε ένα σύστημα για τη δημιουργία σημασιολογικά επεξεργασμένων κειμένων, γνωστών ως αντιφατικές παρεμβάσεις, οι οποίες αλλάζουν την πρόβλεψη του μοντέλου, παρέχοντας έτσι μια μορφή εξηγήσεων με αντιπαράδειγμα για το μοντέλο. Το σύστημα χρησιμοποιεί έναν ειδικό τύπο γραφήματος που είναι γνωστός ως διμερής γράφος (ή διγράφος) μαζί με ένα ΝΔΓ που αναπτυξαμε έτσι ώστε να προσομοιώνει την λύση για το Ορθογώνιο Πρόβλημα Γραμμικής Ανάθεσης. Κατά τη διάρκεια των πειραμάτων μας, αναδεικνύουμε τον ευέλικτο χαρακτήρα του συστήματος και συζητάμε πολλαπλούς συμβιβασμούς όσον αφορά την επεξηγηματικότητα, την ελαχιστοποίηση των αλλαγών και την ταχύτητα. Αξιολογούμε το σύστημά μας σε δύο προβλήματα ΕΦΓ - δυαδική ταξινόμηση συναισθήματος και ταξινόμηση θεμάτων - και δείχνουμε ότι οι παραγόμενες αλλαγές είναι αντιθετικές, νοηματικά ορθές και ελάχιστες, ενώ η όλη διαδικασία παραμένει σημαντικά ταχύτερη σε σχέση με άλλα παρόμοια σύγχρονα συστήματα.

Λέξεις-κλειδιά — Νευρωνικά Δίκτυα Γράφων, Εξηγήσεις με Αντιπαράδειγμα, Συστήματα Εξηγήσεων με Αντιπαράδειγμα, Διμερείς Γράφοι, Ορθογώνιο Πρόβλημα Γραμμικής Ανάθεσης

Abstract

Counterfactual explanations provide reasoning in the form of changes needed to be made in order for a model to make a different decision. When the model in question is a black-box classifier and the input consists of textual data, many counterfactual editors attempt to gain insights about the inner workings of the model by slightly altering the original instances. However most of them are computationally expensive due to the massive space of alternatives one has to search when altering a text.

In this thesis, we propose using the recently thriving deep learning models which specifically operate on graph structured data, called Graph Neural Networks (GNN). We present an editor that generates semantically edited inputs, known as counterfactual interventions, which change the model prediction, thus providing a form of counterfactual explanations for the model. The editor utilizes a special graph type knows as a bipartite graph (or bigraph) along with a GNN that we developed so that it simulates the solution to the Rectangular Linear Assignment Problem (RLAP). During our experiments, we showcase the editor's flexible nature, and discuss multiple trade-offs regarding explainability, minimality and speed. We test our editor on two NLP tasks - binary sentiment classification and topic classification - and show that the generated edits are contrastive, fluent and minimal, while the whole process remains significantly faster than other state-of-the-art counterfactual editors.

Keywords — Graph Neural Networks, Counterfactual Explanations, Counterfactual Editors, Bipartite Graphs, Rectangular Linear Assignment Problem

Ευχαριστίες

Αυτό το έργο δεν θα ήταν δυνατό χωρίς την υποστήριξη πολλών ανθρώπων. Ευχαριστώ πολύ τον επιβλέποντα μου, κ. Στάμου Γεώργιο, για την ευκαιρία που μου έδωσε να εκπονήσω την διπλωματική μου εργασία στο εργαστήριο Συστημάτων Τεχνητής Νοημοσύνης και Μάθησης. Ευχαριστώ επίσης την Μαρία Λυμπεραίου και τον Γιώργο Φιλανδριανό για τη στενή συνεργασία και υποστήριξη τους καθ' όλη τη διάρκεια εξερεύνησης των καινούριων αυτών αντικειμένων.

Πολύ σημαντική για εμένα ήταν ακόμα η συναισθηματική συνεισφορά της οικογένειας και των φίλων μου που ήταν δίπλα μου σε κάθε δυσκολία. Ιδιαίτερα θα ήθελα να ευχαριστήσω τους γονείς μου, καθώς και τους πλέον συναδέλφους μου Γιάννη, Βύρωνα, Γιώργο και Νίκο, με τους οποίους ξεπεράσαμε κάθε εμπόδιο τα τελευταία αυτά χρόνια.

Δημήτρης Λυμπερόπουλος, Οκτώβριος 2024

Contents

List of Figures

Chapter 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Θεωρητικό Υπόβαθρο

Καθώς τα μοντέλα επεξεργασίας φυσικής γλώσσας (ΕΦΓ) γίνονται όλο και πιο αναπόσπαστο μέρος των διαδικασιών λήψης αποφάσεων, η ανάγκη για επεξηγηματικότητα και ερμηνευσιμότητα έχει καταστεί υψίστης σημασίας. Τα μοντέλα αυτά χρησιμοποιούνται ευρέως σε διάφορες εφαρμογές, όπως η ανάλυση συναισθήματος, η ταξινόμηση θεμάτων, η αυτόματη μετάφραση και οι πράκτορες συνομιλίας. Οι αποφάσεις τους έχουν συχνά σημαντικές επιπτώσεις, επηρεάζοντας τα πάντα, από συστάσεις προϊόντων μέχρι εγκρίσεις δανείων και διαγνώσεις υγειονομικής περίθαλψης. Ωστόσο, η φύση μαύρου-κουτιού πολλών εξελιγμένων μοντέλων ΕΦΓ, ιδίως των προσεγγίσεων που βασίζονται στη βαθιά μάθηση, δημιουργεί προκλήσεις για την κατανόηση του τρόπου λήψης συγκεκριμένων αποφάσεων. Αυτή η αδιαφάνεια μπορεί να υπονομεύσει την εμπιστοσύνη και να περιορίσει την ευρύτερη υιοθέτηση αυτών των τεχνολογιών σε κρίσιμους και ευαίσθητους τομείς.

Δεδομένων των υψηλών διακυβευμάτων που υπάρχουν, είναι επιτακτική ανάγκη η ανάπτυξη μεθόδων που μπορούν να παρέχουν σαφείς, αξιοποιήσιμες γνώσεις σχετικά με τη συμπεριφορά των μοντέλων ΕΦΓ. Μια πολλά υποσχόμενη προσέγγιση είναι η δημιουργία εξηγήσεων με αντιπαράδειγμα. Οι εξηγήσεις με αντιπαράδειγμα είναι υποθετικά σενάρια που δείχνουν πώς ελάχιστες αλλαγές στην είσοδο μπορούν να οδηγήσουν σε διαφορετικά αποτελέσματα του μοντέλου. Παρουσιάζοντας αυτά τα εναλλακτικά σενάρια, οι εξηγήσεις αυτές βοηθούν τους χρήστες να κατανοήσουν ποια χαρακτηριστικά έχουν μεγαλύτερη επιρροή στη διαδικασία λήψης αποφάσεων του μοντέλου. Αυτός ο τύπος εξήγησης είναι ιδιαίτερα πολύτιμος επειδή ευθυγραμμίζεται με την ανθρώπινη λογική: η κατανόηση σεναρίων "τι θα συμβεί αν" είναι ένας φυσικός τρόπος για τους ανθρώπους να αντιλαμβάνονται τις αιτιώδεις σχέσεις και να λαμβάνουν τεκμηριωμένες αποφάσεις.

Αυτή η διατριβή εισάγει μια νέα προσέγγιση για τη δημιουργία εξηγήσεων με αντιπαράδειγμα για μοντέλα ΕΦΓ, καλύπτοντας μια κρίσιμη ανάγκη για ερμηνευσιμότητα στον τομέα της Τεχνητής Νοημοσύνης. Αντλούμε έμπνευση από το [50] και παρουσιάζουμε ένα Νευρωνικό Δίκτυο Γράφων (ΝΔΓ) ικανό να επιλύει το πρόβλημα ορθογώνιας γραμμικής ανάθεσης (Rectangular Linear Assignment Problem - RLAP). Χρησιμοποιώντας αυτό το μοντέλο αντί των παραδοσιακών αλγορίθμων ανάθεσης γράφων, όπως ο *Ουγγρικός αλγόριθμος* η εργασία μας επιτυγχάνει σημαντική βελτίωση του χρόνου εκτέλεσης. Παρέχουμε επίσης μια ολοκληρωμένη αξιολόγηση του συστήματός μας σε διάφορα προβλήματα ΕΦΓ, αποδεικνύοντας την ευελιξία και την αποτελεσματικότητά του. Δίνοντας έμφαση στην δημνιουργία ελάχιστων και ευγλωττων παρεμβάσεων, διασφαλίζουμε ότι οι παραγόμενες επεξηγήσεις είναι τόσο κατανοητές όσο και πρακτικά χρήσιμες, ενώ η όλη διαδικασία παραμένει ταχύτερη από άλλους σύγχρονους συντάκτες εξηγήσεων με αντιπαράδειγμα .

1.1.1 Διμερεις Γραφοι

Η δομή ενός γράφου εστιάζει στις σχέσεις μεταξύ οντοτήτων, καθιστώντας την έτσι ένα κατάλληλο μέσο αναπαράστασης δεδομένων σε πολλά πεδία. Σε αυτή τη διατριβή θα χρησιμοποιηθεί ένα ειδικό είδος γράφου, γνωστό ως *διμερής γράφος* ή απλά *διγράφος*.

Στη θεωρία γράφων, ένας διμερής γράφος (ή διγράφος) είναι ένας γράφος του οποίου οι κόμβοι (ή κορυφές) μπορούν να χωριστούν σε δύο διαχωρισμένα και ανεξάρτητα σύνολα S και T έτσι ώστε κανένας κόμβος μέσα στο ίδιο σύνολο να μην είναι γειτονικός. Τυπικά, ένας γράφος $G = (V, E)$ είναι διμερής αν το σύνολο κόμβων του V μπορεί να χωριστεί σε δύο υποσύνολα U, W έτσι ώστε $U \cup W = V$, $U \cap W = \emptyset$ και όπου κάθε ακμή $e_{u \to w} \in E$ συνδέει έναν κόμβο $u \in U$ με έναν κόμβο $w \in W$. Αυτή η δομή καθιστά τους διμερείς γράφους ιδιαίτερα χρήσιμους στη μοντελοποίηση σχέσεων μεταξύ δύο διαφορετικών τύπων οντοτήτων.

Τα δύο σύνολα U και W μπορούν να θεωρηθούν ως χρωματισμός του γράφου με δύο χρώματα: αν χρωματίσει κανείς όλους τους κόμβους στο U με ένα χρώμα και όλους τους κόμβους στο W με ένα διαφορετικό, κάθε ακμή συνδέει κόμβους διαφορετικού χρώματος, όπως απαιτείται στο πρόβλημα χρωματισμού γράφων [86]. Αυτό φαίνεται στο Σχήμα 1.1.1, όπου το σύνολο κόμβων U είναι χρωματισμένο με κόκκινο χρώμα και το σύνολο κόμβων W με μπλε χρώμα. Αντίθετα, ένας τέτοιος χρωματισμός είναι αδύνατος στην περίπτωση ενός μη διμερούς γραφήματος, όπως ένα τρίγωνο: αφού ένας κόμβος χρωματιστεί μπλε και ένας άλλος κόκκινος, η τρίτη κορυφή του τριγώνου συνδέεται με κορυφές και των δύο χρωμάτων, εμποδίζοντας την ανάθεσή της σε κάποιο από τα δύο χρώματα.

Συχνά γράφουμε $G = (U, W, E)$ για να συμβολίσουμε έναν διγράφο του οποίου η διαμέριση αποτελείται από τα σύνολα κόμβων U και W με το E να συμβολίζει το σύνολο των ακμών. Ένας τέτοιος γράφος αναπαρίσταται συνήθως από τον πίνακα διζωνικότητας, ο οποίος είναι ένας πίνακας $(0, 1)$μεγέθους $|U| \times |W|$. Σε αυτόν τον

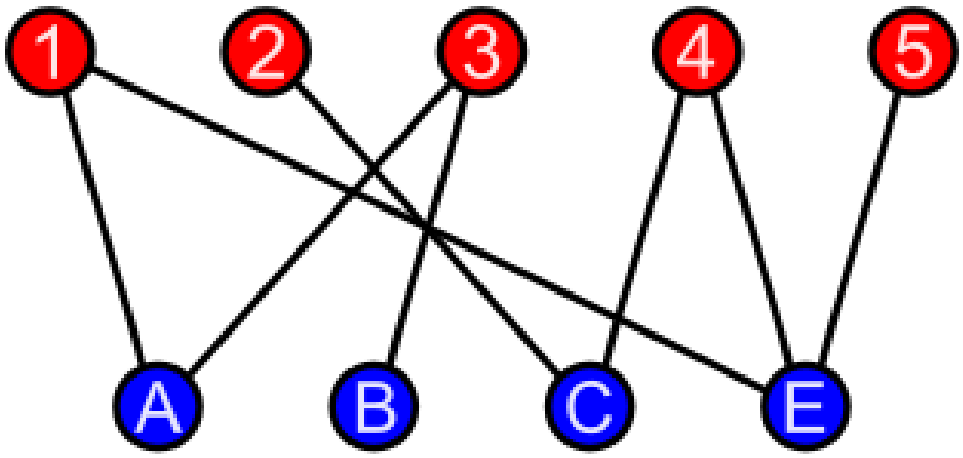


Figure 1.1.1: Ένα παράδειγμα διμερούς γράφου

πίνακα, η τιμή του στοιχείου ij είναι 1 αν $\exists e_{i\to j} \in E : i \in U, j \in W$, ή με απλά λόγια αν οι κόμβοι i και j είναι γειτονικοί. Διαφορετικά, η τιμή του στοιχείου είναι 0. Σε περίπτωση που ο γράφος είναι σταθμισμένος, το στοιχείο ij ισούται με το βάρος της ακμής που συνδέει αυτούς τους δύο κόμβους.

Μια από τις πιο γνωστές εφαρμογές των διμερών γράφων είναι σε προβλήματα αντιστοίχισης, όπως η ανάθεση θέσεων εργασίας, η κατανομή πόρων και η ροή δικτύου. Σε αυτά τα πλαίσια, οι κόμβοι στο ένα σύνολο αντιπροσωπεύουν πράκτορες ή εργασίες, ενώ οι κόμβοι στο άλλο σύνολο αντιπροσωπεύουν θέσεις εργασίας, πόρους ή κόμβους δικτύου. Οι ακμές είναι συνήθως σταθμισμένες, με την τιμή του βάρους να αντιστοιχεί σε ένα *κόστος*, όπως οι ώρες που χρειάζεται ένας πράκτορας για να εκτελέσει μια δεδομένη εργασία κ.λπ. Οι διμερείς γράφοι στην παρούσα διατριβή χρησιμοποιούνται για την αντικατάσταση λέξεων από ένα δεδομένο κείμενο, όπου στην περίπτωση αυτή θα αναφερόμαστε στα δύο σύνολα κόμβων ως *σύνολο κόμβων πηγής* και *σύνολο κόμβων στόχου*.

1.1.2 Νευρωνικά Δίκτυα Γράφων

Δεδομένου ότι η δομή του γράφου αναδύεται φυσικά παντού γύρω μας, εφευρέθηκαν νευρωνικά δίκτυα που λειτουργούν απευθείας σε δεδομένα αυτού του τύπου. Τα γραφήματα είναι μη ευκλείδεια δεδομένα και επομένως τα GNN μπορούν να ομαδοποιηθούν στην ευρύτερη κατηγορία της Γεωμετρικής Μάθησης [7]. Τα Νευρωνικά Δίκτυα Γράφων (GNN) είναι γνωστά για την εκφραστική τους ισχύ και πρόσφατα κερδίζουν δημοτικότητα λόγω των αυξανόμενων δυνατοτήτων τους σε διάφορες εφαρμογές όπως τα συστήματα συστάσεων και το μοριακό δακτυλικό αποτύπωμα [118].

Τα GNN δημιουργήθηκαν γιατι οι περισσότεροι συμβατικοί αλγόριθμοι Machine ή Deep Learning είναι ειδικά κατασκευασμένοι για να καλύπτουν συγκεκριμένο τύπο δεδομένων, όπως εικόνες ή κείμενο, όχι όμως γράφους. Οι περισσότερες αναπαραστάσεις δεδομένων μπορούν να γενικευθούν σε γράφους, αλλά το αντίθετο δεν ισχύει. Στη γενική περίπτωση, τα γραφήματα είναι πιο πολύπλοκα, έχοντας έναν μη σταθερό αριθμό μη ταξινομημένων κόμβων μέσα σε γειτονιές μεταβλητού μεγέθους, και επομένως τα υπάρχοντα μοντέλα δεν μπορούν να τα χειριστούν. Επιπλέον, οι περισσότεροι κοινοί αλγόριθμοι υποθέτουν την ανεξαρτησία στιγμιοτύπων. Αυτό δεν ισχύει όταν εκτελούνται εργασίες σε επίπεδο κόμβου όπου ένα γράφημα είναι η είσοδος του νευρωνικού δικτύου και τα στιγμιότυπα είναι οι κόμβοι του. Τέλος, τα κλασικά Συνελικτικα Νευρωνικά Δίκτυα λειτουργούν σε εικόνες ή γενικότερα κανονικά πλέγματα. Η έλλειψη εντοπιότητας με την παραδοσιακή έννοια στα δεδομένα γράφων, το αυθαίρετο μέγεθος και η αμετοβλητότητα τους σε μεταθέσεις καθιστούν δύσκολη την εκτέλεση της κανονικής συνέλιξης.

Τα Νευρωνικά Δίκτυα Γράφων μπορούν να ταξινομηθούν με διάφορους τρόπους: α) ανάλογα με το επίπεδο του γράφου στο οποίο λειτουργούν σε επιπέδου κόμβου, ακμής ή γραφου, β) ανάλογα με την αρχιτεκτονική που ακολουθούν σε συνελικτικά, επαναλαμβανόμεσα, αυτοκωδικοποιητές και χωροχρονικά και γ) ανάλογα με τον τρόπο εκπαίδευσης σε επιβλεπόμενα, μη επιβλεπόμενα και μερικώς επιβλεπόμενα. Παρακάτω θα αναλύσουμε τις

δύο εκδοχές συνελικτικών δικτύων που θα χρησιμοποιηθούν στο πειραματικό μέρος.

Το **Graph Convolutional Network (GCN)** παρουσιάζει την ιδέα της χρήσης μιας προσέγγισης πρώτης τάξης του ChebNet προκειμένου να μετριαστεί η υπερπροσαρμογή. Στην πραγματικότητα, υποθέτει $K = 1$ και $\lambda_{max} = 2$. Στην ίδια κατεύθυνση το μοντέλο επιβάλλει τον περιορισμό $\theta = \theta_0 = -\theta_1$. Μετά την επιβολή αυτών των περιορισμών, η λειτουργία συνέλιξης είναι:

$$x *_G g_\theta = \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x \tag{1.1.1}$$

Αφού διαπιστώθηκε εμπειρικά ότι ο όρος $I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ προκαλεί αριθμητική αστάθεια, χρησιμοποιήθηκε ένα τέχνασμα *επανακανονικοποίησης*. Ο όρος $D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = \bar{A}$ αντικαταστάθηκε από $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}} = \hat{A}$ όπου $\tilde{A} = I_n + \bar{A}$ και $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Όλα τα παραπάνω μπορούν να περιγραφούν με αυτή τη συμπαγή εξίσωση:

$$H = X *_G g_\theta = f(\hat{A}X\Theta) \tag{1.1.2}$$

όπου το f είναι μια συνάρτηση ενεργοποίησης και επιτρέπονται πολλαπλές είσοδοι και έξοδοι λόγω της χρήσης πινάκων.

Το GCN είναι μια ειδική περίπτωση φασματικής προσέγγισης αφού μπορεί να εκληφθεί και ως χωρική. Στην παρακάτω εξίσωση, μπορούμε να δούμε πώς θα γίνει η συγκέντρωση πληροφοριών εντός της γειτονιάς. Σε αυτήν την περίπτωση ο ίδιος ο κόμβος θεωρείται επίσης ως γείτονάς του εαυτού του, ενός βήματος.

$$h_v = f(\Theta^T(\sum_{u \in N(u) \cup v} \hat{A}_{v,u}x_u)) \quad \forall u \in V \tag{1.1.3}$$

Αυτό το μοντέλο χρησιμοποιείται πολύ συχνά ως μέρος πιο σύνθετων αρχιτεκτονικών στη λογοτεχνία λόγω της απλότητας και της καλής πειραματικής του απόδοσης.

Το **Graph Attention Network (GAT)** [92] υιοθετεί την ιδέα της προσοχής που προτείνεται από το [91] προκειμένου να αποφασίσει ποια μέλη της γειτονιάς ενός κόμβου έχουν πιο σημαντικές πληροφορίες. Στόχος του είναι να μάθει τα σχετικά βάρη μεταξύ γειτονικών κόμβων και επομένως διαφέρει από προηγούμενες προσεγγίσεις όπως το GCN επειδή η έννοια της γειτονιάς δεν είναι προκαθορισμένη ή πανομοιότυπη.

Η συνεκτική λειτουργία ορίζεται ως:

$$h_v^{(k)} = \sigma(\sum_{u \in N(u) \cup v} \alpha_{vu}^{(k)} W^{(k)} h_u^{(k-1)}) \tag{1.1.4}$$

όπου τα βάρη προσοχής για κάθε κόμβο v μπορούν να οριστούν ως:

$$\alpha_{vu}^{(k)} = softmax(LeakyReLU(a^T[W^{(k)}h_v(k-1)||W^{(k)}h_u^{(k-1)}])) \tag{1.1.5}$$

Η μεταβλητή a αντιπροσωπεύει ένα σύνολο παραμέτρων με δυνατότητα εκμάθησης. Η αναπαράσταση των κρυφών επιπέδων αρχικοποιείται με τα χαρακτηριστικά κάθε κόμβου και η συνάρτηση softmax διασφαλίζει ότι τα βάρη της προσοχής αθροίζονται σε ένα.

Ο παραπάνω μηχανισμός ονομάζεται *self-attention*, αλλά το GAT χρησιμοποιεί επιπλέον *multi-head attention* για να σταθεροποιήσει τη μάθηση και να κάνει το μοντέλο πιο εκφραστικό. Οι ακριβείς εξισώσεις βρίσκονται στο [92].

Το GAT είναι αποτελεσματικό αφού από τα ζεύγη κόμβου-γείτονα μπορούν να υπολογιστούν ταυτόχρονα. Επιπλέον, τα μεγέθη της γειτονιάς του είναι αδιάφορα και μπορεί να εφαρμοστεί εύκολα σε επαγωγικά μαθησιακά προβλήματα.

1.1.3 Εξηγήσεις με Αντιπαράδειγμα

Οι εξηγήσεις με αντιπαράδειγμα (counterfactual explanations) στον τομέα της Επεξηγησιμότητας στην Τεχνητή Νοημοσύνη (explainable AI) στοχεύουν να δώσουν μια εξήγηση για το "Τι θα πρέπει να αλλάξει προκειμένου το μοντέλο να λάβει μια διαφορετική απόφαση". Επομένως, μπορούν ουσιαστικά να εξηγήσουν προβλέψεις μεμονωμένων περιπτώσεων, όπου οι αιτίες του προβλεπόμενου αποτελέσματος είναι συγκεκριμένες τιμές χαρακτηριστικών αυτής της περίπτωσης. Είναι αντιθετικές και επιλεκτικές, που σημαίνει ότι βρίσκουν τις ελάχιστες αλλαγές στον χώρο των χαρακτηριστικών. Ταυτόχρονα, είναι εύκολα κατανοητές από τους ανθρώπους και συνήθως προσφέρουν πολλαπλές διαφορετικές απαντήσεις για την ίδια περίπτωση που την εξηγούν εξίσου καλά.

Αντιθετικές Παρεμβάσεις

Η εργασία μας βασίζεται στην δημιουργία Αντιθετικών Παρεμβάσεων ως μια μορφή Εξηγήσεων με Αντιπαράδειγμα για δεδομένα κειμένου. Ουσιαστικά, ο στόχος μας είναι να αντικαταστήσουμε λέξεις από το αρχικό κείμενο με κατάλληλα υποκατάστατα, έτσι ώστε να αλλάξει η πρόβλεψη του ταξινομητή.

Οι αντιφατικές παρεμβάσεις είναι ιδιαίτερα δύσκολες λόγω της μεταβλητής και πολυδιάστατης φύσης του κειμένου. Στην πραγματικότητα, η δημιουργία βέλτιστων γλωσσικών παρεμβάσεων είναι ένα αλγοριθμικά δύσκολο πρόβλημα, που απαιτεί αποτελεσματική βελτιστοποίηση του χώρου αναζήτησης των εναλλακτικών λύσεων [115, 94, 54, 113]. Ένα άλλο εμπόδιο που πρέπει να ξεπεραστεί είναι το γεγονός ότι η παραγωγή αντιπαραδειγμάτων για κείμενο απαιτεί σημασιολογικά σημαντικές επεξεργασίες που μεταβάλλουν την πρόβλεψη του μοντέλου, διατηρώντας παράλληλα την ευχέρεια και τη συνοχή του κειμένου. Έτσι, οι αντιπαραθετικές παρεμβάσεις σε προβλήματα ΕΦΓ πρέπει να ικανοποιούν διάφορα βασικά κριτήρια:

1. **Αντιθετικότητα**: Το αντιθετικό κείμενο x' πρέπει να οδηγεί σε διαφορετική πρόβλεψη του μοντέλου από το αρχικό x. Αυτή η αλλαγή στην πρόβλεψη αναδεικνύει την ευαισθησία του μοντέλου σε ορισμένα μέρη του κειμένου, προσφέροντας πληροφορίες σχετικά με το τι οδηγεί τις αποφάσεις του.
2. **Ελαχιστότητα**: Οι επεξεργασίες που απαιτούνται για τη μετατροπή του x σε x' θα πρέπει να είναι ελάχιστες σύμφωνα με κάποια μετρική.
3. **Ευγλωττία**: Το επεξεργασμένο κείμενο x' πρέπει να είναι γραμματικά ορθό και σημασιολογικά συνεκτικό. Η διατήρηση της ευχέρειας είναι απαραίτητη για να διασφαλιστεί ότι το παραχθέν κείμενο δεν είναι μόνο υπολογιστικά έγκυρο αλλά και ερμηνεύσιμο και με νόημα για τους ανθρώπινους χρήστες.
4. **Αληθοφάνεια**: Το κείμενο που προκύπτει μετά τις επεξεργασίες πρέπει να παραμένει εντός ενός εύλογου εύρους της αρχικής εισόδου. Για παράδειγμα, σε μια εργασία ανάλυσης συναισθήματος, η αλλαγή μιας μόνο λέξης που μεταβάλλει σημαντικά το συναίσθημα αλλά διατηρεί το αρχικό πλαίσιο και νόημα, είναι προτιμότερη από την πλήρη επαναδιατύπωση μιας πρότασης.

Στην παρούσα διατριβή, εστιάζουμε κυρίως στην *αντιθετικότητα*, την *ελαχιστότητα* και την *ευχέρεια*. Η απαίτηση αληθοφάνειας ικανοποιείται επίσης ως παρενέργεια δύο πραγμάτων: της χρήσης αντικαταστάσεων σε επίπεδο λέξης και της υιοθέτησης του αριθμού των επεξεργασμένων λέξεων ως μετρική ελαχιστότητας. Το πρώτο εγγυάται ότι θα αλλάξουν λέξεις και όχι φράσεις, ενώ το δεύτερο εξασφαλίζει ότι οι αλλαγμένες λέξεις θα είναι όσο το δυνατόν λιγότερες.

1.1.4 Ορθογώνιο Πρόβλημα Γραμμικής Ανάθεσης

:Ένα βασικό κομμάτι αυτής της διατριβής, βασίζεται στο Ορθογώνιο Πρόβλημα Γραμμικής Ανάθεσης (RLAP) και την λύση του. Το συγκεκριμένο πρόβλημα αποτελεί μια γενίκευση του προβλήματος γραμμικής ανάθεσης (LAP)- θέλουμε να αναθέσουμε έναν αριθμό n εργασιών σε $m \geq n$ αριθμό πρακτόρων, ελαχιστοποιώντας το συνολικό αντίστοιχο κόστος [6]. Εφαρμογές υπάρχουν, π.χ., στους τομείς της αναγνώρισης αντικειμένων και του προγραμματισμού. Στην περίπτωσή μας, εφαρμόζουμε το πρόβλημα με τη μορφή ενός ταιριάσματος ελάχιστου βάρους σε έναν προκατασκευασμένο διμερή γράφο.

Ας θεωρήσουμε ένα σταθμισμένο διγράφο $G = (V, E)$, όπου το σύνολο ακμών E αποτελείται από όλες τις σταθμισμένες ακμές του γραφήματος, και το σύνολο κόμβων V αποτελείται από το σύνολο πηγής S μεγέθους $|S| = n$ και το σύνολο στόχου T μεγέθους $|T| = m$, έτσι ώστε $S \cup T = V$, $S \cap T = \emptyset$. Η εύρεση βέλτιστων συνδέσεων μεταξύ των κόμβων του G είναι ένα μακροχρόνια αναζητούμενο πρόβλημα διακριτής βελτιστοποίησης της θεωρίας γράφων, όπου η βέλτιστη αντιστοιχία για κάθε κόμβο $s \in S$ πρέπει να προσδιοριστεί μεταξύ ενός

προκαθορισμένου υποψήφιου συνόλου κόμβων $t \in T$. Υποθέτοντας ότι W συμβολίζει το σύνολο βαρών ακμών που αποτελείται από τα βάρη όλων των ακμών $e \in E$, ένα *ταίριασμα ελάχιστου βάρους* $M \subseteq E$ αναζητά ένα υποσύνολο του ελαφρύτερου δυνατού αθροίσματος βαρών ακμών $\sum w_e, w_e > 0 \in W$ που περιέχει εκείνες τις ακμές $e \in E$ που καλύπτουν όλους τους κόμβους του συνόλου $min(|S|, |T|)$ του G. Επομένως, στην περίπτωση του $|S| \leq |T|$, όλοι οι κόμβοι του S θα αντιστοιχιστούν σε έναν κόμβο του T, εάν υπάρχει μια εξερχόμενη ακμή $e_{s \to t}$ από κάθε s σε κάθε $t \neq s$. Υπό αυτές τις απαιτήσεις, διατυπώνουμε το ακόλουθο πρόβλημα βελτιστοποίησης με περιορισμούς:

$$min \sum w_e, \ \ subject\ to\ s \neq t\ \ if\ \exists e_{s \to t} \tag{1.1.6}$$

Η βέλτιστη λύση του παραπάνω προβλήματος μπορεί να βρεθεί με ντετερμινιστικούς αλγόριθμους, όπως ο *Ουγγρικός Αλγόριθμος* [43] με πολυπλοκότητα $O(n^3)$ και ο *Αλγόριθμος του Karp* [36] με πολυπλοκότητα $O(mn \log m)$. Μια πιο πρόσφατη προσέγγιση από τους Liu και λοιποι [50] ωστόσο, προτείνει την χρήση ενός ΝΔΓ για την επίλυσή του, όταν $|S| = |T|$. Το μοντέλο βασίζεται στις αρχιτεκτονικές GAT και GCN (βλ. Ενότητα 1.1.2) και αποτελείται από τρεις μονάδες: τον κωδικοποιητή, τη μονάδα συνέλιξης και τον αποκωδικοποιητή, όπως φαίνεται και στο σχήμα 1.1.2.

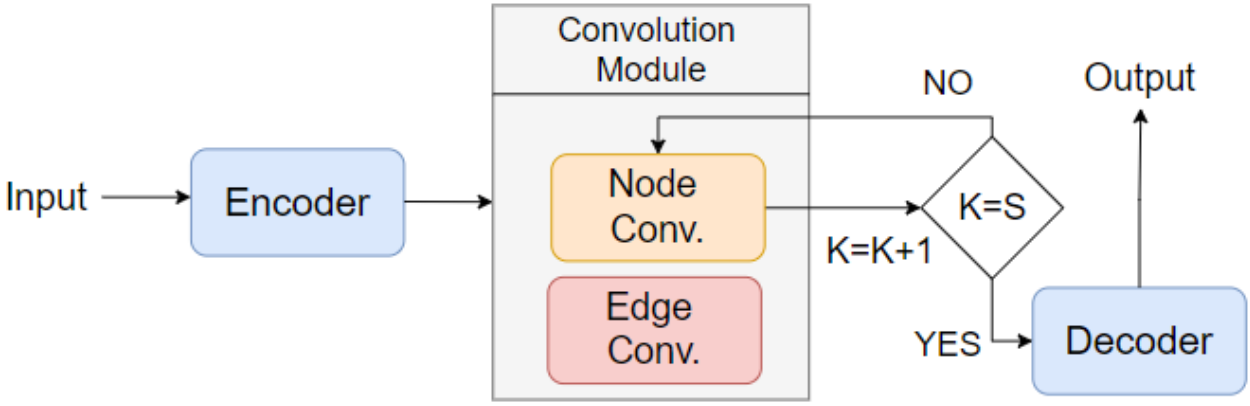


Figure 1.1.2: Η αρχιτεκτονική του προτεινόμενου μοντέλου ΝΔΓ. Στο επίπεδο συνέλιξης κόμβων, τα χαρακτηριστικά των κόμβων ενημερώνονται για συνολικά $S \geq 2$ επαναλήψεις.

Το πρόβλημα ανάθεσης μετατρέπεται σε πρόβλημα δυαδικής ταξινόμησης σε επίπεδο ακμής - η ετικέτα μιας ακμής είναι 1 εάν η ακμή ανήκει στο ταίριασμα ελάχιστου βάρους, διαφορετικά είναι ίση με -1. Έτσι, χρησιμοποιούμε την *Ισορροπημένης Διασταυρούμενης Εντροπίας*:

$$\begin{aligned} L = -\sum_{i=1}^{n}\sum_{j=1}^{m} \big(w \times y_{ij}^{gt} \log(y_{ij}) + (1-w) \times \\ (1 - y_{ij}^{gt}) \log(1 - y_{ij})\big) \end{aligned} \tag{1.1.7}$$

όπου y_{ij} είναι η προβλεπόμενη ετικέτα για την ακμή $i \to j$ που συνδέει τον κόμβο-πηγή i και τον κόμβο-στόχο j, y_{ij}^{gt} είναι η πραγματική ετικέτα της ακμής, και w είναι το βάρος που εξισορροπεί την απώλεια για να αποφευχθεί η κυριαρχία των αρνητικών ετικετών στην εκπαίδευση. Οι παράμετροι n, m υποδηλώνουν το μέγεθος των συνόλων κόμβων πηγής και στόχου, έτσι ώστε $|S| = n, |T| = m$.

Προκειμένου να επιβάλουμε τον περιορισμό αντιστοίχισης ένα προς ένα στο μοντέλο, κατασκευάζουμε πρώτα έναν πίνακα προβλεπόμενης ανάθεσης $Y \in R^{n \times n}$ του οποίου το μέγεθος είναι το ίδιο με το πρόβλημα:

$$Y_{j,k} = \begin{cases} y_{ind(j,k)}, & \text{αν } \exists (j,k) \in E, \\ 0, & \text{διαφορετικά}, \end{cases} \tag{1.1.8}$$

όπου $ind(.,.)$ είναι μια συνάρτηση που απεικονίζει μια ακμή σε έναν ακέραιο δείκτη. Στη συνέχεια, η απώλεια περιορισμών σχεδιάζεται ως εξής:

$$L_1 = ||\mathbf{1} - sum_r(Y)||_2 + ||\mathbf{1} - sum_r(Y^T)||_2, \tag{1.1.9}$$

$$L_2 = ||\mathbf{1} - norm_r(Y)||_2 + ||\mathbf{1} - norm_r(Y^T)||_2, \tag{1.1.10}$$

$$L_C = L_1 + L_2 \tag{1.1.11}$$

Στις παραπάνω εξισώσεις, το $\mathbf{1}$ είναι ένα διάνυσμα με όλα τα στοιχεία του ίσα με 1, το $sum_r(.)$ αθροίζει τις τιμές στον προβλεπόμενο πίνακα ανάθεσης κατά μήκος της γραμμής και το $norm_r(.)$ επιστρέφει ένα διάνυσμα στο οποίο κάθε στοιχείο είναι η 2-νόρμα του αντίστοιχου διανύσματος γραμμής.

Τέλος, η απώλεια δυαδικής ταξινόμησης και η απώλεια περιορισμων συνδυάζονται για να προκύψει η *συνάρτηση απωλειών* που καθοδηγεί τη διαδικασία εκπαίδευσης ως εξής:

$$L = L_A + \alpha L_C \tag{1.1.12}$$

όπου $\alpha > 0$ σταθμίζει τον βαθμό αυστηρότητας των περιορισμών αντιστοίχισης ένα προς ένα που επιβάλλονται.

1.2 Προτεινόμενη Μέθοδος

Σε αυτό το κεφάλαιο προτείνουμε τη μέθοδο με την οποία θα αντιμετωπίσουμε το πρόβλημα των αντιφατικών εξηγήσεων. Επικεντρωνόμαστε σε αντιπραγματικές παρεμβάσεις σε επίπεδο λέξης για να ελέγξουμε τη συμπεριφορά των ταξινομητών κειμένου όταν αντικαθίστανται διαφορετικές λέξεις. Η πρότασή μας περιστρέφεται γύρω από την τοποθέτηση όλων των υλοποιούμενων παρεμβάσεων κάτω από ένα πλαίσιο το οποίο παρουσιάζει τα ακόλουθα χαρακτηριστικά όσον αφορά τις παρεμβάσεις:

- Βελτιστοποίηση: Οι αντικαταστάσεις θα πρέπει να είναι βέλτιστες -ή κατά προσέγγιση βέλτιστες-, δεδομένης κάποιας έννοιας της σημασιολογικής απόστασης.
- Ελεγξιμότητα: σε κάθε δείγμα δεδομένων θα πρέπει να αντικαθίσταται τουλάχιστον ένα σημασιολογικό στοιχείο εισόδου.
- Αποδοτικότητα: η βέλτιστη λύση θα πρέπει να επιτυγχάνεται χωρίς τη χρήση τεχνικών εξαντλητικής αναζήτησης μεταξύ εναλλακτικών αντικαταστάσεων.

Προσεγγίζουμε αυτές τις απαιτήσεις θεωρώντας τις αντιφατικές παρεμβάσεις ως ένα πρόβλημα συνδυαστικής βελτιστοποίησης, επιλύσιμο μέσω αλγορίθμων ανάθεσης γράφων από τη θεωρία γράφων [111]. Για να βελτιώσουμε περαιτέρω τη μέθοδό μας, εξετάζουμε τη χρήση νευρωνικών δικτύων γράφων (ΝΔΓ) [108] ως ταχύτερο προσεγγιστικό υποκατάστατο αυτών των αλγορίθμων [114].

Η μέθοδος που προτείνουμε μπορεί να εφαρμοστεί τόσο σε σενάρια που αφορούν συγκεκριμένα μοντέλα όσο και σε σενάρια γενικού σκοπού, καθώς δεν υπάρχει αυστηρή εξάρτηση από την αλλαγή της εξόδου του ταξινομητή κειμένου. Αυτή η ιδιότητα επιτρέπει τη χρήση των παραγόμενων επεξεργασιών για διαφορετικές εργασίες εκτός από την αλλαγή εξόδου, όπως η σημασιολογική ομοιότητα [54] ή η μη στοχευμένη παραγωγή [106]- παρόλα αυτά, στην παρούσα διατριβή, εστιάζουμε σε εργασίες ταξινόμησης για άμεση σύγκριση με προηγούμενες εργασίες. Για το σκοπό αυτό, συγκρίνουμε την προσέγγισή μας με δύο SoTA συντάκτες [106, 80] χρησιμοποιώντας κατάλληλες μετρικές για την αλλαγή εξόδου, την ευγλωττία και τη σημασιολογική εγγύτητα.

Αρχικά επισημαίνουμε τις κύριες συνεισφορές της παρούσας διατριβής και στη συνέχεια εξηγούμε λεπτομερώς την προτεινόμενη μέθοδο.

1.2.1 Συνεισφορές

Συνοψίζοντας, οι συνεισφορές μας είναι:

- Επιβάλλουμε τη βελτιστοποίηση και την ελέγξιμότητα των λεξικών παρεμβάσεων μετατρέποντάς τες στην εύρεση της βέλτιστης ανάθεσης μεταξύ των κόμβων ενος διγράφου.

- Επιταχύνουμε τη διαδικασία ανάθεσης εκπαιδεύοντας τα ΝΔΓ σε αυτές τις ντετερμινιστικές αντιστοιχίσεις, επιτυγχάνοντας τελικά προηγμένη αποδοτικότητα.
- Επεκτείνουμε ένα υπάρχον πλαίσιο προκειμένου να προσφέρουμε ένα ΝΔΓ που επιλύει το RLAP - εξ όσων γνωρίζουμε, καμία προηγούμενη εργασία δεν έχει αξιοποιήσει ΝΔΓ για την επίλυση του RLAP
- Ο εξαιρετικά αποδοτικός αντιθετικός συντάκτης μαύρου κουτιού μας παρέχει σταθερά επιδόσεις SoTA σε σύγκριση με τις υπάρχουσες μεθόδους λευκού και μαύρου κουτιού σε δύο διαφορετικά σύνολα δεδομένων και σε τέσσερις διαφορετικές μετρικές. Είναι αξιοσημείωτο ότι επιτυγχάνει αυτά τα αποτελέσματα σε λιγότερο από 2% και 20% του χρόνου που απαιτούν οι δύο ανταγωνιστές του, αποδεικνύοντας τόσο ανώτερη αποτελεσματικότητα όσο και αποδοτικότητα.
- Η ευελιξία του προτεινόμενου συντάκτη αποδεικνύεται σε διάφορα σενάρια, καθώς είναι σε θέση να βελτιστοποιηθεί προς μια συγκεκριμένη μετρική ή να δημιουργήσει επεξεργασίες γενικού σκοπού.

1.2.2 Προτεινόμενο Σύστημα

Η ροή της μεθόδου μας, όπως φαίνεται στο Σχήμα 1.2.1) αποτελείται από τρία στάδια. Ένα σύνολο δεδομένων κειμένου D χρησιμεύει ως είσοδος στο σύστημά μας. Στο πρώτο στάδιο, οι λέξεις εξάγονται από το D, με βάση το μέρος του λόγου (POS) στο οποίο ανήκουν, και χρησιμοποιούνται ως σύνολο αρχικών κόμβων S. Το σύνολο-στόχος T είναι είτε ένα αντίγραφο του S, είτε παράγεται από μια εξωτερική λεξιλογική πηγή, όπως το WordNet [61], που περιέχει όλες τις πιθανές υποψήφιες αντικαταστάσεις των λέξεων (κόμβων) της πηγής. Τα σύνολα S και T σχηματίζουν ένα διγράφο G, με τα βάρη των ακμών τους να αντικατοπτρίζουν την ομοιότητα των λέξεων. Στο δεύτερο στάδιο, περνάμε τον διγράφο G ως είσοδο σε ένα προεκπαιδευμένο ΝΔΓ το οποίο εξάγει μια προσεγγιστική λύση του RLAP, με τη μορφή μιας λίστας υποψήφιων ζευγών λέξεων. Κάθε ζεύγος λέξεων, αποτελείται από τη λέξη-πηγή $s_i \in S$ και την υπολογισμένη αντικατάστασή της $t_i \in T$. Στο τρίτο και τελευταίο στάδιο, αξιοποιούμε την αναζήτηση δέσμης [105] για να καθορίσουμε τις τελικές αλλαγές. Η αναζήτηση δέσμης χρησιμοποιεί μια *ευριστική συνάρτηση* για να επιλέξει τις πιο κατάλληλες αντικαταστάσεις από αυτές που επιστρέφει το ΝΔΓ. Οι επιλεγμένες λέξεις από το S αντικαθίστανται στη συνέχεια με το αντίστοιχο ζεύγος τους από το T, παράγοντας ένα αντιθετικό σύνολο δεδομένων D^*.

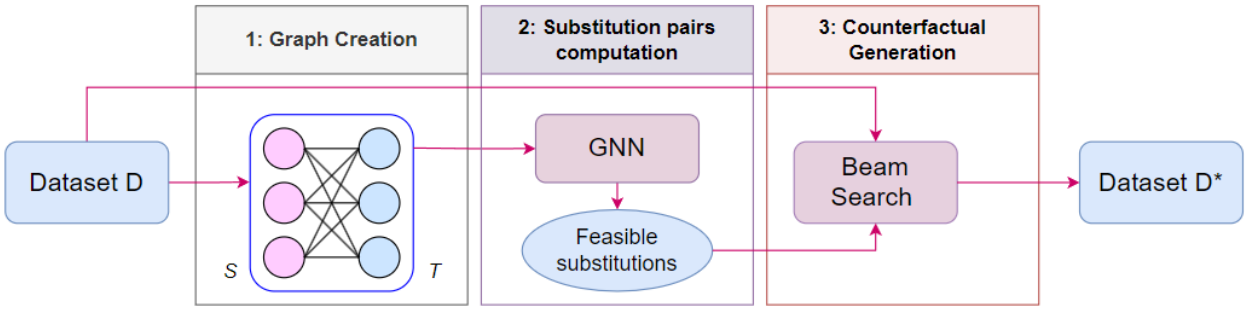


Figure 1.2.1: Η ροή της μεθόδου μας.

Δημιουργία Γράφου

Κατά την κατασκευή του διγράφου G, οι λέξεις εξάγονται από το αρχικό D με βάση το POS τους. Για να ελέγξουμε πόσο καλά γενικεύεται το πλαίσιό μας, χρησιμοποιούμε εξόρυξη λέξεων τόσο συγκεκριμένου POS όσο και γενικού. Η πρώτη σημαίνει ότι επιλέγουμε να αλλάξουμε δυνητικά μόνο τις λέξεις που ανήκουν σε ένα συγκεκριμένο μέρος του λόγου(π.χ. επίθετα, ουσιαστικά, ρήματα κ.λπ.), ενώ η δεύτερη σημαίνει ότι λαμβάνουμε υπόψην όλες τις λέξεις, ανεξάρτητα από το μέρος του λόγου στο οποίο ανήκουν. Για τα βάρη των ακμών, χρησιμοποιούμε δύο διαφορετικές προσεγγίσεις, η καθεμία με διαφορετική διαφάνεια. Για την πρώτη, υιοθετούμε μια πλήρως διαφανή προσέγγιση, υπολογίζοντας τις αποστάσεις χρησιμοποιώντας μια λεξική ιεραρχία: το βάρος μιας ακμής που συνδέει δύο λέξεις καθορίζεται από την τιμή της ομοιότητάς τους, όπως ορίζεται στο WordNet.[1] Στη δεύτερη περίπτωση, για να δημιουργήσουμε διανύσματα ενσωμάτωσης λέξεων εφαρμόζουμε

[1] https://www.nltk.org/howto/wordnet.html.

διάφορα Μεγάλα Γλωσσικά Μοντέλα, και πιο συγκεκριμένα τα AnglE[2] [48, 87], GISTEmbed[3] [88], GinaAI[4] [63] και MUG[5] - στη συνέχεια, θέτουμε το βάρος της ακμής ίσο με την ομοιότητα συνημίτονου των δύο διανυσμάτων ενσωμάτωσης λέξεων. Δεδομένου ότι η χαμηλότερη ομοιότητα σχετίζεται με ελαφρύτερες ακμές, δηλαδή με πιο κατάλληλους υποψήφιους για το M, οι επιλεγμένες λέξεις προς αντικατάσταση θα σχηματίζουν *αντιφατικά ζεύγη λέξεων*. Προκειμένου να διατηρηθεί η σύνταξη στην περίπτωση που δεν έχουμε κάποιον περιορισμό για το μέρος του λόγου, επιβάλλουμε αντικαταστάσεις αποκλειστικά μεταξύ λέξεων με ίδιο POS: έτσι, πειραματιζόμαστε με έναν μηχανισμό φιλτραρίσματος ακμών, ο οποίος θέτει ένα προκαθορισμένο μεγάλο βάρος στις ακμές, ∼10 φορές μεγαλύτερο από τα κανονικά βάρη ακμών που έχουν υπολογιστεί με βάση την ομοιότητα διαδρομών του WordNet ή την ομοιότητα συνημίτονου των διναυσμάτων ενσωματώσεων. Με αυτόν τον τρόπο, αποφεύγουμε τις περιπτώσεις όπου ένα μέρος του λόγου αντικαθίσταται από μια λέξη που ανήκει σε κάποιο διαφορετικό, καθώς μια σημαντικά βαρύτερη ακμή δεν μπορεί να επιλεγεί για να συμμετάσχει στο ελάχιστο σύνολο ακμών ταιριάσματος M. Στην περίπτωση που εξετάζουμε αντικαταστάσεις ενός συγκεκριμένου μέρους του λόγου, ο μηχανισμός αυτός είναι περιττός, αφού όλες οι λέξεις ανήκουν στο ίδιο

Υπολογισμός ζευγών αντικατάστασης

Για τα κατάλληλα ζεύγη αντικατάστασης πρέπει να λύσουμε το RLAP στον διγράφο G. Όπως είδαμε προηγουμένος (Ενότητα 1.1.4οι παραδοσιακές ντετερμινιστικές μέθοδοι το επιτυγχάνουν αυτό σε $O(mn \log m)$. Ενώ αυτές οι μέθοδοι παρέχουν τη βέλτιστη λύση, στερούνται ταχύτητας καθώς το μέγεθος του συνόλου δεδομένων, και συνεπώς το μέγεθος του γράφου, μεγαλώνει. Σε μια προσπάθεια να παράγουμε ζεύγη αντικαταστάσεων σε *σταθερό* χρόνο ανεξάρτητα από το μέγεθος του συνόλου δεδομένων, χρησιμοποιούμε ένα μοντέλο ΝΔΓ, το οποίο προσεγγίζει τη βέλτιστη λύση που βρίσκουν οι ντετερμινιστικοί αλγόριθμοι, ενώ επιταχύνει σημαντικά τη διαδικασία. Όσον αφορά το ΝΔΓ, βελτισοποιήσαμε ως προς το RLAP, το μοντέλο που περιγράφεται στην ενότητα 1.1.4, χρησιμοποιώντας την ακόλουθη διαδικασία:

Αρχικά, δημιουργείται ένα συνθετικό σύνολο δεδομένων που αποτελείται από M δείγματα[6]. Κάθε δείγμα αποτελείται από έναν πίνακα κόστους C στον οποίο τα στοιχεία παράγονται από μια ομοιόμορφη κατανομή στο $(0, 1)$ και την αντίστοιχη βέλτιστη λύση ανάθεσης η οποία λαμβάνεται από τον ουγγρικό αλγόριθμο [42]. Θεωρούμε το RLAP ως ένα πρόβλημα δυαδικής ταξινόμησης και χωρίζουμνε τα στοιχεία του πίνακα ανάθεσης βασικής αλήθειας Y^{gt} [7] σε θετικές ετικέτες και αρνητικές. Για τη Συνάρτηση Απώλειας, χρησιμοποιούμε την Εξίσωση 1.1.12 με $\alpha = 0$, η οποία είναι ισοδύναμη με τη χρήση μόνο της *Ισορροπημένης Διασταυρούμενης Εντροπίας*. Ο λόγος γι' αυτό, είναι το γεγονός ότι στη δική μας έκδοση του RLAP, ο πίνακας κόστους C έχει διαστάσεις $n \times m$, όπου $n \leq m$ και επομένως δεν είναι δυνατή η ακριβής αντιστοίχιση ένα προς ένα. Αφαιρώντας το τμήμα της συνάρτησης απωλειών που αντιστοιχεί σε αυτόν τον περιορισμό, επιχειρούμε να τον αμβλύνουμε σημαντικά, βελτιστοποιώντας έτσι το μοντέλο ΝΔΓ προς τη λύση του RLAP αντί για το LSAP. Όπως και στο [50], η εκπαίδευση διαρκεί συνολικά 20 εποχές, όπου ο ρυθμός μάθησης ορίζεται αρχικά σε 0.003 και μειώνεται κατά 5% μετά από κάθε 5 εποχές.

Χρησιμοποιώντας το προηγούμενο ΝΔΓ, η **αποδοτικότητα** είναι εγγυημένη. Επιλύοντας το πρόβλημα με τον περιορισμό του ελάχιστου $\sum w_e$, βρίσκουμε όλα τα πιο *ανόμοια* $s \to t$ ζεύγη, επιτυγχάνοντας *προσσεγγιστικά* **βελτιστότητα** της αντικατάστασης εννοιών εντός του G και τελικά παράγοντας αντιθετικά ζεύγη αντικατάστασης. Ταυτόχρονα, η **ελεγξιμότητα** είναι *μερικώς* εξασφαλισμένη, αφού ο γράφος G είναι πυκνός (επομένως δεν υπάρχουν ασύνδετοι κόμβοι s) και $|S| \leq |T|$, αφού το T είναι είτε αντίγραφο του S είτε παράγεται με βάση το S χρησιμοποιώντας αντώνυμα από το WordNet (σε κάθε λέξη μπορεί να αντιστοιχούν περισσότερα από ένα αντώνυμα). Σημειώστε εδώ, ότι χρησιμοποιούμε τη λέξη *"μερικώς"*, καθώς υπάρχει συμβιβασμός μεταξύ *ελεγξιμότητας* και *ελάχιστότητας* [8], το οποίο προκύπτει από τη χρήση της αναζήτησης με δέσμη κατά τη διάρκεια της δημιουργίας αντιφατικών παρεμβάσεων. Στην πράξη, υπάρχουν επίσης μερικές εξαιρέσεις στην ελέγξιμότητα, εάν μια έννοια-πηγή δεν υπάρχει στο WordNet.

[2] mixedbread-ai/mxbai-embed-large-v1
[3] avsolatorio/GIST-Embedding-v0
[4] https://jina.ai/embeddings/
[5] Labib11/MUG-B-1.6
[6] Κάθε δείγμα αναπαριστά ένα διγράφο με βάρη στις ακμές.
[7] Y^{gt} είναι ένας πίνακας όπου το στοιχείο y_{ij}^{gt} είναι 1 αν η ακμή που συνδέει τους κόμβους i και j **belongs** στην ελάχιστη αντιστοίχιση, αλλιώς είναι -1.
[8] Η ελαχιστότητα εδώ αναφέρεται στον αριθμό των λέξεων που αλλάζουν.

Δημιουργία Αντιθετικών Παρεμβάσεων

Ως αποτέλεσμα της επίλυσης του RLAP, επιστρέφεται ένα ταίριασμα $M \subset E$, το οποίο υποδεικνύει τις βέλτιστες αντικαταστάσεις σε n έννοιες πηγής. Συμβολίζουμε ως $W_n^M \subset W$ το συνολικό βάρος του M που περιέχει n αρχικές έννοιες. Δεδομένης αυτής της αντιστοίχισης, η αναζήτηση με δέσμη επιλέγει ποιες εννοιολογικές αντικαταστάσεις από το M θα εκτελεστούν *πραγματικά* στο D. Αυτή η διαδικασία επιλογής είναι απαραίτητη καθώς επιθυμούμε οι αλλαγές να είναι *ελάχιστες* όσον αφορά τον αριθμό των λέξεων που τροποποιούνται ανά περίπτωση, διαταράσσοντας μόνο μικρά τμήματα της εισόδου, μια ιδιότητα που έχει υποστηριχθεί ότι κάνει τις εξηγήσεις πιο κατανοητές [1, 62]. Σε αυτό το πλαίσιο, θέσαμε επίσης ένα ανώτατο όριο αντικαταστάσεων σε κάθε περίπτωση κειμένου, πειραματιζόμενοι τόσο με έναν σταθερό όσο και με έναν δυναμικά καθορισμένο αριθμό. Στη δεύτερη περίπτωση, για κάθε παράδειγμα, το ανώτατο όριο είναι ίσο με το 20% του συνολικού αριθμού των λέξεων που περιέχει. Σταματάμε την αναζήτηση όταν η πρόβλεψη του μοντέλου ανατραπεί ή όταν επιτευχθεί το ανώτατο όριο, διατηρώντας έτσι τον αριθμό των αντικαταστάσεων σε χαμηλά επίπεδα.

1.3 Πειραματικό Μέρος

Προκειμένου να αξιολογήσουμε την προτεινόμενη μέθοδο και να τη συγκρίνουμε με άλλες τεχνικές και συστήματα, πραγματοποιήσαμε πολλαπλά πειράματα σε δύο προβλήματα ΕΦΓ. Σε αυτή την ενότητα θα παρουσιαστούν κάποιες προκαταρκτικές πληροφορίες σχετικά με τα σύνολα δεδομένων, τα μοντέλα ταξινομητών και τους ανταγωνιστικούς συντάκτες, καθώς και τις μετρικές που χρησιμοποιήθηκαν στη διαδικασία αξιολόγησης.

Έχοντας θέσει τα βασικά στοιχεία, θα αναλύσουμε τον τρόπο με τον οποίο πειραματιστήκαμε με τα διάφορα μέρη του συστήματός μας, καθώς και τις αντισταθμίσεις που πρέπει να ληφθούν υπόψη και τέλος θα παρουσιάσουμε τα αποτελέσματα της αξιολόγησης. Εκτός από τα ποσοτικά αποτελέσματα, θα παρουσιάσουμε παραδείγματα των επεξεργασμένων κειμένων που βοηθούν στη συνολική κατανόηση της μεθόδου μας και των δυνατοτήτων της.

1.3.1 Σύνολα Δεδομένων και Μετρικές

Σύνολα Δεδομένων

Αξιολογούμε το σύστημά μας και το συγκρίνουμε με άλλους συντάκτες από τη βιβλιογραφία, σε δύο σύνολα δεδομένων στην αγγλική γλώσσα: IMDB, το οποίο περιέχει κριτικές ταινιών και χρησιμοποιείται για δυαδική ταξινόμηση συναισθήματος [58] και μια έκδοση 6 κλάσεων των 20 Newsgroups που χρησιμοποιείται για θεματική ταξινόμηση [44]. Λόγω των υψηλών υπολογιστικών απαιτήσεων των συγκρινόμενων μεθόδων, πήραμε δείγμα 1K περιπτώσεων από κάθε σύνολο δεδομένων. Η εκτέλεση του MiCE σε μόλις 1K δείγματα απαιτούσε πάνω από 47 ώρες (βλ. Πίνακα 1.1), καθιστώντας τα πειράματα σε ολόκληρο το σύνολο δεδομένων μη πρακτικά. Επιλέξαμε το διπλάσιο μέγεθος δείγματος από αυτό που χρησιμοποιήθηκε σε παρόμοιες μελέτες που συνέκριναν τις ίδιες μεθόδους στα ίδια σύνολα δεδομένων [21].

Το σύνολο δεδομένων IMDB Reviews είναι ένα ευρέως χρησιμοποιούμενο σύνολο δεδομένων για εργασίες επεξεργασίας φυσικής γλώσσας (ΕΦΓ), ιδίως για την ανάλυση συναισθήματος. Αποτελείται από 50000 κριτικές ταινιών που λαμβάνονται από τη βάση δεδομένων ταινιών του Διαδικτύου (IMDB), με ίση κατανομή μεταξύ θετικών και αρνητικών ετικετών συναισθήματος. Το σύνολο δεδομένων χωρίζεται σε δύο ίσα μέρη: 25000 κριτικές προορίζονται για εκπαίδευση και οι υπόλοιπες 25000 για δοκιμή. Στα πειράματά μας, πήραμε ένα αντιπροσωπευτικό δείγμα από το σύνολο *δοκιμών*, καθώς δεν στοχεύουμε στην εκπαίδευση ενός νέου μοντέλου, αλλά στην επεξήγηση των υπαρχόντων. Επομένως, δεδομένου ότι τα μοντέλα που χρησιμοποιούμε έχουν ήδη εκπαιδευτεί στο σύνολο εκπαίδευσης, είναι λογικό να αποφύγουμε τη χρήση περιπτώσεων που έχουν ήδη δει και ενδεχομένως απομνημονεύσει.

Κάθε κριτική στο σύνολο δεδομένων συνδέεται με μια δυαδική ετικέτα συναισθήματος: *positive* για κριτικές με βαθμολογία 7 ή μεγαλύτερη (από 10) και *negative* για κριτικές με βαθμολογία 4 ή μικρότερη. Κριτικές με βαθμολογία 5 ή 6 δεν περιλαμβάνονται στο σύνολο δεδομένων για να διασφαλιστεί η σαφής διάκριση μεταξύ θετικών και αρνητικών συναισθημάτων. Τα δεδομένα κειμένου στις κριτικές ποικίλλουν σε μήκος και επισημότητα, αποτυπώνοντας ένα ευρύ φάσμα γλωσσικών στυλ, από την ανεπίσημη έως την επίσημη γραφή. Στην εργασία μας, πριν τα χρησιμοποιήσουμε, τα καθαρίσαμε, αφαιρώντας ειδικούς χαρακτήρες και συνεχόμενα κενά.

Το σύνολο δεδομένων 20 Newsgroups είναι ένα άλλο δημοφιλές σύνολο δεδομένων αναφοράς στον τομέα του NLP και της ταξινόμησης κειμένου. Περιέχει περίπου 20000 έγγραφα που κατανέμονται σε 20 διαφορετικά

θέματα, όπως ο αθλητισμός, η πολιτική, η τεχνολογία ή η επιστήμη. Το σύνολο δεδομένων έχει σχεδιαστεί για την υποστήριξη πειραμάτων σε εργασίες ταξινόμησης κειμένου, ιδίως στο πλαίσιο της ταξινόμησης πολλαπλών κλάσεων.

Στην παρούσα διατριβή, χρησιμοποιούμε την έκδοση 6 κλάσεων του αρχικού συνόλου δεδομένων, η οποία είναι μια συγκεκριμένη διαμόρφωση που ενοποιεί τα 20 θέματα σε 6 ευρύτερες, θεματικές κατηγορίες. Αυτές οι έξι κατηγορίες ομαδοποιούν συναφή θέματα, μειώνοντας την πολυπλοκότητα του προβλήματος ταξινόμησης, διατηρώντας παράλληλα την ποικιλομορφία του περιεχομένου. Όπως και στην περίπτωση του IMDB, πριν χρησιμοποιήσουμε τα δεδομένα, τα καθαρίσαμε αφαιρώντας κενές γραμμές, ειδικούς χαρακτήρες και συνεχόμενα κενά.

Μετρικές

Για να αξιολογήσουμε τις επιδόσεις των διαφόρων συντακτών, αντλούμε έμπνευση από το MiCE [80] και μετράμε τις εξής ιδιότητες:

- **Ποσοστό Αλλαγής Ετικέτας / Flip-Rate** - είναι μια εξαιρετικά δημοφιλής μετρική και επίσης αυτή που προσπαθούν να μεγιστοποιήσουν οι περισσότεροι αντιφατικοί συντάκτες. Πρόκειται για το ποσοστό των περιπτώσεων για τις οποίες μια παρέμβαση οδηγεί σε διαφορετική πρόβλεψη του μοντέλου (αλλαγή ετικέτας). Όσο πιο υψηλό είναι το flip-rate τόσο πιο "αποτελεσματικές" είναι οι παρεμβάσεις που έγιναν.
- **Ελαχιστότητα / Minimality** - τυπικά μετράται με τη χρήση της *απόστασης Levenshtein*. Η απόσταση Levenshtein, γνωστή και ως απόσταση επεξεργασίας, είναι μια μετρική που χρησιμοποιείται για τη μέτρηση της διαφοράς μεταξύ δύο συμβολοσειρών. Αντιπροσωπεύει τον ελάχιστο αριθμό αλλαγών σε επίπεδο ενός χαρακτήρα που απαιτούνται για να μετατραπεί μια συμβολοσειρά σε μια άλλη, όπου οι επιτρεπόμενες αλλαγές είναι η εισαγωγή, η διαγραφή ή η αντικατάσταση ενός χαρακτήρα. Στα πειράματά μας, χρησιμοποιήσαμε την έκδοση αυτής της απόστασης σε επίπεδο λέξης, η οποία αντί για χαρακτήρες, μετράει μεμονωμένες αλλαγές *λέξεων*.
- **Εγγύτητα / Closeness** - αντιπροσωπεύει τη σημασιολογική ομοιότητα μεταξύ της αρχικής και της επεξεργασμένης εισόδου, η οποία μετράται με το BERTScore [117]. Όσο υψηλότερο είναι το BERTScore, τόσο πιο κοντά είναι σημασιολογικά οι δύο προτάσεις.
- **Ευγλωττία / Fluency** - Η ευγλωττία είναι ένα μέτρο του πόσο παρόμοια κατανεμημένη είναι η επεξεργασμένη είσοδος σε σύγκριση με την αρχική, όπου μικρότερες τιμές υποδηλώνουν πιο εύγλωττες παρεμβάσεις. Για την αξιολόγηση της ευγλωττίας, χρησιμοποιούμε ένα προ-εκπαιδευμένο μοντέλο T5-BASE [77].

Για να τονίσουμε τη σημαντική επιτάχυνση που προσφέρει η μέθοδός μας, αναφέρουμε επίσης τους *χρόνους εκτέλεσης* για κάθε συντάκτη.

1.3.2 Ταξινομητές και Ανταγωνιζόμενοι Συντάκτες

Ταξινομητές Κειμένου

Για να μετρήσουμε την αποτελεσματικότητα των παραγόμενων παρεμβάσεων, δοκιμάζουμε τους διάφορους συντάκτες σε δύο ταξινομητές κειμένου. Χρησιμοποιούμε τα ίδια μοντέλα πρόβλεψης με το MiCE [80] σε κάθε σύνολο δεδομένων (IMDB Reviews και 20 Newsgroups). Και τα δύο αυτά μοντέλα βασίζονται στο RoBERTa$_{LARGE}$ [51], και είναι ειδικά εκπαιδευμένα προς τα αντίστοιχα σύνολα δεδομένων. Το πρώτο μοντέλο, είναι ένας δυαδικός ταξινομητής του οποίου η έξοδος είναι είτε 0 είτε 1, με το 0 να δηλώνει *αρνητικές* κριτικές και το 1 να δηλώνει *θετικές*. Το δεύτερο μοντέλο, είναι ειδικά εκπαιδευμένο για την έκδοση 6 κατηγοριών του συνόλου δεδομένων 20 Newsgroups και η έξοδός του κυμαίνεται από 0 έως 5, με κάθε ακέραια τιμή να αντιπροσωπεύει μια κατηγορία από αυτές.

Ανταγωνιζόμενοι Συντάκτες

Συγκρίνουμε το σύστημά μας με δύο SoTA συντάκτες, συγκεκριμένα τους Polyjuice [106] και MiCE [80]. Ο πρώτος είναι ένας αντιθετικός συντάκτης γενικού σκοπού, ενώ ο δεύτερος είναι ένας συντάκτης ειδικά βελτιστοποιημένος για το *ποσοστό αλλαγής ετικέτας* και την *ελαχιστότητα*. Επιλέξαμε αυτούς τους δύο συντάκτες, λόγω

του γεγονότος ότι η μέθοδός μας μπορεί να χρησιμοποιηθεί τόσο ως συντάκτης γενικού σκοπού όσο και ειδικός συντάκτης για κάποιο συγκεκριμένο πρόβλημα, όντας έτσι συγκρίσιμος και με τους δύο.

1.3.3 Περιγραφή Πειραμάτων

Παραλλαγές Συντάκτη

Ο συντάκτης μας υλοποιήθηκε χρησιμοποιώντας πολλές διαφορετικές βιβλιοθήκες της Python. Για το πρώτο στάδιο (κατασκευή γράφου) χρησιμοποιήσαμε τις *Spacy*, *NLTK* και *sentence-transformers*, ενώ για το ΝΔΓ, την *Pytorch Geometric* [20]. Δεδομένου ότι η μέθοδός μας είναι ιδιαίτερα προσαρμόσιμη, πειραματιστήκαμε με διαφορετικές ρυθμίσεις και προσεγγίσεις. Πρέπει επίσης να σημειώσουμε ότι όλα τα πειράματα εκτελέστηκαν στο ίδιο σύστημα, το οποίο αποτελείται από μια *κάρτα γραφικών 16 GB*, ένα*ν συντάκτη Intel i7* και *16 GB μνήμης RAM*.

Προκειμένου να διατηρηθεί το POS σε κάθε αντικατάσταση, εφαρμόζουμε έναν μηχανισμό ποινής (φιλτράρισμα) κατά τον υπολογισμό των βαρών των ακμών του γραφήματος. Αυτός ο μηχανισμός αποδίδει ένα βάρος περίπου 10× μεγαλύτερο από τα *κανονικά* βάρη (όπως ορίζονται από την ομοιότητα διαδρομών του WordNet ή την ομοιότητα συνημίτονου των λεξικών διανυσμάτων ενσωμάτωσης), σε κάθε ακμή που συνδέει λέξεις με διαφορετικό POS. Με αυτόν τον τρόπο, δεδομένου ότι το σύστημά μας προσπαθεί να βρει ένα *ελάχιστο ταίριασμα βάρους*, ακμές με μεγάλα βάρη είναι σχεδόν αδύνατο να επιλεγούν και επομένως οι αντικαταστάσεις που περιλαμβάνουν διαφορετικα μέρη του λόγου έχουν χαμηλή πιθανότητα εμφάνισης. Στην επόμενη ενότητα αναφέρουμε ευρήματα με και χωρίς αυτόν τον μηχανισμό φιλτραρίσματος ακμών.

Διερευνούμε επίσης την επίδραση της χρήσης της ομοιότητας συνημίτονου των διανυσμάτων ενσωματώσεων αντί της ομοιότητας διαδρομής του WordNet μεταξύ δύο λέξεων, κατά τον υπολογισμό του βάρους μιας συγκεκριμένης ακμής στον διγράφο G. Από τη μία πλευρά, οι ντετερμινιστικές ιεραρχίες παρέχουν περισσότερες *εξηγήσιμες* σχέσεις μεταξύ των εννοιών, δικαιολογώντας πλήρως τις αιτιώδεις διαδρομές των αντικαταστάσεων. Από την άλλη πλευρά, τα πρόσφατα αναδυόμενα μοντέλα ενσωμάτωσης μπορούν να αποτυπώσουν καλύτερα τη σχέση και την ομοιότητα δύο λέξεων, σε σύγκριση με το WordNet. Για να διατηρήσουμε το σύστημά μας σχετικά ελαφρύ, χρησιμιοποιούμε τα τέσσερα κορυφαία μοντέλα με τις καλύτερες επιδόσεις που συμμετείχαν στον διαγωνισμό MTEB [68] και των οποίων το μέγεθος δεν υπερβαίνει το *1.25 GB*. Τα μοντέλα με αυτό το μέγεθος κατέλαβαν τις πρώτες θέσεις στον διαγωνισμό και οποιαδήποτε αύξηση του μεγέθους του μοντέλου δεν οδήγησε σε σημαντική βελτίωση της απόδοσης.

Σε μια προσπάθεια να αξιολογήσουμε την ικανότητα του συντάκτη μας να διακρίνει ποιο μέρος του λόγου έχει μεγαλύτερη επιρροή σε ένα συγκεκριμένο σύνολο δεδομένων κατά την αντικατάσταση σχετικών λέξεων, επιβάλλουμε περιορισμούς σχετικά με το ποια POS θα πρέπει να είναι υποψήφια για αντικατάσταση και συγκρίνουμε τα αποτελέσματα με μια έκδοση του πλαισίου μας χωρίς περιορισμούς POS. Το σύνολο δεδομένων IMDB χρησιμοποιείται για την ταξινόμηση συναισθήματος, και ως εκ τούτου τα επίθετα και τα επιρρήματα υπαγορεύουν κυρίως την ετικέτα (συναίσθημα) για κάθε περίπτωση [5]. Έχοντας αυτό κατά νου, περιορίζουμε τον συντάκτη μας να αλλάζει μόνο αυτά τα δύο POS. Τα Newsgroups είναι ένα σύνολο δεδομένων που ανήκει στην κατηγορία ταξινόμησης θεμάτων. Δεδομένου ότι ένα θέμα συνάγεται από την εξέταση των ουσιαστικών σε ένα κείμενο, δίνουμε εντολή στον συντάκτη να λάβει υπόψη του μόνο αυτά.

Για να διατηρηθεί ο αριθμός των επεξεργασιών σχετικά χαμηλός, απαιτείται ένας τρόπος περιορισμού του αριθμού των αντικαταστάσεων ανά περίπτωση δεδομένων, αποδεχόμενος μια πιθανή πτώση του ποσοστού αλλαγής ετικέτας. Για το λόγο αυτό, χρησιμοποιούμε δύο διαφορετικές προσεγγίσεις. Στην πρώτη, επιβάλλουμε έναν στατικό αριθμό μέγιστων επιτρεπόμενων αντικαταστάσεων για κάθε είσοδο, ανεξάρτητα από το μήκος της- μετά από πειραματισμό, ο καλύτερος αριθμός βρέθηκε να είναι ίσος με 10. Στη δεύτερη προσέγγιση, υπολογίζουμε δυναμικά το βέλτιστο ανώτατο όριο (ή κατώφλι) των αντικαταστάσεων με βάση το συνολικό αριθμό των λέξεων του κειμένου. Μετά από διάφορες προσπάθειες, καταλήγουμε να ορίσουμε το όριο αυτό στο 20% του συνολικού αριθμού των λέξεων, το οποίο ουσιαστικά σημαίνει ότι αλλάζουμε κατά μέσο όρο μία λέξη σε κάθε πέντάδα λέξεων.

Δεδομένου ότι η επιλογή των επιλέξιμων αντικαταστάσεων είναι μια διαδικασία γενικού σκοπού, εξετάζουμε τη συμπεριφορά του συντάκτη μας όταν βελτιστοποιείται για σενάρια εναλλαγής ετικετών. Αυτή η βελτιστοποίηση γίνεται με την αλλαγή της ευριστικής συνάρτησης της αναζήτησης δέσμης στο τελευταίο στάδιο του συστήματός μας (βλ. Σχήμα 1.2.1). Για τις επεξεργασίες γενικού σκοπού, η συνάρτηση αυτή είναι η μετρική για την

ευγλωττία που συζητήθηκε στην ενότητα 1.3.1, η οποία βοηθά στην παραγωγή νοηματικά ορθών παρεμβάσεων. Για την εναλλαγή ετικέτας, χρησιμοποιούμε την *αντιθετική πιθανότητα / contrastive probability*, η οποία μετράει την αλλαγή στην πρόβλεψη του μοντέλου για την αρχική ετικέτα, για να καθορίσουμε τις καλύτερες επεξεργασίες (βλ. *GNN w. contrastive* στον Πίνακα 1.1). Τέλος, χρησιμοποιούμε επίσης το μέσο όρο της ευγλωττίας και της αντιθετικής πιθανότητας ως ευριστική συνάρτηση, η οποία οδηγεί σε ευχερείς επεξεργασίες με υψηλό ποσοστό αναστροφής (βλ. *GNN w. fluency_ contrastive* στον Πίνακα 1.1).

Αντισταθμίσεις

Δεδομένου ότι ο συντάκτης μας είναι ένας εξαιρετικά προσαρμόσιμος συντάκτης, υπάρχουν πολλές αντισταθμίσεις που πρέπει να ληφθούν υπόψη κατά τη διάρκεια της δημιουργίας αντιπαραδειγματικών. Οι σημαντικότερες εξ αυτών είναι οι:

- **Ελεγξιμότητα έναντι Ελαχιστότητας** - Οι ελεγχόμενες παρεμβάσεις περιλαμβάνουν την αλλαγή *κάθε* έννοιας που μπορεί να αλλάξει προκειμένου να παρατηρηθεί ένα αποτέλεσμα- για το σκοπό αυτό, θα μπορούσαμε ενδεχομένως να αλλάξουμε όσο το δυνατόν περισσότερες λέξεις προκειμένου να επιτύχουμε έναν στόχο, π.χ. αλλαγή ετικέτας. Ωστόσο, στην περίπτωσή μας, προκειμένου να παράγουμε ελάχιστες παρεμβάσεις, ορίζουμε έναν μέγιστο αριθμό αντικαταστάσεων ανά είσοδο και αξιοποιούμε την *αναζήτηση δέσμης* για την επιλογή των καταλληλότερων αλλαγών. Κατά συνέπεια, η προεπιλεγμένη απαίτηση ελεγξιμότητας θυσιάζεται εν μέρει, καθώς δεν είναι εγγυημένο ότι όλες οι λέξεις που μπορούν να αντικατασταθούν θα αντικατασταθούν πράγματι. Παρ' όλα αυτά, το σύστημά μας εξακολουθεί να παράγει επεξεργασίες για κάθε είσοδο, πράγμα που σημαίνει ότι θα αλλάξει το αρχικό κείμενο, αν και όχι εξ ολοκλήρου. Αυτός είναι ο λόγος για τον οποίο ορίζουμε την ελεγξιμότητα ως την *τροποποίηση τουλάχιστον μιας λέξης του αρχικού δείγματος*. Στα πειράματά μας (βλ. Πίνακα 1.1) αποδεχθήκαμε αυτό το συμβιβασμό, καθώς το ενδιαφέρον μας έγκειται περισσότερο στην ελαχιστότητα σε σύγκριση με την ελεγξιμότητα. Παρ' όλα αυτά, είναι δυνατόν να διασφαλιστεί πλήρως η τελευταία με την άρση των περιορισμών που αναφέρθηκαν παραπάνω (δηλ. μέγιστος αριθμός αντικαταστάσεων και αναζήτηση δέσμης), αν και μια τέτοια προσέγγιση θα είχε ως αποτέλεσμα χειρότερες επιδόσεις όσον αφορά την ελαχιστότητα.

- **Βελτιστότητα έναντι Ταχύτητας Εκτέλεσης** - Στο σύστημά μας, χρησιμοποιούμε τόσο μια ντετερμινιστική (βλ. *Deterministic w. fluency* από τον πίνακα 1.1) όσο και μια προσέγγιση ΝΔΓ (βλ. *GNN w. fluency* από τον πίνακα 1.1) για την επίλυση του RLAP. Με την ντετερμινιστική προσέγγιση εξασφαλίζεται η βέλτιστότητα, καθώς έχει αποδειχθεί ότι οι παραδοσιακοί αλγόριθμοι αντιστοίχισης γράφων βρίσκουν τη βέλτιστη λύση [42, 36]. Ωστόσο, η πολυπλοκότητα αυτών των αλγορίθμων, η οποία είναι $O(mn \log n)$, οδηγεί σε βραδύτερους χρόνους εκτέλεσης καθώς αυξάνεται το μέγεθος του γράφου (το οποίο είναι ανάλογο με τον αριθμό των λέξεων προς αντικατάσταση και επομένως εξαρτάται από το μέγεθος του συνόλου δεδομένων). Αντικαθιστώντας τους ντετερμινιστικούς αλγορίθμους με το εκπαιδευμένο ΝΔΓ (βλ. ενότητα 1.1.4), το σύστημά μας γίνεται σημαντικά ταχύτερο με κόστος μια πτώση στη βελτιστότητα. Αυτό οφείλεται στο γεγονός ότι η λύση που δίνει το ΝΔΓ είναι μια *προσέγγιση* της βέλτιστης.

- **Επεξηγησιμότητα έναντι Ταχύτητας Εκτέλεσης** - Στην εργασία μας, χρησιμοποιούμε το WordNet ως τον προεπιλεγμένο τρόπο υπολογισμού των βαρών ακμής μεταξύ των κόμβων, όπου κάθε βάρος ακμής βασίζεται στο μονοπάτι που συνδέει μια λέξη-πηγή s με μια λέξη-στόχο t στο WordNet. Με την αντιστοίχιση κάθε λέξης σε κάποιο σύνολο του WordNet, αποδίδεται μια ντετερμινιστική θέση έννοιας σε κάθε λέξη, παρέχοντας μια πλήρως διαφανή αντιστοίχιση έννοιας σε μια καλά επεξεργασμένη λεξική δομή. Η αξιοποίηση των διανυσμάτων ενσωμάτωσης λέξεων ρίχνει σκιά στην αντιστοίχιση λέξεων, καθώς μεταβαίνουμε σε μια διανυσματική αναπαράσταση ενός μη ερμηνεύσιμου πολυδιάστατου χώρου μέσω μοντέλων μαύρου κουτιού. Η ομοιότητα στον διανυσματικό χώρο ενσωμάτωσης μεταφράζεται σε σημασιολογική ομοιότητα των φυσικών εννοιών, και αυτός είναι ο λόγος για τον οποίο χρησιμοποιούμε αυτά τα μοντέλα ενσωμάτωσης. Επιπλέον, οι παραλλαγές του συντάκτη μας με αυτά τα μοντέλα είναι σημαντικά ταχύτερες από τις παραλλαγές που χρησιμοποιούν το WordNet, ειδικά αν συνδιαστούν και με το ΝΔΓ αντί για τους ντετερμινιστικούς αλγορίθμους για το RLAP.

1.3.4 Αποτελέσματα

Τα αποτελέσματα των πειραμάτων μας παρουσιάζονται στον Πίνακα 1.1, συμπεριλαμβανομένων των συνόλων δεδομένων IMDB και Newsgroups. Οι προτεινόμενοι συντάκτες μας - ο ντετερμινιστικός και ο βασισμένος

	IMDB					
	Editor	**Fluency ↓**	**Closeness ↑**	**Flip Rate ↑**	**Minimality ↓**	**Runtime ↓**
WordNet	Deterministic w. fluency	0.14	0.969	0.892	0.08	4:09:41
	GNN w. fluency	0.07	0.986	0.861	0.12	3:17:51
	GNN w. fluency & dynamic thresh	0.057	0.986	0.851	0.146	4:18:34
	GNN w. fluency & POS_filter	0.08	0.992	0.862	0.123	**0:32:05**
	GNN w. fluency & edge filter	0.105	0.993	0.845	0.149	3:00:38
	GNN w. fluency_contrastive	0.112	**0.999**	0.914	0.014	2:12:06
	GNN w. contrastive	0.048	0.996	0.927	**0.01**	2:00:15
Embeddings	GNN w. AnglE & contrastive	0.063	0.995	0.944	0.011	0:45:38
	GNN w. GIST & contrastive	0.037	0.995	0.882	0.016	0:58:14
	GNN w. Jina & contrastive	0.047	0.995	0.928	0.017	1:00:56
	GNN w. MUG & contrastive	**0.036**	0.996	0.889	0.013	0:52:19
	Polyjuice	0.394	0.787	0.782	0.705	5:01:58
	MiCE	0.201	0.949	**1.000**	0.173	48:37:56
	Newsgroups					
	Editor	**Fluency ↓**	**Closeness ↑**	**Flip Rate ↑**	**Minimality ↓**	**Runtime ↓**
WordNet	Deterministic w. fluency	0.182	0.951	0.870	0.135	4:20:52
	GNN w. fluency	0.074	0.985	0.826	0.151	3:48:37
	GNN w. fluency & dynamic thresh	0.043	0.984	0.823	0.148	4:47:14
	GNN w. fluency & POS filter	0.044	0.989	0.841	0.143	1:19:57
	GNN w. fluency & edge filter	0.12	0.989	0.834	0.151	3:05:08
	GNN w. fluency_contrastive	0.088	0.979	0.875	0.033	2:45:31
	GNN w. contrastive	0.033	0.989	0.920	0.033	2:02:34
Embeddings	GNN w. AnglE & contrastive	0.005	0.995	0.904	0.027	1:09:13
	GNN w. GIST & contrastive	**0.001**	0.995	0.898	0.02	1:02:55
	GNN w. jina & contrastive	0.013	0.993	0.882	0.025	0:57:31
	GNN w. MUG & contrastive	0.005	**0.996**	0.900	**0.016**	**0:53:04**
	Polyjuice	1.153	0.667	0.8	0.997	6:00:10
	MiCE	0.152	0.922	**0.992**	0.261	47:23:35

Table 1.1: Πειραματικά αποτελέσματα της δημιουργίας αντιφατικών παρεμβάσεων. Αξιολογούμε διάφορες παραλλαγές του συντάκτη μας χρησιμοποιώντας τις μετρικές που περιγράφονται στην ενότητα 1.3.1 και τα συγκρίνουμε με το MiCE και το Polyjuice. Για κάθε μετρική (στήλη) η καλύτερη τιμή επισημαίνεται με **έντονο χρώμα**. Οι χρόνοι εκτέλεσης αναφέρονται σε ολόκληρη τη διαδικασία παραγωγής παρεμβάσεων.

σε ΝΔΓ - υπερτερούν τόσο του MiCE όσο και του Polyjuice σε τρεις από τις τέσσερις μετρικές, δηλαδή **ελαχιστότητα** (minimality), **ευγλωττία** (fluency) και **εγγύτητα** (closeness). Όσον αφορά το flip-rate, το MiCE επιτυγχάνει τα υψηλότερα αποτελέσματα (99% - 100%, στα δύο σύνολα δεδομένων), ακολουθούμενο από την προσέγγισή μας: ο καλύτερος συντάκτης μας επιτυγχάνει τιμές λίγο πάνω από 90% (συγκεκριμένα 94,4% για το IMDB και 92% για το Newsgroups). Ωστόσο, αυτό είναι αναμενόμενο, δεδομένου ότι ο MiCE είναι ο μόνος συντάκτης που έχει πρόσβαση στον ταξινομητή και είναι σε θέση να κατασκευάζει στρατηγικά τις επεξεργασίες που τον επηρεάζουν περισσότερο , ανεξάρτητα από το κείμενο εισόδου.

Τα αποτελέσματα δείχνουν επίσης ότι οι επεξεργασίες μας τείνουν να είναι πιο μικρές όταν η κατασκευή του γράφου βασίζεται σε μοντέλα ενσωμάτωσης αντί του WordNet (περίπου το 10% των αρχικών λέξεων αλλάζει όταν χρησιμοποιείται το WordNet, ενώ με τα μοντέλα ενσωμάτωσης αλλάζει μόνο το 1% των εν λόγω λέξεων). Πιστεύουμε ότι αυτό οφείλεται στο γεγονός ότι τα πιο σύγχρονα μοντέλα ενσωμάτωσης είναι σε θέση να απεικονίζουν καλύτερα την απόσταση των εννοιών σε σύγκριση με το WordNet, και ως εκ τούτου οι αντικαταστάσεις που βασίζονται σε αυτά είναι υψηλότερης ποιότητας, οδηγώντας σε πιο αντιθετικά ζεύγη. Αυτό σημαίνει ότι για τον ίδιο αντίκτυπο στην έξοδο του ταξινομητή απαιτούνται λιγότερες αντικαταστάσεις ενσωμάτωσης σε σύγκριση με αυτές που βασίζονται στο WordNet. Από την άλλη πλευρά, η χρήση μοντέλων ενσωμάτωσης μειώνει τη συνολική διαφάνεια της μεθόδου. Παρά τις μικρές αποκλίσεις, όλες οι παραλλαγές του πλαισίου μας υπερτερούν σταθερά έναντι των προηγούμενων τεχνικών σε κάθε μετρική για το Polyjuice και σε τρεις μετρικές για το MiCE. Επιπλέον, ακόμη και η παραλλαγή γενικού σκοπού του συστήματός μας, η οποία δεν έχει πρόσβαση στον ταξινομητή, αποδίδει καλύτερα αποτελέσματα σε σύγκριση με τον συντάκτη λευκου κουτιού MiCE,

σε μόλις 2% του χρόνου.

Όσον αφορά τον χρόνο εκτέλεσης, οι συντάκτες μας παρουσιάζουν αξιοσημείωτη βελτίωση στην ταχύτητα σε σύγκριση με το MiCE και το Polyjuice. Ο ντετερμινιστικός συντάκτης μας, ο οποίος χρησιμοποιείται ως βάση, απαιτεί περίπου 4 ώρες για κάθε σύνολο δεδομένων, ενώ οι συντάκτες που χρησιμοποιούν το ΝΔΓ που συζητείται στην ενότητα 1.1.4 επιτυγχάνουν ταχύτερη εκτέλεση κατά μέσο όρο (2-4 ώρες). Ο χρόνος εκτέλεσης βελτιώνεται περαιτέρω με τη χρήση μοντέλων ενσωμάτωσης, όπου η εκτέλεση απαιτεί λιγότερο από μία ώρα (52 λεπτά - 1 ώρα για το IMDB, 53 λεπτά - 1 ώρα και 9 λεπτά για το Newsgroups). Αυτή η σημαντική βελτίωση της ταχύτητας είναι ένα από τα κύρια πλεονεκτήματα της μεθόδου μας σε σύγκριση με τους δύο ανταγωνιζόμενους συντάκτες, όπου παρατηρήσαμε περίπου 97% και 83% βελτίωση της ταχύτητας σε σύγκριση με το MiCE και το Polyjuice αντίστοιχα.

Στην συνέχεια συγκρίνουμε τα αποτελέσματα από τις διάφορες παραλλαγές του συντάκτη μας και σχολιάζουμε την αποτελεσματικότητά τους:

1. **Φιλτράρισμα ακμών** - Εξετάζοντας τα αποτελέσματα με και χωρίς τη χρήση του φιλτραρίσματος ακμών παρατηρούμε ότι είναι αρκετά παρόμοια. Αυτό μας οδηγεί στο συμπέρασμα ότι ένας τέτοιος μηχανισμός είναι περιττός και η λειτουργικότητά του καλύπτεται από τη λύση GNN στο πρόβλημα ανάθεσης του γράφου μας.

2. **WordNet έναντι Μοντέλων Ενσωματώσεων** - Όπως φαίνεται από τον Πίνακα 1.1 οι παραλλαγές μας που αξιοποιούν τα μοντέλα ενσωμάτωσης επιτυγχάνουν καλύτερα αποτελέσματα σε όλες τις μετρικές σε σύγκριση με εκείνες που βασίζονται στο WordNet. Όσον αφορά το *χρόνο εκτέλεσης*, τα μοντέλα ενσωμάτωσης επίσης υπερτερούν του WordNet καθώς το τελευταίο απαιτεί κλήσεις σε API για κάθε λέξη/κόμβο του V, οι οποίες επιβραδύνουν σημαντικά τη διαδικασία δημιουργίας του γράφου.

3. **Περιορισμός POS έναντι Γενικών Αντικαταστάσεων** - Όπως παρατηρούμε από τον Πίνακα 1.1, και οι δύο παραλλαγές, με και χωρίς περιορισμό για τα μέρη του λόγου, επιτυγχάνουν παρόμοια αποτελέσματα. Αυτό ισχύει τόσο για το σύνολο δεδομένων IMDB όσο και για το σύνολο δεδομένων Newsgroups, δείχνοντας ότι η παρατηρούμενη ομοιότητα δεν οφείλεται σε κάποιον συγκεκριμένο περιορισμό POS . Η μόνη σημαντική διαφορά παρατηρείται στο χρόνο εκτέλεσης (32 - 60 λεπτά για τους συντάκτες με περιορισμό, 2 - 4 ώρες για τους μη περιορισμένους), κάτι που είναι αναμενόμενο, καθώς όταν εξετάζουμε μόνο ορισμένα μέρη του λόγου κάθε φορά, περιορίζουμε και το πλήθος των λέξεων που θα θεωρηθούν υποψήφιες για αντικατάσταση. Αυτό σημαίνει ότι οι κόμβοι και οι ακμές του γράφου G θα μειωθούν σημαντικά, μειώνοντας έτσι τον χρόνο που απαιτείται για την κατασκευή του γράφου και την εξαγωγή ζευγών αντικαταστάσεων από το GNN.

4. **Στατικός Μέγιστο Αριθμός Αντικαταστάσεων έναντι Δυναμικού** - Τα αποτελέσματα δείχνουν ασήμαντη βελτίωση των μετρικών κατά τη χρήση δυναμικού κατωφλίου σε σύγκριση με τη χρήση στατικού, ενώ ο χρόνος εκτέλεσης αυξάνεται (περίπου κατά 1 ώρα ανά σύνολο δεδομένων). Αυτή η επιβράδυνση είναι αναμενόμενη, καθώς το δυναμικό κατώφλι εισάγει μια επιπλέον γραμμική πολυπλοκότητα για κάθε περίπτωση κειμένου, αντί της πολυπλοκότητας O(1) της στατικής περίπτωσης. Το στατικό κατώφλι είναι η προεπιλεγμένη προσέγγισή μας, εκτός αν αναφέρεται διαφορετικά.

5. **Αντιθετικές έναντι Εύγλωττες και Αντιθετικές Επεξεργασίες** - Ενώ οι συντακτες γενικού σκοπού, στις οποίες χρησιμοποιείται μόνο η ευγλωττία ως ευριστική συνάρτηση, επιτυγχάνουν το χαμηλότερο ποσοστό αλλαγής ετικέτας (flip-rate), παραμένουν καλύτεροι σε όλες τις μετρικές σε σύγκριση με το Polyjuice, έναν άλλο συντάκτη γενικού σκοπού. Αυτό δείχνει ότι το σύστημά μας μπορεί επίσης να χρησιμοποιηθεί ως ένας γενικός, μη στοχευμένος συντάκτης με υψηλής ποιότητας παρεμβάσεις (όσον αφορά τις μετρικές που συζητήθηκαν), ωστόσο ο εκτεταμένος πειραματισμός σχετικά με αυτόν τον ισχυρισμό αφήνεται για μελλοντική εργασία. Οι βελτιστοποιημένες ως προς την εναλλαγή ετικετών παρεμβάσεις, επιτυγχάνουν καλύτερα αποτελέσματα σε *ευγλωττία, εγγύτητα και ελαχιστότητα* σε σύγκριση με τον MiCE, έναν σύγχρονο συντάκτη λευκού κουτιού βελτιστοποιημένο για εναλλαγή ετικετών. Όσον αφορά το flip-rate, ο MiCE επιδεικνύει ανώτερη απόδοση, ξεπερνώντας τη δική μας κατά 7%, αποδεχόμενος μια σημαντική επιβράδυνση 20x στην εκτέλεση.

Ποιοτικά Αποτελέσματα

Σε αυτή την ενότητα παρουσιάζονται ποιοτικές συγκρίσεις με το Polyjuice και το MiCE για να καταδειχθούν οι δυνατότητες της μεθόδου μας όσον αφορά την ελαχιστότητα και το ποσοστό εναλλαγής ετικετών. Για το σκοπό αυτό, επιλέγουμε ένα παράδειγμα του συνόλου δεδομένων IMDB το οποίο έχει αρχικά ταξινομηθεί ως «θετικό» και παράγουμε τα επεξεργασμένα παραδείγματα από το σύστημά μας και τους δύο συντάκτες που αναφέρθηκαν παραπάνω. Συγκεκριμένα για το Polyjuice, δεδομένου ότι ο στόχος του είναι να αλλάξει την πρόβλεψη από *θετική* σε *αρνητική*, χρησιμοποιούμε τον κωδικό ελέγχου [negation], ο οποίος καθοδηγεί τον συντάκτη να δημιουργήσει μια επεξεργασία που είναι η άρνηση του αρχικού κειμένου. Το αρχικό κείμενο μαζί με τα επεξεργασμέναπαρουσιάζονται στο Σχήμα 1.3.1 (οι κόκκινες λέξεις υποδηλώνουν τις αλλαγές που έγιναν από κάθε συντάκτη).

Original:	This movie will likely be too sentimental for many viewers, especially contemporary audiences. Nevertheless I enjoyed this film thanks mostly to the down-to-earth charm of William Holden, one of my favorite stars, and the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors rescues what in lesser hands could've been trite lines. The cinematography of Hong Kong from the period of filming is another highlight of this movie. All in all, a better than average romantic drama, 7/10.
MiCE:	This movie will likely be too harsh for many conservative, conservative audiences. Personally I enjoyed this film thanks mostly to the brilliant acting of William Powell, both of whom have the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors enhances what in less than average hands could've been trite lines. The beautiful performance of Hong Kong from the onset of filming is another highlight of this movie. All in all, a better than average romantic drama, 4/10.
Polyjuice:	This movie will likely be too sentimental for many viewers, especially contemporary audiences. Nevertheless I enjoyed this film thanks mostly to the down-to-earth charm of William Holden, one of my favorite stars, and the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors rescues what in lesser hands could've been trite lines. The cinematography of Hong Kong from the period of filming is another highlight of this movie. All in all, of.
Ours:	This movie will likely be too sentimental for many viewers, especially contemporary audiences. Nevertheless I enjoyed this film thanks mostly to the down-to-earth charm of William Holden, one of my favorite stars, and the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors rescues what in lesser hands could've been trite lines. The cinematography of Hong Kong from the period of filming is another highlight of this movie. All in all, a better than average shameful drama, 7/10.

Figure 1.3.1: Αρχικό κείμενο και επεξεργασμένα κέιμενα από διαφορετικούς συντάκτες.

Όπως βλέπουμε, ο MiCE εκτελεί τον μεγαλύτερο αριθμό αλλαγών στην αρχική είσοδο, με δύο από αυτές τις αλλαγές να είναι σημασιολογικά λανθασμένες (*"conservative, conservative"* και *"both of whom have"*). Παρατηρούμε επίσης ότι οι αλλαγές του δεν είναι εξ ολοκλήρου σε επίπεδο λέξης, γεγονός που επιδεινώνει περαιτέρω την απόδοση του όσον αφορά την *ελαχιστότητα*. Ο Polyjuice από την άλλη πλευρά, κάνει μόνο μία αλλαγή στο τέλος του κειμένου, η οποία όμως δεν έχει καμία σημασιολογική σημασία. Ο συντάκτης μας παρουσιάζει την καλύτερη απόδοση από τους τρεις, αλλάζοντας μόνο μία λέξη, ενώ είναι σημασιολογικά σωστός

και πολύ κοντά στο αρχικό παράδειγμα.

Edits	Minimality ↓	Prediction Flipped
Polyjuice	0.078	False
MiCE	0.256	**True**
Ours	**0.011**	**True**

Table 1.2: Τα μετρικά αποτελέσματα των επεξεργασιών που παρουσιάζονται στην εικόνα 9.3.1. Για κάθε ιδιότητα (στήλη) η καλύτερη τιμή επισημαίνεται με **bold**.

Τα αριθμητικά αποτελέσματα των περιπτώσεων του Σχήματος 1.3.1 όσον αφορά την *ελαχιστότητα* (minimality) και την *αλλαγή ετικέτας* (label-flipping) αναφέρονται στον Πίνακα 1.2. Δεδομένου ότι έχουμε μόνο μία περίπτωση κειμένου, χρησιμοποιούμε τον όρο *prediction flipped* για να δηλώσουμε αν η επεξεργασμένη είσοδος είναι σε θέση να αλλάξει την αρχική πρόβλεψη του ταξινομητή. Σημειώνουμε ότι το Polyjuice δεν είναι σε θέση να αλλάξει την πρόβλεψη, ενώ τόσο το MiCE όσο και το δικό μας σύστημα τα καταφέρνουν. Επίσης, ο δικός μας συντάκτης έχει την καλύτερη (χαμηλότερη) τιμή ελαχιστότητας, με το Polyjuice να είναι δεύτερο και το MiCE το χειρότερο από τα τρία.

1.4 Συμπεράσματα

1.4.1 Συζήτηση

Σε αυτή την εργασία, παρουσιάζουμε ένα σύστημα για τη δημιουργία βέλτιστων και ελεγχόμενων αντιφατικών εξηγήσεων σε επίπεδο λέξεων μέσω αντικαταστάσεων βασισμένων σε γράφους, το οποίο αξιολογούμε σε δύο προβλήματα ταξινόμησης. Ένα βασικό μέρος της μεθόδου μας είναι η επίλυση του Ορθογώνιου Προβλήματος Γραμμικής Ανάθεσης (RLAP), το οποίο αποτελεί επέκταση του πιο ευρέως γνωστού Προβλήματος Γραμμικής Ανάθεσης (LAP). Αποδεικνύεται ότι με κατάλληλες τροποποιήσεις το RLAP μπορεί να επιλυθεί με τους ίδιους αλγορίθμους που χρησιμοποιούνται για την επίλυση του LAP. Αυτοί οι αλγόριθμοι, περιλαμβάνουν τον *Ουγγρικό αλγόριθμο* που επιλύει το RLAP σε $O(n^3)$ και τον *αλγόριθμο του Karp* που το επιλύει σε $O(mn \log m)$. Στην εργασία μας, επεκτείνουμε ένα υπάρχον ΝΔΓ για να φιλοξενήσει το RLAP, το οποίο προσεγγίζει τη βέλτιστη λύση που βρίσκουν οι προηγούμενοι ντετερμινιστικοί αλγόριθμοι, ενώ βελτιώνει σημαντικά το συνολικό χρόνο εκτέλεσης. Εξ όσων γνωρίζουμε, καμία προηγούμενη εργασία δεν έχει αντιμετωπίσει το RLAP χρησιμοποιώντας ΝΔΓ.

Το σύστημά μας αποτελείται από τρία στάδια: στο *πρώτο* στάδιο κατασκευάζουμε έναν διγράφο με κόμβους (λέξεις) πηγής και στόχου, στο *δεύτερο* στάδιο λαμβάνουμε κατάλληλα ζεύγη αντικαταστάσεων επιλύοντας τον RLAP στον γράφο και στο *τρίτο* και τελευταίο στάδιο, χρησιμοποιούμε αναζήτηση δέσμης προκειμένου να παράγουμε ελάχιστες, εύγλωττες και αντιθετικές επεξεργασίες. Σε κάθε στάδιο, χρησιμοποιήσαμε διαφορετικές προσεγγίσεις και ρυθμίσεις, λόγω της ιδιαίτερα παραμετροποιήσιμης φύσης του συστήματος

Κατά τη διάρκεια της κατασκευής του γράφου, πειραματιστήκαμε με έναν μηχανισμό φιλτραρίσματος ακμών, σκοπός του οποίου ήταν να τιμωρεί τις ακμές που συνδέουν διαφορετικά μέρη του λόγου (POS), ο οποίος όμως αποδείχθηκε περιττός με βάση τα αποτελέσματα. Επιλέξαμε επίσης να χρησιμοποιήσουμε μεγάλα γλωσσικά μοντέλα με embeddings αντί του WordNet για τον υπολογισμό της ομοιότητας μεταξύ δύο λέξεων. Τα ευρήματά μας δικαιολογούν την απόφασή μας αυτή, καθώς οι επεξεργασίες που βασίζονται στα embeddings υπερτερούσαν σε όλες τις μετρικές από τις επεξεργασίες που βασίζονται στο Wordnet. Δοκιμάσαμε τις δυνατότητες γενίκευσης του συστήματός μας, παράγοντας αντικαταστάσεις με και χωρίς περιορισμούς POS, και χρησιμοποιήσαμε τόσο ένα δυναμικό όσο και ένα στατικό κατώφλι για να ορίσουμε τον μέγιστο αριθμό αντικαταστάσεων για κάθε περίπτωση κειμένου. Ήταν ενδιαφέρον να διαπιστώσουμε ότι το δυναμικό κατώφλι δεν βελτίωσε την ποιότητα των επεξεργασμένων κειμένων και αντίθετα αύξησε μόνο τον χρόνο εκτέλεσης. Χρησιμοποιήσαμε επίσης διαφορετικές ευριστικές συναρτήσεις στην αναζήτηση δέσμης για να παράγουμε επεξεργασίες βελτιστοποιημένες για διαφορετικές εργασίες, αναδεικνύοντας την ευελιξία της μεθόδου μας.

Τέλος, συγκρίναμε το σύστημά μας με δύο SoTA συντάκτες, συγκεκριμένα τους Polyjuice και MiCE. Ο πρώτος είναι ένας γενικής χρήσης συντάκτης αντιθετικών εξηγήσεων μαυρου κουτιού , ενώ ο δεύτερος είναι ένας συντάκτης αντιθετικών εξηγήσεων λευκού κουτιού που παράγει ελάχιστες επεξεργασίες βελτιστοποιημένες ως

προς την αλλαγή της εξόδου του εκάστοτε ταξινομητή. Χρησιμοποιήσαμε κατάλληλες μετρικές, για τη μέτρηση της *ευγλωττίας*, της *ελαχιστότητας*, της *σημασιολογικής εγγύτητας* και του *ποσοστού αλλαγής ετικέτας*. Τα αποτελέσματα δείχνουν ότι ξεπερνάμε τους δύο παραπάνω συντάκτες στις περισσότερες μετρικές, ενώ είμαστε σημαντικά ταχύτεροι, γεγονός που αποτελεί και ένα από τα κύρια πλεονεκτήματα της μεθόδου μας.

1.4.2 Γενικότερος Αντίκτυπος και Ηθική

Το σύστημά μας προορίζεται να βοηθήσει στην ερμηνεία των μοντέλων ΕΦΓ. Καθώς είναι μια μέθοδος εξήγησης που δεν εξαρτάται από τα μοντέλα (στην προεπιλεγμένη λειτουργία της, δεν είναι βελτιστοποιημένη προς κάποια συγκεκριμένη μετρική), έχει τη δυνατότητα να επηρεάσει την ανάπτυξη συστημάτων ΕΦΓ σε ένα ευρύ φάσμα μοντέλων και εργασιών. Ειδικότερα, οι επεξεργασίες μας μπορούν να βοηθήσουν τους προγραμματιστές που εργάζονται στον τομέα της ΕΦΓ στην διευκόλυνση, την αποσφαλμάτωση και την αποκάλυψη των τρωτών σημείων των μοντέλων. Η μέθοδός μας μπορεί επίσης να βοηθήσει στην επαύξηση των δεδομένων, η οποία οδηγεί σε λιγότερο προκατειλημμένα και πιο ισχυρά συστήματα. Κατά συνέπεια, οι μεταγενέστεροι χρήστες των μοντέλων ΕΦΓ μπορούν επίσης να ωφεληθούν αποκτώντας πρόσβαση σε αυτά τα συστήματα.

Ενώ η εργασία μας επικεντρώνεται στην ερμηνεία μοντέλων ΕΦΓ, θα μπορούσε να χρησιμοποιηθεί καταχρηστικά και σε άλλα πλαίσια. Για παράδειγμα, κακόβουλοι χρήστες θα μπορούσαν να δημιουργήσουν εχθρικά παραδείγματα, όπως ελαφρώς τροποποιημένη ρητορική μίσους, για να παρακάμψουν τους ανιχνευτές τοξικής γλώσσας. Επιπλέον, η χρήση αυτών των επεξεργαστών για την επαύξηση δεδομένων θα μπορούσε ακούσια να οδηγήσει σε λιγότερο ισχυρά και πιο προκατειλημμένα μοντέλα, καθώς οι επεξεργασίες έχουν σχεδιαστεί για να αποκαλύπτουν τις αδυναμίες του μοντέλου. Για να αποφύγουν την ενίσχυση των υφιστάμενων προκαταλήψεων, οι ερευνητές θα πρέπει να εξετάσουν προσεκτικά τον τρόπο επιλογής και επισήμανσης των επεξεργασμένων περιπτώσεων όταν τις χρησιμοποιούν για εκπαίδευση. Ωστόσο, αυτές οι απειλές ισχύουν για οποιονδήποτε επεξεργαστή κειμένου στη βιβλιογραφία του ΕΦΓ και δεν είναι προσαρμοσμένες στην εργασία μας.

1.4.3 Μελλοντικές Κατευθύνσεις

Κλείνοντας αυτή τη διατριβή θα θέλαμε να προτείνουμε μερικές κατευθύνσεις για την περαιτέρω βελτίωση αυτής της εργασίας ή να εμπνεύσουμε διαφορετικές ενδιαφέρουσες προσεγγίσεις. Ως πρώτο βήμα, θα μπορούσαν να ενσωματωθούν περισσότερες εξωτερικές λεξιλογικές πηγές (π.χ. ConceptNet) για την ενίσχυση των πιθανών υποψηφίων αντικαταστάσεων, καθώς και για τον ακριβέστερο υπολογισμό της ομοιότητας μεταξύ δύο λέξεων. Ένα άλλο βήμα θα μπορούσε να είναι ο περαιτέρω πειραματισμός με το ΝΔΓ που χρησιμοποιείται για την επίλυση του RLAP, προκειμένου να βελτιωθεί η έξοδός του σε σύγκριση με τη βέλτιστη λύση που δίνει ο αλγόριθμος του Karp.

Μια ελκυστική κατεύθυνση θα ήταν η διερεύνηση της χρήσης της μεθόδου μας ως συντάκτη γενικού σκοπού. Με αυτόν τον τρόπο, κάποιος θα μπορούσε να αποκτήσει σημαντικές γνώσεις σχετικά με την αποτελεσματικότητά του σε άλλες εργασίες εκτός από την αλλαγή της εξόδου ενός ταξινομητή. Συνιστούμε τη σύγκριση των αποτελεσμάτων με το Polyjuice και το Tailor [81], για την αξιολόγηση της απόδοσής του έναντι άλλων SoTA συντακτών γενικού σκοπού.

Τέλος, η αντιστάθμιση μεταξύ *ελαχιστότητας* και *ποσοστού αλλαγής ετικέτας* δεν εξετάστηκε στην παρούσα διατριβή, αλλά θα μπορούσε να αποτελέσει τη βάση για περαιτέρω έρευνα. Συγκεκριμένα, η αύξηση του μέγιστου αριθμού αντικαταστάσεων ανά περίπτωση κειμένου θα μπορούσε να οδηγήσει σε υψηλότερο ποσοστό αλλαγής ετικέτας, καθώς περισσότερες λέξεις θα άλλαζαν. Ωστόσο, οι επιπλέον αντικαταστάσεις θα αύξαναν το μέγεθος της επεξεργασίας (μετρούμενο με τον αριθμό των αλλαγμένων λέξεων), μειώνοντας έτσι την απόδοση του συντάκτη όσον αφορά την ελαχιστότητα. Οι αποδεκτές απώλειες στην ελαχιστότητα μαζί με τα επιθυμητά κέρδη στο ποσοστό αλλαγής ετικέτας, είναι ανοιχτά προς περαιτέρω διερεύνηση.

Chapter 2

Introduction

As natural language processing (NLP) models become increasingly integral to decision-making processes, the need for explainability and interpretability has become paramount. These models are widely used in various applications, including sentiment analysis, topic classification, machine translation, and conversational agents. Their decisions often have significant impacts, influencing everything from product recommendations to loan approvals and healthcare diagnostics. However, the black-box nature of many sophisticated NLP models, particularly deep learning-based approaches, poses challenges for understanding how specific decisions are made. This opacity can undermine trust and limit the broader adoption of these technologies in critical and sensitive domains.

Given the high stakes involved, there is a pressing need for methods that can provide clear, actionable insights into the behavior of NLP models. One promising approach is the generation of counterfactual explanations. Counterfactual explanations are hypothetical scenarios that show how minimal changes to the input can lead to different model outcomes. By presenting these alternate scenarios, counterfactual explanations help users understand what features are most influential in the model's decision-making process. This type of explanation is particularly valuable because it aligns with human reasoning: understanding "what-if" scenarios is a natural way for people to grasp causal relationships and make informed decisions.

In this work, we propose a graph-based counterfactual editor designed to generate semantically edited inputs, referred to as *counterfactual interventions*, which alter the model's prediction. These interventions serve as counterfactual explanations, offering a clear and intuitive understanding of the model's decision-making process. The primary objective of this research is to develop a method that can create these counterfactual interventions in a way that is contrastive, fluent, and minimal. "Contrastive" means that the changes should effectively alter the model's prediction; "fluent" means that the edited inputs should remain coherent and grammatically correct; and "minimal" means that the changes should be as small as possible to achieve the desired effect.

We evaluate the effectiveness and efficiency of our framework and compare it against existing state-of-the-art counterfactual editors using two NLP tasks, namely binary sentiment classification and topic classification. Our experiments demonstrate that our framework generates edits that are not only contrastive, fluent, and minimal but also produced significantly faster than those from other methods. This speed is crucial for practical applications where timely insights are necessary.

This thesis introduces a novel approach to generating counterfactual explanations for NLP models, addressing a critical need for interpretability in AI. It draws inspiration from [50] and presents a GNN model capable of solving the Rectangular Linear Assignment Problem (RLAP).Using this model instead of the traditional graph assignment algorithms, such as the *Hungarian algorithm* our work achieve a strong improvement in runtime. It also provides a comprehensive evaluation of the editor across different tasks, demonstrating its versatility and effectiveness. By emphasizing minimal and fluent edits, it ensures that the generated explanations are both understandable and practically useful, while the whole process remains faster than others state of the art editors.

The outline of this thesis is as follows:

- We will firstly provide all the background needed in basic Machine Learning algorithms and concepts as well as bipartite graphs in order to be able to explain and justify the idea of Graph Neural Networks. After doing so, we will provide a thorough description of GNN variants relevant to this work.
- We will give a more detailed definition of Counterfactual Explanations and related work. In a similar fashion, we will formally explain the problem of rectangular linear assignment and provide the theoretical background for methods already used to tackle it, such as the Hungarian algorithm.
- Lastly, we will propose our GNN-based editor for counterfactual explanations and highlight the performance of different variants used for its components. We will compare these results with two state of the art counterfactual editors across two NLP tasks and showcase how our editor outperforms them while also being significantly faster. We will also discuss different trade-offs in our editor and how multiple components influence its performance.

Chapter 3

Machine Learning

The topic of machines with human-like capabilities such as thinking has been prevalent among engineers for centuries. In more recent years, after the invention of computers, the field of Artificial Intelligence (AI) has not only been established but also thriving. According to [27] Artificial Intelligence, or the development of computer systems which tackle tasks normally requiring human intelligence, at first focused on solving problems difficult for humans, but elementary for computers. However, it soon became apparent that the opposite problem is also in need of attention, i.e. teaching machines to solve problems which for human beings are intuitive due to their acquired experience.

Machine Learning (ML) is a branch of AI which employs empirical or historical data in order to make predictions with statistical models. In a broader sense, as the name implies, it helps a machine learn through statistics in order to build its experience level without needing to be explicitly programmed to do so [98]. These models perform a type of predictive analysis, processing data using different algorithms, extracting patterns from raw data and finally generating outputs or predictions for a number of tasks.

Machine Learning tasks vary from relatively simple to more convoluted or subjective. For example, some basic tasks involve classifying or grouping data in categories, while others include generating unique brand new data, like images. The more intuitive the task the more challenging it tends to be for computers. The difficulty of the task is also related to the type of input data, which can be images, text, speech, timeseries or even graphs. According to the data representation, a task is grouped with other similar ones belonging to the same field. The most popular ones are Computer Vision, Natural Language Processing, Speech Recognition, Recommendation Systems and more recently Geometric Learning. The task itself is most of the times strongly connected to the machine learning category, as explained in the next section.

Contents

3.1 Learning Categories

Machine Learning algorithms can be categorized according to the experience the model is provided with during training. The three broad types of machine learning algorithms are supervised, unsupervised and reinforcement learning, while semi- and self- supervision can be considered variants of the above.

Supervised Learning

In this type of learning the input data, comprised of multiple features, is associated with a label or target. Let the feature vector be x and the target be y, the model will learn to predict y from x using an array of examples. In most cases, this is achieved by learning the distribution function $p(y|x)$. Some of the most prominent supervised learning tasks are considered to be classification, regression and forecasting.

Unsupervised Learning

Unsupervised learning algorithms do not take labels attached to feature vectors of the dataset into account. Instead, they attempt to implicitly or explicitly learn the probability distribution of the entire dataset $p(x)$ and in turn give insight about the underlying structure of data. Popular tasks of this category include clustering and dimensionality reduction.

Semi-supervised Learning

Semi-supervised learning can be considered a special type of supervised learning where most samples of the dataset used for training are not associated with targets. The small amount of labeled data can be attributed to either difficulty of acquiring said information or desire for better accuracy [102]. Common semi-supervised learning tasks include link prediction in graphs or fraud detection.

Self-supervised Learning

Self-supervised learning is a learning category lying between supervised and unsupervised. It makes use of unlabeled data and leverages supervision signals stemming from the structure of the data itself, by using pseudolabels. This form of learning mostly consists of solving pre-text tasks, i.e. tasks specifically crafted to help a model learn the inner-workings of a dataset, and using the rich information obtained by the originally unlabeled data to later solve other downstream tasks [52], like the ones mentioned in previous sections.

Reinforcement Learning

In reinforcement learning the dataset is not fixed, but rather it receives feedback from changes in its environment. This type of learning will not be considered in this thesis.

3.2 Training a Neural Network

Neural Networks are a subset of machine learning, whose operation resembles that of the human neural system. In this section the general idea of the learning process of such a system will be described as well as the potential challenges.

3.2.1 Basic Concepts

Firstly, we will explain the basic operation of a shallow neural network to provide background information needed to understand similar architectures on graph structured data. The following concepts mostly adhere to supervised learning tasks. The majority of information below was sourced from [27].

In supervised learning, the model uses a fixed amount of N samples from the training dataset $D = \{(x_1, y_1), ..., (x_n, y_n)\}$ and their corresponding labels to compute a function $f : X \to Y$ which maps the input X to the output Y and its trainable parameters are often called weights. In order to evaluate f, a

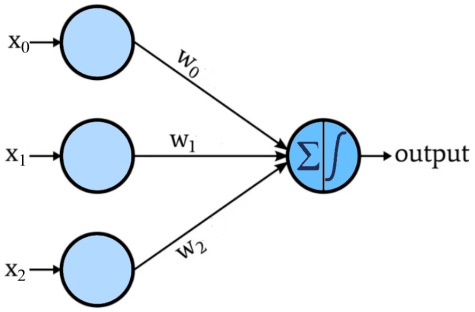


Figure 3.2.1: Shallow Neural Network of One Neuron - Perceptron [38]

function is established to determine the error of training, called loss function L. The goal during training is the calculation of the weights of f in a way that ensures the minimization of the loss function .

The output of each neuron in a neural network is not solely determined by the computation of the weighted sum of the inputs x_i. On the contrary, a different function is employed called **activation function** to define how to transform the weighted sum to an output. An activation function plays a critical role in the network's performance and therefore should be chosen carefully. It can be linear or non-linear and the most commonly used examples can be seen below.

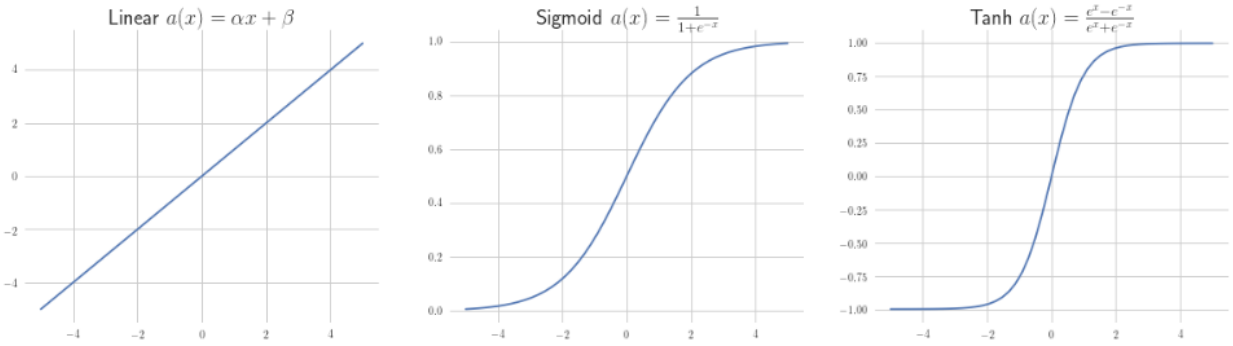


Figure 3.2.2: Examples of activation functions.

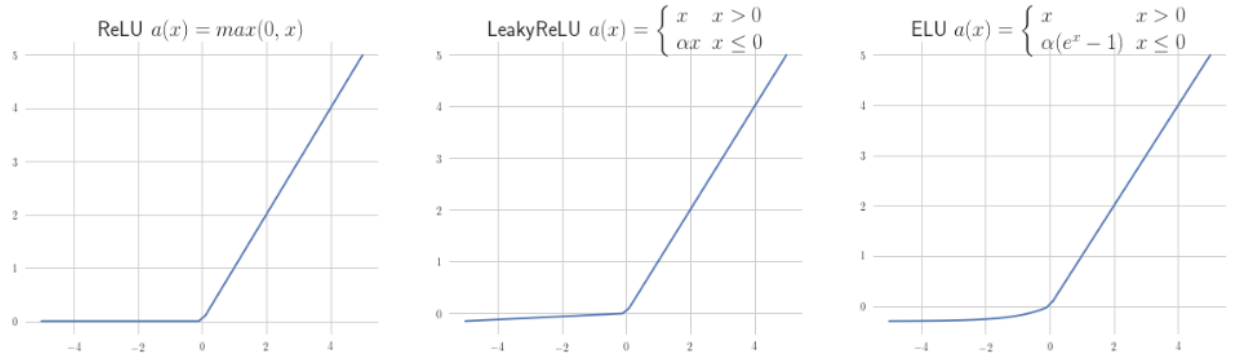


Figure 3.2.3: ReLU activation function and variants.

The linear activation function exhibits a number of limitations, one of the most important being the fact that it makes backpropagation, which will be explained later, impossible since its derivative is a constant. It leaves the weighted sum of the input practically unchanged. In most cases, non-linear activation functions

are used due to the fact that they ultimately allow the stacking of multiple layers (deep networks) and the creation of more complex mappings. Sigmoid and tanh, although being popular non-linearities for certain problems, face the problem of vanishing gradients [103], making training unstable.

The most widely adopted choice of activation function is ReLU and its variants. It overcomes the problems of aforementioned functions and leads to more computationally efficient training. Some of its variants include LeakyReLU and ELU which can be seen in Figure 3.2.3. ReLU also suffers from limitations, such as the dying ReLU problem which causes the existence of non-active neurons [101]. This problem is combated by its variants.

As mentioned previously, the **loss function** or cost function is used to determine how close the model's prediction is to the truth. It maps $Y \times Y$ to a non-negative real number, or $L(y_i, f(x_i))$ for the i_{th} sample. Neural models are trained repeatedly for a number of iterations, called epochs, until meeting an objective or when the maximum amount of iterations has been reached. In general, the total loss of the model in each epoch can be defined as a normalized average of the cost function for each piece of data on the training set. In an optimal scenario, the parameters which minimize this function will be discovered.

The choice of cost function heavily depends on the task carried out. The most notable loss functions are listed below but it is important to note that often custom loss functions are created to cater for more complex tasks.

Mean Squared Error is one of the simplest cost functions and it is often used in regression tasks.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{n} \tag{3.2.1}$$

Mean Absolute Error or *L1 loss* is similar to MSE in the aspect that it ignores the direction of error [99]. This cost function is more robust to outliers.

$$MAE = \frac{\sum_{i=1}^{n}|y_i - f(x_i)|}{n} \tag{3.2.2}$$

Cross Entropy Loss or *Negative Log Likelihood* is a loss function commonly used in classification tasks. It focuses on penalizing predictions that are confident but incorrect and in the case of non-binary classification in M classes can be defined as:

$$NLL = -\sum_{c=1}^{M} p_{y_i,c} log(p_{f(x_i),c}) \tag{3.2.3}$$

The next step after the establishment of the cost function is its minimization. In order to achieve optimization, **gradient-based** methods are employed. The use of gradients is crucial for such problems since derivatives can give us insight on how to scale small changes in the input so as to eventually reach the desirable output [27]. More specifically, the objective function is minimized by iteratively computing the value of the loss function for all training samples and in turn the gradients of the model's parameters and finally updating the parameters in the opposite direction of the gradient.

The most popular gradient method, which adopts the aforementioned process, is *Gradient Descent*. Its definition can be found in Equation 3.2.4, where ϑ represents the model's parameters and ϵ corresponds to the learning rate. The learning rate is a small positive constant which is chosen during training, often by trying several values and keeping the best. Its choice can prove crucial to the model's performance.

$$\theta' = \theta - \epsilon \nabla_\theta L(\theta) \tag{3.2.4}$$

There are many other gradient based optimizers often used in applications, most of which are extensions or inspired by gradient descent. One of the most popular ones is *Stochastic Gradient Descent* (SGD) which differs in the way that it computes gradients and updates parameters for each (x_i, y_i) pair in batches, instead

of the whole dataset. This practice gives it a significant speed advantage. Other honorable mentions include *AdaGrad, RMSProp, AdaDelta* and *Adam*.

Even though the computation of an analytical expression for gradients is simple, its numerical evaluation is quite expensive. For this reason the algorithm of **back-propagation** was proposed [82]. In this approach the chain rule needed to be applied for gradients is computed in a specific order of operations that is highly efficient, making use of computational graphs and storing already computed values.

The process of computing outputs and adjusting weights through back-propagation is often repeated for several epochs until the loss function converges or another threshold is met. The neural network's performance can then be evaluated using a variety of metrics and data which was not encountered during training.

3.2.2 Generalization and Overfitting

A machine learning algorithm must be able to succeed easily on previously unseen data, different from those it was trained with. The ability of a model to perform well on new inputs drawn from the same distribution as the one used to create it is called generalization. To achieve good generalization an algorithm must result in both low train and test error, where train error is defined as the loss during training and test error is the loss obtained from newly observed data, after the training process.

The lack of generalization of a model is often attributed to underfitting or overfitting. Underfitting is caused by high train error, meaning the model has not sufficiently learnt the data distribution whereas overfitting is the product of a large gap between train and test error, given the train error is low. The latter exposes the fact that the model has learnt the data too well, including existing noise, thus having a negative impact on its performance.

It is apparent that the training of a neural network is not just a simple optimization problem, but rather the pursuit of a good trade-off between test and train error. The most prominent technique to handle overfitting is regularization. The concept of regularization is based on Occam's razor, the notion that the simplest solution must be chosen or in this case the smallest in way of parameters. This idea is implemented by adding an extra term $\lambda R(\theta)$ in the loss function which penalizes larger more complex models, while favoring low values of loss. The hyperparameter λ of the term is chosen during training in order to favor the model's performance and R is the regularization function which is often a norm of the weights. The two most prevalent regularizers are L_1 and L_2 norm. In the equations below W represents the weight matrix of the model.

L_1 *Regularization* or *Lasso Regularization* tends to favor sparse solutions, i.e. solutions containing many zero values, by penalizing both uniformly low and high parameter values.

$$R_{L_1}(W) = \sum_{i,j} |W_{i,j}| \tag{3.2.5}$$

L_2 *Regularization* or *Ridge Regularization* punishes high values heavily. It is often referred to as weight decay.

$$R_{L_2}(W) = \sum_{i,j} W_{i,j}^2 \tag{3.2.6}$$

Finally, another way of achieving generalization is by using *dropout*. With this technique, a number of layer outputs are randomly ignored in order to ensure that the model does not rely on specific neurons. By doing so, noise is added to the training process which in turn makes the neural network more robust.

3.3 Deep Learning

With the knowledge of all the basic steps of training a shallow neural network, some principal models will be explained in this section. All of them are examples of Deep Learning algorithms, i.e. neural networks in which multiple layers of neurons are used in order to extract progressively higher level features from the input data. Even though the complexity of said algorithms is increased, the general idea remains the same.

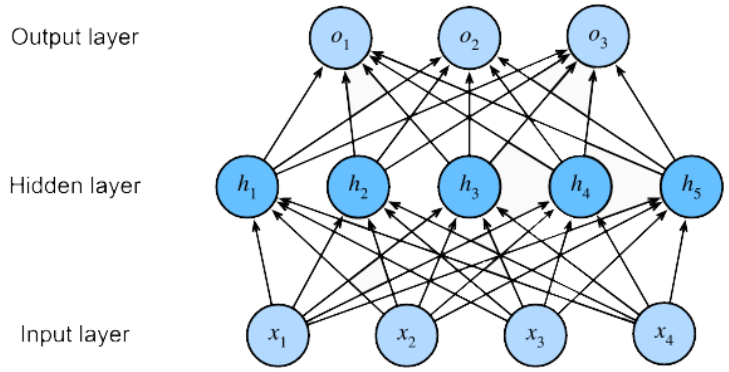


Figure 3.3.1: Multi-layer Perceptron with one hidden layer of 5 units. [116]

3.3.1 Multi-Layer Perceptron (MLP)

The concept of the Multi-Layer Perceptron was introduced as a way of overcoming the limitations of linear models. Specifically, linear models imply monotonicity, i.e. that any change in features always causes change in the same direction for the model's output - decrease or increase [116]. This generic assumption is not true for the majority of problems to be solved. In order to tackle this obstacle, the stacking of multiple fully connected layers was proposed.

The MLP, often referred to as a Feed Forward Neural Network (FFNN), is an artificial neural network in which the connections between nodes do not form a cycle [96]. The information is only processed in one direction from the nodes of the input layer, through the hidden nodes to the output layer nodes. As explained in previous sections, each node computes the weighted sum of inputs and produces an ouput using an activation function, which in this case needs to be non-linear. Otherwise, since the sum of two linear functions is also linear the existence of layers would be unnecessary.

The simplest MLP contains only one hidden layer and is called a *single-layer perceptron*. An example can be seen in Figure 3.3.1. The input layer comprises of the input vectors left unchanged, each of which are passed to all the neurons in the hidden layer. The hidden layer processes the inputs it is given and extracts features from them. If multiple layers exist, the closer the hidden layer is to the output the higher-level the features it extracts. The output layer computes the output of the model by using the processed data passed from the previous layer.

There is no limitation to the number of layers or number of units in each layer. This is a matter of experimentation during the training process.

3.3.2 Convolutional Neural Networks (CNN)

Multi-layer Perceptrons are useful for dealing with tabular data, i.e. data in the form of rows and columns which consist of a number of samples and their corresponding figures. However, if the input data consists of high-dimensional data such as images, an appropriate MLP would have to be enormous in size with millions of parameters. Considering this and the fact that visual data exhibits interactions between features due to the locality of pixels, a new type of neural network was introduced, the Convolutional Neural Network (CNN) [45].

CNNs are rooted in the idea of using a new type of layer which performs convolutions. A *convolutional layer* contains a set of trainable filters. These filters are convolved or in practice cross-correlated with input from the previous layer producing what is called a feature or activation map. They are generally small in size, leading to lower numbers of parameters, since they make the assumptions of translation invariance and locality, which are true for images. The process of cross-correlation consists of simply computing the dot product between filters and the input across both dimensions and subsequently producing a 2-dimensional activation map of that filter. The network is able to learn filters corresponding to different features in specific positions of the input. The local focus of the CNN also aids in the problem of overfitting.

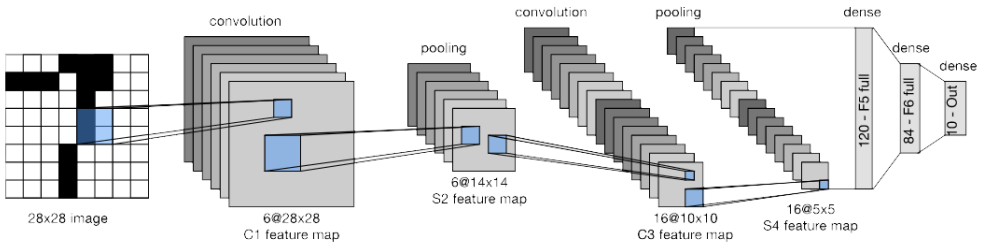


Figure 3.3.2: Typical CNN architecture - LeNet. [116]

In a typical CNN the convolutional layers are followed by pooling layers. The act of pooling serves the purpose of both alleviating the local sensitivity of convolutional layers and spatially downsampling representations [116]. Pooling filters are non trainable and perform the deterministic operation of aggregating elements present in a fixed-size window of the feature maps they receive. The types of aggregation most commonly used are the computation of maximum or average, resulting in the so-called max and mean pooling filters respectively.

In Figure 3.3.2 we can see the architecture of the first and simplest CNN. Nowadays, the CNNs used in applications are much deeper, meaning they consist of tens of alternating convolutional and pooling layers, followed by fully connected -or dense- ones. The dense layers contribute in learning combinations of the features extracted and serve as a classification head.

3.4 Natural Language Processing

Natural Language Processing (NLP) is a field at the intersection of computer science, artificial intelligence, and linguistics, focused on enabling machines to understand, interpret, and generate human language. With the increasing availability of digital text data and the growing demand for intelligent systems capable of interacting with humans in natural language, NLP has become a critical area of research and application. In this work, we will address two common NLP problems, namely binary sentiment classification and topic classification. Below, we provide an overview of the fundamentals required, in order to better understand those problems.

3.4.1 Embeddings

A term which we will refer to a lot in this dissertation is *Embeddings*. An embedding is a relatively low-dimensional space used for the translation of higher-dimensional vectors. The goal of embeddings is to bring semantically similar objects closer together in the embedding space, by effectively capturing the semantics of the input. The dimensionality reduction offered by this method makes machine learning easier if the original input comprises of sparse high-dimensional vectors [14].

As far as NLP is concerned, embeddings serve as word or sentence representations. They generally are real-valued vectors representing the semantic meaning of words in such a way that maps words with similar meaning closer together in the vector space [104]. The generation of embeddings is possible through an array of methods such as dimensionality reduction on the word co-occurrence matrix, probabilistic models or even neural networks. The most popular approaches include Word2vec [60], GloVe [74], BERT [75] and Principal Component Analysis (PCA).

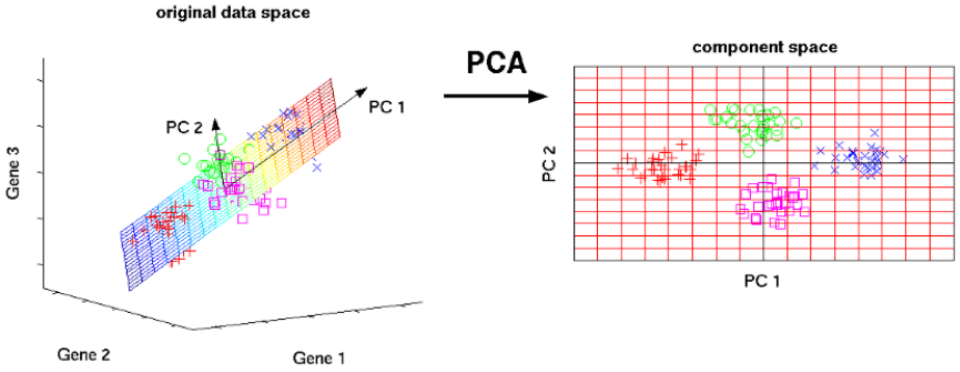


Figure 3.4.1: Visualization of Embedding using PCA. [33]

In the sections below we will explain some of the most notable embedding methods:

Principal Component Analysis

PCA is a standard mathematical technique used for dimensionality reduction. It focuses on finding dimensions of the input data which are highly correlated and therefore can be represented as one. PCA is defined as an orthogonal linear transformation which maps data to a new coordinate space. The dimensions of this space are computed as the eigenvectors corresponding to the greatest eigenvalues of the covariance features matrix. These vectors are chosen to be orthogonal, meaning the features are independent, and result in the least amount of error for approximating the data.

PCA captures linear correlations and therefore if the dataset has underlying non-linearities this approach will fail. Although simple in conception, if not performed appropriately it can lead to information loss [100].

Autoencoder

The *Autoencoder* is a dimensionality reduction method using neural networks. It succeeds in overcoming PCA's limitations since it basically is a non-linear generalization of it. Specifically, if it were to be linear it would produce any orthogonal basis in a non-deterministic way.

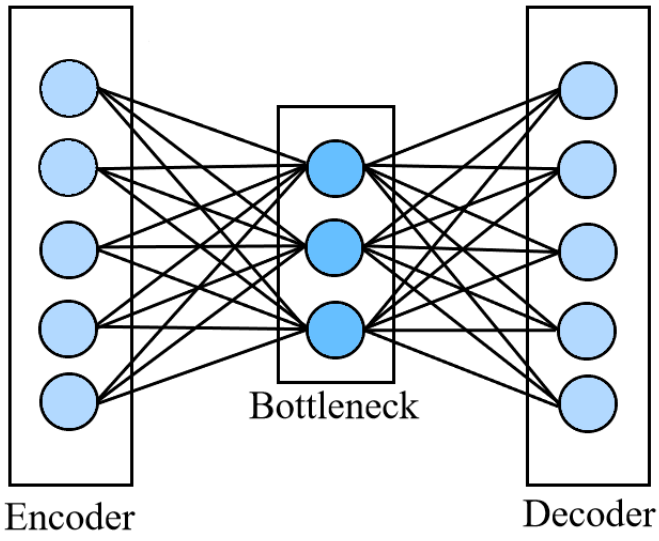


Figure 3.4.2: Autoencoder Architecture.

The autoencoder is a bottleneck architecture whose framework is composed by an *Encoder* and a *Decoder*. The first component transforms the high-dimensional input into a latent low-dimensional code, whereas the second reconstructs the input data using this code [30]. Both components are neural networks which are iteratively optimized using a custom loss function, called reconstruction loss, which computes the distance between the input and reconstructed data by the decoder.

The encoder and decoder can be stacks of different types of layers, even convolutional if the input is an image. There are various different models adopting the autoencoder architecture such as the undercomplete, sparse, contrastive and variational autoencoder. These models primarily differ on the construction of the loss function. For example, in the Variational Autoencoder there is a regularization term aiming to tackle overfitting. This feature combined with the encoding of the input as a distribution - and not points - makes this particular variant well-suited not only for the embedding of existing data but also for the generation of new ones.

This architecture provides a trainable way of embedding data. Its most important feature is the idea of the bottleneck, which only lets vital semantic information go through and eventually be reconstructed. Control of the bottleneck layer by size reduction helps to alleviate overfitting. The choice of depth and dimensions is also crucial.

Trained Embeddings as Part of a Larger Model

Another way of producing embeddings is implicitly while training a neural network for another task. A way to achieve this is by adding a special type of layer, defined by the library used to train, with dimension d in order to create a d-dimensional embedding. Alternatively, the embeddings could be extracted by any other given layer deemed fit for capturing data semantics.

This approach provides specific embeddings which capture semantic similarity of the input data determined by the training task. In many applications, like the one presented in this thesis, this is a desirable trait. However, in most cases the training of the larger model is more computationally expensive than the training of embeddings separately [14].

3.5 Large Language Models

Large Language Models (LLMs) are advanced machine learning models that are designed to understand and generate human language. LLMs are typically built using deep learning techniques, and they are trained on vast amounts of text data to develop the ability to generate coherent and contextually appropriate text. These models have become foundational in natural language processing (NLP) tasks due to their ability to capture the complexities of language, such as syntax, semantics, and even pragmatics. Recent works show that LLMs are able to also solve riddles and puzzles [70], showcasing their reasoning capabilities [71].

3.5.1 LLM Architecture

At the core of LLMs are neural network architectures, particularly transformer-based models. The transformer model, introduced by Vaswani et al. in [91], revolutionized NLP by allowing models to process sequences of text in parallel rather than sequentially, making them more efficient and scalable.

The LLMs used in this work, such as GPT [112] and BERT [15] are composed of multiple components that allow them to process, generate, and understand natural language. This section delves deeper into the mathematical formulations of the key parts of their architecture.

Tokenization and Embedding Layer

Tokenization: Text is split into tokens, which can be words, subwords, or characters. Let $x = (x_1, x_2, \ldots, x_n)$ represent an input sequence of n tokens.

Embedding: Each token x_i is transformed into a dense vector $e_i \in \mathbb{R}^d$ using an embedding matrix $E \in \mathbb{R}^{V \times d}$, where V is the vocabulary size and d is the embedding dimension. The embedding for the entire input sequence can be represented as:

$$E(x) = (e_1, e_2, \ldots, e_n)$$

where $e_i = E[x_i]$.

Self-Attention Mechanism

The self-attention mechanism is central to transformer-based LLMs. For each token in the input sequence, the attention mechanism computes a weighted sum of the other tokens, determining how much focus the model should place on each token when processing the sequence.

For a given input $x = (x_1, x_2, \ldots, x_n)$, the attention mechanism relies on three matrices: the query (Q), key (K), and value (V) matrices. These matrices are computed as:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

where $X \in \mathbb{R}^{n \times d}$ is the input embedding matrix, and $W^Q, \in \mathbb{R}^{d \times d_q}$, $W^K, \in \mathbb{R}^{d \times d_k}$, $W^V, \in \mathbb{R}^{d \times d_v}$ are learned parameter matrices that project the input embeddings into the query, key, and value spaces, respectively.

The attention scores are computed by taking the dot product of the query with the keys, scaled by the square root of the dimensionality d_k, followed by a softmax operation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

This equation computes a weighted sum of the value vectors, where the weights are determined by the similarity between the query and key vectors. The softmax function ensures that the attention weights sum to 1, giving a probabilistic interpretation to the weights. An visual representation of the self-attention mechanism can be seen in Figure 3.5.1.

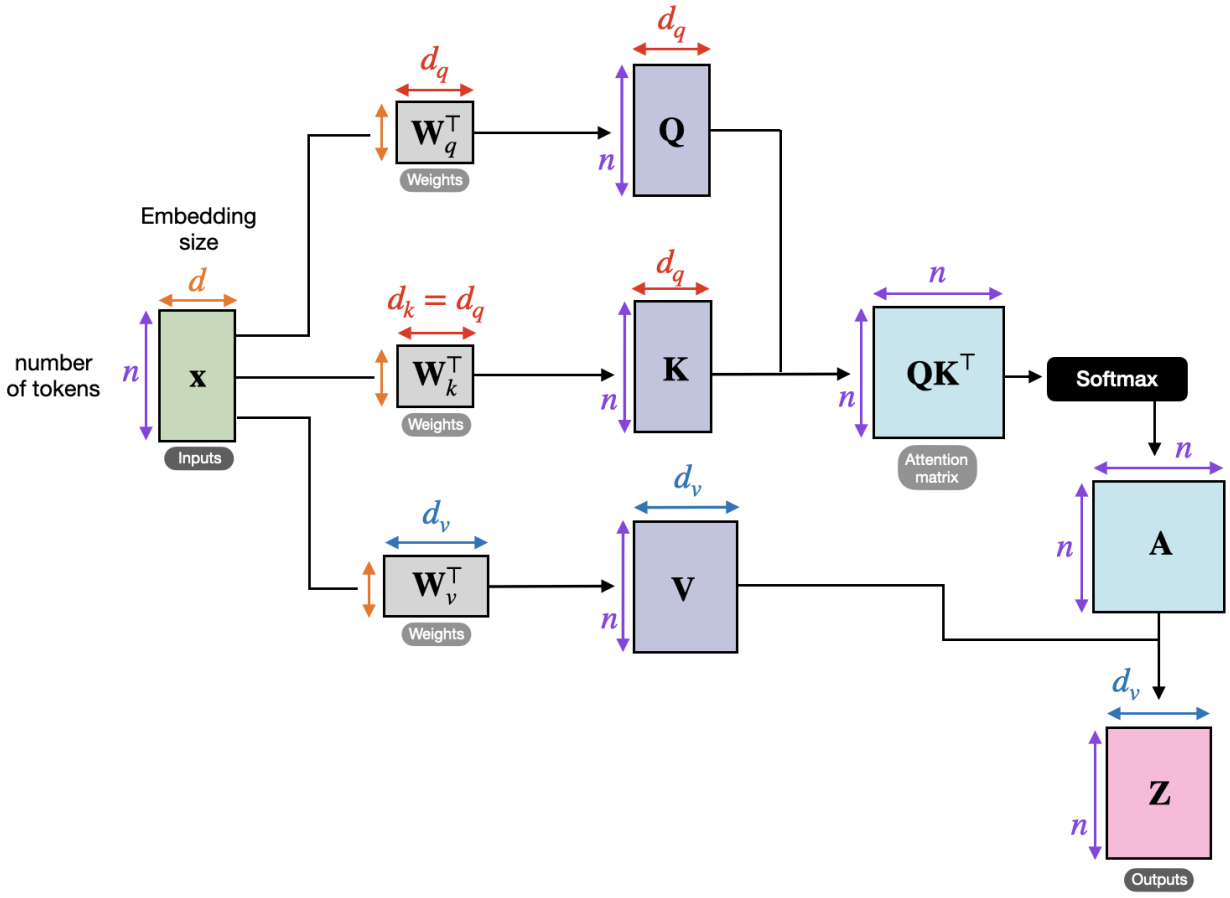


Figure 3.5.1: Self-Attention Mechanism

Multi-Head Attention

Instead of computing a single set of attention scores, LLMs use multi-head attention to capture different types of relationships between tokens. In multi-head attention, multiple attention mechanisms (heads) are run in parallel, and their outputs are concatenated:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W^O$$

where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

and $W^O \in \mathbb{R}^{hd_k \times d}$ is a learned output projection matrix. The use of multiple heads allows the model to attend to different parts of the input in different ways, increasing its representational power.

Position Encoding

Transformers do not have an inherent notion of word order, so position encoding is added to capture the sequential nature of the input. A positional encoding matrix $P \in \mathbb{R}^{n \times d}$ is added to the input embeddings. The positional encoding is typically defined using sinusoidal functions:

$$P_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right), \quad P_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d}}\right)$$

where i is the position and j is the dimension. This encoding allows the model to differentiate between positions in the sequence while maintaining generalization to longer or shorter sequences.

Feed-Forward Layer

Each transformer block includes a feed-forward network (FFN) applied independently to each position. The FFN consists of two linear transformations with a non-linearity in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

where $W_1 \in \mathbb{R}^{d \times d_{ff}}$ and $W_2 \in \mathbb{R}^{d_{ff} \times d}$ are learned weight matrices, $b_1 \in \mathbb{R}^{d_{ff}}$ and $b_2 \in \mathbb{R}^d$ are bias terms, and $\max(0, \cdot)$ represents the ReLU activation function. The FFN is applied separately to each token in the sequence.

Final Output Layer

The final output of the transformer model is passed through a linear transformation and softmax layer to generate probabilities for each class or token prediction. Let h_i represent the hidden state of the i-th token after the transformer layers. The probability distribution over the vocabulary is computed as:

$$P(x_i|h_i) = \text{softmax}(h_i W^T + b)$$

where $W \in \mathbb{R}^{V \times d}$ is the learned weight matrix, and $b \in \mathbb{R}^V$ is the bias term.

3.5.2 Pretraining and Fine-Tuning

LLMs are typically pretrained on large text corpora using one of two main objectives:

- **Masked Language Modeling (MLM):** This objective is used in models like BERT. A random subset of tokens in the input is masked, and the model is trained to predict the masked tokens based on the surrounding context. Formally, for a masked token x_i, the loss is computed as:

$$\mathcal{L}_{\text{MLM}} = -\log P(x_i|x_{\text{mask}})$$

- **Causal Language Modeling (CLM):** This objective is used in models like GPT, where the model is trained to predict the next word in the sequence given the previous context. The loss function for a sequence of tokens $x = (x_1, x_2, \ldots, x_n)$ is:

$$\mathcal{L}_{\text{CLM}} = -\sum_{i=1}^{n} \log P(x_i|x_1, x_2, \ldots, x_{i-1})$$

After pretraining, the model can be fine-tuned on a specific task by minimizing a task-specific loss function, such as classification or sequence labeling.

3.5.3 Computational Complexity

The computational complexity of the attention mechanism in transformers is $O(n^2 d)$, where n is the sequence length and d is the dimensionality of the embeddings. This quadratic complexity arises because the attention mechanism computes interactions between all pairs of tokens in the input sequence.

Chapter 4

Graphs

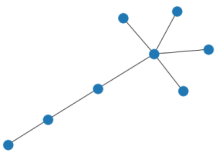
4.1 Graph Theory Basics

A graph, denoted G, is a non-linear data structure comprising of a set of objects called nodes or vertices which can be connected to one another, forming an ordered pair called an edge. In other words:

$$G(V, E) = \{(u, v) : u, v \in V, (u, v) \in E\} \tag{4.1.1}$$

where V is the set of vertices and E is the set of edges with $|V| = N, |E| = M$. In literature, the term "graph" is often used as a synonym for a simple graph, i.e. a graph without any self-loops (edges connecting a vertex to itself) and no more than one edge connecting any pair of vertices.

A visual representation of a graph can be seen in Figure 4.1.1a. However, graphs can also be described using an adjacency matrix A, a square array of dimensions $N \times N$ whose non-zero elements indicate the existence of a link between vertices. In some cases links between nodes can be assigned weights, which hold a relevant meaning to what the graph represents. Thus, the weight value for the corresponding edge would be present for each node pair in the adjacency matrix.



(a) Visual Representation

```
[[0 1 0 0 0 0 0 0]
 [1 0 1 0 0 1 1 1]
 [0 1 0 0 0 0 0 0]
 [0 0 0 0 1 1 0 0]
 [0 0 0 1 0 0 0 0]
 [0 1 0 1 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]]
```

(b) Adjacency Matrix

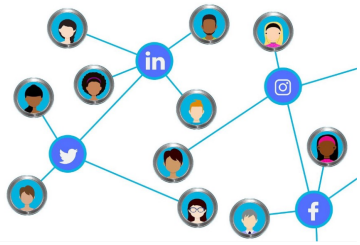
Figure 4.1.1: Representation of undirected graph.

A graph may additionally have node and/or edge attributes. In this case, each node (edge) is characterized by a feature vector of dimension D, resulting in a node (edge) feature matrix X of dimension $N \times D$ (X^e of dimension $M \times D$).

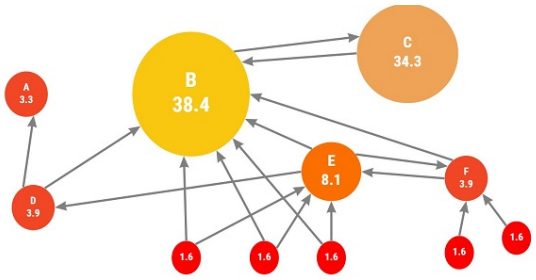
Graphs are categorized according to the direction of connections present between nodes. More specifically, an edge between nodes u, v is called undirected if both ordered pairs (u, v) and (v, u) belong in the set of edges if u and v are connected whereas is called directed if only one of them is present. Therefore, a graph is said to be undirected when all its edges are undirected and directed (or digraph) if at least one of them is not. A property of an undirected graph is the symmetry of its adjacency matrix.

The neighborhood $N(u)$ of an entity u in a graph is defined as the set of nodes adjacent to it. A path is a sequence of vertices $u_1, u_2, ..., u_n$ such that u_i and u_{i+1} are neighbors for all $1 \leq i \leq n - 1$. An undirected

(a) Molecule [26]



(b) Social Network [79]



(c) PageRank on websites [69]

Figure 4.1.2: Graph reprentation examples

graph is connected if every pair of nodes are connected through a path. In the case of digraphs, there are two distinct versions of connectedness, weak and strong. A directed graph is weakly connected if for every pair of vertices u and v there exists a path either from u to v or from v to u. In contrast, a strongly connected digraph should contain paths leading both ways.

A distinct feature of graphs that sets them apart from other types of data is the fact that they are generally non-euclidean. Non planar graphs cannot be mapped on a two-dimensional, non-curved space because they cannot be drawn on the plane so that links intersect only at their endpoints. This attribute which stems from the dependency of objects in graphs creates limitations in many established data processing models and results in the need for new ones.

4.2 Bipartite Graphs

In graph theory, a bipartite graph (or bigraph) is a graph whose nodes (or vertices) can be divided into two disjoint and independent sets S and T such that no two nodes within the same set are adjacent. Formally, a graph $G = (V, E)$ is bipartite if its node set V can be partitioned into two subsets U, W such that $U \cup W = V$, $U \cap W = \emptyset$ and where each edge $e_{u \to w} \in E$ connects a node $u \in U$ to a node $w \in W$. This structure makes bipartite graphs particularly useful in modeling relationships between two distinct types of entities.

The two sets U and W may be thought of as a coloring of the graph with two colors: if one colors all nodes in U with one color, and all nodes in W with a different one, each edge has endpoints of differing colors, as is required in the graph coloring problem [86]. This is shown in Fig. 4.2.1, where the node set U is colored in red and the node set W is colored in blue. In contrast, such a coloring is impossible in the case of a

non-bipartite graph, such as a triangle: after one node is colored blue and another red, the third vertex of the triangle is connected to vertices of both colors, preventing it from being assigned either color.

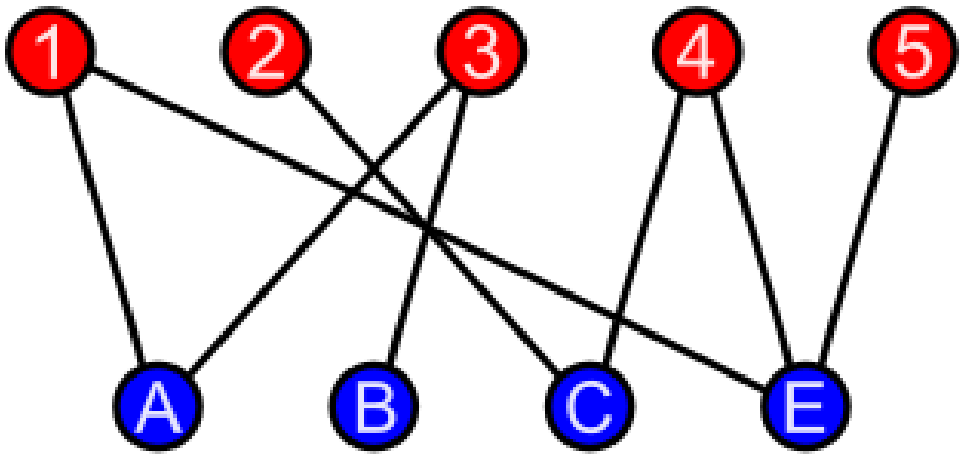


Figure 4.2.1: An example bipartite graph

We often write $G = (U, W, E)$ to denote a bipartite graph whose partition has the parts U and W with E denoting the edges of the graph. If a bipartite graph is not connected, it may have more than one bipartition; in this case, the (U, W, E) notation is helpful in specifying one particular bipartition that may be of importance in an application. If $|U| = |W|$ that is, if the two subsets have equal cardinality, then G is called a balanced bipartite graph. If all vertices on the same side of the bipartition have the same degree, then G is called biregular.

A bipartite graph $G = (U, W, E)$ is usually represented by its biadjacency matrix, which is a $(0, 1)$matrix of size $|U| \times |W|$. In that matrix, the value of the element ij is 1 if $\exists e_{i \to j} \in E : i \in U, j \in W$, or in simple terms if nodes i and j are adjacent. Otherwise, the element value is 0. In case the graph is weighted, element ij is equal to the weight of the edge connecting those two nodes.

One of the most prominent applications of bipartite graphs and the one for which they are used in this dissertation is in matching problems, such as job assignment, resource allocation, and network flow. In these contexts, nodes in one set represent agents or tasks, while nodes in the other set represent jobs, resources, or network nodes. The edges are usually weighted, with the weight value corresponding to a *cost*, such as the hours needed for an agent to execute a given task, etc. Bipartite graphs in this work are employed to replace words from a given text, where in this case we will refer to the two node sets as *source node set* and *target node set*.

Chapter 5

Graph Neural Networks (GNN)

It has been established from the previous section that the graph structure naturally emerges all around us. For this reason, neural networks operating directly on graph data were invented. Graphs are non-euclidean data and thus GNNs can be grouped in the broader category of Geometric Learning [7].

Graph Neural Networks (GNN) are known for their expressive power and recently have been gaining popularity due to their growing capabilities in various applications like recommendation systems and molecular fingerprinting. GNN applications range from chemistry and physics to traffic networks. These models can extract features from knowledge graphs and perform several graph mining tasks. They are widely used in computer vision for visual reasoning and semantic segmentation, in NLP for relation extraction or text classification and in combinatorial optimization to solve graph-related problems [118].

In the following sections we will review their unique features that make them so influential, explain how they are used and offer a review of the most important GNN variants.

Contents

5.1 Unique Characteristics

5.1.1 Motivation

The first question to be answered is why GNNs were created, even though there is already a substantial amount of neural network architectures. Most other data representations can be generalized to graphs but the opposite in not true. Most conventional Machine or Deep Learning algorithms are specifically crafted to cater to a certain type of data, such as images or text. Images can be perceived as fixed-size grid graphs and text or even speech can be thought of as line graphs. But in the general case, graphs are more complex, having a non-fixed number of unordered nodes within neighborhoods of variable size, and therefore existing models cannot handle them.

Another reason why machine learning is more complex on graphs is that most common algorithms assume instance independence. This is not true when performing node-level tasks where one graph is the input of the neural network and the instances are its nodes. Vertices are obviously related to one another with directed or undirected links.

The main motivation behind GNNs are Convolutional Neural Networks. Classic CNNs operate on images, or more broadly regular grids. As explained previously, they take advantage of the spatial locality of pixels which are nodes on the grid by sliding rectangular kernels with a small receptive field over the image to produce feature maps. The lack of locality in the traditional sense in graph data, their arbitrary size and invariance make regular convolution difficult to perform.

5.1.2 Permutation Invariance

Graphs are represented with adjacency matrices, a format that is permutation invariant. A Graph Neural Network is a transformation on nodes, edges or global-context that should preserve the graph symmetries - or permutation invariances - through the process of optimization.

More formally, for a graph with n number of nodes, their features can be represented by the feature matrix $X = [x_1, ..., x_n]^T$ where x_i are the feature vectors of each node. As explained the order of the nodes should not be important. However, by using this representation an ordering is unavoidably enforced. In order to overcome this problem the network should learn a function which is not affected by this ordering and therefore preserves permutation invariance.

A *permutation invariant* function f can be defined as:

$$f(PX) = f(X) \tag{5.1.1}$$

where X is any input matrix - in this case the feature matrix of a graph - and P is a permutation matrix. Permutation matrices contain exactly one non-zero element in each row and column and intuitively they scramble the rows of X resulting in different orderings.

A more thorough examination of the above function leads to the conclusion that operators like f are very limited. Basically such functions belong to a two-dimensional vector space containing summing the diagonal and summing the off diagonal of X. For example, a graph network consisting of only invariant layers would not even be able to discriminate between two graphs with the same number of nodes and edges, because it observes nodes as sets and not individually [59].

For all the above, in practice we want to combine invariant functions with equivariant functions which reflect the permutation of the node features in the output. *Equivariant operators* can be defined as:

$$f(PX) = Pf(X) \tag{5.1.2}$$

It is notable that a permutation in the node ordering could also be reflected on the adjacency matrix A of the graph. In this case the permutation will be enforced as PAP^T and result in $f(X, A)$ for invariance and $Pf(X, A)$ for equivariance.

Figure 5.1.1: Permutation in the adjacency matrix.

Left: Original Graph, Right: Graph obtained by permutation PAP^T

In practice graph neural networks often work on neighborhoods and not isolated nodes. For this reason, if we consider only the immediate 1-hop neighbors as the neighborhood N_i of i, the neighborhood feature matrix X_{N_i} would be a set of the neighbors' collective features. Each N_i would be processed separately using a local function g and finally the output would be a stack of the intermediate local results. To achieve f to be permutation equivariant, g should be invariant.

5.1.3 Weisfeiler-Lehman Test

The Weisfeiler-Lehman (WL) Graph Isomorphism Test [95] is used for the discrimination of non-isomorphic graphs. Specifically, it is able to distinguish between graphs that are not isomorphic, but cannot exactly provide proof that two graphs are in fact isomorphic. This capability gives significant insight on the graph's structure and aids in graph comparison.

In terms of graph theory, two graphs G and H are called isomorphic when an isomorphism exists between them. An isomorphism of graphs is a bijection between the vertex sets of G and H

$$f : V(G) \to V(H) \tag{5.1.3}$$

such that any two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H [97].

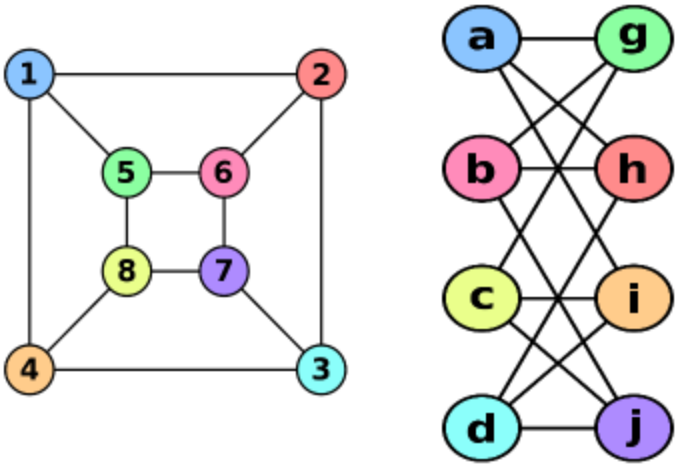


Figure 5.1.2: Two isomorphic graphs [97]

The principal idea of the Weisfeiler-Lehman test is to replace the label of each vertex with a multiset label consisting of the original and the sorted set of labels belonging to members of the neighborhood. This process

of relabeling is repeated for a given number of times simultaneously for all the graphs. If the outcome is graphs with different node labels, one can safely say that the input graphs were not isomorphic.

The problem that the WL test tackles is very challenging with a computational complexity in the class NP-intermediate. However, the WL test provides its answer in polynomial time. This test is also relevant to GNNs since it closely resembles the general idea of the GNN learning process of message passing. In fact, it was proven that a Graph Neural Network can at best be as good as the WL test at solving the graph isomorphism problem [109]. Thus, GNNs and the WL test are often compared based on their expressivity.

5.2 Taxonomy

Developments in the field of Geometric Learning in recent years have resulted in the creation of an abundance of different GNN models. Each one specializes in a certain task or offers a specific optimization. In order to explain the many GNN variants in the next section, we will explain on which bases they can be categorized:

5.2.1 Task Type

GNN variants tackle an array of graph analytics tasks each of which may focus on different attributes of the graph structure.

- **Node-Level tasks** require models which predict some property for each node in a graph, like its identity or role. With a broader perspective, node-level models build node representations. Analogous tasks would be image segmentation in computer vision or parts-of-speech prediction in NLP. In the supervised case, the most common tasks include node regression and node classification.
- **Edge-Level tasks**, in a similar fashion, call for models to predict the property or presence of edges. Subsequently, these tasks include edge classification and link prediction. The topic of this dissertation is also an edge-level task, where we try to predict whether an edge belongs to a desired set of edges or not.
- **Graph-Level tasks** entail the prediction of a single property for the whole graph or more vaguely attempt to extract graph representations. In order to do so, the model utilized is combined with pooling or readout operations. *Graph pooling* is a form of down-sampling which creates coarser graphs, whereas *Readout* instantly collapses node representations into a singular global graph representation. These tasks include graph classification or regression.

5.2.2 Architecture

Another way of categorizing Graph NNs is by prioritizing the type of framework or architecture they are using. This is the taxonomy [107] suggests and it is the following:

- **Convolutional graph neural networks (ConvGNNs)** inspired by classic CNNs are based on using the convolution operation, as defined for graph data. A basic outline of the function of the majority of these variants is: they define a node's neighborhood, aggregate information between neighboring nodes and finally generate each node's representation. Just like CNNs in standard Deep Learning, multiple graph convolutional layers are stacked in order to extract high-level node features the closer we get to the ouput. ConvGNNs are some of the most important building blocks for more advanced models. ConvGNNs can further be labelled as **spatial** or **spectral**. The distinction lies on whether the model makes use of spatial graph convolutions - spatial - or treats graphs as signals in the frequency domain - spectral.
- **Recurrent graph neural networks (RecGNNs)** make use of the idea of using information from previous units as influence for the current one, just like in classic Machine Learning. Specifically, they assume a node constantly trades information within its neighboorhood until a stable equilibrium is reached. This idea of *message passing* is one of the first in the field of GNNs and has been the basis for most other models, especially spatial-based convolutional ones.

- **Graph autoencoders (GAEs)** are based on the same premise as the Autoencoder explained in previous sections, i.e. are unsupervised learning frameworks which encode graphs/nodes into a low-dimensional space and attempt to reconstruct them. GAEs are mostly used on tasks like: network embedding - creating node representations through adjacency matrix reconstruction - and graph generation in a step by step manner.
- **Spatial-temporal graph neural networks (STGNNs)** are architectures which work on spatial-temporal graphs, i.e. graphs that change in structure overtime. To do so, they consider both spatial dependency as well as temporal dependency, often combining main ideas from convolutional and recurrent graph nets. They have recently become more significant due to their involvement in applications like traffic forecasting.

5.2.3 Training Type

Last but least, it is common to separate GNN variants according to what kind of signals they are trained with. This is is not a strict grouping since some models can be applied to multiple of the categories below.

- **Semi-supervised learning** tasks on graphs primarily work on graphs which are partially labeled, meaning only some nodes have a target class. The task in this case is node classification, which is achieved in a more robust way. Link prediction can also be carried out in a semi-supervised manner, in a similar fashion.
- **Supervised learning** tasks are based on the fact that the data is labeled. So they comprise of classification and regression tasks in node, edge or graph level.
- **Unsupervised learning** tasks are characterized by a lack of labels for the graph attributes. Some examples of tasks are node clustering and graph embedding. For the latter instance, the embedding can be learned in a purely unsupervised way using an end-to-end encoder-decoder based framework or by contrastive learning using negative samples. There also exist a lot of Graph Self-Supervised learning applications, which are mostly encoder-decoder based [52], and can be loosely grouped in this category.

5.3 GNN Models

In this section we will describe some of the most important Graph Neural Network architectures. Starting from the framework which introduced the concept of GNNs, we will expand to the many GNN variants and finally to general GNN frameworks.

5.3.1 Original Graph Neural Network

The original Graph Neural Network was presented by Scarselli et al. [84]. It falls under the category of RecGNNs and is an extension of prior recurrent models in order for them to work on graphs. It is based on the use of information diffusion within the graph in a recurrent manner until eventually a stable equilibrium is achieved.

Using notation from [118], the original model tries to learn a state embedding $h_v \in \mathbb{R}^d$ for every node v which includes not only its own information but also the neighborhood's N_v. It is defined as:

$$h_v = f(x_v, x_{co[v]}, h_{N_v}, x_{N_v}) \tag{5.3.1}$$

where x_v represents the features of node v, $x_{co[v]}$ are the features of its corresponding edges while h_{N_v} and x_{N_v} refer to the set of neighbors of v and represent their collective states and features respectively.

The function f is called *local transition function* and it is a parametric function common for all nodes which performs the state update by utilizing the neighborhood's information. In practice the neighborhood's information is summed in order to preserve invariance [107].

The state embedding can be used to produce an output related to the task. The output o_v is defined as:

$$o_v = g(h_v, x_v) \tag{5.3.2}$$

where g is the *local output function* and it describes how the output will be produced. The term function is used loosely and can even refer to an FFNN.

A compact representation of the above equations after stacking them for all nodes of the graph can be seen in Equation 5.3.3. H, O, X, X_N are the matrices corresponding to states, outputs, features and neighborhood features after the stacking, while F, G are the *global* versions of f, g which are also constructed by the combination of the local functions for each node.

$$\begin{aligned} H &= F(H, X) \\ O &= G(H, X_N) \end{aligned} \tag{5.3.3}$$

The state in each iteration of this recurrent model is computed using Banach's theorem [39], since H is the fixed point of the above equation. In order to find a unique solution, F must be a contraction map. H is randomly initialized and if the criterion for F is met, then the convergence is exponentially fast regardless of the initial value. After the fixed point is found, the last step node hidden states are forwarded to a readout layer extracting a global representation. The state update is defined as:

$$H^{t+1} = F(H^t, X) \tag{5.3.4}$$

In the case of supervised learning, the loss function looks like this:

$$\sum_{i=1}^{n} (t_i - o_i)^2 \tag{5.3.5}$$

As common in neural network applications, the cost function may include a penalty term to control other properties of the model. The optimization is gradient-based and determines the weights of the parametric functions f, g.

Overall, the learning algorithm iteratively updates the states using 5.3.3 until convergence to the fixed point. Then, gradients are computed and the weights are updated accordingly.

The original GNN presented crucial ideas which are utilized by many of the approaches which will be explained later and so is some of the terminology. However, it has some important limitations:

- The requirement of f being a contraction map limits the model's abilities.
- GNN is computationally expensive due to the iterative updates towards the fixed point.
- It is unsuitable for node-level tasks because of the fixed point requirement. The representation obtained is smooth and therefore not informative for each node.
- There is no representation of edge features.

5.3.2 Variants

In this section we will present some of the most important GNN variants presented in order to alleviate the aforementioned challenges or cater to unique cases. We will group them according to their architecture, as proposed in Section 5.2.2. STGNNs and RecGNNs will not be further elaborated on since they are out of the scope of this thesis.

Spectral-based ConvGNNs

As mentioned before, spectral ConvGNNs treat the graph as a signal and therefore are strictly mathematical approaches. They make use of the Laplacian matrix L of the graph, which captures its key properties, and use the Fourier transform to project the graph to the orthonormal space defined by the eigenvectors obtianed after factorizing the Laplacian. The formal definition of the Laplacian matrix can be seen below, where D is the degree matrix and A the adjacency matrix of the graph.

$$L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{5.3.6}$$

All spectral convolutional approaches for GNNs use the graph convolution defined in 5.3.7, but they differ in the choice of the filter g_θ. In this equation x is the input signal to be convolved, U is the matrix of eigenvectors ordered by eigenvalues resulting from the factorization of the normalized Laplacian matrix of the input graph and g_θ is the convolution filter, for which $g_\theta = diag(U^T g)$ is true.

$$x *_G g_\theta = U g_\theta U^T x \tag{5.3.7}$$

Spectral Convolutional Neural Network (Spectral CNN) [8] assumes that the filter is a diagonal matrix with learnable parameters. However, this initial approach suffers from computational inefficiency, dependence on the input graph's structure and non-spatial locality.

Chebyshev Spectral CNN (ChebNet) [12] approximates g_θ by Chebyshev polynomials of the diagonal matrix of eigenvalues Λ ($L = U \Lambda U^T$). The convolution can be defined as:

$$x *_G g_\theta = \sum_{k=0}^{K} \theta_k T_k(\tilde{L}) x \tag{5.3.8}$$

where $\tilde{L} = 2L/\lambda_{max} - I_n$ with λ_{max} being the largest eigenvalue of L and I_n the identity matrix of dimension n. $T_k(x)$ represents the Chebysev polynomial for the k-th order and in the above equation it can be proven by induction that:

$$T_k(\tilde{L}) = U T_k(\tilde{\Lambda}) U^T \tag{5.3.9}$$

$T_k(x)$ can be computed as:

$$T_k(x) = 2x T_{k-1}(x) - T_{k-2}(x) \quad \text{with} \quad T_0(x) = 1 \quad \text{and} \quad T_1(x) = x \tag{5.3.10}$$

This model improves on the spatial locality problem of Spectral CNN. Specifically, the filters defined by ChebNet are K-localized, since the operation is a K^{th}-order polynomial of the Laplacian, allowing this model to obtain local features regardless of the graph size.

Graph Convolutional Network (GCN) presents the idea of using a first order approximation of ChebNet in order to alleviate overfitting. In fact, it assumes $K = 1$ and $\lambda_{max} = 2$. In the same direction, the model enforces the constraint $\theta = \theta_0 = -\theta_1$. After imposing these restraints on Equation 5.3.9 and taking into consideration 5.3.10, the convolution operation is:

$$x *_G g_\theta = \theta(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) x \tag{5.3.11}$$

After empirically finding that the term $I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ causes numerical instability, a *renormalization* trick was used. The term $D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = \bar{A}$ was replaced with $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} = \hat{A}$ where $\tilde{A} = I_n + \bar{A}$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. All of the above can be described by this compact equation:

$$H = X *_G g_\theta = f(\hat{A} X \Theta) \tag{5.3.12}$$

where f is an activation function and multiple inputs and ouputs are allowed due to the matrix form.

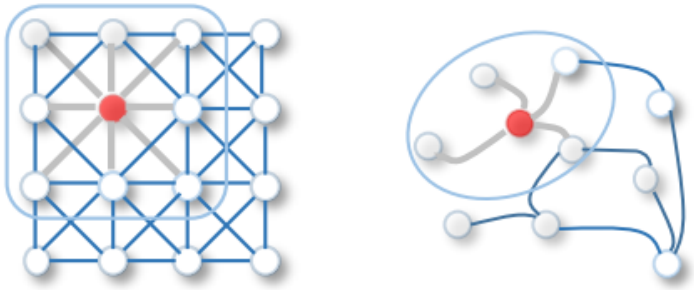


Figure 5.3.1: Comparison of 2D and Graph Convolution [107]

Left: 2D Convolution in CNNs, Right: Graph Convolution

The GCN is a special case of a spectral approach since it can also be perceived as a spatial one. In the equation below, we can see how the aggregation of information within the neighborhood would be performed. In this case the node itself is also regarded as its 1-hop neighbor.

$$h_v = f(\Theta^T(\sum_{u \in N(u) \cup v} \hat{A}_{v,u} x_u)) \quad \forall u \in V \tag{5.3.13}$$

This model is very frequently used as a part of more complex architectures in literature due to its simplicity and good experimental performance.

Spatial-based ConvGNNs

Spatial approaches define the graph convolution operation taking into account the spatial locality of nodes, i.e. the neighborhood, making them adjacent to conventional CNNs. However, they are also heavily inspired by the RecGNN pioneering ideas. Specifically, spatial ConvGNNs convolve the central and neighboring nodes' features, as seen in Figure 5.3.1, to obtain the updated representation which ultimately leads to information propagation along the edges of the graph.

Diffusion Convolutional Neural Networks (DCNN) [3] conceive the convolution as a diffusion process and define the diffusion convolution function as:

$$H^{(k)} = f(W^{(k)} \odot P^k X) \tag{5.3.14}$$

where f is an activation function and $P = D^{-1}A$ and is called the *probability transition matrix*. During the information distribution each node exchanges information with one of its neighbors with a certain transition probability provided by P. After an amount of iterations an equilibrium is reached and the diffusion is over. The update rule of the original GNN is not used here, instead each hidden representation $H^{(k)}$ is computed independently and all of them are concatenated to obtain the final one.

GraphSAGE [29] learns a function that generates embeddings by sampling and aggregating features from a node's local neighborhood. This framework introduces the idea of sampling the node's neighborhood in order to obtain a fixed number of neighbors in each iteration. The convolution process can be summarized in:

$$h_v^{(k)} = \sigma(W^{(k)} \cdot f_k(h_v^{(k-1)}, \{h_u^{(k-1)}, \forall u \in S_{N(v)}\})) \tag{5.3.15}$$

where f_k is an aggregation function, σ is the sigmoid function and $S_{N(v)}$ is a random sample of the neighborhood of v. We assume that h_v is initialized with the node's original features. The choice of aggregation function is important since it should be permutation invariant, such as a mean, sum or max function.

GraphSAGE is practically an extension of the GCN to inductive unsupervised learning. It is proposed that it should be trained using an unsupervised graph-based loss, but can also be trained in a supervised manner if needed. The model provides low-dimensional embeddings for downstream tasks which reflect local and global characteristics of the graph. It focuses on attributed graphs - with features - but can also work without node attributes by using handcrafted structural information. GraphSAGE does not train the embeddings, but the aggregator function with which inference is performed.

This approach is more efficient for unseen data than older transductive approaches and it is also invariant to orthogonal transformations. With adjustments it is an instance of the Weisfeiler-Lehman test.

Graph Attention Network (GAT) [92] adopts the idea of attention proposed by [91] in order to decide which members of a node's neighborhood have more important information. It aims to learn the relative weights between adjacent nodes and therefore differs from previous approaches like GCN and GraphSAGE because the concept of the neighborhood is not pre-defined or identical.

The convolutional operation is defined as:

$$h_v^{(k)} = \sigma(\sum_{u \in N(u) \cup v} \alpha_{vu}^{(k)} W^{(k)} h_u^{(k-1)}) \tag{5.3.16}$$

where the attention weights for each node v can be defined as:

$$\alpha_{vu}^{(k)} = softmax(LeakyReLU(a^T [W^{(k)} h_v^(k-1) || W^{(k)} h_u^{(k-1)}])) \tag{5.3.17}$$

The variable a represents a set of learnable parameters. The hidden representation is initialized with the features of each node and the softmax function ensures that attention weights sum to one.

The mechanism above is called *self-attention*, but GAT additionally uses *multi-head attention* in order to stabilize learning and make the model more expressive. The exact equations can be found in [92].

GAT is efficient since the node-neighbor pairs can be computed simultaneously. Moreover, it is indifferent to neighborhood sizes and can be applied to inductive learning problems easily.

Graph Isomorphism Network (GIN) [109] is the first spatial approach that addresses the inability of previous spatial models to discriminate between different graph structures based on the embeddings produced. In order to do that, GIN uses a simple technique, adding a learnable weight parameter for the central node of the convolution. The operation is defined below where $\epsilon^{(k)}$ is the weight.

$$h_v^{(k)} = MLP((1 + \epsilon^{(k)}) h_v^{(k-1)} + \sum_{u \in N(u)} h_u^{(k-1)}) \tag{5.3.18}$$

GIN is proved to be as powerful as the WL graph isomorphism test, i.e. produces different node embeddings when dealing with non-isomorphic graphs This makes this model maximally powerful and therefore the most expressive among the variants. The most discriminative GNN should use an injective multiset function in order to distinguish rooted subtree structures. For this reason, GIN uses the Multi-Layer Perceptron and the sum function as the aggregator. For each layer, node embeddings are summed and the result is concatenated. Thus, the expressiveness of the sum operator is combined with the memory of previous iterations by using concatenation. All things considered, one should keep in mind that the theoretical power of the GIN does not always translate in practice

GAEs

In this section we will present the most popular graph autoencoder architectures presented in the same paper [40].

Graph AutoEncoder (GAE) improves on previous autoencoder-based techniques by leveraging the information provided by node features. It uses two GCN layers to capture node structural and node feature information at the same time as seen in the equation for the encoder below.

$$Z = enc(X, A) = Gconv(ReLU(Gconv(A, X; \Theta_1)); \Theta_2) \tag{5.3.19}$$

where Z is the graph's embedding matrix.

The decoder aims to utilize the embeddings in order to reconstruct the graph adjacency matrix. The model is trained with a CrossEntropy loss between the real adjacency matrix A and the reconstructed adjacency matrix $\hat{A}$. Its equation is visible below, where z_v refers to the embedding of node v.

$$\hat{A}_{v,u} = dec(z_v, z_u) = \sigma(z_v^T z_u) \tag{5.3.20}$$

Variational Graph AutoEncoder (VGAE) is the variational version of the previous model which learns the data distribution. It mainly attempts to alleviate overfitting, but additionally gives the opportunity for the model to be used for generative tasks.

Aside from employing distributions, the main difference from GAE is the use of the Kullback-Leibler divergence function which measures the distance between two distributions as a regularizer, in a similar fashion to the conventional VAE.

Special GNN variants

All of the variants above are made for static homogeneous graphs with node features. But graphs in real life are not always that simplified. For this reason, dedicated efforts have been made to create GNN variants for complex graphs. These efforts can significantly contribute to the adoption of GNNs in a broader range of applications [56].

Heterogeneous graphs are commonly used to represent the relations between papers or authors. For this reason, many GNN approaches, such as the GAT-filter, have been modified to accommodate them. This is possible with the use of meta-paths [31] which capture various relations between nodes with different semantics. The original graph is split into a set of homogeneous graphs which are dealt with separately. In a similar manner, if a graph is bipartite the graph convolution operation is split into two components, one for each set of nodes.

Multi-dimensional graphs are graphs containing different types of edges.In this case, it is necessary to consider both within and across-dimension interactions, i.e. relations between nodes which are of the same type as well as are not. Towards this direction, [57] was presented. Moreover, the type of graph containing two types of relations denoted as positive and negative is called signed and is a common representation used in social network theory. For those types of graphs we cannot treat the positive and negative subgraphs independently because they interact. Works like [13] leverage the balance theory to model these interactions.

Hypergraphs, also often representing author and paper networks, are graphs containing hyperedges, i.e. edges connecting any number of nodes. These structures can be transformed into simple graphs by pairwise relation extraction, as proposed by [19, 110].

Finally, graphs can also have temporal characteristics meaning they can evolve overtime. Dynamic graphs are graphs with many real world applications and require learning multiple models for different snapshots in time. An interesting recent dynamic or STGNN model is [73].

5.3.3 General Frameworks

The existence of such an abundance of GNN variants led to the creation of various general frameworks which group them together. This way, we can study them more efficiently and draw significant conclusions. In this section, we will coarsely describe some of the most important ones belonging to the category of spatial ConvGNNs.

Mixture model network (MoNet) [65] is a framework combining several non-euclidean models, such as CNNs for manifolds and GNNs. A new pseudo-coordinate system is created, with its origin being each point on a manifold or each vertex on a graph. The neighboring points/nodes are defined accordingingly. GCN can be formulated as an instance of this framework.

Message Passing Neural Network (MPNN) [24] is probably the most well-known family of GNNs. This framework uses two phases: the message passing phase and the readout phase for graph-level tasks.

In the first phase, information is aggregated to each node's neighbors by using a message function M_k which is essentially the graph convolution. Then, an update function U_k is used to update the hidden state. This is a K-step process defined by the equation below.

$$h_v^{(k)} = U_k(h^{(k-1)}, \sum_{u \in N(v)} M_k(h_u^{(k-1)}, x_{vu}^e)) \tag{5.3.21}$$

The readout phase is optional depending on whether node or graph-level inference is required. In the second case, the representation $h_v^{(K)}$ is passed to a readout function in order to create a graph representation.

$$h_G = R(h_v^{(k)} | u \in G) \tag{5.3.22}$$

where R is the readout function.

MPNN can describe a variety of different spatial convolutional GNNs which generally adopt this message passing process. This is done by defining the functions M_k, U_k and R in appropriate ways. One of the networks belonging to this group is GCN and also several other models used on molecular structures which have not been mentioned.

It is notable that GIN has been proven to be more powerful than all MPNN-based methods at least in theory.

Non-Local Neural Network (NLNN) [93] uses a non-local operation to compute the hidden state at a position as a weighted sum of features at all possible positions in space, time or spacetime. Consequently, the NLNN essentially groups all "self-attention" methods, including GAT.

Graph Network (GN) [4] is an even more general framework which groups models according to the type of task they fulfill, i.e. node-level, edge-level and graph-level. It generalizes other frameworks, including the aforementioned ones.

This framework is based on the use of a computational unit called GN block. This block defines three update and three aggregation functions, each referring to a different level. By choosing these functions, a significant array of models or other frameworks can be defined.

Chapter 6

Counterfactual Explanations

The field of AI interpretability has been gaining increasing attention due to the growing realization that in order to confidently be able to count on the impressive outputs provided by intelligent systems, appropriate explanations are needed [2]. AI applications are taking over almost every aspect of everyday life and it is significant to assure that these models and the respective input datasets are not biased. AI Explainability is crucial not only for detecting biases [89, 53], hallucinations [28], and robustness issues [41] in several tasks, but also for increasing social acceptance, establishing safety measures for applications like self-driving cars and finally for the enhancement of the machine learning systems themselves using the information learnt [64].

In this thesis, we will focus on a specific explainability technique called Counterfactual Explanations. In the following sections we will define this type of AI explanations and describe the way they relate to our proposed framework.

Contents

6.1 Definitions

A general definition for a *counterfactual explanation* - or simply a *counterfactual* - is that it describes the causation of a situation by assuming "If X had not happened, Y would not have occurred". Its name derives from the need to imagine a counterfeit reality which contradicts our perception [64].

In terms of AI explainability, the counterfactual aims to provide an explanation for "What would need to change in order for the model to make a different decision". Therefore, they essentially can explain predictions of individual instances, where the causes of the predicted outcome are particular feature values of this instance.

Counterfactuals are contrastive and selective, meaning they find minimal changes in the feature space. Thus, they are human-friendly. However, there are usually multiple different counterfactual explanations for the same instance which explain it equally well. This creates contradictions which can be overcome simply by providing all possible truths and letting the end user decide on establishing a criterion to choose the best one.

There are both *model-agnostic* and *model-specific* counterfactual explanation methods. Model-agnostic approaches have a chief advantage over solely using interpretable models, which is their flexibility. They can be tested on several different machine learning models and evaluated on the explainability of an array of tasks. The method which we will be focusing on is also model-agnostic.

A good counterfactual explanation should first and foremost be able to produce the predefined prediction. It should provide a minimal explanation, i.e. the closest one in terms of features as determined by some distance metric. Moreover, its features should have possible values. Finally, in some cases it is desirable for a CE method to be efficient, meaning that an optimal solution should be reached using non-exhaustive search techniques. In this work, these requirements are met by viewing counterfactuals as a combinatorial optimization problem, solvable via graph assignment algorithms from graph theory [111]

6.2 Counterfactual Interventions

In this section, we will elaborate on the counterfactual explanation method which inspired this work. Essentially, our goal is to replace words from the original text with appropriate substitutes, so that the classifier's prediction is changed. This method, known as *counterfactual interventions*, provides a form of Counterfactual Explanations when we are dealing with textual data.

Counterfactual interventions aim to change the prediction of the model, by slightly altering the original input. Formally, given an input x and a model f such that $f(x) = y$, a counterfactual intervention seeks a perturbed input x' such that $f(x') = y' \quad \text{with} \quad y' \neq y \quad$ and the perturbation $\delta(x, x')$ is minimal according to some distance metric δ. The goal is to generate x' that is close to x in terms of its features, yet sufficiently different to change the model's output. In our work, we focus on semantic changes, following the paradigm of word-level perturbations.

Counterfactual interventions are particularly challenging due to the discrete and high-dimensional nature of text. In fact, creating optimal linguistic interventions is an algorithmically challenging problem, requiring efficient optimization of the search space of alternatives [115, 94, 54, 113]. Another obstacle that needs to be overcome is the fact that generating counterfactuals for text requires semantically meaningful edits that alter the model's prediction while maintaining the text's fluency and coherence. Thus, counterfactual interventions in NLP must satisfy several key criteria:

1. **Contrastiveness**: The counterfactual text x' should result in a different model prediction than the original text x. This change in prediction highlights the model's sensitivity to certain parts of the text, offering insights into what drives the model's decisions.
2. **Minimality**: The edits required to transform x into x' should be minimal according to some metric.
3. **Fluency**: The edited text x' must be grammatically correct and semantically coherent. Maintaining fluency is essential to ensure that the counterfactual is not only computationally valid but also interpretable and meaningful to human users.

4. **Plausibility**: The counterfactual should remain within a plausible range of the original input. For example, in a sentiment analysis task, changing a single word that significantly alters the sentiment but retains the original context and meaning, is preferable to completely rewriting a sentence.

In this dissertation, we focus mainly on *contrastiveness*, *minimality* and *fluency*. Plausibility requirement is also met as the side-effect of two things: the use of word-level perturbations and the adoption of number of edited words as the minimality metric. The former guarantees that words and not phrases will be changed, while the latter ensures that changed words will be as few as possible.

6.3 Related Work

6.3.1 Textual Counterfactuals

After examining the concepts of counterfactual interventions as a form of Counterfactual Explanations, we are going to present other relevant work based on textual counterfactuals, use of Large Language Models (LLMs) and the utilization of graph structures and algorithms.

Exposing vulnerabilities present in SoTA models has been an active area of research [90], endorsing the probing of opaque models through adversarial/counterfactual inputs. Granularity of perturbations ranges from character [18] to word level [23, 78] or even sentence level [34]. In our work, we focus on semantic changes, following the paradigm of word-level perturbations.

Manual creation of adversarial examples has been explored [22, 37, 67] with the purpose of changing the true label. Automatic text generation initially implemented via paraphrases [32], and most recently using masked language modelling [46, 80, 47], targets predicted label changes in binary/multi-label classification or textual entailment setups. Similarity-driven substitutions based on word embedding distance [35, 119] ensure optimality in local level for classification tasks, while constraint perturbations guarantee controllability of adversarials [66]. Those works partially preserve some desiderata of our approach; however, they are model-specific and thus constrained.

General purpose counterfactual generators fine-tune LLMs to offer diverse perturbations, applicable in multiple granularities [106, 25, 81]. Prompting on LLMs opens novel trajectories for textual counterfactuals [10, 83], even though explainability of interventions is completely sacrificed, due to the unpredictability of LLM decision-making. Overall, utilizing LLMs is computationally expensive, while produced substitutions may not be optimal as far as word distance is concerned [21]. On the other hand, interventions through the use of graph-related optimizations [115, 54] have recently emerged, showcasing that advanced performance and explainability of interventions are on par with computational efficiency.

6.3.2 Counterfactual explanations using GNNs

The utilization of GNNs in computing counterfactual explanations has been rather underexplored in literature, since the creation of the graph itself may be a strenuous process. Nevertheless, some recent endeavors [17, 16] suggest leveraging GNNs for computing counterfactuals for visual classifiers: by representing images as graphs, counterfactual explanations are equivalent to calculating the graph edit distance (GED) between two graphs. In that case, the closest graph to a reference one serves as the counterfactual. Finally, the edits suggested to transit from the reference graph to the counterfactual one serve as the explanations regarding what it needs to be changed to perform the desired transition. Recent literature scrutinizes both supervised GNNs for GED calculation [17], as well as unsupervised autoencoders [16] for more computationally efficient computation.

Chapter 7

Rectangular Linear Assignment Problem

One core component of this work is lying on the Rectangular Linear Assignment Problem (RLAP) and its solution. The rectangular assignment problem is a generalization of the linear assignment problem (LAP); one wants to assign a n number of tasks to $m \geq n$ number of agents, minimizing the total corresponding costs [6]. Applications are, e.g., in the fields of object recognition and scheduling. In our case, we apply the problem in the form of a weighted minimum matching on a pre-constructed bipartite graph (see Sec. 4.2).

Contents

7.1 Problem Formulation

The Rectangular Assignment Problem can be formally defined as follows:

Given two sets T (tasks) and A (agents), where $|T| = n$ and $|A| = m$ and a cost matrix C of size $n \times m$ where each element c_{ij} represents the cost of assigning task i to agent j, the goal is to find an optimal assignment that minimizes the total cost. An assignment can be represented by a binary matrix Y of the same size, where $y_{ij} = 1$ if task i is assigned to agent j and $y_{ij} = 0$ otherwise. The problem can be expressed mathematically as:

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} y_{ij} \tag{7.1.1}$$

with the following constraints:

1. Each task is assigned to at most one agent:

$$\sum_{i=1}^{n} x_{ij} \leq 1, \quad \forall i \in T \tag{7.1.2}$$

2. Each agent is assigned to at most one task:

$$\sum_{i=1}^{n} x_{ij} \leq 1, \quad \forall j \in A \tag{7.1.3}$$

When $n = m$ the problem is *linear*, while when $n \neq m$ the problem is considered *rectangular*. However, it has been proven that each rectangular assignment problem can be converted to a linear one, by adding extra rows or columns with zero or infinite values [11].

As part of this dissertation, we attempt to solve the RLAP applied on a bipartite graph, which is known as the *minimum weighted bipartite graph matching*. Let's consider a weighted bipartite graph $G = (V, E)$, where the edge set E consists of all the weighted edges in the graph, and the node set V consists of the source set S of cardinality $|S| = n$ and the target set T of cardinality $|T| = m$, such that $S \cup T = V$, $S \cap T = \emptyset$. Finding optimal connections between nodes of G has been a long sought discrete optimization problem of graph theory, where the optimal match for each node $s \in S$ needs to be determined among a predefined candidate set of nodes $t \in T$. Assuming that W denotes the edge weight set consisting of the weights of all edges $e \in E$, a *min weight matching* $M \subseteq E$ searches for a subset of the lightest possible sum of edge weights $\sum w_e, w_e > 0 \in W$ containing those edges $e \in E$ that cover all nodes of the $min(|S|, |T|)$ set of G. Therefore, in the case of $|S| \leq |T|$, all nodes in S will be assigned to a node in T, should an outcoming edge $e_{s \to t}$ exists from each s to any $t \neq s$. Under these requirements, we formulate the following constraint optimization problem:

$$min \sum w_e, \ \textit{subject to } s \neq t \ \textit{if } \exists e_{s \to t} \tag{7.1.4}$$

Equation 7.1.4 is simply a different form of equation 7.1.1, thus our problem can be converted to RLAP and subsequently to LAP. Therefore, the same algorithms can be applied in both cases. With that in mind, in the following sections we will examine some of them, along with a recently emerged non-deterministic approach which is based on Graph Neural Networks (GNNs).

7.2 Deterministic Approaches

A naive solution to this constraint optimization problem would be the exhaustive search of all possible (s, t) combinations, by examining all possible $m!$ permutations of T until the optimal solution of $min \sum w_e$ is reached. This yields an exponential complexity of $O(m^n)$, supposing that G is complete, i.e. each pair of $s - t$ nodes is connected so that $E = S \times T, |E| = nm$. To understand this complexity, assume the bipartite graph of Figure 7.2.1 with $S = \{A, B, C\}$ of cardinality $|S| = 3$ and $T = \{1, 2, 3, 4\}$ of cardinality $|T| = 4$, the following node combinations occur:

Source node A can take $|T| = 4$ values: A-1, A-2, A-3, A-4. Node B can independently of A take $|T| = 4$ values: B-1, B-2, B-3, B-4. Finally, C independently of A and B can also take $|T| = 4$ values: C-1, C-2, C-3, C-4. Therefore, all combinations for the $|S| = 3$ source nodes are $4 \times 4 \times 4 = 4^3 = |T|^{|S|}$

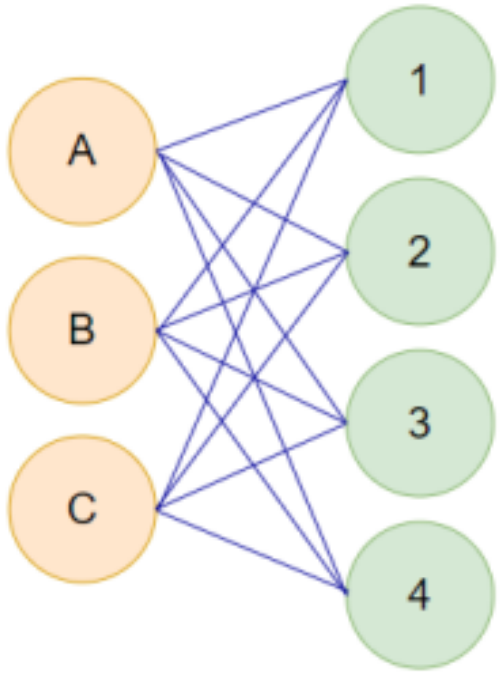


Figure 7.2.1: Example graph

7.2.1 Hungarian Algorithm

The Hungarian algorithm [43], also known as the Kuhn-Munkres algorithm, is a combinatorial optimization method designed to solve the assignment problem efficiently in polynomial time. Originally developed by Harold Kuhn in 1955 and later improved by James Munkres, this algorithm finds an optimal assignment that minimizes the total cost or maximizes the total profit of assigning a set of source nodes S to a set of target nodes T.

The Hungarian algorithm consists of several steps to achieve an optimal assignment:

1. **Subtract Row Minimums:**

   For each row in the cost matrix C, subtract the smallest element in that row from all elements in the same row. This step transforms the cost matrix such that each row has at least one zero.

$$c_{ij} \leftarrow c_{ij} - \min_k(c_{ik}), \quad \forall i = 1, 2, \ldots, n, \quad \forall j = 1, 2, \ldots, m$$

2. **Subtract Column Minimums:**

   After processing rows, for each column in the cost matrix, subtract the smallest element in that column from all elements in the same column. This step ensures that each column has at least one zero.

$$c_{ij} \leftarrow c_{ij} - \min_k(c_{kj}), \quad \forall j = 1, 2, \ldots, m, \quad \forall i = 1, 2, \ldots, n$$

3. **Cover All Zeros with Minimum Number of Lines:**

Using a minimum number of horizontal and vertical lines, cover all zeros in the resulting matrix. The minimum number of lines required to cover all zeros gives an indication of the progress towards finding an optimal assignment.

4. **Test for Optimality:**

   If the minimum number of lines required to cover all zeros equals the smaller of n or m, an optimal assignment exists among the zeros. If this condition is met, move to step 7. Otherwise, proceed to step 5.

5. **Create Additional Zeros:**

   Identify the smallest element in the matrix that is not covered by any line. Let this minimum uncovered value be δ. Subtract δ from all uncovered elements and add δ to all elements that are covered by two lines. This operation introduces additional zeros into the matrix while maintaining existing zeros.

$$c_{ij} \leftarrow \begin{cases} c_{ij} - \delta, & \text{if } (i,j) \text{ is not covered by any line,} \\ c_{ij} + \delta, & \text{if } (i,j) \text{ is covered by two lines.} \end{cases}$$

6. **Repeat Steps 3-5:**

   Repeat the process of covering all zeros and creating additional zeros until the minimum number of lines required to cover all zeros equals the smaller of n or m.

7. **Make the Optimal Assignment:**

   Construct the optimal assignment by selecting zeros such that no two selected zeros are in the same row or column. This set of zeros represents the optimal assignment. Each zero in this set corresponds to an source-target node pair with minimized total cost.

The Hungarian algorithm has a time complexity of $O(n^3)$, where $n = \min(n, m)$. This cubic time complexity makes it efficient for moderate-sized problems and significantly faster than the naive brute-force approach previously discussed. For very large problems however, especially those involving thousands of source and target nodes, alternative or approximate methods may be necessary to achieve computational feasibility.

7.2.2 Karp's Algorithm

Karp's algorithm [36] is based on a combination of graph theory and optimization techniques. It leverages the concept of shortest augmenting paths in a bipartite graph, where one set of vertices represents the source nodes and the other set represents target nodes. The edges between vertices have weights corresponding to the assignment costs from the cost matrix C.

The key idea behind Karp's algorithm is to iteratively build an optimal assignment by augmenting paths — specifically, paths that can reduce the overall cost when source nodes are reassigned to different target nodes. This is done by transforming the assignment problem into a series of minimum-cost matching problems, each of which can be solved more efficiently using data structures that support fast updates and queries.

Karp's algorithm involves the following steps to solve the RLAP:

1. **Initialization:**

   Initialize a feasible dual solution for the assignment problem. This involves assigning potential values u_i for each source node i and v_j for each target node j, such that for every assigned pair (i, j), the reduced cost $\tilde{c}_{ij} = c_{ij} - u_i - v_j \geq 0$.

2. **Construct Initial Matching:**

   Construct an initial matching by finding a feasible assignment that minimizes the total reduced cost. The initial matching can be obtained using a greedy approach or a simpler algorithm like the Hungarian algorithm to find a partial matching.

3. **Iteratively Improve the Matching:**

   Karp's algorithm iteratively improves the current matching by finding augmenting paths and adjusting the dual variables to reduce the total cost:

   - **Find Augmenting Paths:** Use a shortest path algorithm, such as Dijkstra's algorithm, to find the shortest augmenting path in the residual graph. The residual graph is constructed by considering the current matching and the reduced costs. Each edge in the graph corresponds to a feasible assignment with zero reduced cost.
   - **Update Matching:** If an augmenting path is found, augment the matching along this path. This involves reversing the direction of the edges in the path, effectively swapping the assignments to reduce the overall cost.
   - **Adjust Dual Variables:** Update the dual variables u_i and v_j to maintain the feasibility of the reduced cost matrix. This step ensures that the current assignment remains optimal with respect to the new potential values.

4. **Terminate When Optimal Matching is Found:**

   The algorithm terminates when no further augmenting paths can be found, indicating that an optimal assignment has been reached.

Karp's algorithm achieves a time complexity of $O(mn \log m)$, where n is the number of source nodes and m is the number of target nodes. This is primarily due to the use of efficient data structures and algorithms for maintaining the augmenting paths and updating the dual variables. The algorithm leverages Fibonacci heaps or similar priority queue structures to achieve logarithmic time updates, allowing it to handle large-scale problems more efficiently compared to the Hungarian algorithm.

7.3 GNN Approach

Graph Neural Networks (GNNs) [85] have emerged as a powerful tool for learning representations of graph-structured data, making them particularly well-suited for applications in which relationships between entities can be naturally expressed as graphs. In the context of linear assignment problems [9], a recent approach introduces a GNN to solve the linear sum assignment problem (LSAP), where n agents need to be assigned to n jobs under one-to-one matching constraints, while the cumulative cost remains minimal [50]. To solve the problem, the authors first construct a weighted bipartite graph $G = (S, T, E)$, where S is the source node set, T is the target node set and E is the edge set, and then solve the minimum matching problem. The GNN they propose is based on GAT and GCN architectures (see Sec. 5.3.2) and consists of three modules: the encoder, the convolution module and the decoder (Figure 7.3.1). The task is converted to an edge-level binary classification task; an edge's label is 1 if the edge belongs to the minimum matching edge set, otherwise it is equal to -1.

7.3.1 Encoder/Decoder

Given the bipartite graph G, the encoder module applies a Multi-Layer Perceptron (MLP) to each edge to transform the attributes of the constructed graph into latent representations, thus forming the embedding features. Note that initially the attribute of each edge is simply its weight so that $e_{ij} = w_{ij}$, where e_{ij} denotes the attributes of the edge connecting nodes i and j and w_{ij} is the weight of this edge. Also, the raw attributes of the nodes are initialized as zero-valued vectors. The transformed graph is then passed to the convolutional module as input to update its state. The decoder coupled with the encoder reads out the edge attributes from the output graph and predicts each edge label through an update function. Similarly, the update function is designed as an MLP and mapped to each edge to form edge labels through a sigmoid activation.

7.3.2 The convolution module

The convolution module is comprised of a node convolution layer and an edge convolution layer. For the i^{th} node in the graph, the node convolution layer collects the information from adjacent edges and its

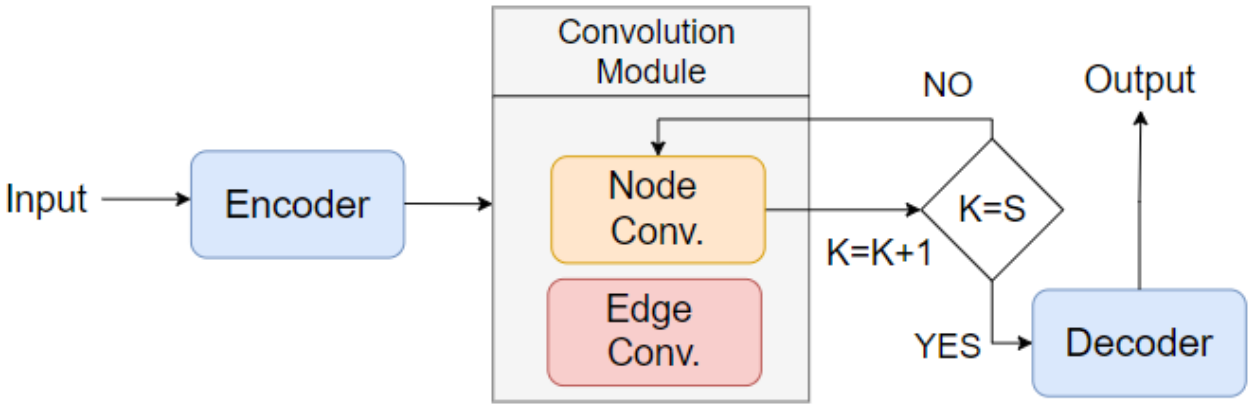


Figure 7.3.1: The architecture of the proposed GNN model. In the node convolution layer, node attributes are updated for a total of $S \geq 2$ iterations.

1^{st} order neighboring nodes by adaptive aggregation weights and updates its attributes. For each edge, the edge convolution layer aggregates the attribute vectors of the two nodes that the edge connects, and updates the edge attribute vector. Although the reception field of the convolution module regards 1^{st}-order neighborhoods, the messages on each node can reach all other nodes after two iterations of convolution, since the graph is bipartite consisting of two node sets (see Section 7.2), and each node from one set connects with all other nodes of the other set. As a result, the reception field of the convolution module can cover the whole graph after the 2^{nd} iteration.

The edge convolution layer first collects information about each edge based on its two adjacent nodes using the aggregation function:

$$\overline{e}_{ij} = [v_i \odot c^u, v_j \odot c^u, e_{ij} \odot c^e] \tag{7.3.1}$$

where e_{ij} denotes the attributes of the edge connecting node i and node j, v_i and v_j the attributes of i^{th} and i^{th} nodes and $\odot$ indicates the element-wise multiplication of two vectors. The operator $[\cdot, \cdot, \cdot]$ concatenates its input vectors channel-wise, while the vectors c^u and c^e are the node and edge channel attention vectors with the same dimensions as node attributes and edge attributes respectively. We must also clarify that $\overline{e}_{ij}$ is an intermediate vector representing the concatenated features the edge $i \rightarrow j$ and not the *updated edge attribute vector*. After the aggregation function, an update function ρ^e designed as an MLP takes the concatenated features as input and outputs the updated feature, so that: $e_{ij} \leftarrow \rho^e(\overline{e}_{ij})$.

The node convolution layer collects information from adjacent edges and 1^{st}-order neighborhoods for each node. Specifically, for the i^{th} node in the bipartite graph G we apply the following function:

$$\overline{v}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \rho_1^v([e_{ij} \odot c^e, w_{ij}(v_j \odot c^u)]), \tag{7.3.2}$$

$$e_{ij} \in \mathcal{E}_i \ \textit{and} \ v_j \in \mathcal{V}_i$$

where ρ_1^v denotes the function to transform its input to an embedding feature. $\mathcal{E}_i$ denotes the attribute set of all edges associated with node v_i in G, and $\mathcal{V}_i$ represents the attribute set of 1^{st}-order adjacent nodes to node v_i. For node v_i, w_{ij} is the weight measuring the contribution of its adjacent node v_j during feature aggregation, and is computed as $w_{ij} = \tau([v_i, v_j])$. The collected embedding features are then concatenated with the current attributes of node v_i and are passed to another transformation function that outputs the updated attributes for node v_i using the formula $v_i \leftarrow \rho_2^u([\overline{v}_i, v_i])$. Functions ρ_1^v, ρ_2^v and τ are all specified as MLP modules, each of them with a different architecture and parameters.

7.3.3 Loss Function

In order to solve the linear assignment problem, we consider it as a binary classification task and divide the elements in the ground-truth assignment matrix Y^{gt} into positive labels and negative ones. Considering the

case that for each node, there is at most one positive edge among its adjacent edges and the rest are negative ones, to avoid the negative labels dominating the training, Balanced Cross Entropy is utilized as the loss function:

$$L = -\sum_{i=1}^{n}\sum_{j=1}^{m}\big(w \times y_{ij}^{gt}\log(y_{ij}) + (1-w)\times (1-y_{ij}^{gt})\log(1-y_{ij})\big) \tag{7.3.3}$$

where y_{ij} is the predicted label for edge $i \to j$ which connects source node i and target node j, y_{ij}^{gt} is the corresponding ground-truth vector element indicating the edge as positive or negative, and w is the weight which balances the loss to avoid the negative labels dominating the training. Parameters n, m denote the cardinality of source and target nodes sets, so that $|S| = n, |T| = m$.

In order to impose the one-to-one matching constraint in the model, we first construct a predicted assignment matrix $Y \in R^{n\times n}$ whose size is the same as the problem:

$$Y_{j,k} = \begin{cases} y_{ind(j,k)}, & \text{if } \exists (j,k) \in E, \\ 0, & \text{otherwise,} \end{cases} \tag{7.3.4}$$

where $ind(.,.)$ is a bijection mapping an edge to an integer index. Then the soft constraint loss is designed as follows:

$$L_1 = ||\mathbf{1} - sum_r(Y)||_2 + ||\mathbf{1} - sum_r(Y^T)||_2, \tag{7.3.5}$$

$$L_2 = ||\mathbf{1} - norm_r(Y)||_2 + ||\mathbf{1} - norm_r(Y^T)||_2, \tag{7.3.6}$$

$$L_C = L_1 + L_2 \tag{7.3.7}$$

In the above equations, $\mathbf{1}$ is a one-valued vector; $sum_r(.)$ sums the values in predicted assignment matrix along the row-wise; $norm_r(.)$ returns a vector in which each element is the 2-norm of corresponding row-vector. In training, Eq. 7.3.5 drives the prediction to satisfy the constraints in Eq. 7.1.2 and 7.1.3. As a part of supervision signals in training, Eq. 7.3.6 guides the proposed network to output sparse predictions.

Finally, the binary classification loss and the constraint loss are combined to obtain the *loss function* that guides the training process as follows:

$$L = L_A + \alpha L_C \tag{7.3.8}$$

where $\alpha > 0$ weights the degree of the soft one-to-one matching constraints imposed.

This GNN-based approach offers significant speed-up to the solution of the LSAP, at the cost of optimality; while the inference time of the model is approximately 7ms regardless of the graph size, the solution given is not the optimal one but rather an approximation of it. In our work, we experiment with both Karp's algorithm and an extended version of the aforementioned GNN model, fine-tuned towards solving RLAP.

Chapter 8

Proposal

In this section, we propose the method with which we will tackle the problem of Counterfactual Explanations. We focus on word-level counterfactual interventions to test the behaviour of textual classifiers when different words are perturbed. Our proposal revolves around placing all implemented interventions under a framework which presents the following characteristics regarding interventions: [55]

- Optimality: Substitutions should be optimal -or approximately optimal-, respecting a given notion of semantic distance.
- Controllability: at least one input semantic should be substituted in each data sample.
- Efficiency: an optimal solution should be reached using non-exhaustive search techniques among alternative substitutions.

We approach these requirements by viewing counterfactual interventions as a combinatorial optimization problem, solvable via graph assignment algorithms from graph theory [111]. To further enhance our method, we consider the use of Graph Neural Networks (GNNs) [108] as a faster approximate substitute of these algorithms [114].

Our proposed method can be applied to both model-specific and general purpose scenarios, since there is no strict reliance on changing the final label. This property allows for generated edits to be used for different tasks apart from label-flipping, such as semantic similarity [54] or untargeted generation [106]; nevertheless, in this dissertation, we focus on classification tasks for direct comparison with prior work. To this end, we compare our approach with two SoTA editors [106, 80] using appropriate metrics for label-flipping, fluency and semantic closeness.

We first highlight the main contributions of this thesis and then explain the proposed method in detail.

8.1 Contributions

To sum up our contributions are:

- We impose optimality and controllability of word interventions translating them in finding the optimal assignment between graph nodes.
- We accelerate the assignment process by training GNNs on these deterministic matchings, ultimately achieving advanced efficiency.
- We extend an existing framework in order to offer a GNN that solves the RLAP; to the best of our knowledge, no prior work has leveraged GNN modules to solve RLAP
- Our highly efficient black-box counterfactual editor consistently delivers SoTA performance compared to existing white-box and black-box methods on two diverse datasets and across four distinct metrics.

Remarkably, it achieves these results in less than 2% and 20% of the time required by its two competitors, demonstrating both superior efficacy and efficiency.

- The versatility of our proposed editor is demonstrated in different scenarios, since it is able to be optimized towards a specific metric or perform general-purpose fluent edits.

Our work is thoroughly presented in our paper named "Optimal and efficient text counterfactuals using Graph Neural Networks" [55].

8.2 Proposed Method

The workflow of our method, as shown in Figure 8.2.1) comprises of three stages. A textual dataset D serves as the input to our workflow. In the first stage, words are extracted from D, based on their part of speech (POS), and used as the source node set S. The target set T is either a copy of S, or else produced from an external lexical source such as WordNet [61], containing all possible candidate substitutions of the source words (nodes). The S and T sets form a bipartite graph G (described in Section 4.2), with their in-between edge weights reflecting word similarity. In the second stage, we pass the constructed G as input to a pre-trained GNN which outputs an approximate RLAP solution, in the form of a list of candidate word pairs. Each word pair, consists of the source word $s_i \in S$ and its computed substitution $t_i \in T$. In the third and final stage, we harness beam search [105] to define the final changes. Beam search uses a *heuristic function* to choose the most suitable substitutions from those returned by the GNN. The selected words from S are then substituted with their respective pair from T, producing a counterfactual dataset D^*.

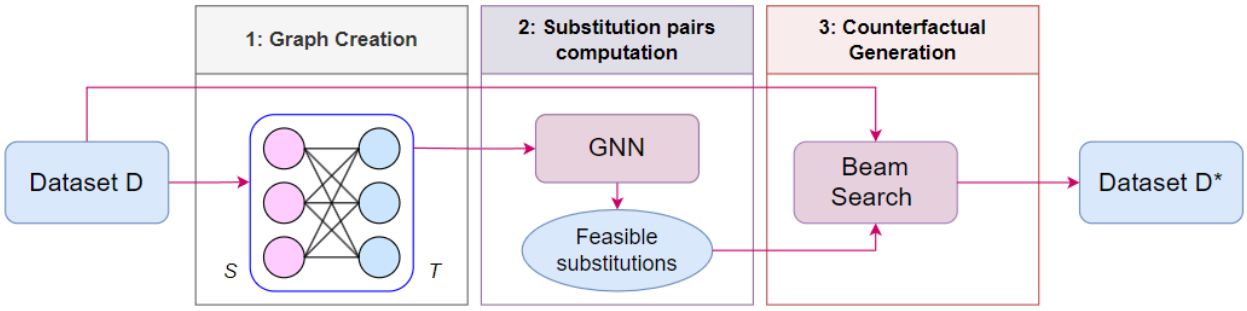


Figure 8.2.1: The pipeline of our method. In the first stage, we construct a bipartite graph using words as nodes, and in the second stage we utilize a GNN to get feasible substitutions that approximately solve the RLAP. In the final stage, we use beam search to change appropriate words of the original dataset, thus getting a new counterfactual dataset. [55]

8.2.1 Graph creation

When constructing the bipartite G, words are extracted from the original D based on their POS. To test how well our framework generalizes, we use both POS-specific and POS-agnostic word extraction. The former means that we only select to potentially change words that belong to a specific POS (i.e. adjectives, nouns, verbs, etc.), while the latter means that we regard all words, irrespective of their POS. For the edge weights, we employ two different approaches, each varying in transparency. For the first one, we adopt a fully transparent approach by calculating the distances using a lexical hierarchy: the weight of an edge connecting two words is determined by their similarity value as defined in WordNet.[1] In the second case, we apply different LLMs to generate word embeddings, namely AnglE[2] [48, 87], GISTEmbed[3] [88], GinaAI[4] [63] and MUG[5]; then, we set the edge weight equal to the cosine similarity of the two word embedding vectors. Since lower similarity is associated with lighter edges, i.e. more suitable candidates for M, the selected words to be

[1] path_similarity function between synsets corresponding to the words (https://www.nltk.org/howto/wordnet.html).
[2] mixedbread-ai/mxbai-embed-large-v1
[3] avsolatorio/GIST-Embedding-v0
[4] https://jina.ai/embeddings/
[5] Labib11/MUG-B-1.6

substituted will form *contrastive word pairs*. In order to preserve syntax in the POS-agnostic case, we force substitutions between same-POS words exclusively: thus, we experiment with an edge filtering mechanism, which sets a predefined large weight to edges, ∼10 times bigger than the normal edge weights as instructed from WordNet path similarity or cosine similarity of embeddings. This way, we avoid cases where a POS is substituted with a word of different POS, since a significantly heavier edge cannot be selected to participate in minimum matching edge set M. In the POS-specific case, this mechanism is redundant since all words are of the same POS.

8.2.2 Substitution pairs computation

For appropriate substitution pairs we need to solve RLAP on the constructed graph G. As previously discussed (Section 7.2), traditional deterministic approaches achieve this in $O(mn \log n)$. While these methods provide the optimal solution, they lack speed as the dataset size, and therefore graph size grows larger. In an attempt to produce substitution pairs in *stable* time regardless of the dataset size, we use a GNN model, which approximates the optimal solution found by deterministic algorithms, while significantly speeding up the process.Regarding the GNN, we fine-tuned the model described in Section 7.3, using the following procedure:

Initially, a synthetic dataset that consists of M samples[6] is created. Each sample is composed of a cost matrix C in which the elements are generated from a uniform distribution on $(0, 1)$ and the corresponding optimal assignment solution which is obtained by the Hungarian algorithm [42]. We consider the RLAP as a binary classification task and divide the elements in the ground-truth assignment matrix Y^{gt} [7] into positive labels and negative ones. For the Loss Function, we use Equation 7.3.3 with $\alpha = 0$, which is equivalent of using only *Balanced Cross Entropy*. The reason for this, is the fact that in our version of RLAP, cost matrix C is of dimensions $n \times m$, where $n \leq m$ and therefore an exact one-to-one matching is not possible. By removing the part of the loss function which corresponds to this constraint, we attempt to greatly soften it, thus optimizing the GNN model towards RLAP solution instead of LSAP. As in [50], training takes 20 epochs in total, where the learning rate is set as 0.003 initially and declined by 5% after every 5 epochs.

Using the fine-tuned GNN, **efficiency** is guaranteed. By solving the problem with the constraint of minimum $\sum w_e$, we find all most *dissimilar* $s \rightarrow t$ pairs, achieving *approximate* **optimality** of concept substitution within G and ultimately producing contrastive substitution pairs. At the same time, **controllability** is *partially* ensured since the graph G is dense (therefore there are no disconnected s nodes) and $|S| \leq |T|$, since T is either a copy of S or produced based on S using antonyms from WordNet (more than one antonym may correspond to each word). Note here, that we use the word *"partially"* as there is a trade-off between *controllability* and *minimality* [8], which stems from using beam search during counterfactual generation. In practice, there are also a few exceptions in controllability, if a source concept cannot be mapped on WordNet.

8.2.3 Counterfactual Generation

As a result of solving RLAP, a matching $M \subset E$ is returned, indicating the optimal substitutions to n source concepts. We denote as $W_n^M \subset W$ the total weight of M that contains n source concepts. Given this matching, beam search selects which conceptual substitutions from M will *actually* be performed on D. This selection process is necessary since we desire changes to be *minimal* in terms of number of words altered per instance, perturbing only small portions of input, a property which has been argued to make explanations more intelligible [1, 62]. In this context, we also set an upper limit of substitutions on each text instance, experimenting with both a fixed and a dynamically set number. In the second case, for each instance, the upper limit is equal to the 20% of the total number of words it contains. We stop the search when the model's prediction is flipped or when the upper limit is reached, thus keeping the number of edits low.

[6] Each sample represents a weighted bipartite graph.

[7] Y^{gt} is a matrix where element y_{ij}^{gt} is 1 if the edge connecting nodes i and j **belongs** to the minimum matching, else it is -1.

[8] Minimality here refers to the number of words changed.

Chapter 9

Experiments

In order to evaluate the proposed method and compare it to other techniques and frameworks, we carried multiple experiments across two NLP tasks. In this section, some preliminary information will be presented about the datasets, the classifier models and the antagonizing editors, as well as the metrics used in the evaluation process.

With the basics put in place, we will analyze how we experimented with different parts of our framework, and what trade-offs should be taken into account and finally present evaluation results. In addition to the quantitative results, we will present instances of the edited texts that assist in the overall understanding of our method and its capabilities.

Contents

9.1 Preliminaries

9.1.1 Dataset

We evaluate our framework and compare it with other editors from literature, on two English-language datasets: IMDB, which contains movie reviews and is used for binary sentiment classification [58] and a 6-class version of the 20 Newsgroups used for topic classification [44]. Due to the high computational demands of the compared methods, we sampled 1K instances from each dataset. Running MiCE on just 1K samples required over 47 hours (see Table 9.1), making full dataset experiments impractical. We chose twice the sample size used in similar studies comparing the same methods on the same datasets [21].

IMDB Reviews

The IMDB Reviews Dataset is a widely used benchmark dataset for natural language processing (NLP) tasks, particularly sentiment analysis. It consists of 50,000 movie reviews taken from the Internet Movie Database (IMDB), with an equal split between positive and negative sentiment labels. The dataset is divided into two equal parts; 25,000 reviews are designated for training, and the remaining 25,000 are reserved for testing. In our experiments, we took a representative sample from the *test set*, since we are not aiming to train a new model, but rather to explain existing ones. Therefore, since the models we use are already trained on the training set, it makes sense to avoid using instances they have previously seen and possibly memorized.

Each review in the dataset is associated with a binary sentiment label; *positive* for reviews with a rating of 7 or above (out of 10) and *negative* for reviews with a rating of 4 or below. Reviews with a rating of 5 or 6 are not included in the dataset to ensure a clear distinction between positive and negative sentiments. The text data in the reviews vary in length and complexity, capturing a wide range of linguistic styles, from informal to formal writing. In our work, before using them, we cleaned them up, by removing special characters and trailing spaces. An example of a text instance from IMDB Reviews Dataset after the cleaning process can be seen in Figure 9.1.1.

> When I see a movie, I usually seek entertainment. But of course if I know what genre the move is, then I will seek what it is meant to do. For example, if it is a deep film, I expect the film to rile thoughts up in my cranium and make me ponder what it is saying. But Who's That Girl? is not a deep film. But it is entertaining, nonetheless. It's a campy sort of film that's a joy to watch. There's barely a boring moment in the film and there are plenty of humorous parts. I've watched it when I was younger. The cast is always entertaining as usual. I had a small crush on Griffin Dunne even though he wasn't the typical male heartthrob at the time. Haviland Morris also stars. And late Austrian actress Bibi Besch is here too! Overall, a delight!

Figure 9.1.1: An example of a review labeled as 'positive' from the IMDB Reviews Dataset

20 Newsgroups

The 20 Newsgroups dataset is another popular benchmark dataset in the field of NLP and text classification. It contains approximately 20,000 documents spread across 20 different topics such as sports, politics, technology, or science. The dataset is designed to support experiments on text classification tasks, particularly in the context of multi-class classification.

In this dissertation, we use the 6-class version of the original dataset, which is a specific configuration that consolidates the 20 topics into 6 broader, thematic categories. These six categories group related topics together, reducing the granularity of the classification task while still preserving the diversity of content. These 6 categories are:

1. **Comp (Computers and Technology)**: Contains documents related to computer hardware, software, and discussions around technology.

2. **Rec (Recreation and Sports)**: Includes documents focused on recreational activities, such as sports (e.g., baseball, hockey) and hobbies.
3. **Sci (Science)**: Groups documents related to scientific topics, such as space, electronics, and medical science.
4. **Talk (Talk and Politics)**: Aggregates documents discussing politics, religion, and various societal issues.
5. **Misc (Miscellaneous)**: Contains documents that do not fit neatly into other categories, such as discussions on automobiles and motorcycles.
6. **Alt (Alternative)**: Includes documents that cover alternative topics not well-defined by the other categories, such as alt.atheism and alt.misc.

This version provides a more simplified classification task compared to the original 20 categories, making it useful for evaluating the performance of text classification algorithms in situations where there are fewer target classes. This version of the dataset helps researchers and practitioners test the generalization capabilities of their models over more generalized topic categories, making it particularly relevant for applications where distinguishing between broad subject areas is required. Its popularity also extends to the field of Counterfactual Explanations, with many counterfactual editors being evaluated on this dataset, which is why we also chose it along with IMDB Reviews Dataset. We also perform the same cleaning process, removing empty lines, special characters and trailing spaces. An example of a document from the dataset after the cleaning process is shown in Figure 9.1.2.

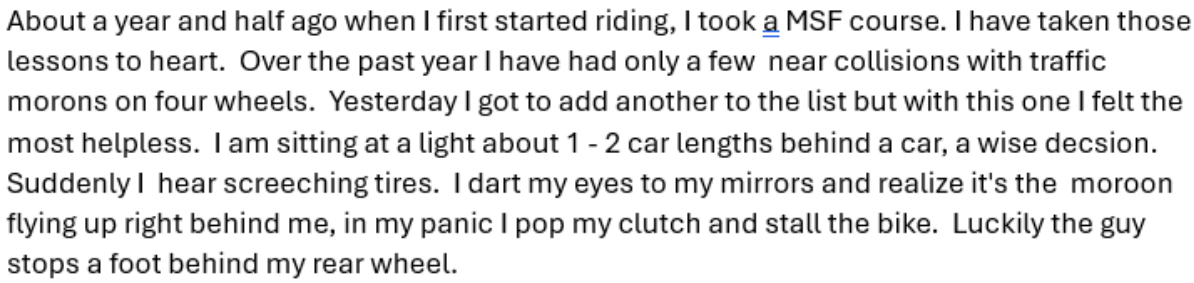


Figure 9.1.2: An instance labeled as 'Rec' from the 6-class version of 20 Newsgroups Dataset

9.1.2 Evaluation Metrics

To assess the performance of the different editors, we draw inspiration from MiCE [80] and measure properties such as *flip-rate*, *minimality*, *closeness* and *fluency*. The metrics that correspond to each of these properties are explained below. To highlight the significant speedup offered by our method, we also report *execution times* for each editor.

Flip-Rate

Flip-Rate is an extremely popular metric and also the one most counterfactual editors try to maximize. Formally, flip-rate is the percentage of instances for which an edit results in different model prediction (label-flipping):

$$FlipRate = \frac{\sum_{i=1}^{N} \mathbb{I}(y_i \neq y_i^{edit})}{N}$$

where N is the total number of input (text) instances, y_i is the prediction of the model for the original input, y_i^{edit} is the model's prediction for the edited input and $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition is true, and 0 otherwise. Since flip-rate shows how "effective" the produced edits are, it has become one of the primary evaluation metrics used in Counterfactual Editors.

Minimality

In this dissertation, minimality is measured using the *Levenshtein distance*. Typically, Levenshtein distance also known as edit distance, is a metric used to measure the difference between two strings. It represents the minimum number of single-character edits required to transform one string into another, where the allowable operations are insertion, deletion, or substitution of a single character. In our experiments, we employee the word-level version of this distance, which instead of character, measures single *word* edits.

To compute the word-level Levenshtein distance between two strings s_1 and s_2, where each string has n and m words respectively, a dynamic programming approach is typically employed. A matrix D of size $(n+1) \times (m+1)$ is created, where each entry $D[i][j]$ represents the Levenshtein distance between the first i words of s_1 and the first j words of s_2. The matrix is filled using the following recurrence relations:

$$D[i][0] = i, \quad D[0][j] = j$$

$$D[i][j] = \min \begin{cases} D[i-1][j] + 1, & \text{(represents word deletion)} \\ D[i][j-1] + 1, & \text{(represents word insertion)} \\ D[i-1][j-1] + \delta(s_1[i], s_2[j]), & \text{(represents word substitution)} \end{cases}$$

where $\delta(s_1[i], s_2[j])$ is 0 if $s_1[i] = s_2[j]$, otherwise it is 1. The value at $D[n][m]$ will be the Levenshtein distance between the two strings.

To deal with the imbalance caused due to different word lengths of text instances, we adopt a normalized version of the word-level Levenshtein distance, with a range of [0, 1] — the Levenshtein distance divided by the number of words in the original input. Thus, minimality can be computed as:

$$Minimality = \frac{D[n][n]}{n}$$

Closeness

Closeness represents the semantic similarity between the original and edited input, measured by BERTScore [117]. BERTScore is a metric for evaluating the similarity between two text sequences, commonly used for tasks like text generation, machine translation, and summarization. Unlike traditional evaluation metrics such as BLEU [72] or ROUGE [49], which rely on exact n-gram overlap, BERTScore leverages contextual embeddings from pre-trained language models like BERT [15] to capture semantic similarity between text sequences at a deeper level.

To compute BERTScore, the following steps are undertaken:

1. **Token Embedding:** Given a candidate text sequence C and a reference text sequence R, each token in the sequences is embedded into a high-dimensional vector space using a pre-trained BERT model. This results in two sets of token embeddings: $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n\}$ for the candidate and $\{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_m\}$ for the reference, where n and m are the lengths of the candidate and reference sequences, respectively.

2. **Similarity Calculation:** Compute the cosine similarity between each pair of token embeddings from the candidate and reference sequences:

$$\text{sim}(\mathbf{c}_i, \mathbf{r}_j) = \frac{\mathbf{c}_i \cdot \mathbf{r}_j}{\|\mathbf{c}_i\| \|\mathbf{r}_j\|}$$

   for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.

3. **Precision, Recall, and F1 Score:** Calculate the precision, recall, and F1 score using the similarity scores. Precision is computed by averaging the maximum similarity for each token in the candidate sequence with respect to all tokens in the reference sequence:

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\max_{j}\text{sim}(\mathbf{c}_i, \mathbf{r}_j)$$

Recall is computed by averaging the maximum similarity for each token in the reference sequence with respect to all tokens in the candidate sequence:

$$\text{Recall} = \frac{1}{m}\sum_{j=1}^{m}\max_{i}\text{sim}(\mathbf{c}_i, \mathbf{r}_j)$$

The BERTScore F1 is then computed as the harmonic mean of Precision and Recall:

$$\text{BERTScore}_{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

BERTScore effectively captures semantic meaning and contextual information by using embeddings from large pre-trained language models, making it a robust metric for evaluating produced edits against the original inputs. The higher BERTScore is, the more semantically close the two sentences are.

Fluency

Fluency is a measure of how similarly distributed the edited input is compared to the original. To evaluate fluency, we first take a pretrained T5-BASE model [77] and compute the loss value for both the edited and original input. Afterwards, we report their *loss_ratio* - i.e., *edited / original*. Since we aim for a value of 1.0, which indicates equivalent losses for the original and edited texts, the final measure of fluency is defined as $|1 - loss\_ratio|$, where smaller values indicate more fluent edits.

9.1.3 Classifier Models

In order to measure the effectiveness of produced edits, we test the different editors on two text classifiers. We use the same predictor models with MiCE [80] in each dataset (IMDB Reviews and 20 Newsgroups). Both of these models are based on RoBERTa$_{LARGE}$ [51], and fine-tuned towards their respective datasets. The first model, is a binary classifier whose output is either 0 or 1, with 0 denoting *negative* reviews and 1 denoting *positive* ones. The second model, is fine-tuned on the 6-class version of the 20 Newsgroups dataset and its output ranges from 0 to 5, with each integer value representing a category from those discussed in Sec. 9.1.1.

9.1.4 Counterfactual Editors

We compare our framework with two SoTA editors, namely Polyjuice [106] and MiCE [80]. The former is a general purpose counterfactual editor, while the latter is a task-specific editor optimized towards *flip-rate* and *minimality*. We chose these two editors, due to the fact that our framework can be used as both a general-purpose and a task-specific editor, thus being comparable to both of them.

Polyjuice

Polyjuice is a versatile data augmentation model designed to generate a wide range of counterfactuals and text perturbations for various natural language processing (NLP) tasks. It enables the creation of semantically diverse text variations that can be used to improve model robustness, perform adversarial testing, or study model biases. Since it is not optimized towards a specific task nor has it access to the inner workings of the predictor models, it is considered a *general-purpose, black-box* counterfactual editor.

Polyjuice is built on top of the GPT-2 architecture [76], a generative language model known for its ability to produce coherent and contextually relevant text. The key innovation in Polyjuice lies in its ability to conditionally generate counterfactuals based on specific control codes. These control codes guide the generation process, allowing the model to produce text variations with desired attributes.

Key features of Polyjuice include:

- **Conditional Generation:** Polyjuice can generate text conditioned on various attributes such as sentiment, tense, or specific words. This is achieved through the use of control codes, which are provided as input to the model along with the original text. For instance, by specifying a control code for sentiment, Polyjuice can generate a version of the text with a different emotional tone.
- **Diverse Counterfactuals:** The model is capable of producing a wide range of counterfactuals that differ significantly from the original text while still being grammatically correct and contextually appropriate. These variations can include changes in factual content, stylistic alterations, or shifts in tone.

MiCE

Minimal Contrastive Editing (MiCE) is a method aimed at generating minimal edits to input text that lead to a change in the prediction of a machine learning model. The primary objective of MiCE is to provide counterfactual explanations by identifying the smallest possible modification to an input text that alters the model's output.

MiCE operates by making minimal modifications to a given input text, ensuring that the resulting edited text is fluent, semantically coherent, and sufficient to change the model's prediction. This contrasts with other counterfactual generation methods that may produce extensive or unrealistic changes, thereby compromising the usefulness of the explanation.

The method revolves around the following key principles:

- **Minimality:** The primary goal of MiCE is to achieve the smallest possible edit that can flip the model's decision. This minimality ensures that the generated counterfactuals are concise and focused on the most critical aspects of the input.
- **Contrastiveness:** The edits made by MiCE are designed to produce a clear contrast in the model's predictions. By comparing the original and edited inputs, users can gain insights into which features or words are pivotal in driving the model's output.
- **Fluency and Coherence:** MiCE ensures that the edited text remains fluent and semantically coherent, maintaining readability and interpretability for human users. This is crucial for the explanations to be meaningful and actionable.

MiCE produces edits, using a two-stage approach. In Stage 1, it prepares a highly-contextualized EDITOR model to associate edits with given end-task labels (i.e., labels for the task of the predictor model) such that the contrast label y_c is not ignored in MICE's second stage. Intuitively, this is done by masking the spans of text that are "important" for the given target label (as measured by the predictor model's gradients) and training our EDITOR to reconstruct these spans of text given the masked text and target label as input. In Stage 2 of MiCE, contrastive edits $e(x)$ are generated using the EDITOR model from Stage 1 (here, x is the original input). Specifically, it generates candidate edits $e(x)$ by masking different percentages of x and giving masked inputs with prepended contrast label y_c to the EDITOR; binary search is employed to find optimal masking percentages and beam search is used to keep track of candidate edits that result in the highest probability of the contrast labels $p(y_c|e(x))$ given by the predictor model.

This framework uses knowledge of the inner workings of the predictor model to generate edits specifically optimized for label-flipping and therefore it is considered a *task-specific, white-box* counterfactual editor.

9.2 Editor Experiments

9.2.1 Editor Variants

Our editor was implemented using many different Python libraries. For the first stage (graph construction) we used *Spacy*, *NLTK* and *sentence-transformers*, while for the GNN, *Pytorch Geometric* [20] was employed. Since our method is a highly customizable one, we experimented with different setups and approaches. We

must also note that all experiments were run on the same system consisting of a *16 GB GPU*, an *Intel i7 CPU* and *16 GB RAM*.

In order to preserve the POS in each substitution, we apply a penalty mechanism (filtering) when computing edge weights of the graph. This mechanism assigns a weight approximately 10× bigger than the *normal* weights (as defined from WordNet path similarity or embedding cosine similarity), to each edge that connects different-POS words. This way, since our framework is trying to find a *minimum weight matching*, edges with large weights are almost impossible to be chosen and therefore substitutions involving different POS have a low occurrence probability. We report findings with and without this edge-filtering mechanism in the next section.

We also investigate the effect of using cosine similarity of embeddings in place of WordNet path similarity between two words, when computing the weight of a specific edge in the bipartite graph G. On the one hand, deterministic hierarchies provide more *explainable* relationships between concepts, fully justifying causal pathways of substitutions. On the other hand, recently-emerged embedding models can better capture the relationship and similarity of two words, compared to WordNet. To keep our framework relatively lightweight, we deploy the top four best performing models that participated in an embedding benchmark competition [68] and whose size does not exceed *1.25 GB*. Models with that size occupied the top spots in the competition and any increase in model size did not result in significant improvements in performance.

In an attempt to evaluate our editor's ability to distinguish which POS is more influential to a specific dataset when related words are substituted, we impose restrictions regarding which POS should be candidates for substitutions, and compare the results with a POS-unrestricted version of our framework. The IMDB dataset is used for sentiment classification, and therefore adjectives and adverbs are presumed to mainly dictate the label (sentiment) for each instance [5]. With that in mind, we limit our editor to change only those two POS. Newsgroups is a dataset which belongs to the topic classification category. Since a topic is deduced by examining the nouns in a text, we instruct the editor to take into account only those.

To keep the number of edits relatively low, a way to limit the number of substitutions per data instance is required, accepting a potential drop in flip-rate. For this reason, we use two different approaches. In the first one, we enforce a static number of maximum substitutions allowed for each textual input, regardless of its length; after experimentation, the best number was found to be 10. In the second approach, we dynamically compute the optimal upper limit (or threshold) of substitutions based on the total number of words in the text. After different attempts, we end up defining that limit as 20% of the total number of words, which basically means that we change on average one in every five words.

Since the selection of eligible substitutions is a general-purpose process (only defined by the graph), we examine the behaviour of our editor when optimized for label-flipping scenarios. This optimization is done by altering the heuristic function of beam search in the last stage of our framework (see Figure 8.2.1). For general-purpose edits, this function is the metric for *fluency* discussed in Subsection 9.1.2, which assists the production of fluent edits. For label flipping, we use *contrastive probability*, which regards the change to the model prediction for the original label, to determine the best edits (see *GNN w. contrastive* in Table 9.1). Finally, we also use the average of fluency and contrastive probability as the heuristic function, which results in fluent edits with high flip-rate (see *GNN w. fluency_ contrastive* in Table 9.1).

9.2.2 Trade-Offs

Since our editor is a highly customizable one, there are many trade-offs which must be considered during counterfactual generation. Below, we discuss those with the highest importance amongst them.

Controllability vs. Minimality

Controllable interventions involve changing *any* semantic that can be changed in order to observe an outcome; to this end, we could potentially alter as many words as possible in order to reach a goal, e.g. label-flipping. However, in our case, in order to produce minimal edits, we set a maximum number of substitutions per textual input and leverage *beam search* to select the most appropriate changes. As a consequence, the default controllability requirement is partially sacrificed, since it is not guaranteed that all words that can be substituted will be indeed substituted. Nevertheless, our framework still produces edits for each input,

meaning that it will change the original text, although not entirely; this is why we impose as controllability to *modify at least one word of the original data sample*. In our experiments (see Table 9.1) we have accepted this trade-off since our interest lies more heavily with minimality compared to controllability. Despite that, it is possible to fully ensure controllability by arsing the limitations mentioned above (i.e. max number of substitutions and beam search), although such an approach would results in worse performance regarding minimality.

Optimality vs. Execution Speed

In our framework, we use both a deterministic (see *Deterministic w. fluency* from Table 9.1) and a GNN approach (see *GNN w. fluency* from Table 9.1) to solve RLAP. With the deterministic approach, optimality is ensured, since traditional graph matching algorithms have been proved to find the optimal solution [42, 36]. However, the complexity of those algorithms, which is $O(mn \log n)$, results to slower runtimes as graph size increases (which is analogous to the number of words to be substituted and therefore depends on the dataset size). By replacing the deterministic algorithms with the trained GNN (see Section 7.3), our framework becomes significantly faster at the cost of optimality. This is due to the fact that the solution given by the GNN is an *approximation* of the optimal one.

Explainability vs. Execution Speed

In our work, we utilize WordNet as the default way of computing edge weights between nodes, where each edge weight is based on the path that connects a source word s with target word t in WordNet. By mapping each concept to WordNet synsets, a deterministic concept position is assigned to each word, providing a fully transparent concept mapping to a well-crafted lexical structure. The utilization of word embeddings casts a shadow on word mapping, since we transit to a vector representation of an uninterpretable multi-dimensional space via black-box models. Similarity in the embedding space translates to semantic similarity of physical concepts, acting as our guarantee towards employing embedding models.

In combination with the deterministic solution to RLAP, WordNet mapping guarantees *explainability* of edits, since all paths $s \to t$ are tractable, and the choice of edges is fully transparent due to the deterministic selection process of graph matching algorithms [6]. By obtaining the resulting matching M we gain full access to the set of edits to perform $S \to T$ transition. A sacrifice in explainability is imposed when using the GNN instead of the deterministic graph assignment algorithms: the GNN introduces an uncertainty to the edge selection, since we cannot be entirely sure *why* a specific edge was chosen. Although we have trained the GNN to output the RLAP solution, the model itself still remains a black-box structure that hides the exact criteria which decide whether an edge will be selected or not. Still, in some applications the speedup offered by the GNN outweighs this drop in explainability, while the opposite may hold in cases where trustworthiness is of utmost importance.

9.3 Results

9.3.1 Overall Performance

The results of our experiments are shown in Table 9.1, including both IMDB and Newsgroups datasets. Our proposed editors—deterministic and GNN-powered—outperform both MiCE and Polyjuice across the three of the four metrics namely **minimality**, **fluency** and **closeness**. Regarding flip-rate, MiCE achieves the highest results (99% - 100%, across the two datasets), followed by our approach: our best editor reaches values slightly above 90% (specifically 94.4% for IMDB and 92% for Newsgroups). However, this is expected, since MiCE is the only editor that has white-box access to the classifier and it is able to strategically construct edits that affect the classifier the most, regardless of the input text.

Results also show that our edits tend to be more minimal when graph construction is based on embeddings models instead of WordNet (approximately 10% of the original tokens are changed when WordNet is employed, while with embedding models only 1% of the said tokens change). We believe this is due to the fact that SoTA embedding models are able to better depict concept distance compared to WordNet, and therefore substitutions based on them are of higher quality, leading to more contrastive pairs. This means that for

	IMDB					
	Editor	**Fluency ↓**	**Closeness ↑**	**Flip Rate ↑**	**Minimality ↓**	**Runtime ↓**
WordNet	Deterministic w. fluency	0.14	0.969	0.892	0.08	4:09:41
	GNN w. fluency	0.07	0.986	0.861	0.12	3:17:51
	GNN w. fluency & dynamic thresh	0.057	0.986	0.851	0.146	4:18:34
	GNN w. fluency & POS_filter	0.08	0.992	0.862	0.123	**0:32:05**
	GNN w. fluency & edge filter	0.105	0.993	0.845	0.149	3:00:38
	GNN w. fluency_contrastive	0.112	**0.999**	0.914	0.014	2:12:06
	GNN w. contrastive	0.048	0.996	0.927	**0.01**	2:00:15
Embeddings	GNN w. AnglE & contrastive	0.063	0.995	0.944	0.011	0:45:38
	GNN w. GIST & contrastive	0.037	0.995	0.882	0.016	0:58:14
	GNN w. Jina & contrastive	0.047	0.995	0.928	0.017	1:00:56
	GNN w. MUG & contrastive	**0.036**	0.996	0.889	0.013	0:52:19
	Polyjuice	0.394	0.787	0.782	0.705	5:01:58
	MiCE	0.201	0.949	**1.000**	0.173	48:37:56
	Newsgroups					
	Editor	**Fluency ↓**	**Closeness ↑**	**Flip Rate ↑**	**Minimality ↓**	**Runtime ↓**
WordNet	Deterministic w. fluency	0.182	0.951	0.870	0.135	4:20:52
	GNN w. fluency	0.074	0.985	0.826	0.151	3:48:37
	GNN w. fluency & dynamic thresh	0.043	0.984	0.823	0.148	4:47:14
	GNN w. fluency & POS filter	0.044	0.989	0.841	0.143	1:19:57
	GNN w. fluency & edge filter	0.12	0.989	0.834	0.151	3:05:08
	GNN w. fluency_contrastive	0.088	0.979	0.875	0.033	2:45:31
	GNN w. contrastive	0.033	0.989	0.920	0.033	2:02:34
Embeddings	GNN w. AnglE & contrastive	0.005	0.995	0.904	0.027	1:09:13
	GNN w. GIST & contrastive	**0.001**	0.995	0.898	0.02	1:02:55
	GNN w. jina & contrastive	0.013	0.993	0.882	0.025	0:57:31
	GNN w. MUG & contrastive	0.005	**0.996**	0.900	**0.016**	**0:53:04**
	Polyjuice	1.153	0.667	0.8	0.997	6:00:10
	MiCE	0.152	0.922	**0.992**	0.261	47:23:35

Table 9.1: Experimental results of counterfactual generation. We evaluate different versions of our framework using the metrics described on subsection 9.1.2, and we compare it with MiCE and Polyjuice. For each metric (column) the best value is highlighted in **bold**. Reported runtimes refer to inference.

the same impact on the classifier's output, less embedding substitutions are required compared to WordNet-based ones. On the other hand, using embedding models reduces the overall transparency of the method. Despite minor discrepancies, all our framework variants consistently outperform previous techniques across every metric for Polyjuice and three metrics for MiCE. Moreover, even the general-purpose variation of our framework, which lacks access to the classifier, yields better results compared to the white-box MiCE, in just 2% of the time.

As far as runtime is concerned, our editors show a remarkable improvement in speed compared to MiCE and Polyjuice. Our deterministic editor, which is used as a baseline, requires approximately 4 hours for each dataset, while editors that use the GNN discussed in Section 7.3 achieve faster execution on average (2-4 hours). Runtime is further improved with the use of embedding models, where execution requires less than an hour (52 minutes - 1 hour for IMDB, 53 minutes - 1 hour and 9 minutes for Newsgroups). This significant speed improvement is one of the main advantages of our framework compared to the two SoTA editors, where we observed approximately 97% and 83% speed improvement compared with MiCE and Polyjuice respectively.

9.3.2 Variant Comparisons

Here we will compare the results from the different variations of our editor, as well as comment on their effectiveness.

Edge Filtering

By examining the results with and without the use of edge filtering we observe that they are quite similar. This leads us to assume that such a mechanism is redundant and its functionality is covered by the GNN solution to our graph assignment problem.

WordNet vs. Embeddings

As seen from Table 9.1 our variants that leverage the embedding models achieve better results in all metrics compared to our WordNet-based variants. Regarding *GPU inference*, the embedding models also outperform WordNet in terms of speed, since the latter requires API calls for each word/graph node of V, which greatly slow down the graph creation process.

POS-restricted vs. Unrestricted Substitutions

As we observe from Table 9.1, both editors, with and without POS filtering, achieve very similar results. This holds true for both IMDB and Newsgroups datasets, showing that the observed similarity is not due to a specific POS restriction. The only significant difference is seen in runtime (32 - 60 minutes for restricted editors, 2 - 4 hours for unrestricted ones), which is to be expected since when we only consider certain POS at a time, we also limit the amount of words that will be considered as candidates for substitution. This means that the graph nodes and edges of G will be significantly reduced, thus decreasing the time needed for graph construction and GNN inference.

Static vs. Dynamic Threshold

Results show insignificant improvement in metrics when using dynamic threshold compared to when using static threshold, while the runtime is increased (approximately by 1 hour per dataset). This slow-down is expected since dynamic threshold introduces an extra linear complexity for each text instance, in place of the O(1) complexity of the static case. Static is our default approach unless stated otherwise.

Contrastive vs fluent contrastive edits

While the general-purpose edits, in which only fluency is used as the heuristic function, achieve the lowest flip-rate, they remain better in all metrics compared to Polyjuice, another general-purpose editor. This shows that our framework can also be used as a general, untargeted editor with high-quality edits (regarding discussed metrics); extensive experimentation on this claim is left for future work. The label-flipping optimized edits, achieve better results in *fluency, closeness and minimality* compared to MiCE, a SoTA white-box editor optimized for label-flipping. Therefore, in terms of flip-rate, MiCE demonstrates superior performance, exceeding ours by 7%, accepting a significant 20x slowdown in execution.

9.3.3 Qualitative Results

Qualitative comparisons with Polyjuice and MiCE are presented in this Section to demonstrate the capabilities of our framework regarding minimality and flip-rate. For that purpose, we choose an instance of the IMDB dataset which is originally classified as 'positive' and acquire the edited instances from our framework and the two editors mentioned above. Specifically for Polyjuice, since its goal is to change the prediction from *positive* to *negative*, we use the control code [negation], which guides the editor to generate an edit that is the negation of the original text. The original along with the edited inputs (red words denote changes made by each editor) are shown in Figure 9.3.1.

As we can see, MiCE performs the highest number of interventions on the original input, with two of those changes being semantically incorrect (*"conservative, conservative"* and *"both of whom have"*). We also notice that its changes are not entirely word-level, which further deteriorates the editor's performance regarding *minimality*. Polyjuice on the other hand, makes only one change at the end of the text, which however has no semantic meaning; such edits may betray the presence of a counterfactual editor or a neural model in general, coming in contrast with the requirement of "imperceptible edits" that commonly involves counterfactual interventions. Our editor presents the best performance out of the three, changing only one word, while being semantically correct and very close to the original instance.

Original: This movie will likely be too sentimental for many viewers, especially contemporary audiences. Nevertheless I enjoyed this film thanks mostly to the down-to-earth charm of William Holden, one of my favorite stars, and the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors rescues what in lesser hands could've been trite lines. The cinematography of Hong Kong from the period of filming is another highlight of this movie. All in all, a better than average romantic drama, 7/10.

MiCE: This movie will likely be too harsh for many conservative, conservative audiences. Personally I enjoyed this film thanks mostly to the brilliant acting of William Powell, both of whom have the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors enhances what in less than average hands could've been trite lines. The beautiful performance of Hong Kong from the onset of filming is another highlight of this movie. All in all, a better than average romantic drama, 4/10.

Polyjuice: This movie will likely be too sentimental for many viewers, especially contemporary audiences. Nevertheless I enjoyed this film thanks mostly to the down-to-earth charm of William Holden, one of my favorite stars, and the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors rescues what in lesser hands could've been trite lines. The cinematography of Hong Kong from the period of filming is another highlight of this movie. All in all, of.

Ours: This movie will likely be too sentimental for many viewers, especially contemporary audiences. Nevertheless I enjoyed this film thanks mostly to the down-to-earth charm of William Holden, one of my favorite stars, and the dazzling beauty of Jennifer Jones. There are some truly heartwarming scenes between the pair and the talent of these two actors rescues what in lesser hands could've been trite lines. The cinematography of Hong Kong from the period of filming is another highlight of this movie. All in all, a better than average shameful drama, 7/10.

Figure 9.3.1: Original input and edited inputs from different editors. The changes that each editor performed are highlighted in red color.

Edits	Minimality ↓	Prediction Flipped
Polyjuice	0.078	False
MiCE	0.256	**True**
Ours	**0.011**	**True**

Table 9.2: Metric results of the edits presented in Figure 9.3.1. For each property (column) the best value is highlighted in **bold**.

Numeric results of Figure 9.3.1 instances regarding *minimality* and *label-flipping* are reported in Table 9.2. Since we only have one textual instance, instead of *flip-rate* we use the term *prediction flipped* to denote whether the edited input is able to change the original prediction of the classifier. Note that Polyjuice is unable to flip the prediction, while both MiCE and our framework succeed. Also, our editor is the best as far as minimality is concerned, with Polyjuice being second and MiCE being the worst out of the three.

Chapter 10

Conclusion

10.1 Discussion

In this work, we present a framework for generating optimal and controllable word-level counterfactuals via graph-based substitutions, which we evaluate on two classification tasks. One key component of our method is solving the Rectangular Linear Assignment Problem (RLAP), which is an extension of the more commonly seen Linear Assignment Problem (LAP). It is shown that with appropriate modifications RLAP can be solved with the same algorithms used to solve LAP. These algorithms, include the *Hungarian Algorithm* which solves RLAP in $O(n^3)$ and *Karp's Algorithm* which solves it in $O(mn \log m)$. In our work, we extend an existing GNN to accommodate RLAP which approximates the optimal solution found by the previous deterministic algorithms, while significantly improving the overall execution time. To the best of our knowledge, no previous work has tackled the RLAP using GNNs.

Our editor consists of three stages; in the *first* stage we construct a bipartite graph with source and target nodes (words), in the *second* stage we obtain suitable substitution pairs by solving the RLAP on the graph and in the *third* and final stage, we employ beam search in order to produce minimal, fluent and contrastive edits. In each stage, we used different approaches and setups, due to the highly customizable nature of the editor.

During the graph construction, we experimented with an edge filtering mechanism whose purpose was to penalize edges connecting different parts-of-speech (POS), which however was shown to be redundant based on the results. We also opted to use Large Embedding Models instead of WordNet to compute the similarity between two words. Our findings justify this decision, since embedding-based edits outperformed the Wordnet-based ones in all metrics. We tested the generalization capabilities of our editor by producing POS-restricted and unrestricted substitutions, and used both a dynamic and a static threshold to define the max number of substitutions for each text instance. It was interesting to find out that dynamic threshold did not improve the quality of edits and instead only increased the execution time. We also used different heuristic functions in beam search to produce edits optimized for different tasks, showcasing the flexibility of our editor.

Finally, we compared our framework with two SoTA editors, namely Polyjuice and MiCE. The former is a black-box general-purpose counterfactual editor, while the latter is a white-box counterfactual editor that produces minimal edits optimized for label-flipping. We used appropriate metrics, to measure *fluency*, *minimality*, *closeness* and *flip-rate*. Results show that we surpass them in most metrics, while being considerably faster, which is one of the main advantages of our method.

10.2 Broader Impact and Ethics

Our framework is intended to aid the interpretation of NLP models. As a model-agnostic explanation method by design (not optimized towards a certain metric in the default case), it has the potential to impact NLP system development across a wide range of models and tasks. In particular, our edits can assist developers

working on the NLP field in facilitating, debugging and exposing model vulnerabilities. The framework can also assist in data augmentation which results in less biased and more robust systems. As a consequence, downstream users of NLP models can also be benefited by gaining access to those systems.

While our work focuses on interpreting NLP models, it could be misused in other contexts. For instance, malicious users might generate adversarial examples, such as slightly altered hate speech, to bypass toxic language detectors. Additionally, using these editors for data augmentation could inadvertently lead to less robust and more biased models, as the edits are designed to expose model weaknesses. To avoid reinforcing existing biases, researchers should carefully consider how they select and label edited instances when using them for training. However, such threats are applicable to any text editor in NLP literature and are not tailored on our work.

10.3 Future Work

In closing this thesis we would like to suggest a few directions to further improve on this work or inspire different interesting approaches. As a first step, more external lexical sources (e.g. ConceptNet) could be integrated to enhance the possible substitution candidates, as well as computing more accurately the similarity between two words. Another step could be the further experimentation with the GNN used to solve RLAP, in order to improve its output compared with the optimal solution given by Karp's algorithm.

An appealing direction would be the exploration of the general-purpose version of our editor. This way, someone could gain important insights regarding its effectiveness in other tasks apart from label-flipping. We recommend comparing results with Polyjuice and Tailor [81], to evaluate its performance against SoTA general-purpose editors.

Finally, the trade-off between *minimality* and *flip-rate* has not been addressed in this dissertation, but could provide the basis for further research. Specifically, increasing the max number of substitutions per text instance could results in higher flip-rate since more words would change. However, the extra substitutions would increase the size of the edit (measured by number of changed words), thus lowering the editor's performance regarding minimality. The acceptable losses in minimality along with the desired gains in flip-rate, are open for further investigation.

Chapter 11

Bibliography

[1] Alvarez-Melis, D., Daumé III, H., Wortman Vaughan, J., and Wallach, H. "Weight of Evidence as a Basis for Human-Oriented Explanations". In: *Workshop on Human-Centric Machine Learning at the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.* Oct. 2019. URL:

[2] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information fusion* 58 (2020), pp. 82–115.

[3] Atwood, J. and Towsley, D. "Diffusion-convolutional neural networks". In: *Advances in neural information processing systems* 29 (2016).

[4] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[5] Benamara, F., Cesarano, C., Picariello, A., Reforgiato Recupero, D., and Subrahmanian, V. "Sentiment analysis: Adjectives and adverbs are better than adjectives alone". In: *ICWSM* (Nov. 2005).

[6] Bijsterbosch, J. and Volgenant, T. "Solving the Rectangular assignment problem and applications". In: *Annals OR* 181 (Dec. 2010), pp. 443–462. DOI: 10.1007/s10479-010-0757-3.

[7] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[8] Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. "Spectral networks and locally connected networks on graphs". In: *arXiv preprint arXiv:1312.6203* (2013).

[9] Burkard, R. and Çela, E. "Linear assignment problems and extensions". English. In: *Handbook of Combinatorial Optimization*. 1st ed. Supplement Volume A. Netherlands: Kluwer Academic Publishers, 1999, pp. 75–149.

[10] Chen, Z., Gao, Q., Bosselut, A., Sabharwal, A., and Richardson, K. "DISCO: Distilling Counterfactuals with Large Language Models". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 5514–5528. DOI: 10.18653/v1/2023.acl-long.302. URL:

[11] Cheng, F.-H., Hsu, W.-H., and Chen, C.-A. "Fuzzy approach to solve the recognition problem of handwritten chinese characters". In: *Pattern Recognition* 22.2 (1989), pp. 133–141. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(89)90060-5. URL:

[12] Defferrard, M., Bresson, X., and Vandergheynst, P. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in neural information processing systems* 29 (2016).

[13] Derr, T., Ma, Y., and Tang, J. "Signed graph convolutional networks". In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2018, pp. 929–934.

[14] Developers, G. *Foundational Courses - Embeddings*. [Online; accessed 23-September-2022]. 2022.

[15] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North Amer-*

ican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Ed. by J. Burstein, C. Doran, and T. Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL:

[16] Dimitriou, A., Chaidos, N., Lymperaiou, M., and Stamou, G. "Graph Edits for Counterfactual Explanations: A Comparative Study". In: *Explainable Artificial Intelligence*. Ed. by L. Longo, S. Lapuschkin, and C. Seifert. Cham: Springer Nature Switzerland, 2024, pp. 100–112. ISBN: 978-3-031-63797-1.

[17] Dimitriou, A., Lymperaiou, M., Filandrianos, G., Thomas, K., and Stamou, G. *Structure Your Data: Towards Semantic Graph Counterfactuals*. 2024. International Conference on Machine Learning (ICML):

[18] Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. "HotFlip: White-Box Adversarial Examples for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Ed. by I. Gurevych and Y. Miyao. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 31–36. DOI: 10.18653/v1/P18-2006. URL:

[19] Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. "Hypergraph neural networks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 3558–3565.

[20] Fey, M. and Lenssen, J. E. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[21] Filandrianos, G., Dervakos, E., Menis Mastromichalakis, O., Zerva, C., and Stamou, G. "Counterfactuals of Counterfactuals: a back-translation-inspired approach to analyse counterfactual editors". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by A. Rogers, J. Boyd-Graber, and N. Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 9507–9525. DOI: 10.18653/v1/2023.findings-acl.606. URL:

[22] Gardner, M. et al. "Evaluating Models' Local Decision Boundaries via Contrast Sets". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Ed. by T. Cohn, Y. He, and Y. Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 1307–1323. DOI: 10.18653/v1/2020.findings-emnlp.117. URL:

[23] Garg, S. and Ramakrishnan, G. "BAE: BERT-based Adversarial Examples for Text Classification". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by B. Webber, T. Cohn, Y. He, and Y. Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 6174–6181. DOI: 10.18653/v1/2020.emnlp-main.498. URL:

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

[25] Gilo, D. and Markovitch, S. "A General Search-Based Framework for Generating Textual Counterfactual Explanations". In: *AAAI Conference on Artificial Intelligence*. 2022. URL:

[26] GitHub, c. *GitHub - chemplexity/molecules: chemical graph theory library for JavaScript*. en. [online]. 2022. URL:

[27] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

[28] Grigoriadou, N., Lymperaiou, M., Filandrianos, G., and Stamou, G. "AILS-NTUA at SemEval-2024 Task 6: Efficient model tuning for hallucination detection and analysis". In: *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*. Ed. by A. K. Ojha, A. S. Doğruöz, H. Tayyar Madabushi, G. Da San Martino, S. Rosenthal, and A. Rosá. Mexico City, Mexico: Association for Computational Linguistics, June 2024, pp. 1549–1560. DOI: 10.18653/v1/2024.semeval-1.222. URL:

[29] Hamilton, W., Ying, Z., and Leskovec, J. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[30] Hinton, G. E. and Salakhutdinov, R. R. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[31] Hussein, R., Yang, D., and Cudré-Mauroux, P. "Are meta-paths necessary? Revisiting heterogeneous graph embeddings". In: *Proceedings of the 27th ACM international conference on information and knowledge management*. 2018, pp. 437–446.

[32] Iyyer, M., Wieting, J., Gimpel, K., and Zettlemoyer, L. "Adversarial Example Generation with Syntactically Controlled Paraphrase Networks". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*

(Long Papers). Ed. by M. Walker, H. Ji, and A. Stent. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1875–1885. DOI: 10.18653/v1/N18-1170. URL:

[33] James Im. *Introduction to PCA (Principal Component Analysis) - Medium*. [Online; accessed 23-September-2022]. 2018.

[34] Jia, R. and Liang, P. "Adversarial Examples for Evaluating Reading Comprehension Systems". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by M. Palmer, R. Hwa, and S. Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2021–2031. DOI: 10.18653/v1/D17-1215. URL:

[35] Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. *Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment*. 2020. arXiv: 1907.11932 [cs.CL]. URL:

[36] Karp, R. *An Algorithm to Solve the mxn Assignment Problem in Expected Time O (mn log n)*. Tech. rep. UCB/ERL M78/67. EECS Department, University of California, Berkeley, Sept. 1978. URL:

[37] Kaushik, D., Hovy, E., and Lipton, Z. C. *Learning the Difference that Makes a Difference with Counterfactually-Augmented Data*. 2020. arXiv: 1909.12434 [cs.CL]. URL:

[38] Keim, R. *How to Train a Basic Perceptron Neural Network*. en. [online]. 2019. URL:

[39] Khamsi, M. A. and Kirk, W. A. *An introduction to metric spaces and fixed point theory*. John Wiley & Sons, 2011.

[40] Kipf, T. N. and Welling, M. "Variational graph auto-encoders". In: *arXiv preprint arXiv:1611.07308* (2016).

[41] Koulakos, A., Lymperaiou, M., Filandrianos, G., and Stamou, G. *Enhancing adversarial robustness in Natural Language Inference using explanations*. 2024. arXiv: 2409.07423 [cs.CL]. URL:

[42] Kuhn, H. W. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. DOI: https://doi.org/10.1002/nav.3800020109. eprint: URL:

[43] Kuhn, H. W. "The Hungarian method for the assignment problem". In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[44] Lang, K. "NewsWeeder: learning to filter netnews". In: *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*. ICML'95. Tahoe City, California, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 331–339. ISBN: 1558603778.

[45] LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. "Object recognition with gradient-based learning". In: *Shape, contour and grouping in computer vision*. Springer, 1999, pp. 319–345.

[46] Li, D., Zhang, Y., Peng, H., Chen, L., Brockett, C., Sun, M.-T., and Dolan, B. "Contextualized Perturbation for Textual Adversarial Attack". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Ed. by K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou. Online: Association for Computational Linguistics, June 2021, pp. 5053–5069. DOI: 10.18653/v1/2021.naacl-main.400. URL:

[47] Li, L., Ma, R., Guo, Q., Xue, X., and Qiu, X. *BERT-ATTACK: Adversarial Attack Against BERT Using BERT*. 2020. arXiv: 2004.09984 [cs.CL]. URL:

[48] Li, X. and Li, J. "AnglE-optimized Text Embeddings". In: *arXiv preprint arXiv:2309.12871* (2023).

[49] Lin, C.-Y. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL:

[50] Liu, H., Wang, T., Lang, C., Feng, S., Jin, Y., and Li, Y. "GLAN: A graph-based linear assignment network". In: *Pattern Recognition* 155 (June 2024), p. 110694. DOI: 10.1016/j.patcog.2024.110694.

[51] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL:

[52] Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., and Yu, P. "Graph self-supervised learning: A survey". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[53] Lymperaiou, M., Filandrianos, G., Thomas, K., and Stamou, G. *Counterfactual Edits for Generative Evaluation*. 2023. arXiv: 2303.01555 [cs.CV]. URL:

[54] Lymperaiou, M., Manoliadis, G., Menis Mastromichalakis, O., Dervakos, E. G., and Stamou, G. "Towards Explainable Evaluation of Language Models on the Semantic Similarity of Visual Concepts". In: *Proceedings of the 29th International Conference on Computational Linguistics*. Ed. by N. Calzolari

et al. Gyeongju, Republic of Korea: International Committee on Computational Linguistics, Oct. 2022, pp. 3639–3658. URL:

[55] Lymperopoulos, D., Lymperaiou, M., Filandrianos, G., and Stamou, G. *Optimal and efficient text counterfactuals using Graph Neural Networks*. 2024. arXiv: 2408.01969 [cs.CL]. URL:

[56] Ma, G., Ahmed, N. K., Willke, T. L., and Yu, P. S. "Deep graph similarity learning: A survey". In: *Data Mining and Knowledge Discovery* 35.3 (2021), pp. 688–725.

[57] Ma, Y., Wang, S., Aggarwal, C. C., Yin, D., and Tang, J. "Multi-dimensional graph convolutional networks". In: *Proceedings of the 2019 siam international conference on data mining*. SIAM. 2019, pp. 657–665.

[58] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Ed. by D. Lin, Y. Matsumoto, and R. Mihalcea. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL:

[59] Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. "Invariant and equivariant graph networks". In: *arXiv preprint arXiv:1812.09902* (2018).

[60] Mikolov, T., Chen, K., Corrado, G., and Dean, J. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[61] Miller, G. A. "WordNet: a lexical database for English". In: *Commun. ACM* 38.11 (Nov. 1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL:

[62] Miller, T. "Explanation in artificial intelligence: Insights from the social sciences". In: *Artificial Intelligence* 267 (2019), pp. 1–38. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2018.07.007. URL:

[63] Mohr, I., Krimmel, M., Sturua, S., Akram, M. K., Koukounas, A., Günther, M., Mastrapas, G., Ravishankar, V., Martínez, J. F., Wang, F., et al. "Multi-Task Contrastive Learning for 8192-Token Bilingual Text Embeddings". In: *arXiv preprint arXiv:2402.17016* (2024).

[64] Molnar, C. *Interpretable machine learning*. Lulu. com, 2020.

[65] Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. "Geometric deep learning on graphs and manifolds using mixture model cnns". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5115–5124.

[66] Morris, J. X., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., and Qi, Y. *TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP*. 2020. arXiv: 2005.05909 [cs.CL]. URL:

[67] Mozes, M., Kleinberg, B., and Griffin, L. D. "Identifying Human Strategies for Generating Word-Level Adversarial Examples". In: *Conference on Empirical Methods in Natural Language Processing*. 2022. URL:

[68] Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. *MTEB: Massive Text Embedding Benchmark*. 2023. arXiv: 2210.07316 [cs.CL]. URL:

[69] Networkx. *PageRank algorithm | NetworkX Guide*. en. [online]. 2022. URL:

[70] Panagiotopoulos, I., Filandrianos, G., Lymperaiou, M., and Stamou, G. *AILS-NTUA at SemEval-2024 Task 9: Cracking Brain Teasers: Transformer Models for Lateral Thinking Puzzles*. 2024. arXiv: 2404.01084 [cs.CL]. URL:

[71] Panagiotopoulos, I., Filandrianos, G., Lymperaiou, M., and Stamou, G. *RISCORE: Enhancing In-Context Riddle Solving in Language Models through Context-Reconstructed Example Augmentation*. 2024. arXiv: 2409.16383 [cs.CL]. URL:

[72] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. ACL '02. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL:

[73] Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T., and Leiserson, C. "Evolvegcn: Evolving graph convolutional networks for dynamic graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5363–5370.

[74] Pennington, J., Socher, R., and Manning, C. D. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL:

[75] Pires, T., Schlinger, E., and Garrette, D. "How multilingual is multilingual BERT?" In: *arXiv preprint arXiv:1906.01502* (2019).

[76] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. "Language Models are Unsupervised Multitask Learners". In: (2019).
[77] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL:
[78] Ren, S., Deng, Y., He, K., and Che, W. "Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Ed. by A. Korhonen, D. Traum, and L. Màrquez. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 1085–1097. DOI: 10.18653/v1/P19-1103. URL:
[79] Ribeiro Neto, J. *Social Network — Big Data Success Case*. en. [online]. 2022. URL:
[80] Ross, A., Marasović, A., and Peters, M. "Explaining NLP Models via Minimal Contrastive Editing (MiCE)". In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Ed. by C. Zong, F. Xia, W. Li, and R. Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 3840–3852. DOI: 10.18653/v1/2021.findings-acl.336. URL:
[81] Ross, A., Wu, T., Peng, H., Peters, M., and Gardner, M. "Tailor: Generating and Perturbing Text with Semantic Controls". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by S. Muresan, P. Nakov, and A. Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3194–3213. DOI: 10.18653/v1/2022.acl-long.228. URL:
[82] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.
[83] Sachdeva, R., Tutek, M., and Gurevych, I. "CATfOOD: Counterfactual Augmented Training for Improving Out-of-Domain Performance and Calibration". In: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Y. Graham and M. Purver. St. Julian's, Malta: Association for Computational Linguistics, Mar. 2024, pp. 1876–1898. URL:
[84] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. "The graph neural network model". In: *IEEE transactions on neural networks* 20.1 (2008), pp. 61–80.
[85] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.
[86] Scheinerman, E. *Mathematics: A Discrete Introduction*. Cengage Learning, 2012. ISBN: 9780840049421. URL:
[87] Sean, L., Aamir, S., Darius, K., and Julius, L. *Open Source Strikes Bread - New Fluffy Embeddings Model*. 2024. URL:
[88] Solatorio, A. V. "GISTEmbed: Guided In-sample Selection of Training Negatives for Text Embedding Fine-tuning". In: *arXiv preprint arXiv:2402.16829* (2024). arXiv: 2402.16829 [cs.LG]. URL:
[89] Stoikou, T., Lymperaiou, M., and Stamou, G. *Knowledge-Based Counterfactual Queries for Visual Question Answering*. 2023. arXiv: 2303.02601 [cs.CL]. URL:
[90] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV]. URL:
[91] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).
[92] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).
[93] Wang, X., Girshick, R., Gupta, A., and He, K. "Non-local neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7794–7803.
[94] Wang, X., Xiong, Y., and He, K. "Detecting textual adversarial examples through randomized substitution and vote". In: *Conference on Uncertainty in Artificial Intelligence*. 2021. URL:
[95] Weisfeiler, B. and Leman, A. "The reduction of a graph to canonical form and the algebra which appears therein". In: *NTI, Series* 2.9 (1968), pp. 12–16.
[96] Wikipedia contributors. *Feedforward neural network — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-September-2022]. 2022.

[97] Wikipedia contributors. *Graph isomorphism — Wikipedia, The Free Encyclopedia*. [Online; accessed 26-September-2022]. 2022.
[98] Wikipedia contributors. *Machine learning — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-September-2022]. 2022.
[99] Wikipedia contributors. *Mean absolute error — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-September-2022]. 2022.
[100] Wikipedia contributors. *Principal component analysis — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-September-2022]. 2022.
[101] Wikipedia contributors. *Rectifier (neural networks) — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-September-2022]. 2022.
[102] Wikipedia contributors. *Semi-supervised learning — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-September-2022]. 2022.
[103] Wikipedia contributors. *Vanishing gradient problem — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-September-2022]. 2022.
[104] Wikipedia contributors. *Word embedding — Wikipedia, The Free Encyclopedia*. [Online; accessed 23-September-2022]. 2022.
[105] Wikipedia contributors. *Beam search — Wikipedia, The Free Encyclopedia*. [Online; accessed 1-September-2024]. 2024. URL:
[106] Wu, T., Ribeiro, M. T., Heer, J., and Weld, D. "Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by C. Zong, F. Xia, W. Li, and R. Navigli. Online: Association for Computational Linguistics, Aug. 2021, pp. 6707–6723. DOI: 10.18653/v1/2021.acl-long.523. URL:
[107] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.
[108] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32 (2019), pp. 4–24. URL:
[109] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826* (2018).
[110] Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., and Talukdar, P. "HyperGCN: Hypergraph Convolutional Networks for Semi-Supervised Learning and Combinatorial Optimisation". In: *arXiv* (2019).
[111] Yan, J., Yin, X.-C., Lin, W., Deng, C., Zha, H., and Yang, X. "A Short Survey of Recent Advances in Graph Matching". In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ICMR '16. New York, New York, USA: Association for Computing Machinery, 2016, pp. 167–174. ISBN: 9781450343596. DOI: 10.1145/2911996.2912035. URL:
[112] Yenduri, G. et al. *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. 2023. arXiv: 2305.10435 [cs.CL]. URL:
[113] Yin, K. and Neubig, G. "Interpreting Language Models with Contrastive Explanations". In: *Conference on Empirical Methods in Natural Language Processing*. 2022. URL:
[114] Yow, K. S. and Luo, S. *Learning-Based Approaches for Graph Problems: A Survey*. Apr. 2022.
[115] Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., and Sun, M. "Word-level Textual Adversarial Attacking as Combinatorial Optimization". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Ed. by D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault. Online: Association for Computational Linguistics, July 2020, pp. 6066–6080. DOI: 10.18653/v1/2020.acl-main.540. URL:
[116] Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. "Dive into Deep Learning". In: *arXiv preprint arXiv:2106.11342* (2021).
[117] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. "BERTScore: Evaluating Text Generation with BERT". In: *ArXiv* abs/1904.09675 (2019). URL:
[118] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81.

[119] Zhu, H., Zhao, Q., and Wu, Y. "BeamAttack: Generating High-quality Textual Adversarial Examples through Beam Search and Mixed Semantic Spaces". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. 2023. URL: