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

Exploring Text Counterfactual Explanations: A Multi-Metric Evaluation Approach for Counterfactual Editors

Diploma Thesis

by

Karavangelis Athanasios

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπουσα: Παρασκευή Τζούβελη
Ε.Δ.Ι.Π. Ε.Μ.Π.

Αθήνα, Ιούνιος 2023

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

Exploring Text Counterfactual Explanations: A Multi-Metric Evaluation Approach for Counterfactual Editors

Diploma Thesis

by

Karavangelis Athanasios

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπουσα: Παρασκευή Τζούβελη
Ε.Δ.Ι.Π. Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Ιουνίου, 2023.

....	....	....
Γεώργιος Στάμου	Στέφανος Κόλλιας	Αθανάσιος Βουλόδημος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2023

....
Καραβαγγελης Αθανασιος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Εν μέσω της εκθετικής ανάπτυξης και των επιτευγμάτων στη μηχανική μάθηση (ML) καθώς και του βαθύτατου αντίκτυπού της σε κρίσιμους τομείς, η ανάγκη για ερμηνευσιμότητα των μοντέλων είναι υψίστης σημασίας. Μια γέφυρα για αυτό το χάσμα μοντέλου-ανθρώπου παρέχεται από την Εξηγήσιμη Τεχνητή Νοημοσύνη (Explainable AI - XAI), η οποία έχει σημειώσει ταχεία πρόοδο τα τελευταία χρόνια, προσθέτοντας διαφάνεια στις διαδικασίες μηχανικής μάθησης. Στην παρούσα εργασία, εστιάζουμε στις εξηγήσεις με αντιπαράδειγμα, μια μέθοδο που παρέχει γνώσεις σχετικά με τη διαδικασία λήψης αποφάσεων των μοντέλων μηχανικής μάθησης, διερευνώντας εναλλακτικά σενάρια και υποθετικούς μετασχηματισμούς. Συγκεκριμένα, ασχολούμαστε με τη δημιουργία εξηγήσεων με αντιπαράδειγμα σε κείμενο και την αξιολόγηση των συντακτών αντιπαραδειγμάτων, οι οποίοι αξιοποιούν τα μοντέλα και τις εργασίες στην επεξεργασία φυσικής γλώσσας (NLP) για τη δημιουργία παραλλαγών των προτάσεων κειμένου. Η προσέγγισή μας περιλαμβάνει τον πειραματισμό με πολλαπλούς συντάκτες αντιπαραδειγμάτων από την πρόσφατη βιβλιογραφία, μοντέλα αλλά και μεθόδους παραγωγής, προκειμένου να κατανοήσουμε τους εσωτερικούς μηχανισμούς τους και να καταστήσουμε τις αποφάσεις τους κατανοητές. Για να το επιτύχουμε αυτό, παρουσιάζουμε ένα σύστημα επεξεργασίας αντιπαραδειγμάτων όπου παράγουμε αντιθετικές εξηγήσεις με αντιπαράδειγμα συνδυάζοντας τους συντάκτες αντιπαραδειγμάτων με έναν ταξινομητή και στη συνέχεια επιλέγοντας την ελάχιστη δυνατή επεξεργασία που αντιστρέφει την αρχική πρόβλεψη του ταξινομητή. Επιπλέον, αξιοποιούμε μεθόδους δημιουργίας αντιπαραδειγμάτων που χρησιμοποιούνται σε τρέχουσες ακαδημαϊκές δημοσιεύσεις και εισάγουμε μια νέα μέθοδο παραγωγής εξηγήσεων με αντιπαράδειγμα με τη χρήση ετικετών μέρους του λόγου ως περιορισμό στην παραγωγή τους. Εξετάζουμε επίσης πολλαπλές τεχνικές αξιολόγησης και μετρικές που μας επιτρέπουν να εξάγουμε πολύτιμα συμπεράσματα τα οποία καλύπτουν πολυάριθμες πτυχές της γέννησης εξηγήσεων με αντιπαράδειγμα. Συνοπτικά, τα πειράματά μας απέδωσαν πολύτιμα συμπεράσματα και γνώσεις. Καταφέρνουμε να αποκαλύψουμε κρυμμένα χαρακτηριστικά και μοτίβα των συντακτών αντιπαραδειγμάτων, να εξηγήσουμε τα αποτελέσματά τους και να διερευνήσουμε διάφορες πτυχές της δημιουργίας αντιπαραδειγμάτων. Τα πειράματά μας αναδεικνύουν επίσης βελτιώσεις στην απόδοση των μεθόδων γέννησης αντιπαραδειγμάτων μέσω μιας συστηματικής διερεύνησης των δομικών συστατικών και μεθοδολογιών που χρησιμοποιούν. Ως εκ τούτου, οι συνεισφορές αυτής της διατριβής, συμπεριλαμβανομένης της αξιοποίησης και της παρουσίασης νέων μεθόδων στον τομέα της δημιουργίας εξηγήσεων με αντιπαράδειγμα και μίας ολοκληρωμένης ανάλυσης σχετικά με την αξιολόγηση των συντακτών αντιπαραδειγμάτων αποδεικνύεται ότι αποτελούν μια πολλά υποσχόμενη οδό για μελλοντική έρευνα.

Λέξεις-κλειδιά — Εξηγήσιμη Τεχνητή Νοημοσύνη, Εξηγήσεις με Αντιπαράδειγμα, Αντιφατικά Κείμενα, Μοντέλα Μηχανικής Μάθησης, Παραγωγή Κειμένου, Αξιολόγηση με Πολλαπλές Μετρικές

Abstract

Amidst the exponential growth and breakthroughs in machine learning (ML) and its profound impact on critical domains, the need for interpretability of the models is paramount. A bridge for this model-human gap is provided by Explainable AI (XAI), which has seen rapid progress in recent years, adding transparency to machine learning processes. In this work, we focus on counterfactual explanations, a method that provides insights into the decision-making process of machine learning models by exploring alternative scenarios and hypothetical transformations. Specifically, we are concerned with the generation of text counterfactual explanations and the evaluation of counterfactual editors, which leverage natural language processing (NLP) models and tasks to generate perturbations of text sentences. Our approach involves experimenting with multiple counterfactual editors from the recent literature, models, and generation methods in order to understand their inner mechanisms and make their decisions comprehensive. In order to achieve this, we present a counterfactual editing system where we generate counterfactual, contrastive edits combining counterfactual editors with a predictor and then selecting the most minimal edit that flips the predictor's original prediction. Moreover, we utilize methods of counterfactual generation used in current academic publications and introduce a novel method of generating counterfactual edits using part-of-speech tags to constrain the generation. We also explore multiple evaluation techniques and metrics that allow us to extract valuable conclusions that cover numerous aspects of counterfactual generation. In summary, our experiments have yielded valuable conclusions and insights. We manage to unveil hidden characteristics and patterns of counterfactual editors, explain their results, and explore various aspects of counterfactual generation. Our experiments showcase performance enhancements in counterfactual generation methods through a systematic exploration of their structural components and methodologies. Therefore, the contributions of this thesis including the utilization and introduction of novel methods in the field of counterfactual generation and a comprehensive analysis on the evaluation of counterfactual editors prove to be a promising avenue for future research.

Keywords — Explainable AI, Counterfactual Explanations, Text Counterfactuals, Machine Learning Models, Text Generation, Multi-metrics Evaluation

Ευχαριστίες

Κάθε λογής έργο, όπως και αυτό, απαιτεί τη συμβολή πολλών ανθρώπων χωρίς τους οποίους σίγουρα δε θα ήταν δυνατή η εκπόνηση του. Γι'αυτό λοιπόν, ευχαριστώ πολύ τον επιβλέποντα μου, κ. Στάμου Γεώργιο, για την πολύτιμη στήριξη του στην εκπόνηση της εργασίας αυτής. Ευχαριστώ επίσης την συνεπιβλέπουσα κα. Τζόυβελη Παρασκευή, τη Μαρία Λυμπεραίου, τον Γεώργιο Φιλανδριανό και τον Ορφέα-Μένη Μαστρομιχαλάκη, για την καθοδήγηση, τις ιδέες και τις αμέτρητες συμβουλές που μου έδωσαν καθ'όλη τη διάρκεια αυτής της εξερεύνησης ενός νέου αντικειμένου για εμένα, καθώς και τη διάθεση τους να μοιραστούν τη δουλειά τους μαζί μου.

Επίσης, δε θα μπορούσα να μην ευχαριστήσω τους γονείς μου Μαρίνα και Νίκο, καθώς και τα αδέρφια μου Εβελίνα, Ζέτα και Αλέξανδρο για την αγάπη και τη στήριξη που μου παρέχουν πάντοτε. Ακόμα, θα ήθελα να ευχαριστήσω την κοπέλα μου, Αθηνά, η οποία είναι το στήριγμα μου σε ότι κάνω, είναι ο άνθρωπος που δείχνει καθημερινά αμέριστη συμπαράσταση και κατανόηση. Θα ήθελα επίσης να δώσω ένα ευχαριστώ στους φίλους μου Μιχάλη και Κωνσταντίνο που με στηρίζουν σε κάθε μου βήμα με το δικό τους μοναδικό τρόπο, με γέλιο και ευχάριστη διάθεση. Τέλος, ένα ξεχωριστό ευχαριστώ στους συμφοιτητές και φίλους Γιάννη και Αντρέα με τους οποίους ζήσαμε μαζί αυτό το ακαδημαϊκό ταξίδι από την αρχή ως το τέλος.

Καραβαγγέλης Αθανάσιος, Ιούνιος 2023

Contents

List of Figures

Chapter 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Θεωρητικό υπόβαθρο

Τις τελευταίες δεκαετίες, η μηχανική μάθηση (ML) έχει σημειώσει αξιοσημείωτες εξελίξεις, έχοντας επιδείξει εντυπωσιακές ικανότητες σε διάφορες εφαρμογές από την αναγνώριση εικόνας και ομιλίας έως την επεξεργασία φυσικής γλώσσας και τα συστήματα συστάσεων. Αυτές οι καινοτομίες έχουν βρει εφαρμογή σε πολλές πτυχές της καθημερινής μας ζωής, από τις έξυπνες συσκευές και τους βοηθούς με τεχνητή νοημοσύνη μέχρι τη δημιουργία τέχνης, τα αυτοκινούμενα αυτοκίνητα και την τεχνητή νοημοσύνη στην υγειονομική περίθαλψη. Ωστόσο, εν μέσω αυτών των αξιοσημείωτων εξελίξεων, είναι εξαιρετικά σημαντικό να διερευνήσουμε το επίπεδο εμπιστοσύνης που μπορούμε να οικοδομήσουμε με αυτά τα εργαλεία. Προκύπτει συνεπώς ένα θεμελιώδες ερώτημα: Τι μπορούμε να κάνουμε για να ενεργοποιήσουμε συστήματα τεχνητής νοημοσύνης που μπορούν να εξηγούν τις αποφάσεις που λαμβάνουν τα μοντέλα;

Ο τομέας της Εξηγήσιμης Τεχνητής Νοημοσύνης (Explainable AI - XAI) στοχεύει να απαντήσει ακριβώς σε αυτό και να παράσχει πληροφορίες πίσω από τις ενέργειες και τις αποφάσεις των μοντέλων ML ώστε να επιτρέπει στους ειδικούς του τομέα να εντοπίζουν πιθανές προκαταλήψεις [3], σφάλματα ή ακούσιες συνέπειες. Σε αυτή τη διατριβή, διερευνούμε την έννοια των εξηγήσεων με αντιπαράδειγμα, οι οποίες παρέχουν ένα ισχυρό πλαίσιο για την υλοποίηση της εξηγησιμότητας και μας επιτρέπουν να παρατηρήσουμε πώς οι αλλαγές στο κείμενο επηρεάζουν τις αποφάσεις που λαμβάνει το μοντέλο και να αποκαλύψουμε τους υποκείμενους παράγοντες που οδηγούν τις προβλέψεις του μοντέλου.

Στην παρούσα εργασία, επικεντρωνόμαστε στην αξιολόγηση των εξηγήσεων που παράγονται από συντάκτες κειμενικών εξηγήσεων με αντιπαράδειγμα με πολλαπλές μεθόδους, μοντέλα και περιορισμούς. Χρησιμοποιούμε διάφορες μεθόδους και μετρικές αξιολόγησης προκειμένου να ερμηνεύσουμε τη συμπεριφορά αυτών των στοιχείων και να δώσουμε εξηγήσεις για τις αποφάσεις τους. Το έργο μας παρακινείται κυρίως από την έρευνα των Φιλανδριανού και λοιποί [20], όπου οι συγγραφείς αξιολογούν επίσης κειμενικές εξηγήσεις με αντιπαράδειγμα. Χρησιμοποιώντας δύο νέες μεθόδους παραγωγής και αξιολόγησης, που παρουσιάστηκαν για πρώτη φορά στην εν λόγω έρευνα, εξετάζοντας έτσι μια πρωτότυπη διαδικασία αξιολόγησης σε συνδυασμό με πιο παραδοσιακές. Μια άλλη εστίαση αυτής της εργασίας, είναι η υλοποίηση ενός συστήματος δημιουργίας επεξηγήσεων με αντιπαράδειγμα, όπου συνδυάζουμε συντάκτες κειμενικών αντιπαραδειγμάτων με ένα σύστημα πρόβλεψης κλάσης, προκειμένου να συγκρίνουμε δίκαια τα μοντέλα και τις μεθόδους στο έργο της αντιθετικής αντιπαραθετικών εξηγήσεων.

Στην παρούσα διατριβή αρχικά παρέχουμε όλο το θεωρητικό υπόβαθρο σε βασικές έννοιες της μηχανικής μάθησης και εξηγούμε πώς οι εξηγήσεις αντιφατικών γεγονότων είναι ένα σημαντικό μέσο για την ερμηνευσιμότητα. Έπειτα, δίνουμε έναν λεπτομερή ορισμό των εξηγήσεων με αντιπαράδειγμα και παρέχουμε τα κύρια κίνητρα πίσω από τη χρήση τους. Μετά από αυτό, συζητάμε τις μεθόδους αξιολόγησης και τις μετρικές που ακολουθούνται από την πρόσφατη βιβλιογραφία, εστιάζοντας σε αυτές που χρησιμοποιούμε στην εργασία μας. Επίσης, προτείνουμε τη χρήση των μεθόδων που παρουσιάστηκαν στο [20] μαζί με τη δική μας μέθοδο που χρησιμοποιεί στοχευμένα ένα μέρος του λόγου κάθε φορά, ως κριτήριο για τις αλλαγές στις κειμενικές προτάσεις και εξηγούμε πώς μπορούν να συμβάλουν στην αξιολόγηση της εξήγησης αντιφατικών γεγονότων. Τέλος, παρουσιάζουμε τον τρόπο με τον οποίο δομούμε και υλοποιούμε το σύστημα δημιουργίας κειμενικών εξηγήσεων με αντιπαράδειγμα και παρουσιάζουμε τα αποτελέσματα που προέκυψαν τόσο από ποσοτική όσο και από ποιοτική άποψη. Μέσω αυτής της διαδικασίας, θα παρέχουμε πολυάριθμες εξηγήσεις με αντιπαράδειγμα μαζί με επεξηγήσεις για τις αποφάσεις των συντακτών εξηγήσεων, πολύτιμα συμπεράσματα αλλά και πιθανούς περιορισμούς.

1.1.1 Επεξεργασία Φυσικής Γλώσσας

Παραγωγή Κειμένου και Transformers

Η παραγωγή κειμένου είναι ένας ενεργός τομέας έρευνας στον τομέα της Επεξεργασίας Φυσικής Γλώσσας (NLP), που επικεντρώνεται στην ανάπτυξη αλγορίθμων και μοντέλων ικανών να παράγουν συνεκτικό και σχετικό με το περιεχόμενο κείμενο. Δύο εξέχουσες προσεγγίσεις στην παραγωγή κειμένου είναι τα πιθανοτικά μοντέλα, όπως τα μοντέλα n-gram και τα κρυφά μοντέλα Markov, και τα μοντέλα που βασίζονται σε νευρωνικά δίκτυα, όπως τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) και οι παραλλαγές τους, όπως τα δίκτυα μακράς βραχυπρόθεσμης μνήμης (LSTM). Οι πρόσφατες εξελίξεις στη βαθιά μάθηση οδήγησαν στην εμφάνιση μοντέλων μετασχηματιστών (Transformers), τα οποία έχουν επηρεάσει σημαντικά τις εργασίες παραγωγής κειμένου. Τα μοντέλα αυτά συνήθως εκπαιδεύονται χρησιμοποιώντας μεγάλα σώματα δεδομένων κειμένου, είτε με επίβλεψη είτε χωρίς επίβλεψη. Η αξιολόγηση των μοντέλων παραγωγής κειμένου είναι ένα δύσκολο έργο, καθώς απαιτεί μια ολοκληρωμένη αξιολόγηση της παραγόμενης εξόδου από άποψη συνάφειας, συνοχής, γραμματικής και συνολικής ποιότητας. Οι δεοντολογικοί προβληματισμοί είναι επίσης εξαιρετικά σημαντικοί στην έρευνα για την παραγωγή κειμένου.

Οι **Transformers** είναι ένα σημαντικό στοιχείο της επεξεργασίας φυσικής γλώσσας (NLP) για την παραγωγή κειμένου. Αξιοποιούν τον μηχανισμό προσοχής για να συλλάβουν τις παγκόσμιες εξαρτήσεις και να επιτρέψουν την παράλληλη επεξεργασία ακολουθιών εισόδου. Το πλαίσιο **κωδικοποιητή-αποκωδικοποιητή**, ο μηχανισμός αυτό-προσοχής και τα νευρωνικά δίκτυα τροφοδότησης προς τα εμπρός είναι όλα βασικά συστατικά της αρχιτεκτονικής του μετασχηματιστή. Η **αυτοπροσοχή** είναι ένα βασικό χαρακτηριστικό του Transformer, που επιτρέπει στο μοντέλο να εστιάζει σε σχετικά μέρη της ακολουθίας εισόδου και να υπολογίζει βαθμολογίες προσοχής μεταξύ όλων των θέσεων. Το **GPT** και το **T5** είναι δύο ισχυρά μοντέλα που βασίζονται στον Transformer και έχουν επιδείξει εντυπωσιακές ικανότητες στη δημιουργία συνεκτικού και σχετικού με το πλαίσιο κειμένου, και η περαιτέρω πρόοδος τους έχει τη δυνατότητα να βελτιώσει περαιτέρω την ικανότητά μας να παράγουμε κείμενο που μοιάζει με ανθρώπινο καθώς και τη δημιουργία ευφυών συστημάτων.

Κατηγοριοποίηση Κειμένου

Η κατηγοριοποίηση κειμένου είναι μια θεμελιώδης εργασία στην Επεξεργασία Φυσικής Γλώσσας (NLP) που περιλαμβάνει την ανάθεση προκαθορισμένων κατηγοριών ή ετικετών σε έγγραφα κειμένου με βάση το περιεχόμενό τους. Παραδοσιακοί αλγόριθμοι μηχανικής μάθησης, όπως ο Naive Bayes, οι Μηχανές Διανυσμάτων Υποστήριξης (SVM) και τα Δέντρα Αποφάσεων, έχουν εφαρμοστεί με επιτυχία σε αυτό το έργο. Τα νευρωνικά δίκτυα, όπως τα συνελικτικά νευρωνικά δίκτυα (CNN) και τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN), έχουν επίσης αναδειχθεί ως ισχυρά μοντέλα για την ταξινόμηση κειμένου. Προεκπαιδευμένα γλωσσικά μοντέλα όπως το BERT (Bidirectional Encoder Representations from Transformers) [12] και το RoBERTa [46] έχουν προωθήσει σημαντικά τον τομέα της ταξινόμησης κειμένου. Η αξιολόγηση των μοντέλων ταξινόμησης κειμένου γίνεται συνήθως με τη χρήση μετρικών όπως η ακρίβεια, η ακρίβεια, η ανάκληση και η βαθμολογία F1. Η ταξινόμηση κειμένου βρίσκει εφαρμογές σε ένα ευρύ φάσμα πεδίων, όπως η ανάλυση συναισθήματος για την παρακολούθηση των μέσων κοινωνικής δικτύωσης, το φιλτράρισμα ανεπιθύμητης αλληλογραφίας για συστήματα ηλεκτρονικού ταχυδρομείου, η κατηγοριοποίηση θεμάτων για την οργάνωση περιεχομένου και τα συστήματα συστάσεων με βάση το συναίσθημα.

Ανάλυση Συναισθήματος

Η ανάλυση συναισθήματος, γνωστή και ως εξόρυξη γνώμης, είναι ένας ερευνητικός τομέας της Επεξεργασίας Φυσικής Γλώσσας (NLP) που επικεντρώνεται στην εξαγωγή υποκειμενικών πληροφοριών, στάσεων και απόψεων που εκφράζονται σε δεδομένα κειμένου. Περιλαμβάνει τη χρήση υπολογιστικών μεθόδων για την αυτόματη ταξινόμηση κειμένου σε κατηγορίες συναισθημάτων, όπως θετικά, αρνητικά ή ουδέτερα.

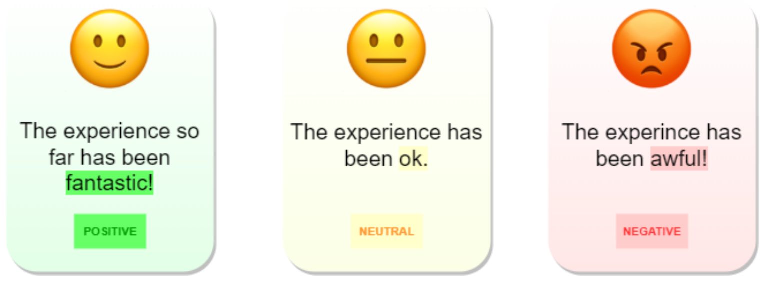

The experience so far has been fantastic!
POSITIVE
The experience has been ok.
NEUTRAL
The experince has been awful!
NEGATIVE

Figure 1.1.1: Ένα παράδειγμα το ποίο δείχνει πως μία πρόταση σημειώνεται από το κατάλληλο μοντέλο ως θετική, ουδέτερη η αρνητική ανάλογα με το κυρίαρχο συναίσθημά της.

Οι παραδοσιακές προσεγγίσεις στην ανάλυση συναισθήματος στηρίζονται σε μεθόδους που βασίζονται σε λεξικά, αλλά οι αλγόριθμοι μηχανικής μάθησης, όπως οι μηχανές διανυσμάτων υποστήριξης (SVM), Naive Bayes και η λογιστική παλινδρόμηση, έχουν χρησιμοποιηθεί ευρέως για εργασίες ταξινόμησης συναισθήματος. Τα μοντέλα βαθιάς μάθησης έχουν επιδείξει αξιοσημείωτη επιτυχία στην ανάλυση συναισθήματος αξιοποιώντας την ικανότητά τους να συλλαμβάνουν περίπλοκα μοτίβα και πληροφορίες πλαισίου από δεδομένα κειμένου. Η ανάλυση συναισθήματος είναι η διαδικασία ταξινόμησης των συναισθημάτων σε κείμενο με τη χρήση CNN και RNN, με βαθιά γλωσσικά μοντέλα όπως το RoBERTa να χρησιμοποιούνται για τη σύλληψη πιο δύσκολων τομέων δεδομένων. Για τη βελτίωση της απόδοσης της ανάλυσης συναισθήματος έχουν διερευνηθεί προηγμένες τεχνικές, όπως η ανάλυση συναισθήματος με βάση τις πτυχές και οι βαθιές αναπαραστάσεις λέξεων με βάση τα συμφραζόμενα.

Επισήμανση Μέρους του Λόγου

Η επισήμανση μέρους του λόγου (POS - part-of-speech) είναι μια θεμελιώδης εργασία στην Επεξεργασία Φυσικής Γλώσσας (NLP) που περιλαμβάνει την ανάθεση γραμματικών ετικετών στις λέξεις μιας πρότασης, υποδεικνύοντας τον συντακτικό τους ρόλο και τη λειτουργία τους. Οι παραδοσιακές προσεγγίσεις για την επισήμανση POS έχουν χρησιμοποιήσει μεθόδους βασισμένες σε κανόνες, στοχαστικά μοντέλα και ακόμη και ανθρώπινους σχολιαστές. Τα μοντέλα βαθιάς μάθησης, ιδίως τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) και οι αρχιτεκτονικές που βασίζονται σε μετασχηματιστές, έχουν κερδίσει σημαντική προσοχή στην επισήμανση POS λόγω της ικανότητάς τους να συλλαμβάνουν τις διαδοχικές εξαρτήσεις και τις πληροφορίες που σχετίζονται με το πλαίσιο. Η ακριβής επισημείωση POS έχει σημαντικές επιπτώσεις σε επόμενες εργασίες NLP, διευκολύνοντας την ακριβέστερη συντακτική ανάλυση, τη βελτιωμένη αναγνώριση ονομαστικών οντοτήτων και τα βελτιωμένα συστήματα αυτόματης μετάφρασης.

1.1.2 Νευρωνικά Δίκτυα

Στη μάθηση με επίβλεψη, ένα νευρωνικό δίκτυο εκπαιδεύεται χρησιμοποιώντας έναν σταθερό αριθμό N δειγμάτων από το σύνολο δεδομένων εκπαίδευσης $D = (x_1, y_1), ..., (x_n, y_n)$, όπου το x_i αντιπροσωπεύει την είσοδο και το y_i την αντίστοιχη ετικέτα. Το μοντέλο στοχεύει στον υπολογισμό μιας συνάρτησης $f : X\text{ß}Y$ που αντιστοιχίζει την είσοδο X στην έξοδο Y, και οι εκπαιδεύσιμες παράμετροι του συχνά αναφέρονται ως **βάρη**. Η απόδοση της f αξιολογείται με τη χρήση μιας **συνάρτησης απώλειας** (loss function) L, η οποία μετρά τη διαφορά μεταξύ της προβλεπόμενης εξόδου και της πραγματικής ετικέτας. Επίσης, σε ένα νευρωνικό δίκτυο, η έξοδος κάθε νευρώνα δεν καθορίζεται αποκλειστικά από το σταθμισμένο άθροισμα των εισόδων x_i. Αντ' αυτού, χρησιμοποιείται μια συνάρτηση ενεργοποίησης για να μετατρέψει το σταθμισμένο άθροισμα σε έξοδο. Η επιλογή της **συνάρτησης ενεργοποίησης** είναι κρίσιμη, καθώς επηρεάζει σημαντικά την απόδοση του δικτύου. Κατά τη διάρκεια της εκπαίδευσης, ο στόχος είναι η προσαρμογή των βαρών του f με τρόπο που να ελαχιστοποιεί τη συνάρτηση απώλειας. Για αυτή καθώς και τις παρακάτω παραγράφους χρησιμοποιούμε αναφορές από το [23].

Όσον αφορά την **εκπαίδευση των νευρωνικών δικτύων**, αυτή περιλαμβάνει την αρχικοποίηση των παραμέτρων, τη διάδοση προς τα εμπρός και τη σύγκριση των προβλέψεων με τις τιμές-στόχους. Η διαδικασία εκπαίδευσης των νευρωνικών δικτύων περιλαμβάνει τον υπολογισμό των κλίσεων επίπεδο προς επίπεδο, την ενημέρωση των παραμέτρων του δικτύου μέσω ενός αλγορίθμου βελτιστοποίησης και τη διέλευση ολόκληρου του συνόλου δεδομένων εκπαίδευσης μέσω πολλαπλών επαναλήψεων ή **εποχών**. Είναι επίσης σημαντικό να παρακολουθείται η απόδοση του μοντέλου σε αθέατα δεδομένα και να αξιολογείται σε ένα ξεχωριστό σύνολο δοκιμών. Ακολουθώντας αυτά τα βήματα, τα νευρωνικά δίκτυα μπορούν να εκπαιδευτούν ώστε να εξάγουν σημαντικές πληροφορίες από τα δεδομένα και να παρέχουν ακριβείς προβλέψεις.

Ακόμη, τα **προεκπαιδευμένα νευρωνικά δίκτυα** αποτελούν πολύτιμο πλεονέκτημα στη βαθιά μάθηση, καθώς έχουν προ-εκπαιδευτεί σε μεγάλα σύνολα δεδομένων και περιέχουν εκπαιδευμένα βάρη και παραμέτρους. Μπορούν να αποδώσουν καλά σε ένα ευρύ φάσμα εργασιών, ακόμη και με περιορισμένα δεδομένα εκπαίδευσης, και χρησιμοποιούνται συνήθως στη μάθηση με μεταφορά. Τα προεκπαιδευμένα μοντέλα έχουν φέρει επανάσταση στον τομέα της επεξεργασίας φυσικής γλώσσας, παρέχοντας έτοιμες προς χρήση, ισχυρές και αποτελεσματικές λύσεις για διάφορες εργασίες. Συνήθως εκπαιδεύονται σε μεγάλα σώματα δεδομένων κειμένου και μπορούν να χρησιμοποιηθούν για την εξαγωγή χαρακτηριστικών, την εξαγωγή συμπερασμάτων και την αξιολόγηση μετρήσεων.

1.1.3 Εξηγήσεις με Αντιπαράδειγμα

Μια εξήγηση με αντιπαράδειγμα ή αντιφατική εξήγηση είναι μια δήλωση που περιγράφει την αιτιώδη συνάφεια μιας κατάστασης υποθέτοντας ότι αν δεν είχε συμβεί το A, δε θα είχε συμβεί το B. Προκύπτει από την ανάγκη να φανταστούμε μια πλαστή πραγματικότητα που έρχεται σε αντίθεση με την αντίληψή μας. Ένα παράδειγμα αντιφατικής εξήγησης είναι η υπόθεση ότι αν δεν είχα χάσει το λεωφορείο, δε θα είχα αργήσει στη δουλειά. Για να τεθεί αυτό το παράδειγμα σε ένα ερμηνεύσιμο πλαίσιο στην ΤΝ, είναι απαραίτητο να σημειωθεί ότι το "γεγονός" είναι η πρόβλεψη ενός μοντέλου για μια συγκεκριμένη είσοδο και τα "αίτια" είναι κάποιες τιμές χαρακτηριστικών για την ίδια είσοδο που οδήγησαν το μοντέλο σε μια συγκεκριμένη πρόβλεψη. Το ερώτημα στο οποίο σκοπεύει να απαντήσει ένα αντιφατικό γεγονός είναι τι θα έπρεπε να αλλάξει για να κατηγοριοποιηθεί κάτι στην κλάση X αντί για την κλάση Υ.

Μια αντιφατική εξήγηση είναι επιλεκτική και αντιθετική, και σε αυτή διαφοροποιείται κάθε φορά ένα μικρό κομμάτι του αρχικού κειμένου και έτσι καθίσταται φιλική όσον αφορά την κατανόηση από τον αναγνώστη. Η διαδικασία αξιολόγησης μιας αντιφατικής εξήγησης ξεκινά με τον τελικό χρήστη να καθορίζει μια επιθυμητή πρόβλεψη ώστε να παραχθεί μια "ελάχιστη" εξήγηση, που είναι δηλαδή όσο πιο κοντινή στην αρχική περίπτωση όσον αφορά την ομοιότητα των τιμών των χαρακτηριστικών (features). Είναι ζωτικής σημασίας να δημιουργούνται πολυάριθμες αντιφατικές εξηγήσεις, ώστε να υπάρχει ποικιλομορφία και η δυνατότητα επιλογής της βέλτιστης ανάμεσα σε πολλαπλές επιλογές.

1.1.4 Κειμενικοί Συντάκτες Αντιπαραδειγμάτων

Η παρούσα διατριβή διερευνά συστήματα που στοχεύουν στην ελάχιστη δυνατή επεξεργασία μιας δεδομένης περίπτωσης κειμένου προκειμένου να αλλάξουν την πρόβλεψη ενός ταξινομητή. Αυτά τα συστήματα ονομάζονται κειμενικοί συντάκτες αντιπαραδειγμάτων (Counterfactual Editors). Οι συντάκτες αυτοί χρησιμοποιούν διάφορες μεθοδολογίες για την επίτευξη του επιθυμητού αποτελέσματος, όπως για παράδειγμα η δημιουργία αντιπαραδειγμάτων που εξαρτάται άμεσα από την έξοδο ενός συγκεκριμένου ταξινομητή. Μία από αυτές τις υλοποιήσεις είναι ο συντάκτης **MiCE** [75]. Ακόμη, υπάρχουν ορισμένοι συντάκτες όπως ο **Polyjuice** [102] που προσπαθούν να αλλάξουν τη σημασιολογική της πρότασης δημιουργώντας έτσι ποικίλα αντιπαραδείγματα που μπορούν να χρησιμοποιηθούν στην επέκταση δεδομένων (data augmentation) ή τα "υπό συνθήκες αντιπαραδείγματα". Επιπλέον, ορισμένοι επεξεργαστές προορίζονται για τη δημιουργία αντιφατικών παραδειγμάτων, τα οποία αποσκοπούν στον εντοπισμό και την αποκάλυψη τρωτών σημείων ενός ταξινομητή. Ένα πλαίσιο που ενθυλακώνει πολλές τέτοιες υλοποιήσεις είναι το TextAttack [60], το οποίο περιλαμβάνει πολλές μεθόδους γέννησης - συντάκτες, όπως το **TextFooler** [35]. Αυτές οι μέθοδοι είναι πιο απλές και χρησιμοποιούν διάφορες τεχνικές για τη δημιουργία παραδειγμάτων, όπως η εναλλαγή λέξεων με χρήση gradient descent ή η χρήση ενσωματώσεων λέξεων (word embeddings).

Original
Perfect performance by the actor → Positive (99%)
Adversarial
Spotless performance by the actor → Negative (100%)

Table 1.1: Εξήγηση με αντιπαράδειγμα που δημιουργείται με τη χρήση του TextFooler [35] για έναν ταξινομητή συναισθήματος βασισμένο στο BERT. Αντικαθιστώντας τη λέξη "perfect" με τη συνώνυμη της "spotless" αλλάζει εντελώς η πρόβλεψη του μοντέλου, παρόλο που η υποκείμενη έννοια της πρότασης δεν έχει αλλάξει. [60]

1.1.5 Αξιολόγηση Κειμενικών Συντακτών Αντιπαραδειγμάτων

Παρά το γεγονός ότι δεν έχουν καθιερωθεί καθολικές μέθοδοι για την αξιολόγηση της απόδοσης ενός συντάκτη αντιπαραδειγμάτων, υπάρχουν αρκετές βασικές αλήθειες και κοινές μετρικές με βάση τις οποίες μπορούμε να συγκρίνουμε τους συντάκτες και πόσο αποτελεσματικά λειτουργούν. Θα αναφερθούμε σε αυτές των οποίων κάνουμε χρήση εμείς στη διατριβή αυτή. Αρχικά, μία πολύ σημαντική μετρική είναι το "**ποσοστό αντιστροφής**" ή flip rate, η οποία ισούται με το ποσοστό των αντιπαραδειγμάτων που καταφέραμε να ταξινομηθούν σε διαφορετική κλάση από την αρχική, επί των συνολικών. Μία άλλη πολύ σημαντική μετρική είναι η λεγόμενη "**απόσταση Λέβενσταϊν**" [95] ή αλλιώς "ελαχιστότητα" (minimality) η οποία εκφράζει κατά πόσο είναι οι ελάχιστες δυνατές οι αλλαγές που έκανε ο συντάκτης. Ειδικότερα:

$$lev(a,b) = \begin{cases} |a| & \text{if } b = 0 \\ |b| & \text{if } a = 0 \\ lev(\text{tail}(a), \text{tail}(b)) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} lev(\text{tail}(a), b) \\ lev(a, \text{tail}(b)) \\ lev(\text{tail}(a), \text{tail}(b)) \end{cases} & \text{otherwise} \end{cases}$$

Ακόμη υπάρχει μία κατηγορία μετρικών που αξιολογεί την "**ευχέρεια λόγου**" ή αλλιώς fluency. Ωστόσο, ένα συχνό μέσο υπολογισμού της μετρικής αυτής είναι η **περιπλοκότητα**(perplexity) ενός μεγάλου γλωσσικού μοντέλου σε επίπεδο συμβόλων. Αυτή η μέθοδος έχει εφαρμοστεί σε διάφορες εργασίες NLP [87] [36] [92] και απαιτεί ένα γλωσσικό μοντέλο M_D εκπαιδευμένο σε ένα μεγάλο σύνολο δεδομένων D και τον υπολογισμό της μέσης περιπλοκότητας για μια δεδομένη ακολουθία κειμένου $x = x_1, x_2, ..., x_N$ ως εξής:

$$PPL(x) = exp\left\{\frac{1}{T}\sum_{t=1}^{T} \log p_{\mathcal{M}_{\mathcal{D}}}(x_t|x_{1:t-1})\right\}.$$

Υπάρχει βέβαια και η **ακριβής περιπλοκότητα** (fine perplexity) η οποία περιλαμβάνει τη λεπτομερή ρύθμιση (fine-tuning) του γλωσσικού μοντέλου M_D στη πραγματική κατανομή κειμένων L για τον εντοπισμό περιπτώσεων εκτός κατανομής με τη χρήση του τύπου PPL. Η μέθοδος αυτή έχει χρησιμοποιηθεί ως μέθοδος κάλυψης για να ανιχνευθούν και να μετρηθούν περιπτώσεις εκτός κατανομής [39].

1.2 Προτάσεις

1.2.1 Συνεισφορά

- Παράγουμε εξηγήσεις με αντιπαράδειγμα με τη χρήση πολλαπλών αντιφατικών συντακτών, που είναι πολύ σημαντικοί στην πρόσφατη βιβλιογραφία. Ως εκ τούτου, διερευνούμε τα πλεονεκτήματα και τις αδυναμίες που παρουσιάζει ο καθένας από αυτούς για μελλοντική έρευνα. Επικεντρώνουμε το ενδιαφέρον μας στην αξιολόγηση και την εξήγηση των παραγόμενων εξηγήσεων με αντιπαράδειγμα με αυτοματοποιημένες μετρήσεις και ποιοτικά κριτήρια.
- Εμπνευσμένοι από το [20] αξιοποιούμε μια νέα προσέγγιση για την παραγωγή αντιπαραδειγμάτων που είναι η μέθοδος της ανατροφοδότησης αντιπαραδειγμάτων στους συντάκτες για τη δημιουργία νέων αντιπαραδειγμάτων (Counterfactuals of counterfactuals), καθώς και τη νέα μετρική που ονομάζεται **ασυνέπεια**. Η εργασία μας επεκτείνει σημαντικά το προτεινόμενο πλαίσιο δημιουργίας εξηγήσεων με αντιπαράδειγμα, καθώς πειραματιζόμαστε με περισσότερα σενάρια για τους συντάκτες και τις μεθόδους δημιουργίας και το αξιοποιούμε για τη διεξαγωγή μίας βαθύτερης διαδικασίας αξιολόγησης.
- Παρουσιάζουμε έναν απλό αλλά ισχυρό τρόπο παρέμβασης στη σημασιολογία ενός αντιπαραδείγματος με τη χρήση ετικετών σε μέρη του λόγου (POS tags). Αξιολογούμε αυτή τη μέθοδο και εξάγουμε πολύτιμα συμπεράσματα σχετικά με την παραγωγή αντιπαραδειγμάτων με περιορισμούς.
- Τα συγκριτικά αποτελέσματα που εξάγονται από αυτή τη μελέτη βοηθούν στην εξήγηση των αποφάσεων ορισμένων μεθόδων και μοντέλων που χρησιμοποιούνται ευρέως στην παραγωγή εξηγήσεων με αντιπαράδειγμα.

1.2.2 Υλοποίηση του Προτεινόμενου Συστήματος Παραγωγής κι Αξιολόγησης Αντιπαραδειγμάτων

Βασικό υπόβαθρο

Για τα πειράματα μας χρησιμοποιήσαμε δύο **σύνολα δεδομένων** της αγγλική γλώσσας, το IMDb [51] που χρησιμοποιείται στην αναγνώριση συναισθήματος και το NewsGroups [41] που χρησιμοποιείται για την εργασία της θεματικής ταξινόμησης. Ακόμη, εκτελέσαμε πειράματα με **τρεις διαφορετικούς συντάκτες**, τους MiCE [75], Polyjuice [102] και TextFooler [35]. Ένας ακόμη σημαντικός παράγοντας στο σύστημα και τα πειράματα μας είναι οι **μέθοδοι μασκαρίσματος**. Χρησιμοποιήσαμε τυχαίο μασκάρισμα, τη μέθοδο μασκαρίσματος προσοχής, που χρησιμοποιεί τον μηχανισμό προσοχής του ταξινομητή για να βρει ποιες λέξεις επηρεάζουν περισσότερο την πρόβλεψη του ταξινομητή καθώς και την κατάταξη σπουδαιότητας λέξης.

Ακόμη τα **γλωσσικά μοντέλα** που χρησιμοποιούνται είναι ο Transformer T5 [70] για τον MiCE και ο Transformer GPT-2 [100] για τον Polyjuice. Ο συντάκτης TextFooler δε χρησιμοποιεί κάποιο μοντέλο αλλά ενσωματώσεις λέξεων σε συνδυασμό με διάφορους περιορισμούς. Επίσης, όσον αφορά τον **ταξινομητή**, αυτός είναι βασισμένος στο μοντέλο RoBERTa Large [46] και έχει εκπαιδευτεί ξεχωριστά στα δύο σύνολα δεδομένων που χρησιμοποιούμε. Τέλος, οι **μετρικές** που χρησιμοποιούμε είναι: ελαχιστότητα, ασυνέπεια, ποσοστό αντιστροφής, βασική περιπλοκότητα και ακριβής περιπλοκότητα

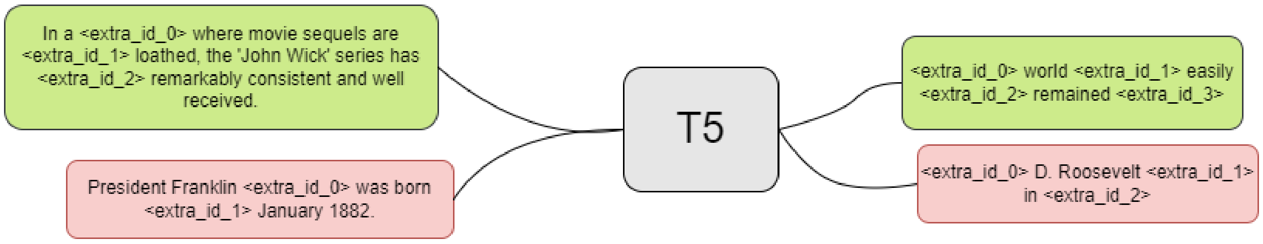

In a <extra_id_0> where movie sequels are <extra_id_1> loathed, the 'John Wick' series has <extra_id_2> remarkably consistent and well received.
President Franklin <extra_id_0> was born <extra_id_1> January 1882.
T5
<extra_id_0> world <extra_id_1> easily <extra_id_2> remained <extra_id_3>
<extra_id_0> D. Roosevelt <extra_id_1> in <extra_id_2>

Figure 1.2.1: Παραδείγματα των αποτελεσμάτων που παράγει το μοντέλο T5 για δύο προτάσεις με κάποιες "καλυμμένες" λέξεις. Η ένδειξη κάλυψης συμβολίζεται ως "<extra_id_i>" όπου το i είναι ο αύξων αριθμός κάθε καλυμμένης λέξης εκκινώντας από το μηδέν.

1.2.3 Η αρχιτεκτονική του συστήματος

Η αρχιτεκτονική που χρησιμοποιήθηκε για τα πειράματα αποτελείται από τρία στοιχεία: έναν συντάκτη αντιπαραδειγμάτων, έναν ταξινομητή και τη διαδικασία αξιολόγησης των αντιπαραδειγμάτων. Για τον συντάκτη χρησιμοποιούμε προ-εκπαιδευμένα μοντέλα για τους MiCE και Polyjuice ενώ ο TextFooler δεν απαιτεί κάποιο μοντέλο. Για τον ταξινομητή χρησιμοποιούμε προ-εκπαιδευμένους ταξινομητές με τη βιβλιοθήκη AllenNLP [22]. για τα σύνολα δεδομένων IMDb και NewsGroups. Εάν περισσότερες από ένα αντιπαραδείγματα ταξινομούνται στην κλάση αντίθεσης, χρησιμοποιείται ένα τρίτο βήμα για την αξιολόγηση των αντιπαραδειγμάτων που βασίζεται στη μετρική της ελαχιστότητας. Τέλος, η έξοδος μορφοποιείται κατάλληλα για να χρησιμοποιηθεί ως είσοδος για το επόμενο βήμα της διαδικασίας επανάληψης που χρησιμοποιούμε.

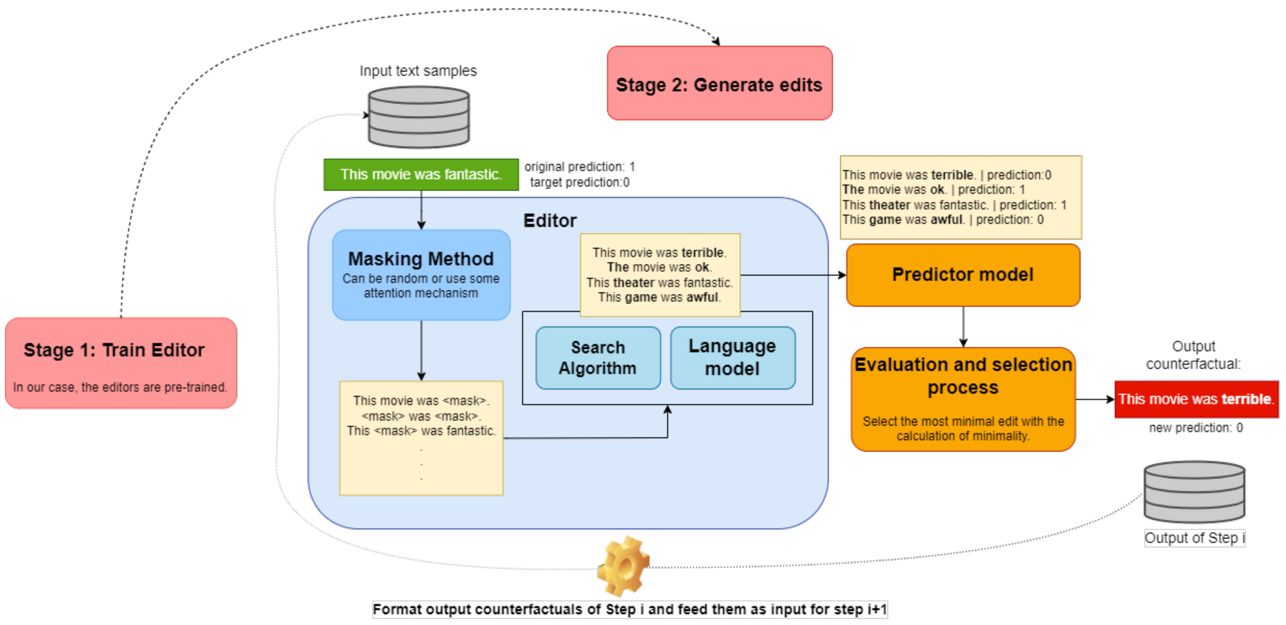

Stage 1: Train Editor
In our case, the editors are pre-trained.
Input text samples
Stage 2: Generate edits
This movie was fantastic.
original prediction: 1
target prediction:0
Editor
Masking Method
Can be random or use some attention mechanism
This movie was <mask>.
<mask> was <mask>.
This <mask> was fantastic.
This movie was terrible.
The movie was ok.
This theater was fantastic.
This game was awful.
Search Algorithm
Language model
This movie was terrible. | prediction:0
The movie was ok. | prediction: 1
This theater was fantastic. | prediction: 1
This game was awful. | prediction: 0
Predictor model
Evaluation and selection process
Select the most minimal edit with the calculation of minimality.
Output counterfactual:
This movie was terrible.
new prediction: 0
Output of Step i
Format output counterfactuals of Step i and feed them as input for step i+1

Figure 1.2.2: Μια επισκόπηση της αρχιτεκτονικής του συστήματος και του τρόπου με τον οποίο τον συνδυάζουμε με τον ταξινομητή για να διεξάγουμε τα πειράματά μας. Στο σχήμα, μπορούμε επίσης να δούμε πώς χρησιμοποιούμε την έξοδο του Editor για να δημιουργήσουμε μια νέα είσοδο στο σύστημά μας.

Στόχος του παρόντος συστήματος είναι η παραγωγή αντιπαραδειγμάτων που καταφέρνουν να ταξινομηθούν σε διαφορετική κλάση από την αρχική με τον ελάχιστο δυνατό τρόπο, δηλαδή με τους συντάκτης να επιφέρουν τις ελάχιστες αλλαγές στις προτάσεις κειμένου.

1.2.4 Ενσωμάτωση της Μεθόδου Στοχευμένων Μερών του Λόγου

Θέλοντας να πετύχουμε την παραγωγή όσο το δυνατόν πιο "ελάχιστων" αντιπαραδειγμάτων, προσπαθούμε να μειώσουμε τις υποψήφιες λέξεις για μασκάρισμα με τη μέθοδο στοχευμένων μερών του λόγου. Με τον τρόπο αυτό μασκάρουμε μόνο λέξεις που ανήκουν στο στοχευμένο μέρος του λόγου. Η πλειονότητα των πειραμάτων που πραγματοποιήθηκαν σε αυτή τη διατριβή δημιουργούν αντιπαραδείγματα με βάση αυτή τη μέθοδο. Ωστόσο, για να το πετύχουμε αυτό με επιτυχία πρέπει να ενσωματώσουμε και να υλοποιήσουμε τη μέθοδο κατάλληλα σε κάθε έναν από τους συντάκτες, με τρόπο που να μην επηρεάζει τα άλλα στοιχεία του συντάκτη. Έτσι, το πραγματοποιούμε στη φάση του μασκαρίσματος κάθε επεξεργαστή, προσθέτοντας με αυτόν τον τρόπο έναν περιορισμό για τις λέξεις που μπορεί να καλύψει το σύστημα μασκαρίσματος (masker). Η βιβλιοθήκη που χρησιμοποιούμε για την ανάκτηση των ετικετών POS κάθε λέξης είναι η Spacy [31], καθώς παρέχει στο χρήστη το μέρος του λόγου της κάθε λέξης με αβίαστο τρόπο. Η τυποποίηση του παραπάνω προβλήματος παρουσιάζεται ως εξής:

Έστω S η πρόταση εισόδου και $TARGET$ η στοχευόμενη ετικέτα μέρους του λόγου. Τότε, το $tok(S)$ αναπαριστά τη διαδικασία tokenization (αναγνώριση τμημάτων) που εφαρμόζεται σε μια πρόταση εισόδου S και παράγει μια λίστα από λέξεις $T = [t_1, t_2, ..., t_n]$, όπου n είναι ο αριθμός των λέξεων. Χρησιμοποιούμε μια συνάρτηση $pos(t)$ που επιστρέφει την ετικέτα μέρους του λόγου μίας λέξης t και δημιουργούμε μια νέα λίστα από λέξεις T' που επιλέγονται από την T, η οποία θα περιέχει μόνο τις λέξεις με την ετικέτα POS $TARGET$, το διατυπώνουμε ως εξής:

$$T' = [t_i | t_i \in T, pos(t_i) = TARGET]$$

Στη συνέχεια, $masker(T', S)$ είναι η συνάρτηση που αντικαθιστά όλα τα tokens της πρότασης εισόδου S που βρίσκονται στο T' με μία ετικέτα μάσκας για να δημιουργήσει μια τροποποιημένη μασκαρισμένη πρόταση M.

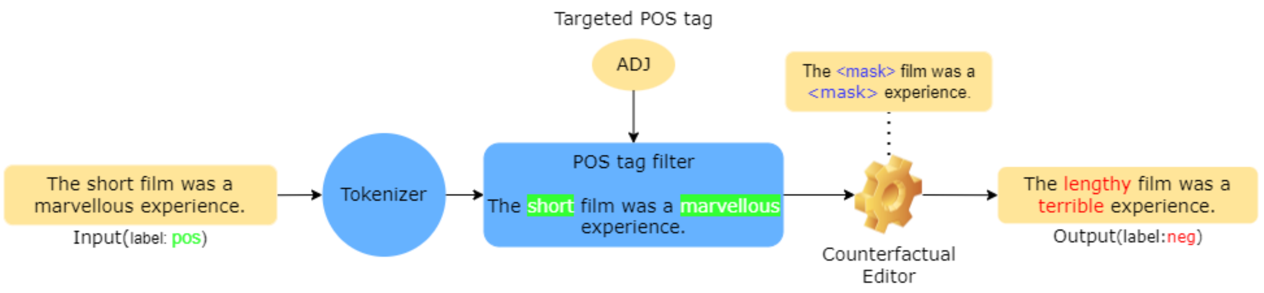

Targeted POS tag
ADJ
The <mask> film was a <mask> experience.
The short film was a marvellous experience.
Input(label: pos)
Tokenizer
POS tag filter
The short film was a marvellous experience.
Counterfactual Editor
The lengthy film was a terrible experience.
Output(label:neg)

Figure 1.2.3: Ένα παράδειγμα που εξηγεί τη διαδικασία δημιουργίας αντιπαραδειγμάτων με μια στοχευμένη ετικέτα μέρους του λόγου. Παρατηρούμε ότι η στοχευμένη ετικέτα POS στο παράδειγμα είναι "ADJ", δηλαδή επίθετο, και επομένως τα λεκτικά σημεία που είναι επίθετα στοχεύονται για τροποποίηση. Στην έξοδο του επεξεργαστή, παρατηρούμε ότι ο συντάκτης έχει παράξει αλλαγές μόνο για αυτά τα λεκτικά σημεία.

1.3 Πειραματικό μέρος

1.3.1 Επισκόπηση των Πειραμάτων

Αρχικά, θεωρούμε σημαντικό να δώσουμε ένα περίγραμμα των πειραμάτων που διεξάγονται, ώστε να αποσαφηνιστεί γύρω από ποιον άξονα εκτελούμε τα πειράματα και πώς συνδυάζουμε τις μεθοδολογίες που συζητήθηκαν στα προηγούμενα κεφάλαια. Συνολικά, χρησιμοποιούμε τρεις επεξεργαστές, το MiCE, το Polyjuice και το TextFooler. Συγκεκριμένα για το MiCE, χρησιμοποιούμε δύο τρόπους μασκαρίσματος, το τυχαίο μασκάρισμα και το μασκάρισμα προσοχής. Στο εξής, θα αναφερόμαστε στο MiCE με τη χρήση τυχαίας μασκαρίσματος ως *MiCERandom* και στο MiCE με τη χρήση κάλυψης προσοχής, απλά ως MiCE. Επιπλέον, διεξάγουμε όλα τα πειράματά μας και στα δύο σύνολα δεδομένων, IMDb και NewsGroups. Σε όλα τα πειράματα που πραγματοποιήθηκαν, υλοποιούμε την έννοια των "Counterfactuals of counterfactuals" για 10 βήματα. Τα 10 βήματα αυτά είναι 10 συνεχόμενες διαδικασίες ανατροφοδοσίας της εξόδου των συντακτών ως νέα είσοδο σε αυτούς. Επιπλέον, χρησιμοποιούμε και τους τρεις συντάκτες και το MiCERandom για να δημιουργήσουμε αντιφατικές επεξεργασίες με τη μέθοδο των στοχευμένων ετικετών μέρους του λόγου. Τέλος, εκτελούμε ορισμένα πειράματα για να επιθεωρήσουμε την επίδραση του αριθμού των δεσμών όταν χρησιμοποιούμε αναζήτηση δέσμης στο MiCE.

Για να συνοψίσουμε όλα τα παραπάνω, παρουσιάζουμε τον παρακάτω πίνακα πειραμάτων, ο οποίος αποτελεί έναν οργανωμένο τρόπο περιγραφής όλων των πειραμάτων μας.

Editors	**Experiment Types**				
	Out-of-the-box	**ADJ**	**NOUN**	**VERB**	**Beam - search***
MiCE	✓	✓	✓	✓	✓
MiCERandom	✓	✓	✓	✓	–
Polyjuice	✓	✓	✓	✓	–
TextFooler	✓	✓	✓	✓	–

* Beam-search experiments are conducted with multiple numbers of beams, namely 1,5,15,30,60 and 120

Table 1.2: Ένα περίγραμμα των πειραμάτων μας. Out-of-the-box σημαίνει τη χρήση των συντακτών χωρίς καμία παρέμβαση από εμάς, ADJ, NOUN, και VERB είναι οι στοχευμένες ετικέτες μέρους του λόγου και Beam-Search είναι μια προσπάθεια με πειράματα στο MiCE. Όλα τα πειράματα διεξάγονται τόσο για το σύνολο δεδομένων IMDb όσο και για το NewsGroups.

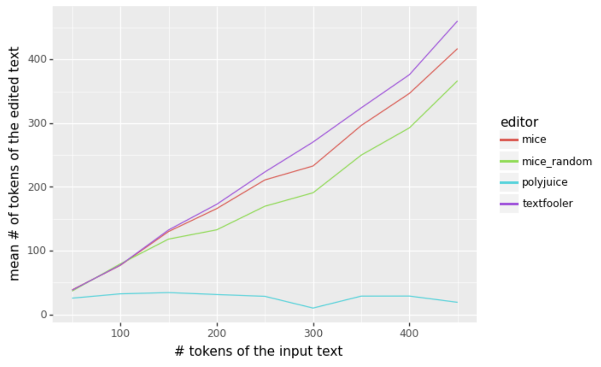
1.3.2 Ποσοτικά αποτελέσματα

Σε αυτό το κομμάτι της εργασίας μας, αξιολογούμε τα ποσοτικά αποτελέσματα χρησιμοποιώντας τις μετρικές που περιγράψαμε προηγουμένως και τα παρουσιάζουμε σε μορφές πινάκων και διαγραμμάτων για να μας βοηθήσουν να κατανοήσουμε τις αποφάσεις που ελήφθησαν από τις διάφορες μεθόδους.

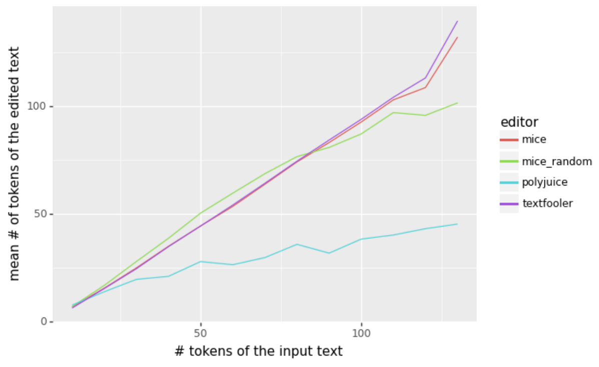
Ελαχιστότητα

Η κύρια παρατήρηση της σύγκρισης μεταξύ των συντακτών είναι ότι ο TextFooler παράγει αντιπαραδείγματα με τις πιο ελάχιστες δυνατές αλλαγές και στα δύο σύνολα δεδομένων λόγω της ντετερμινιστικής προσέγγισης και της χρήσης πολλαπλών περιορισμών. Επιπλέον, ο MiCE που χρησιμοποιεί μασκάρισμα προσοχής (gradient masking), το οποίο βρίσκει και μασκάρει τις πιο επιδραστικές λέξεις για τον ταξινομητή, αντιστρέφει την αρχική κλάση της πρότασης με λιγότερες τροποποιήσεις σε σχέση με τους συντάκτες που χρησιμοποιούν τυχαίο μασκάρισμα όπως είναι οι MiCERandom και Polyjuice. Καθώς αυξάνονται τα βήματα ανατροφοδότησης, η ελαχιστότητα των αντιπαραδειγμάτων μειώνεται, υποδεικνύοντας ότι οι συντάκτες τείνουν να εκτελούν λιγότερες επεξεργασίες μετά από κάθε βήμα ανατροφοδότησης.

Ακόμη όσον αφορά τον Polyjuice, παρατηρούμε ότι γεννά αντιπαραδείγματα με σημαντικά μεγαλύτερες τιμές ελαχιστότητας. Το τυχαίο μασκάρισμα που χρησιμοποιεί για να βρει πού πρέπει να γίνουν αλλαγές σε συνδυασμό με μεγάλου μήκους προτάσεις, οδηγεί σε εκθετικά μεγαλύτερο χώρο αναζήτησης. Σε πειράματα, όπως και στο [20], διαπιστώθηκε ότι διαγράφει πάνω από το 70% της αρχικής εισόδου στο σύνολο δεδομένων IMDb και το 50% του αρχικού κειμένου στο σύνολο δεδομένων NewsGroups στα δύο πρώτα βήματα επεξεργασίας. Αυτή η συμπεριφορά οφείλεται στη διαδικασία αξιολόγησής του συντάκτη, η οποία επιλέγει επεξεργασίες με διαφορετικές ετικέτες από το αρχικό κείμενο, και στο εσωτερικό του μοντέλο, το GPT-2, το οποίο δεν είναι καθόλου περιορισμένο σε σύγκριση με το T5 του MiCE.



(a) Μέσος αριθμός λέξεων του αντιπαραδείγματος σε σύγκριση με τον μέσο αριθμό λέξεων της εισόδου στο IMDb.



(b) Μέσος αριθμός λέξεων του αντιπαραδείγματος σε σύγκριση με τον μέσο αριθμό λέξεων της εισόδου στο NewsGroups.

Figure 1.3.1: Μέσος αριθμός λέξεων του αντιπαραδείγματος σε σύγκριση με τον μέσο αριθμό λέξεων της εισόδου.

Όσον αφορά τις αλλαγές με συγκεκριμένο μέρος του λόγου, όλοι οι συντάκτες δημιουργούν αντιπαραδείγματα με σημαντικά πιο ελάχιστες επεξεργασίες και στα δύο σύνολα δεδομένων σε σύγκριση με όταν λειτουργούν χωρίς κάποιο περιορισμό. Αυτό οφείλεται στο γεγονός ότι ο περιορισμός μας για τα μέρη του λόγου προκαλεί την απόκρυψη λιγότερου κειμένου και την τροποποίηση περισσότερου κειμένου. Στο IMDb, τα επίθετα παράγουν αντιπαραδείγματα με πιο ελάχιστο τρόπο από ότι με ρήματα ή ουσιαστικά, ενώ στο NewsGroups, τα ουσιαστικά συμβάλλουν σε πιο ελάχιστες επεξεργασίες από ότι τα ρήματα και τα επίθετα. Αυτή η συμπεριφορά, οφείλεται κυρίως στην εργασία για την οποία χρησιμοποιείται το κάθε σύνολο δεδομένων αλλά και στην κατανομή των μερών του λόγου στο κείμενο. Τέλος, τα συμπεράσματα που εξάγαμε για τη σύγκριση ανάμεσα στους συντάκτες φαίνεται να επικυρώνονται και με τη μέθοδο στοχευμένων μερών του λόγου.

Ασυνέπεια

Στον πίνακα 1.3, βλέπουμε ότι υπάρχουν αξιοσημείωτες διαφορές μεταξύ των συντακτών, οι οποίες μπορούν να αποδοθούν στις μεθόδους και τα μοντέλα που χρησιμοποιεί κάθε συντάκτης. Ο TextFooler είναι ο πιο συνεπής από τους επεξεργαστές, έχοντας πολύ χαμηλές τιμές ασυνέπειας. Οι πιο υψηλές τιμές στους άλλους συντάκτες οφείλονται στη χρήση γλωσσικών μοντέλων, τα οποία έχει αποδειχθεί ότι είναι πιο ευαίσθητα στις διαφοροποιήσεις στην είσοδό τους [59]. Το σύνολο δεδομένων IMDd παρουσιάζει υψηλή τιμή ασυνέπειας για το

Polyjuice στο πρώτο βήμα των επεξεργασιών, αλλά το σύνολο δεδομένων NewsGroups δείχνει ότι το Polyjuice είναι συνεπές για μικρότερες εισόδους.

Στο σχήμα 1.3.2 η υψηλή τιμή ασυνέπειας για το MiCE σε ζυγό αριθμό βημάτων δείχνει ότι ο συντάκτης δυσκολεύεται να μετακινηθεί από την αρχική κλάση μιας εισόδου σε μια διαφορετική. Από την άλλη πλευρά, όπως επισημαίνεται και στο [20] η επιστροφή στην αρχική κλάση είναι ευκολότερη για τον συντάκτη, καθώς συνήθως τα τμήματα της αρχικής εισόδου που συμβάλλουν στην πρόβλεψη της αρχικής κλάσης παραμένουν ακόμη στην πρόταση κειμένου, με αποτέλεσμα να απαιτούνται λιγότερες επεξεργασίες για την επιστροφή στην αρχική ετικέτα.

Σε γενικές γραμμές, οι συντάκτες γίνονται πιο συνεπείς όσο αυξάνονται τα βήματα ανατροφοδότησης. Μετά τη δημιουργία επεξεργασιών στα πρώτο βήματα, φαίνεται ότι απαιτούνται λιγότερες τροποποιήσεις στα επόμενα βήματα, καθώς υπάρχουν πολλαπλές περιπτώσεις όπου υπάρχουν "υπολείμματα κειμένου" από προηγούμενες επεξεργασίες που συμβάλλουν στην πρόβλεψη του ταξινομητή.

Table 1.3: Τα αποτελέσματα για τη μετρική της ασυνέπειας.

	IMDb			
	MiCE	MiCERandom	Polyjuice	TextFooler
inc@1↑	0.86	2.42	6.21	**0.01**
inc@2↑	5.95	5.81	4.65	**0.33**
inc@3↑	4.65	6.37	3.98	**0.36**
inc@5↑	4.87	7.58	2.9	**0.47**
inc@9↑	4.73	8.11	2.22	**0.49**
	NewsGroups			
inc@1↑	1.23	2.66	0.53	**0.04**
inc@2↑	2.53	4.3	1.27	**0.36**
inc@3↑	2.44	4.37	1.28	**0.27**
inc@5↑	2.46	4.62	1.24	**0.27**
inc@9↑	2.42	4.94	1.2	**0.25**

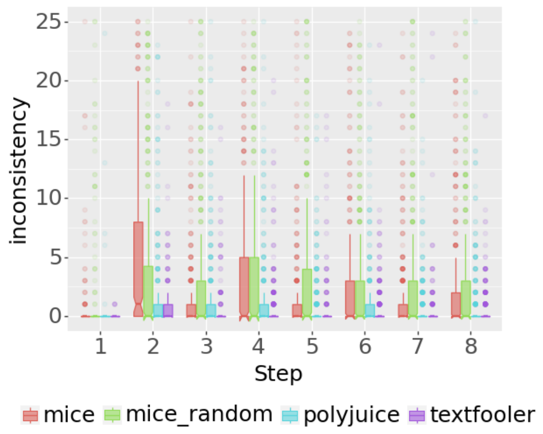


Figure 1.3.2: Διάγραμμα όπου απεικονίζουμε την ασυνέπεια για 10 βήματα στο σύνολο δεδομένων IMDb.

Τέλος, με τη μέθοδο στοχευμένων μερών του λόγου παρατηρήσαμε ότι οι συντάκτες γίνονται πιο συνεπείς από ότι είναι όταν δεν περιορίζονται σε μια συγκεκριμένη ετικέτα POS. Οι πολλαπλές τιμές ασυνέπειας κοντά στο 0 μας δείχνουν ότι οι συντάκτες με τη μέθοδο αυτή, διαφοροποιούν με τέτοιο τρόπο την είσοδο έτσι ώστε να είναι πράγματι η ελάχιστη δυνατή. Παράδειγμα αυτού ο TextFooler, όπου το ποσοστό των προτάσεων με τιμή ασυνέπειας 0 ξεπερνά το 90% για τα μονά βήματα και το 60% για τα ζυγά βήματα.

Γενικά, η μετρική της ασυνέπειας της ελαχιστότητας βοηθά στην αποκάλυψη χαρακτηριστικών του συντάκτη που δεν μπορεί να αποκαλύψει η ελαχιστότητα και απεικονίζει τις αδυναμίες που έχουν οι συντάκτες όσον αφορά την ελαχιστότητα. Επίσης δείχνουμε ότι με μεθόδους όπως αυτή των στοχευμένων μερών του λόγου δείχνουμε ότι είναι δυνατό να "καταπολεμήσουμε" αποτελεσματικά ορισμένες από αυτές τις αδυναμίες και να έχουμε πιο συνεπείς συντάκτες.

Ποσοστό αντιστροφής

Το ποσοστό αντιστροφής, σε συνδυασμό με τη μέθοδο της παραγωγής αντιπαραδειγμάτων από αντιπαραδείγματα, μας βοηθάει να διαφωτίσουμε παραλείψεις και επαναλαμβανόμενα μοτίβα στους συντάκτες. Στον πίνακα 1.4 παρουσιάζουμε τα αποτελέσματά σας. Κάνοντας πειράματα στους συντάκτες για πάνω από ένα βήματα βλέπουμε ότι η απόδοση των συντακτών είναι διαφορετική και συγκεκριμένα επέρχεται σημαντική μείωσή της. Ωστόσο, παρατηρούμε ότι η μείωση αυτή ξεκινάει από το βήμα 2 που αντιστοιχεί σε μετάβαση προς την αρχική κλάση. Αυτό επιβεβαιώνει τη συμπεριφορά του MiCE που δυσκολεύεται να γυρίσει στην αρχική του κλάση. Στη σύγκριση των συντακτών παρατηρούμε σταδιακή μείωση για τους MiCE και MiCERandom, σε αντίθεση με Polyjuice και TextFooler όπου βλέπουμε σταθερή αύξηση. Η αυξημένη επίδοση του Polyjuice ίσως οφείλεται στο μειωμένο μέγεθος των πρώτων βημάτων που οδηγεί σε μικρότερο χώρο αναζήτησης για το μοντέλο του συντάκτη.

Για να εξηγήσουμε τους λόγους για τα παραπάνω διερευνούμε το Σχήμα 1.3.3 που μας δείχνει την επιρροή στο ποσοστό σωστών προβλέψεων από τον ταξινομητή. Για το παρακάτω διάγραμμα αλλά και στα πειράματα θεωρούμε ότι η αντιστροφή επιτυγχάνεται όταν η πρόβλεψη του ταξινομητή είναι άνω του 0.5. Στο διάγραμμα παρατηρούμε επίσης διαφορές σε μονά και ζυγά βήματα όπως είδαμε και στην ανάλυση για τη μετρική της ασυνέπειας. Επίσης, παρατηρούμε πως η πιθανότητα να βρεθούμε στην κλάση-στόχο σταδιακά μειώνεται στον MiCERandom καθώς οι αλλαγές που προκαλεί ο συντάκτης φαίνεται να προκαλούν θόρυβο που σταδιακά οδηγεί σε περισσότερες μη επιθυμητές προβλέψεις.

Table 1.4: Τα αποτελέσματα τπυ ποσοστού αντιστροφής με τους 4 συντάκτες.

	MiCE	MiCERandom	Polyjuice	TextFooler
	IMDb			
Flip Rate@1↑	**1.0**	0.9953	0.8747	0.6241
Flip Rate@2↑	0.8422	0.8419	**0.9107**	0.6984
Flip Rate@3↑	0.891	0.7163	**0.9392**	0.7193
Flip Rate@5↑	0.8677	0.6279	**0.9592**	0.7517
Flip Rate@9↑	0.8561	0.5674	**0.9668**	0.7865
	NewsGroups			
Flip Rate@1↑	0.89	0.79	0.726	**0.941**
Flip Rate@2↑	0.9188	0.715	0.9131	**1.0**
Flip Rate@3↑	0.8806	0.6395	0.9074	**1.0**
Flip Rate@5↑	0.8574	0.5972	0.9237	**1.0**
Flip Rate@9↑	0.8322	0.5444	0.9659	**1.0**

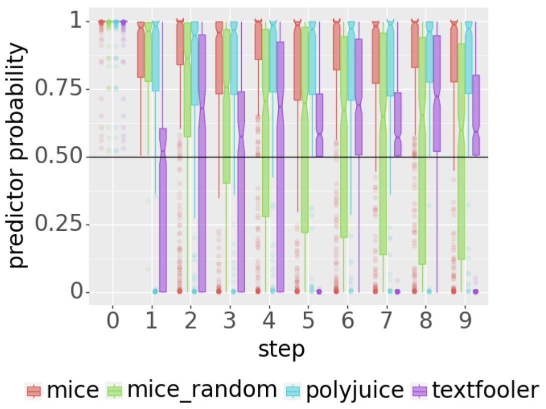


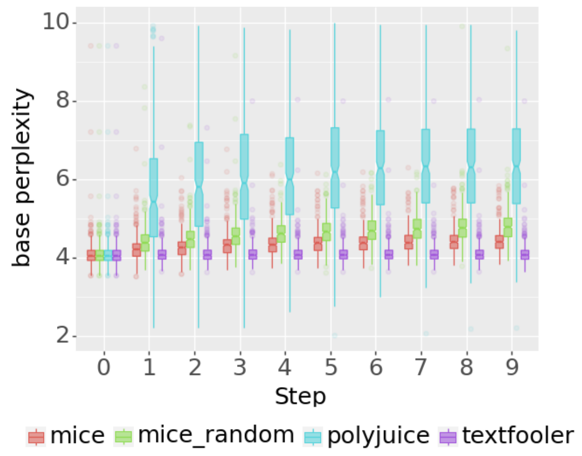
Figure 1.3.3: Πιθανότητα πρόβλεψης της κλάσης στόχου στο σύνολο δεδομένων IMDb

Στα αποτελέσματα του ποσοστού αντιστροφής όταν χρησιμοποιούμε τη μέθοδό μας για στοχευμένα μέρη του λόγου, βλέπουμε μείωση του ποσοστού αντιστροφής των πρώτων βημάτων λόγω του περιορισμού που επιφέρουμε στους συντάκτες. Είναι σημαντικό να σημειωθεί ποια ετικέτα μέρους του λόγου έχει καλύτερες επιδόσεις όσον αφορά το ποσοστό αντιστροφής σε κάθε σύνολο δεδομένων, με τα επίθετα να είναι τα επικρατέστερα στο IMDb και τα ουσιαστικά να είναι η βέλτιστη μέθοδος στο NewsGroups. Αυτή η διαφορά στην απόδοση φαίνεται να σχετίζεται με την εργασία, καθώς το IMDb χρησιμοποιείται για ανάλυση συναισθήματος και τα επίθετα επηρεάζουν περισσότερο το συναίσθημα από ότι τα ρήματα και τα ουσιαστικά που είναι πιο ουδέτερα. Το NewsGroups από την άλλη χρησιμοποιείται για ταξινόμηση θεμάτων όπου τα ουσιαστικά είναι ζωτικής σημασίας.

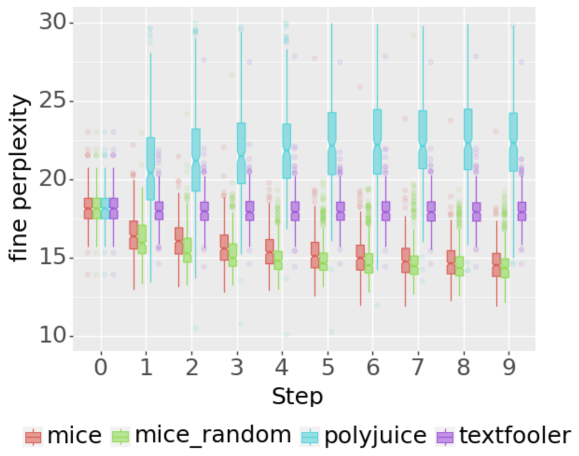
Περιπλοκότητα

Τα αποτελέσματα για τη **βασική περιπλοκότητα**, τη μία από τις δύο μετρικές που αξιολογούν την ευχέρεια λόγου των αντιπαραδειγμάτων, έδειξαν ότι το TextFooler παρήγαγε το πιο εύγλωττο κείμενο, ενώ οι MiCE, MiCERandom και Polyjuice είχαν αυξημένες τιμές περιπλοκότητας καθώς αυξάνονταν τα βήματα. Το Σχήμα 1.3.4a δείχνει τη συνοχή του TextFooler και την επιδείνωση των MiCE, MiCERandom και Polyjuice με αυτή τη σειρά. Η επίδοση του TextFooler δε μας εκπλήσσει αφού αποδεικνύεται και μέσα από τα αποτελέσματα αυτής της μετρικής ότι όσο πιο αυστηρούς κανόνες χρησιμοποιούμε για τη γέννηση αντιφατικού κειμένου τόσο λιγότερα παρεκκλίνει από μια συγκεκριμένη κατανομή. Ακόμη, η σύγκριση μεταξύ των δύο συντακτών που χρησιμοποιούν τυχαίο μασκάρισμα, MiCERandom και Polyjuice, έδειξε ότι ο πρώτος έχει καλύτερη απόδοση από τον δεύτερο. Αυτό οφείλεται στα εσωτερικά γλωσσικά μοντέλα που χρησιμοποιούν οι συντάκτες, αφού το GPT-2 του Polyjuice παράγει κείμενο με θεωρητικά "απρόβλεπτο" τρόπο σε σχέση με το T5 που είναι ένα πολύ πιο ελεγχόμενο μοντέλο.

Η **ακριβής περιπλοκότητα** είναι μία μετρική που δείχνει πόσο διαφορετικό είναι το παραγόμενο κείμενο σε σύγκριση με το σύνολο δεδομένων που χρησιμοποιείται για τη ρύθμιση του γλωσσικού μοντέλου. Τα αποτελέσματα του Σχήματος 1.3.4 δείχνουν ότι ο TextFooler είναι ο πιο σταθερός συντάκτης, ενώ οι MiCE και MiCERandom παράγουν πιο σωστό γλωσσικά κείμενο όταν αξιολογούνται με την ακριβή περιπλοκότητα. Ωστόσο, αυτά τα αποτελέσματα είναι φαινομενικά αφού οι συντάκτες MiCE και MiCERandom εκπαιδεύονται αποκλειστικά στο σύνολο δεδομένων IMDb, κι έτσι οδηγούνται όπως φαίνεται σε συμπεριφορά υπερπροσαρμογής (overfitting) [20]. Το Polyjuice αντιθέτως, το οποίο εκπαιδεύεται σε πολλαπλά σύνολα δεδομένων [102], παράγει πιο ποικιλόμορφο κείμενο από το MiCE και για αυτό φαίνεται να έχει μεγαλύτερες τιμές στο διάγραμμα.



(a) Βασική Περιπλοκότητα στο σύνολο δεδομένων IMDb.



(b) Ακριβής Περιπλοκότητα στο σύνολο δεδομένων IMDb.

Figure 1.3.4: Περιπλοκότητα στο IMDb.

Όσον αφορά την παραγωγή αντιπαραδειγμάτων με στοχευμένα μέρη του λόγου βλέπουμε ότι αυτή η μέθοδος οδηγεί σε πιο εύγλωττους συντάκτες. Το TextFooler είναι ο συντάκτης που παράγει τις πιο εύγλωττες προτάσεις με τη χρήση μερών του λόγου, ενώ οι MiCE και MiCERandom φαίνεται να έχουν καλές επιδόσεις και να γίνονται πιο εύγλωττοι και πιο συνεπείς από τις αρχικές τους εκδόσεις. Αυτές οι παρατηρήσεις δείχνουν να συμπίπτουν με τις χαμηλές τιμές ασυνέπειας που παρουσιάζουν αυτοί οι συντάκτες. Επίσης, η χρήση στοχευμένων μερών του λόγου βοηθά στον περιορισμό της συμπεριφοράς υπερπροσαρμογής στους συντάκτες MiCE για τα δεδομένα του IMDb, καθώς οι τιμές της ακριβής περιπλοκότητας παραμένουν γενικά υψηλότερες από αυτές των αρχικών εκδόσεων.

Αναζήτηση δέσμης στο MiCE

Το MiCE είναι ένα μοντέλο που χρησιμοποιεί πολυωνυμική δειγματοληψία ως μέθοδο αναζήτησης, η οποία επιλέγει τυχαία το επόμενο σύμβολο με βάση την κατανομή πιθανότητας σε ολόκληρο το λεξιλόγιο που δίνει το μοντέλο. Στα πειράματα μας, **η αναζήτηση δέσμης με 120 δέσμες έχει καλύτερη επίδοση** από την πολυωνυμική δειγματοληψία στο σύνολο δεδομένων IMDb όσον αφορά την **ελαχιστότητα και την ασυνέπεια**. Στο σύνολο δεδομένων NewsGroups, η αναζήτηση δέσμης φτάνει σε απόδοση αλλά δεν ξεπερνά την πολυωνυμική δειγματοληψία. Αυτό ίσως οφείλεται στο μικρότερος μέγεθος των προτάσεων στο NewsGroups που ευνοεί την αναζήτηση αντικατάστασης λέξη κατά λέξη σε αντίθεση με την αναζήτηση δέσμης που εξερευνά και μεγαλύτερες ακολουθίες λέξεων. Στο Σχήμα 1.5, είναι ενδιαφέρον να παρατηρήσουμε πώς οι τιμές της ελαχιστότητας και της ασυνέπειας μειώνονται καθώς αυξάνεται ο αριθμός των δεσμών, καθώς ο αλγόριθμος εξερευνά μεγαλύτερο χώρο αναζήτησης και έτσι έχουμε περισσότερες υποψήφιες αντικαταστάσεις που μπορούν να οδηγήσουν πιο εύκολα σε ελάχιστες αλλαγές.

	IMDb						
	MiCE	**Greedy**	**5 beams**	**15 beams**	**30 beams**	**60 beams**	**120 beams**
inc@1↑	0.86	3.5	3.59	1.91	3.35	2.27	**0.57**
inc@2↑	5.95	13.73	11.75	9.25	11.85	8.78	**3.29**
inc@3↑	4.65	11.62	10.09	8.2	9.91	7.5	**3.01**
inc@5↑	4.87	11.38	9.65	8.3	9.56	7.03	**3.04**
inc@9↑	4.73	10.51	9.0	8.19	8.34	6.43	**2.9**
	NewsGroups						
inc@1↑	**1.23**	4.32	3.64	3.04	2.74	2.36	2.13
inc@2↑	**2.53**	5.19	4.41	3.97	3.69	3.52	2.88
inc@3↑	**2.44**	5.18	4.31	3.83	3.43	3.21	2.67
inc@5↑	**2.46**	5.29	4.24	3.69	3.23	3.09	2.48
inc@9↑	2.42	5.33	3.99	3.38	2.98	2.9	**2.31**

Table 1.5: Τα αποτελέσματα για τη μετρική της ασυνέπειας για όλες τις διαφορετικές παραλλαγές του MiCE.

Επίσης, όσον αφορά το **ποσοστό αντιστροφής**. τα πειράματα έδειξαν ότι ο MiCE με πολυωνυμική δειγματοληψία ήταν η καλύτερη επιλογή για το πρώτο βήμα στο IMDb, αλλά η παραλλαγή του MiCE με χρήση 120 δεσμών αποδίδει καλύτερα στα μεταγενέστερα βήματα. Η άπληστη αναζήτηση (greedy search) πέτυχε πολύ υψηλά ποσοστά αντιστροφής και στα δύο σύνολα δεδομένων, γεγονός που υποδηλώνει ότι η επιλογή λέξεων με την υψηλότερη πιθανότητα οδηγεί σε πιο "επιθετικές" επεξεργασίες στο κείμενο που τελικά αντιστρέφουν το αποτέλεσμα της πρόβλεψης. Επιπλέον, το ποσοστό ανατροπής στο IMDb μειώθηκε καθώς αυξανόταν ο αριθμός των δοκών.

Σχετικά με την ποιότητα του λόγου στα αντιπαραδείγματα με αναζήτηση δέσμης βλέπουμε πως όσο υψηλότερος είναι ο αριθμός των δεσμών τόσο πιο ποικιλόμορφο κείμενο παράγεται. Ακόμη με την πάροδο των βημάτων της διαδικασίας παραγωγής αντιπαραδειγμάτων, η ευχέρεια λόγου (fluency) μειώνεται σε ένα μικρό βαθμό. Όσον αφορά την ακριβή περιπλοκότητα και το φαινόμενο υπερπροσαρμογής που παρατηρήσαμε στο IMDb σε προηγούμενα πειράματα, παρατηρήσαμε πως το MiCE με πολυωνυμική δειγματοληψία παρουσιάζει σε χαμηλότερο βαθμό αυτό το φαινόμενο σε σχέση με τις παραλλαγές του συντάκτη με αναζήτηση δέσμης.

Τέλος, είναι σημαντικό να σημειωθεί ότι για να πειραματιστούμε με τόσο μεγάλο αριθμό δεσμών όπως οι 120, πρέπει να συμβιβαστούμε με κάποιους περιορισμούς, όπως η διάρκεια του πειράματος και οι υπολογιστικές απαιτήσεις.

1.3.3 Ποιοτικά αποτελέσματα

Σε αυτό το μέρος της διατριβής παρουσιάζονται πολλαπλά ποιοτικά αποτελέσματα που βοηθούν στην παρατήρηση συμπεριφορών, μοτίβων και ιδιαιτεροτήτων στο παραγόμενο κείμενο κάθε συντάκτη. Λόγω της φύσης των αποτελεσμάτων και τους πολλαπλούς συντάκτες, προσπαθούμε να παρουσιάσουμε τα αποτελέσματα που παρουσιάζουν το μεγαλύτερο ενδιαφέρον για τα συμπεράσματα που εξάγονται, αλλά και να εξηγήσουμε τα χαρακτηριστικά των συντακτών με παραδείγματα που δεν είναι μακροσκελή, προκειμένου να επικεντρωθούμε στις εξηγήσεις που εξάγουμε για τους συντάκτες και όχι στην ιδιαιτερότητα κάθε παραδείγματος. Στους πίνακες που θα παρουσιάσουμε οι λέξεις που τροποποιούνται σε κάθε επεξεργασία παρουσιάζονται με έντονη γραφή.

Τυχαίο Μασκάρισμα και Μασκάρισμα Προσοχής

Στα παραδείγματα των Πινάκων 1.6 και 1.7, μπορούμε να δούμε την κύρια διαφορά μεταξύ των δύο συντακτών και την "ισχυρή" επίδραση που μπορεί να έχει στα αντιπαραδείγματα μας μετά από πολλαπλά βήματα. Συγκεκριμένα, το MiCE το οποίο χρησιμοποιεί μάσκα προσοχής, ουσιαστικά "γνωρίζει" ποιες λέξεις είναι πιο σημαντικές για τον συντάκτη και αντικαθιστά όσο το δυνατόν λιγότερες από αυτές για να επιτύχει την αντιστροφή. Με αυτόν τον τρόπο, βλέπουμε γιατί το MiCE είναι πολύ πιο συνεπές από τον MiCERandom, ο οποίος προσπαθεί να αντιστρέψει το αποτέλεσμα του ταξινομητή τροποποιώντας τυχαίες λέξεις. Με αυτό τον τρόπο, το MiCERandom όχι μόνο παράγει επεξεργασίες που είναι λιγότερο ελάχιστες αλλά και λιγότερο εύγλωττες, καθώς μπορούμε να παρατηρήσουμε ότι το κείμενο παραμορφώνεται όλο και περισσότερο

Ένα άλλο φαινόμενο που παρατηρούμε, το οποίο έχει παρατηρηθεί και από τους Φιλανδριανός και λοιποί [20] είναι η εισαγωγή των "λανθασμένων λευκών διαστημάτων", η οποία πολλαπλασιάζεται μετά από ορισμένα βήματα. Αυτό φαίνεται να προκαλείται από τους εσωτερικούς μηχανισμούς ή το μοντέλο του MiCE, καθώς δε συμβαίνει με τον Polyjuice ή τον TextFooler. Επίσης, μια άλλη κοινή ελαττωματική συμπεριφορά που παρατηρούμε είναι η διάσπαση των tokens και των "ψευδαισθήσεων" (hallucinations) όπως στα βήματα 9 και 10 των επεξεργασιών με το MiCERandom και τη λέξη "Unbelievably".

Μετά και από τα 10 βήματα, παρατηρούμε πως τα αντιπαραδείγματα του MiCERandom έχουν μετατοπιστεί σε μεγάλο βαθμό από την αρχική είσοδο, ενώ ο MiCE έχει παραμείνει πολύ πιο συνεπής, παρά την επανάληψη της λέξης "Script". Αυτό συμβαίνει επειδή ο προβλεπτικός μηχανισμός κατατάσσει σταθερά την τελευταία λέξη της πρότασης ως το πιο επιδραστικό, οδηγώντας τον MiCE να διαφοροποιεί την πρόταση κυρίως αντικαθιστώντας μόνο αυτή τη λέξη.

Table 1.6: Τα παραγόμενα αντιπαραδείγματα για 10 βήματα στον MiCE σε ένα δείγμα από το IMDb.

Step	**MiCE**
1	What a script, what a story, what a **mess**!
2	What a script, what a story, what a **filmmaker**!
3	What a script, what a story, what a **disappointment**!
4	What a script, what a story, what a **thriller**!
5	What a script, what a **script**, what a **suck**!
6	What a script, what a script, what a **director**!
7	What a script, what a script, what a **mess**!
8	What a script, what a script, what a **script**!
9	What a script, what a script, what a **disaster**!
10	What a script, what a script, what a **script**!

Table 1.7: Τα παραγόμενα αντιπαραδείγματα για 10 βήματα στον MiCERandom σε ένα δείγμα από το IMDb.

Step	**MiCERandom**
1	What **a** script**, what a** story, **what** a mess!
2	What **great** script **&** story, **but** a mess!**!!!!!!**
3	What **fantastic** script & story, but **GREAT FUN**!!!!!!!**!!**
4	What fantastic script & story but **NO** FUN!!!!!!!!!
5	What fantastic script **. Unbelievable** story but **amazing story**!!!!!!!!!
6	What **unbelievable** script . Unbelievable **unbelievable** story!!!!!!!!!
7	What **incredible** script . Unbelievable unbelievable!!!!!!!!!**!!!!!!**
8	What incredible script . Unbelievable unbelievable **acting.**!!!!!!!!!!!!!!!
9	What incredible script . **Un believably** unbelievable acting.!!!!!!!!!!!!!!!
10	What incredible script . Un **believ** believably unbelievable acting.!!!!!!!!!!!!!!!

Σύγκριση των 4 συντακτών

Στους Πίνακες 1.8 και 1.9, παρουσιάζουμε ένα παράδειγμα εισόδου από το σύνολο δεδομένων IMDb και πώς τα αντιπαραδείγματα διαφέρουν από συντάκτη σε συντάκτη. Πρώτον, παρατηρούμε ότι οι MiCERandom και Polyjuice δεν παραμένουν συνεπείς με τις λέξεις που επιλέγουν να τροποποιήσουν, δείχνοντας την τυχαιότητά τους. Ο Polyjuice φαίνεται να κάνει την πιο επιθετική αλλαγή μεταξύ των άλλων στο βήμα 3, όπου τροποποιείται ολόκληρο το κείμενο δημιουργώντας μία εντελώς διαφορετική πρόταση. Αυτή η συμπεριφορά του Polyjuice είναι συνηθισμένη και επικυρώνει τις αυξημένες τιμές ασυνέπειας καθώς παράγει πιο ποικιλόμορφο κείμενο από τους άλλους συντάκτες. Από την άλλη πλευρά, ο TextFooler δεν επηρεάζει καθόλου τη δομή της πρότασης καθώς επιλέγει πολύ "αυστηρές" τροποποιήσεις. Τα αντιπαραδείγματα του TextFooler στον πίνακα 7.16 διαθέτουν μια ασυνέπεια τιμής 0 σε όλα τα βήματα και επικυρώνουν τη συνέπεια του συντάκτη. Για το MiCE, το μοτίβο προστιθέμενων κενών διαστημάτων, καθώς και η διάσπαση των λέξεων λαμβάνουν χώρα και πάλι στα τελευταία βήματα αυτής της επεξεργασίας.

Table 1.8: Τα αντιπαραδείγματα από τους MiCE και MiCERandom για 10 βήματα σε ένα δείγμα από το IMDb.

Step	**MiCE**	**MiCERandom**
1	Read the book, **forget** the movie!	Read the book, forget the **movie**!
2	Read the book, **enjoy** the movie!	Read the book, forget the **computer**!
3	Read the book, **not** the movie!	Read the book **then** forget the computer!
4	Read the book, not the **screen**!	Read the book then forget the **TV**!
5	Read the book, not the **film**!	Read the book **,** forget the **movie**!
6	Read the book, not the **dictionary**!	Read the book , forget the **blah**!
7	Read the book, not the **movie**!	Read the book , forget the **duh**!
8	Read the book, not the **snob**!	Read the book , forget the **dash**!
9	Read the book, not the **Sn atchbox**!	**...**the book , forget the dash!
10	Read the **Book**, not the Sn **atchers**!	... **classic** book, forget the dash!

Table 1.9: Τα αντιπταδείγματα για τους Polyjuice and TextFooler για 10 βήματα σε ένα δείγμα από το IMDb.

Step	**Polyjuice**	**TextFooler**
1	**Read** the book, **forget** the movie!	Read the book, **forget** the movie!
2	**Watch** the book, **watch** the movie!	Read the book, **missed** the movie!
3	**Don't waste your time**, **or your money.**	Read the **accountancy**, missed the movie!
4	**You do**n't waste your time, or money.	**Dsl** the accountancy, missed the movie!
5	**Spend** your time, or money.	Dsl the accountancy, **forget** the movie!
6	Spend your time, **money,** or money.	Dsl the accountancy, **missed** the movie!
7	Spend your time, money or money **wisely**.	Dsl the accountancy, **forget** the movie!
8	Spend your time, money**, and** wisely.	Dsl the accountancy, **missed** the movie!
9	Spend your time, money, and **dollars** wisely.	Dsl the accountancy, **forget** the movie!
10	Spend your time, money and **time on this masterpiece**.	Dsl the accountancy, **missed** the movie!

Στο σημείο αυτό είναι δόκιμο να παρουσιάσουμε και τον Πίνακα 1.10 όπου χρησιμοποιείται ο MiCE με αναζήτηση δέσμης με 120 δέσμες.

Table 1.10: Τα αντιπαραδείγματα που παράγει ο συντάκτης MiCE με αναζήτηση δέσμης με 120 δέσμες.

Step	**MiCE with 120 beams**
1	Read the book, **forget** the movie!
2	Read the book, **savor** the movie!
3	Read the book, **avoid** the movie!
4	Read the book, **LOVE** the movie!
5	Read the book, **avoid** the movie!
6	Read the book, **enjoy** the movie!
7	Read the book, **skip** the movie!
8	Read the book, **LOVE** the movie!
9	Read the book, **skip** the movie!
10	Read the book, **LOVE** the movie!

Παρόλο που τόσο αυτή η έκδοση του συντάκτη όσο και η αρχική του MiCE χρησιμοποιούν τη μέθοδο μασκαρίσματος προσοχής, η διαφορά τους έγκειται στο γεγονός ότι αυτός ο συντάκτης διερευνά πολύ περισσότερες διαφορετικές αντικαταστάσεις καθώς επιλέγει λέξεις που συνολικά παράγουν την πιο πιθανή ακολουθία λέξεων. Ως εκ τούτου, παρατηρούμε ότι από το βήμα 3 και μετά ο συντάκτης αυτός είναι σε θέση να επιλέγει ρήματα που αντιστρέφουν το κύριο συναίσθημα της πρότασης, και στη συνέχεια να αντιστρέφει εύκολα την πρόβλεψη για την πρόταση. Αντίθετα, ο MiCE τροποποιεί την πρόταση με τη λέξη "not" στο βήμα 3, γεγονός που φαίνεται να μην είναι ιδανικό και να επηρεάζει τη δημιουργία επεξεργασιών αργότερα. Επιπλέον, σε αυτό το παράδειγμα παρατηρούμε μια επανάληψη των λέξεων "LOVE", "avoid" και "skip". Αυτό αποδεικνύει ότι όταν αυτές οι λέξεις επιλέγονται από τον συντάκτη, ο προγνωστικός δείκτης επηρεάζεται έντονα προς την κατεύθυνση της στοχευμένης κλάσης.

Κατάτμηση κειμένου από τον Polyjuice

Στους Πίνακες 1.9 και 1.12, βλέπουμε δύο παραδείγματα από το σύνολο δεδομένων IMDb όπου παρατηρούμε ότι το Polyjuice αφαιρεί την πλειονότητα του κειμένου εισόδου, αλλάζοντας δραματικά τη δομή και τη σημασιολογία της πρότασης. Αυτό επικυρώνει τη συμπεριφορά του Polyjuice που διερευνήσαμε στο υποκεφάλαιο της Ελαχιστότητας, ότι δηλαδή το Polyjuice αφαιρεί κατά μέσο όρο πάνω από 70% του κειμένου εισόδου.

Table 1.11: Ένα παράδειγμα από το IMDb όπου ο Polyjuice αφαιρεί μεγάλο κομμάτι της είσοδου.

Step	**Polyjuice**
1	This movie was **horrible** and corny. James Agee is rolling in his grave.**This movie was nothing at all like the book and made a mockery of it. No one should see this movie unless they want to gag.**
2	This movie was **fun** and corny. James Agee is rolling in his grave.
3	This movie was **boring** and **depressing**. James Agee is rolling in his grave.
4	This movie was **good** and **moving**. James Agee is **at** his **best**.

Table 1.12: Ένα παράδειγμα από το NewsGroups όπου ο Polyjuice αφαιρεί μεγάλο κομμάτι της είσοδου.

Step	**Polyjuice**
1	**On all 1.44Mb drives (both Mac and PC), the disk spins at a constant RPM. On 800k Mac disk drives, the spin rate of the disk is varied so that the tracks pass under the head at a constant speed; a slower rotation for the outer tracks, and a faster rotation for the inner tracks.** A PC needs **special controller hardware** to make this happen.
2	A PC needs **to be able** to make this **game** happen.
3	A PC needs to be **installed** to **run** this game happen.
4	A PC needs to be installed to **play** this game **correctly** .

Αξιολόγηση αντιπαραδειγμάτων με στοχευμένο μέρος του λόγου

Στους πίνακες 1.13, 1.14 και 1.15, τα στοχευμένα μέρη του λόγου είναι το επίθετο και το ουσιαστικό σε ένα δείγμα NewsGroups. Καθώς το σύνολο δεδομένων IMDb εξειδικεύεται στην ανάλυση συναισθήματος τα επίθετα επηρεάζουν σημαντικά την ταξινόμηση της πρότασης, ενώ στο NewsGroups, το οποίο χρησιμοποιείται για την ταξινόμηση θεμάτων, τα ουσιαστικά παίζουν σημαντικό ρόλο στο αποτέλεσμα του ταξινομητή. Τα παραδείγματα καταδεικνύουν πόσο αποτελεσματικοί είναι οι συντάκτες όταν στοχεύουμε σε μια συγκεκριμένη ετικέτα POS, και απεικονίζουν επίσης τη σημαντική μείωση της ελαχιστότητας και επακόλουθη αύξηση της συνέπειας για τους συντάκτες. Ωστόσο, αυτό δυνητικά δείχνει πως ένας συντάκτης με περιορισμούς δημιουργεί λιγότερο ποικίλες επεξεργασίες συγκριτικά με την παραγωγή αντιπαραδειγμάτων χωρίς περιορισμούς όπου γεννάται πιο ποικιλόμορφο κείμενο.

Στα παρακάτω παραδείγματα, παρατηρούμε πώς η στοχευμένη μέθοδος των ετικετών μέρους του λόγου μειώνει σημαντικά την ελαχιστότητα των αντιπαραδειγμάτων. Επιπλέον, παρατηρούμε ότι παρόλο που το δείγμα δεν έχει μικρό μήκος, οι συντάκτες καταφέρνουν να αντιστρέψουν την πρόβλεψη του ταξινομητή τροποποιώντας μόνο μία ή δύο λέξεις. Ως εκ τούτου, οι περιορισμοί που επιβάλλουμε στους συντάκτες τους κάνουν να διαταράσσουν την πρόταση εισόδου πιο επιθετικά προκειμένου να επιτύχουν την επιθυμητή κλάση.

Table 1.13: Παράδειγμα από ΤΟ IMDb με τον συντάκτη MiCE όταν στοχεύουμε τα επίθετα για αλλαγές.

Step	**MiCE ADJ**
1	The **biggest** heroes, is one of the **greatest** movies ever. A good story, great actors and a brilliant ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
2	The **great carrier,** heroes, is one of the **worst horror** movies ever. A good story, great actors and a brilliant ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
3	The great carrier, heroes, is one of the **best** horror movies ever. A good story, great actors and a **surprisingly satisfying** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
4	The **original** carrier, heroes, is one of the best horror movies ever. A good story, great actors and a surprisingly **predictable** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
5	The original carrier, heroes, is one of the best horror movies ever. A good story, great actors and a surprisingly **satisfying** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
6	The **liminal** carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's **romplimlim** carrier.
7	The **romplimpig** carrier, heroes, is one of the **best** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's **romplimpig** carrier.
8	The romplimpig carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.
9	The romplimpig carrier, heroes, is one of the **greatest** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.
10	The romplimpig carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.

Table 1.14: Παράδειγμα από ΤΟ IMDb με τον συντάκτη TextFooler όταν στοχεύουμε τα επίθετα για αλλαγές.

Step	**TextFooler ADJ**
1	The **biggest** heroes, is one of the **greatest** movies ever. A **good** story, **great** actors and a **brilliant** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
2	The **longer** heroes, is one of the **more** movies ever. A **ok** story, **dramatic** actors and a **creditable** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
3	The longer heroes, is one of the more movies ever. A **okay** story, dramatic actors and a creditable ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
4	The longer heroes, is one of the more movies ever. A okay story, dramatic actors and a creditable ending is what makes this film the jumping start of the director Thomas Vinterberg's **unbelievable** carrier.
5	The longer heroes, is one of the more movies ever. A okay story, dramatic actors and a **laudable** ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
6	The longer heroes, is one of the **further** movies ever. A okay story, dramatic actors and a laudable ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
7	The longer heroes, is one of the further movies ever. A okay story, dramatic actors and a **praiseworthy** ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
8	The longer heroes, is one of the further movies ever. A okay story, **disastrous** actors and a praiseworthy ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
9	The longer heroes, is one of the further movies ever. A okay story, **catastrophic** actors and a praiseworthy ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
10	The **plus** heroes, is one of the **alternatively** movies ever. A okay story, catastrophic actors and a praiseworthy ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.

Ακόμη, στα εν λόγω παραδείγματα παρατηρούμε ότι οι στοχευμένες ετικέτες μέρους του λόγου μας παρέχουν ένα επίπεδο ελέγχου στα παραγόμενα αντιπαραδείγματα και ταυτόχρονα εξασφαλίζουν υψηλή κειμενική συνοχή των προτάσεων. Επιπλέον, οι αλλαγές που επιβάλλουν οι συντάκτες μάς δείχνουν την ευελιξία των αντιπαραδειγμάτων που εξασφαλίζει η μέθοδος μας όταν τροποποιούμε κάθε φορά διαφορετικές ετικέτες μερών του λόγου.

Table 1.15: Παράδειγμα από το NewsGroups με τον συντάκτη TextFooler ενώ στοχεύουμε τα ουσιαστικά για αλλαγές.

Step	TextFooler NOUN
1	This is very curious **being** that they are both built by Mercury in the very same factory. Steve
2	This is very curious **continual** that they are both built by Mercury in the very same factory. Steve
3	This is very curious continual that they are both built by Mercury in the very same **factories**. Steve
4	This is very curious **indefatigable** that they are both built by Mercury in the very same factories. Steve
5	This is very curious indefatigable that they are both built by Mercury in the very same **fabrication**. Steve
6	This is very curious indefatigable that they are both built by Mercury in the very same **manufacture**. Steve
7	This is very curious indefatigable that they are both built by Mercury in the very same **fabrication**. Steve
8	This is very curious indefatigable that they are both built by Mercury in the very same **manufacture**. Steve
9	This is very curious indefatigable that they are both built by Mercury in the very same **fabrication**. Steve
10	This is very curious indefatigable that they are both built by Mercury in the very same **manufacture**. Steve

Είναι επίσης σημαντικό να σημειωθεί ότι οι αλλαγές του TextFooler στον Πίνακα 1.15 μπορεί να φαίνονται ασήμαντες εκ πρώτης όψεως όσον αφορά τη μεταβολής του νοήματος της πρότασης. Για παράδειγμα, από το 5ο βήμα και έπειτα παρατηρούμε την εναλλαγή των λέξεων, "fabrication" και "manifacture" οι οποίες είναι σχεδόν συνώνυμες και έχουν μικρή απόσταση στις ενσωματώσεις που χρησιμοποιεί ο TextFooler αλλά για τον ταξινομητή αρκούν για να αλλάξει η πρόβλεψη για το κυρίαρχο θέμα της πρότασης. Με αυτόν τον τρόπο επομένως, ο TextFooler μας αποκαλύπτει την ευαισθησία του ταξινομητή σε μικρές αλλαγές.

Επιπλέον, επικυρώνουμε τόσο σε αυτό όσο και στα παραπάνω παραδείγματα ότι η συνοχή του παραγόμενου κειμένου παραμένει γενικά σταθερή καθώς διαταράσσουμε την πρόταση, οπότε η γλωσσική ποιότητα των αντιπαραδειγμάτων είναι στις περισσότερες περιπτώσεις επαρκής.

1.3.4 Χρόνος Εκτέλεσης: Ένας Σημαντικός Περιορισμός

Ίσως, ο σημαντικότερος περιορισμός των πειραμάτων είναι η διάρκεια των πειραμάτων, η οποία αποτελεί συχνά έναν αναμενόμενο περιορισμό σε εργασίες μηχανικής μάθησης. Για να αντιμετωπίσουμε αυτόν τον περιορισμό, πειραματιστήκαμε μόνο με ένα μέρος των συνόλων δοκιμών των συνόλων δεδομένων, γεγονός που μπορεί να μη μας περιόρισε όσον αφορά τα αποτελέσματά μας αλλά μας εμποδίζει από την παραγωγή περισσότερων αντιπαραδειγμάτων και από την εξαγωγή μίας πιο συνολικής εικόνας. Ενδιαφέρον έχουν κάποιοι αριθμοί σχετικά με τη διάρκεια όπως ο μέσος χρόνος που απαιτείται για τη δημιουργία ενός αντιπαραδείγματος, που είναι 20 και 9,5 δευτερόλεπτα για το IMDb και το NewsGroups αντίστοιχα. Ακόμη, οι συνολικές ώρες GPU που χρειάστηκαν για όλα τα πειράματά μας ήταν 1670 (!) που μεταφράζονται σε 69,5 ημέρες. Το Σχήμα 1.3.5 δείχνει πώς ο απαιτούμενος χρόνος κατανέμεται μεταξύ των διαφόρων κατηγοριών πειραμάτων και πώς ορισμένες μέθοδοι επηρεάζουν σημαντικά τη διάρκεια των πειραμάτων. Επίσης, σημαντικά σημεία παρατήρησης είναι ότι το MiCE με 120 δέσμες χρειάζεται περίπου τετραπλάσιο χρόνο εκτέλεσης από το MiCE με πολυωνυμική δειγματοληψία, ενώ η μέθοδος των στοχευμένων ετικετών POS μειώνει σημαντικά τον απαιτούμενο χρόνο εκτέλεσης σε όλους τους συντάκτες.

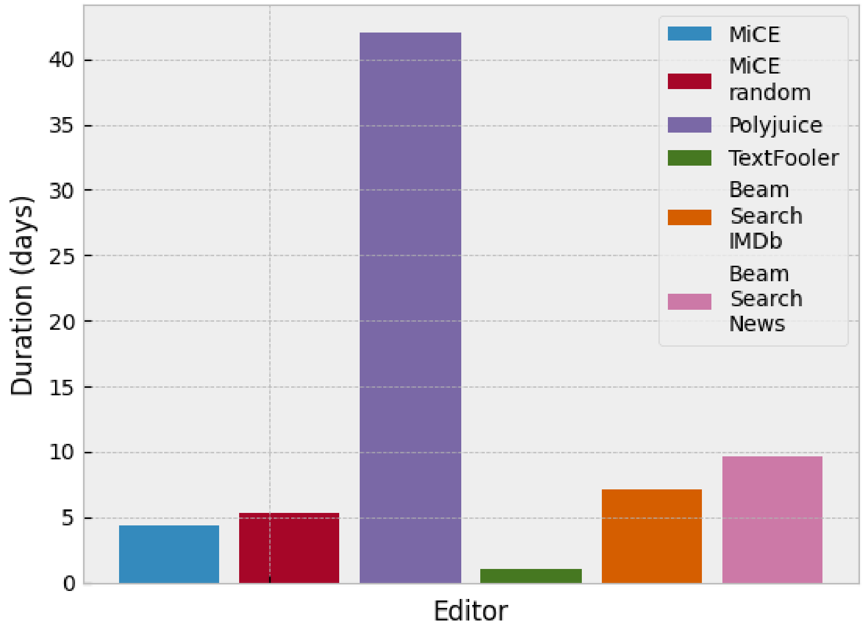


Figure 1.3.5: Κατανομή του συνολικού χρόνου εκτέλεσης στις 6 κατηγορίες πειραμάτων που περιλαμβάνουν τα πειράματα στους 4 συντακτών και τα πειράματα στο MiCE με αναζήτηση δέσμης και στα 2 σύνολα δεδομένων.

1.4 Συμπεράσματα

1.4.1 Συζήτηση

Η παρούσα εργασία διερεύνησε τη χρήση διαφορετικών συντακτών αντιφατικών εξηγήσεων για το έργο της δημιουργίας αντιφατικών προτάσεων με σκοπό την αξιολόγηση των διαφορετικών στοιχείων και μεθόδων αυτών των συντακτών με τη χρήση πολλαπλών μετρήσεων και ποιοτικής ανάλυσης. Για τη διεξαγωγή των πειραμάτων μας, χρησιμοποιήσαμε τη νέα μέθοδο των Counterfactuals of counterfactuals [20] και για την αξιολόγηση διερευνήσαμε τα αποτελέσματα της πρόσφατα εισαχθείσας μετρικής που ονομάζεται ασυνέπεια. Επιπλέον, παρακινούμενοι από την υποεργασία της δημιουργίας συντακτών που επιφέρουν αλλαγές με τον πιο ελάχιστο τρόπο, εισαγάγαμε μια παραλλαγή των αντιφατικών εξηγήσεων όπου οι συντάκτες περιορίζονται στο να δημιουργούν αντιπαραδείγματα ενώ στοχεύουν σε μια συγκεκριμένη ετικέτα μέρους του λόγου. Στα αποτελέσματα, παρατηρήσαμε ότι κάθε συντάκτης παρουσιάζει πολλαπλές αδυναμίες και πλεονεκτήματα, τα οποία εξηγούνται από τις επιδόσεις τους σε μία ή περισσότερες μετρικές, αλλά και από την ποιοτική ανάλυση. Συγκεκριμένα, η μέθοδος κάλυψης που χρησιμοποιείται εμφανίζει σημαντικό αντίκτυπο στις παραγόμενες αντιφατικές εξηγήσεις, με τους συντάκτες που χρησιμοποιούν μέθοδο μασκαρίσματος που έχει οποιαδήποτε γνώση για τις τάσεις του ταξινομητή να έχουν καλύτερες επιδόσεις ως προς όλες τις μετρικές και την ποιότητα του παραγόμενου κειμένου.

Επιπλέον, για πιο ντετερμινιστικές προσεγγίσεις στην παραγωγή αντιφατικών τα πειράματα έδειξαν ότι οι συντάκτες στις εκδόσεις με ετικέτες POS γίνονται πιο συνεπείς, αλλά με το κόστος της αποτυχίας να αναστρέψουν ορισμένες από τις εξηγήσεις. Ωστόσο, τα επίπεδα ευχέρειας ήταν συγκρίσιμα και σε ορισμένες περιπτώσεις ακόμη και υψηλότερα από τις επεξεργασίες χωρίς στοχευμένες ετικέτες POS. Επιπλέον, μέσω αυτής της μεθόδου συλλέχθηκαν πολύτιμες πληροφορίες σχετικά με τη σημασία κάθε ετικέτας μέρους του λόγου σε συνδυασμό με την εργασία κάθε συνόλου δεδομένων. Αυτή η γλωσσολογική ανάλυση έδειξε ότι για την ανάλυση συναισθήματος και το σύνολο δεδομένων IMDb, η διαταραχή των επιθέτων είναι ζωτικής σημασίας για την επιτυχή αντιπαραβολή. Από την άλλη πλευρά, το σύνολο δεδομένων NewsGroups αποδεικνύεται ότι αποδίδει μεγαλύτερη σημασία στα ουσιαστικά που μπορούν να επηρεάσουν αποτελεσματικότερα το κύριο θέμα του κειμένου.

Τέλος, η μέθοδος των αντιφατικών των αντιφατικών αποδείχθηκε ανεκτίμητη για την ερευνητική κοινότητα, καθώς διερευνά καταστάσεις που δεν είχαν προηγουμένως παρατηρηθεί από τις παραδοσιακές μεθόδους δημιουργίας αντιφατικών. Το παρόν κείμενο εξετάζει τη χρήση των εξηγήσεων αντιπαραδειγμάτων για την εξήγηση της συμπεριφοράς των συντακτών αντιπαραδειγμάτων και των μοντέλων και αλγορίθμων παραγωγής κειμένων. Υποστηρίζεται ότι με κατάλληλες μεθόδους αξιολόγησης και τη χρήση εξηγήσεων με αντιπαραδείγματα, μπορούμε να παρέχουμε επαρκείς εξηγήσεις για την απόδοση διαφόρων επεξεργαστών, μοντέλων και μεθόδων παραγωγής κειμένου. Μέσω των πειραμάτων μας, δημιουργήσαμε χιλιάδες αντιφατικές εξηγήσεις και μπορέσαμε να αξιοποιήσουμε τις ετικέτες μέρους του λόγου για να εισάγουμε μια νέα αποτελεσματική μέθοδο παραγωγής αντιφατικών κειμένων. Τα αποτελέσματα της εργασίας μας παρέχουν περαιτέρω κίνητρα για μελλοντική έρευνα στο ίδιο πεδίο, με πρόσθετες μετρικές, συντάκτες και εργασίες.

Διαπιστώσαμε ότι ο περιορισμός των μοντέλων μπορεί να οδηγήσει με αποδείξεις σε καλύτερα ποσοτικά αποτελέσματα για πολλαπλές μετρήσεις, αλλά ποιο είναι το "κόστος" όσον αφορά την ποιότητα των αποτελεσμάτων; Μπορούμε να χρησιμοποιήσουμε την εξήγηση των υφιστάμενων μοντέλων για να δημιουργήσουμε τελικά βέλτιστους συντάκτες/μοντέλα ή θα πρέπει να τη χρησιμοποιήσουμε για να αποφασίσουμε τα συμβιβαστικά ανάλογα με την εκάστοτε εργασία;

1.4.2 Περιορισμοί

Κατά τη διάρκεια αυτής της διατριβής, αντιμετωπίσαμε αρκετούς περιορισμούς που προέκυψαν κυρίως από τις μεγάλες υπολογιστικές απαιτήσεις της εργασίας μας. Όπως περιγράφουμε στην ενότητα 7.2.4, τα πειράματα που διεξήχθησαν απαιτούσαν σημαντικό χρονικό διάστημα και μερικές φορές παράλληλη εκτέλεση σε πολλές GPU. Αυτός ο περιορισμός επηρέασε άμεσα το μέγεθος των συνόλων δεδομένων στα οποία πειραματιστήκαμε, γεγονός που ευτυχώς δεν εμπόδισε τα αποτελέσματα και τα συμπεράσματά μας. Επιπλέον, με βάση αυτόν τον περιορισμό επιλέξαμε να μην εξαντλήσουμε όλους τους διαθέσιμους αντιπαραθετικούς συντάκτες, καθώς αυτό θα απαιτούσε σημαντικά περισσότερες υπολογιστικές απαιτήσεις και χρόνο. Τέλος, τα πειράματά μας καταφεύγουν στη γλώσσα των αγγλικών, καθώς δεν πειραματιστήκαμε με σύνολα δεδομένων από άλλες γλώσσες. Ωστόσο, αναμένουμε ότι τα συμπεράσματα που προέκυψαν καθ' όλη τη διάρκεια αυτής της εργασίας ισχύουν για όλες τις γλώσσες και ότι οι μέθοδοι που χρησιμοποιήθηκαν σε αυτή την εργασία μπορούν να λειτουργήσουν χωρίς τροποποιήσεις και σε άλλες γλώσσες.

1.4.3 Μελλοντικές κατευθύνσεις

Κλείνοντας αυτή τη διατριβή θα θέλαμε να προτείνουμε διάφορους δρόμους για περαιτέρω βελτίωση ή εναλλακτικές εφαρμογές και προσεγγίσεις που θα μπορούσαν να εμπνεύσουν μελλοντική έρευνα. Πρώτον, καθώς σε αυτή την εργασία τα σύνολα δεδομένων με τα οποία εργαστήκαμε εξειδικεύονται στις εργασίες της ανάλυσης συναισθήματος και της ταξινόμησης θεμάτων, θα ήταν ενδιαφέρον να διερευνήσουμε τον τρόπο με τον οποίο οι συντάκτες αποδίδουν σε συνδυασμό και με άλλες εργασίες, όπως η αναγνώριση ονομαστικών οντοτήτων (NER), όπου το ενδιαφέρον θα ήταν να διαταραχθούν μόνο οι ονομαστικές οντότητες, ή η ταξινόμηση με βάση τις πτυχές όπου το συναίσθημα κατηγοριοποιείται σε σχέση με συγκεκριμένες οντότητες-στόχους ή συγκεκριμένες πτυχές. Επιπλέον, μια άλλη ενδιαφέρουσα προσέγγιση θα ήταν να πειραματιστούμε με διαφορετικούς ταξινομητές προκειμένου να καταγραφεί πώς αποδίδουν σε συνδυασμό με πολλαπλά σύνολα δεδομένων, επιδιώκοντας την αποκάλυψη τυχόν υφιστάμενων προκαταλήψεων ή μεροληψιών. Τέλος, αυτό που θα μπορούσε επίσης να διερευνηθεί είναι η ασυνέπεια των άλλων μετρικών εκτός από την ελαχιστότητα. Όπως δείχνουν τα αποτελέσματά μας, η προσέγγιση των αντιπαραδειγμάτων από αντιπαραδείγματα των συγγραφέων του [20] σε συνδυασμό με την έννοια της ασυνέπειας μας επιτρέπουν να αποκαλύψουμε τους πιθανούς περιορισμούς μιας μετρικής και συνεπώς η χρήση μιας παρόμοιας προσέγγισης με άλλες μετρικές που σχετίζονται με τη βιβλιογραφία θα μπορούσε να αποδειχθεί επωφελής. Μια άλλη ιδέα για μελλοντική έρευνα θα ήταν επίσης το αντίστροφο πρόβλημα, δηλαδή να επιχειρήσουμε να χρησιμοποιήσουμε τις εξηγήσεις και τους περιορισμούς κάθε επεξεργαστή προκειμένου να προσπαθήσουμε να βελτιώσουμε τις υπάρχουσες υλοποιήσεις ή ακόμη και να δημιουργήσουμε έναν νέο συντάκτη από το μηδέν με βάση το ποιες μέθοδοι και μοντέλα αποδίδουν καλύτερα. Για παράδειγμα, προκειμένου να δημιουργηθεί ένας συντάκτης αντιπαραδειγμάτων ο οποίος να είναι πιο συνεπής με τις τροποποιήσεις που κάνει, θα μπορούσαμε να διερευνήσουμε τις υλοποίηση ενός συντάκτη που θα είναι ρυθμισμένος (fine-tuned) με την προσέγγιση της ανατροφοδότησης του [20] και την ασυνέπεια της ελαχιστότητας, η οποία φαίνεται ότι θα μπορούσε να παρέχει ελπιδοφόρα αποτελέσματα και παραγόμενες αντιπαραδείγματα. Τέλος, οι χιλιάδες παραγόμενες επεξηγήσεις με αντιπαραδείγματα θα μπορούσαν να αξιοποιηθούν προκειμένου να δημιουργηθεί ενδεχομένως ένα σύνολο δεδομένων για την περαιτέρω αξιολόγηση τους ή για τη χρήση τους σε άλλες εργασίες, όπως η επαύξηση δεδομένων.

Chapter 2

Introduction

Over the past few decades, machine learning (ML) has undergone remarkable advancements, revolutionizing the field of artificial intelligence. ML algorithms have exhibited impressive capabilities across various applications, from image and speech recognition to natural language processing and recommendation systems. Additionally, sophisticated techniques like generative adversarial networks (GANs) [24] and reinforcement learning have pushed the boundaries of what ML can achieve, generating realistic images and videos and enabling optimal decision-making. These innovations have found application in multiple aspects of our everyday lives, from smart devices and AI-powered assistants to art-creation, self-driving cars, and AI in healthcare. However, amidst these remarkable developments, it is of utmost importance to investigate the level of trust we can build with these tools. Thus, a fundamental question arises: What can we do to enable AI systems that can explain the decisions taken by models?

The field of Explainable AI (XAI) aims to answer just that and provide insights behind the actions and decisions of ML models. The explanation is crucial in the realm of AI because as ML algorithms become increasingly complex, their decision-making processes can become less transparent. In domains such as healthcare, finance, and law, where critical decisions are made based on AI-generated recommendations, interpretability, and transparency are paramount. Stakeholders, including end-users, regulators, and ethical committees, demand a clear understanding of the factors influencing AI decisions to ensure fairness, accountability, and trustworthiness. Moreover, explainability facilitates user acceptance and enables domain experts to identify potential biases [3], errors, or unintended consequences. By providing explanations for ML decisions, AI systems can bridge the gap between the black-box nature of complex models and human understanding, empowering users to make informed judgments, refine models, and mitigate risks.

In this thesis, we investigate the concept of counterfactuals, which provide a powerful framework for implementing explainability in AI systems. Counterfactuals, in the context of a text, refer to hypothetical alterations or "what-if" transformations applied to the input text that result in alternative versions of the output. These alterations allow us to observe how changes in the text impact the decisions made by the model and uncover the underlying factors driving the model's predictions, shedding light on the intricate workings of the AI system. To illustrate the concept of counterfactuals, consider a scenario where an automated news summarization system generates a summary for an article about a political event. The original summary reads: "The candidate's speech received widespread acclaim for its inspiring message." To investigate the factors contributing to this positive sentiment, a counterfactual edit could involve replacing "inspiring" with "divisive." By generating this counterfactual version, we can discern how altering the sentiment of the summary affects its reception and assess the model's sensitivity to such changes.

In this work, we focus on the evaluation of counterfactual explanations generated by different counterfactual editors with multiple methods, models, and constraints. We use multiple evaluation methods and metrics in order to interpret the behavior of these components and provide explanations for their decisions. Our work is mainly motivated by the research of Filandrianos et al. [20], where the authors also evaluate counterfactual edits in order to explain decisions made by models. Using two novel methods of generation and evaluation, first presented in [20] we examine an original evaluation process combined with more traditional ones. Moreover, this thesis proposes a linguistics-inspired method of counterfactual generation using specific part-of-speech tags to add an extra constraint to edit generation. In this way, we seek to provide another aspect in the generation and evaluation of counterfactual explanations, examine its efficiency, and how it differentiates from other generation tactics.

In addition, this thesis provides a broad exploration of Counterfactual Explanations and the motivation for their use, as long as methods of counterfactuals other than text. We also discuss relevant work and demonstrate the range of existing systems that specify in counterfactual generation. Another focus of this work is the implementation of a counterfactual generation system where we combine state-of-the-art text counterfactual editors with a predictor, in order to fairly compare models and methods on the task of contrastive counterfactual explanations. An important aspect of counterfactual generation and a main examination point of our work is the minimality of counterfactual edits.

The outline of this thesis is as follows:

- We firstly provide all the essential background in basic Machine Learning concepts and explain how counterfactual explanations are a significant means to interpretability.
- We give a detailed definition of Counterfactual Explanations and provide the main motivations behind them. After doing so, we discuss the evaluation methods and metrics that are followed by recent literature, focusing on the ones that we employ in our work.
- We propose the use of the methods introduced in [20] along with our targeted part-of-speech tagging method and explain how they can all contribute towards the evaluation of counterfactual explanation.
- Lastly, we show how we structure and implement our counterfactual generation system and present the results obtained from both a quantitative and qualitative standpoint. Through this process, we provide numerous counterfactual edits along with explanations for the decisions of the counterfactual editors, valuable conclusions but also potential limitations.

Chapter 3

Machine Learning

Over the course of centuries, researchers have been captivated by the pursuit of replicating the remarkable capabilities and intelligence exhibited by humans through technological advancements. This quest has been further fueled by the advent of computers and the subsequent establishment of Artificial Intelligence (AI) as a dedicated field of study. According to [23] AI encompasses the development of computer systems capable of performing tasks that typically demand human intelligence. At its inception, AI primarily focused on solving problems that posed challenges to humans but were relatively straightforward for machines. However, as the field evolved, the importance of addressing the converse challenge became evident – that of teaching machines to handle tasks that humans find intuitive due to their accumulated experiences. This endeavor gave rise to the domain of Machine Learning, which plays a pivotal role in enabling machines to acquire knowledge, learn from data, and navigate tasks previously deemed exclusive to human intellect.

Machine Learning (ML) is a branch of AI that leverages data in order to make predictions with the use of statistical methods and algorithms. As the name of the field implies, machine learning algorithms structure models based on sample data, known as training data, to reach a specific level of knowledge without being explicitly programmed to do so [78] [97]. These models process input data using multiple methods and can extract multiple patterns from raw data in order to generate outputs for various tasks.

To accomplish these tasks effectively, machine learning heavily relies on data that should be diverse, representative, and properly labeled. This data serves as the input from which the algorithm learns and extracts patterns and features. It can be in the form of text, images, audio, or any other structured or unstructured data. Machine learning then utilizes this data effectively for a wide range of tasks, starting with the fundamental ability to classify and group data based on patterns and similarities. These two tasks form the foundation of many other advanced machine learning applications.

In the field examined by our work, which is Natural Language Processing (NLP), machine learning can be utilized for sentiment analysis, text generation, text classification, and semantic analysis. Computer vision tasks are also a vast category and include image recognition, object detection, facial recognition, and even autonomous driving. Machine learning is also employed in recommendation systems, fraud detection, e-learning [85], predictive analytics, and many more fields. These diverse applications demonstrate the versatility of machine learning and its potential to extract insights, automate processes, and improve decision-making across numerous domains, driven by the power of data analysis and pattern recognition.

Contents

3.1 Natural Language Processing

In this section, we focus on Natural Language Processing (NLP), specifically text generation, text classification, sentiment analysis, and POS tagging. These tasks are essential in enabling computers to generate coherent text, categorize documents, understand emotions in text, and assign grammatical tags to words. Each subsection explores the underlying techniques and practical applications of these NLP tasks, providing valuable insights into the field's advancements and their impact in various domains.

3.1.1 Text generation

Text generation is an active area of research within the field of Natural Language Processing (NLP), focusing on the development of algorithms and models capable of generating coherent and contextually relevant text. It encompasses a range of techniques and methodologies aimed at producing human-like textual output based on given prompts, conditions, or learned patterns.

One prominent approach in text generation is the use of probabilistic models, such as n-gram models [80] or hidden Markov models (HMMs) [1]. These models estimate the likelihood of generating the next word or sequence of words based on the previous context. While they can capture local dependencies and generate text relatively quickly, they often struggle with capturing long-range dependencies and producing coherent and meaningful output. Another widely adopted approach is the use of neural network-based models, particularly recurrent neural networks (RNNs) [16] and their variants, such as long short-term memory (LSTM) [29] networks. RNNs excel in capturing sequential dependencies and have shown success in various text generation tasks. By processing input text step-by-step and updating hidden states, RNNs can generate text that is contextually informed and coherent.

Recent advancements in deep learning have led to the emergence of Transformer models [89], which have significantly impacted text generation tasks. Transformers leverage the attention mechanism, allowing for parallel processing and capturing global dependencies across input sequences. This architecture has revolutionized the field by achieving state-of-the-art results in machine translation, summarization, and text generation. In the context of text generation, models are typically trained using large corpora of text data, either in a supervised or unsupervised manner. Supervised approaches involve training the model on paired input-output examples, while unsupervised approaches focus on learning from unannotated text. Pre-training on vast amounts of data followed by fine-tuning for specific tasks has proven effective in improving the quality and coherence of the generated text. Evaluation of text generation models is a challenging task, as it requires a comprehensive assessment of the generated output in terms of relevance, coherence, grammaticality, and overall quality. Metrics such as perplexity, BLEU score, or human evaluation through crowdsourcing are commonly employed to measure the performance of text generation models.

Ethical considerations are also of utmost importance in text generation research. Issues related to bias, fairness, and the potential for malicious use must be addressed to ensure responsible and ethical deployment of text generation algorithms.

Transformers

In this segment, we provide some information on Transformers as they are a crucial component of our work which we present in the next chapters. Transformers have emerged as a powerful architecture for text generation within the field of Natural Language Processing (NLP). Unlike traditional recurrent neural networks (RNNs) that process text sequentially, Transformers leverage the attention mechanism to capture global dependencies and enable parallel processing of input sequences. Introduced by Vaswani et al. in 2017 [89], the Transformer architecture has significantly impacted text generation tasks. Unlike traditional recurrent neural networks (RNNs), Transformers leverage the attention mechanism, enabling parallel processing and capturing global dependencies within input sequences.

The Transformer architecture includes the **encoder-decoder framework**, self-attention mechanism and feed-forward neural networks. The encoder is the component that takes the input sequence and transforms it into a set of high-dimensional representations, capturing the contextual information of each token in the sequence. The decoder then generates the output sequence based on the encoder's representations and previously generated tokens. **Self-attention** is a key feature of the Transformer, allowing the model to focus on relevant parts of the input sequence. It computes attention scores between all positions, weighing their importance during generation. By capturing long-range dependencies and contextual information, self-attention enables effective processing of the sequence. Lastly, Transformers employ **feed-forward neural networks** as a way to transform the representations learned by the self-attention mechanism. These networks consist of multiple layers of fully connected layers with non-linear activation functions. They help to capture and refine the learned representations, enabling the model to better model complex relationships in the data.

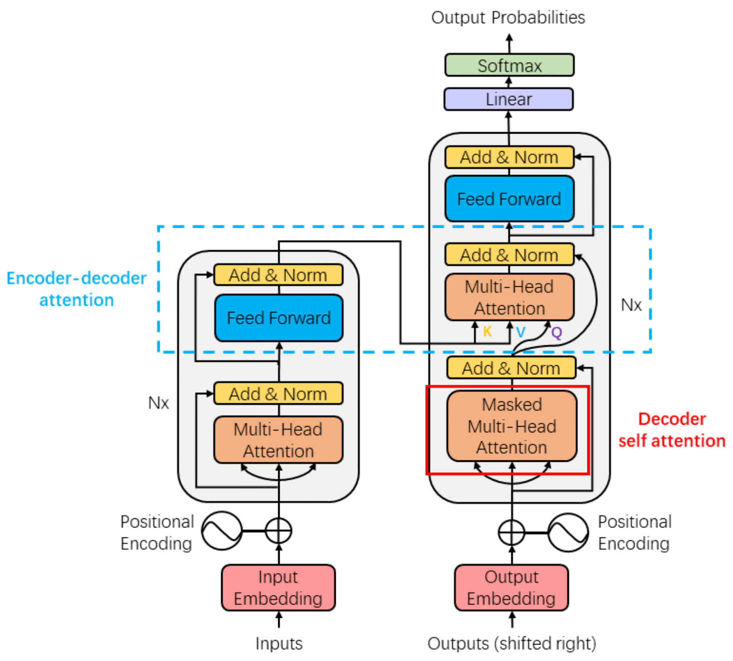


Figure 3.1.1: An overview of the Transformer's architecture. The figure is obtained from the relevant paper [89].

An important Transformer-based model is **GPT**, which includes models such as GPT-1, GPT-2 (which we use later in this work), and GPT-3, which has demonstrated impressive capabilities in generating coherent and contextually relevant text. By pre-training on large-scale datasets and fine-tuning for specific tasks, GPT models have displayed very prominent results. For example, GPT-3, with its 175 billion parameters, has demonstrated impressive capabilities in various text generation tasks. It can generate human-like articles, compose poetry, answer questions, and even write code snippets. The upsurging AI-powered chatbot ChatGPT [62] launched by OpenAI in 2022, an instantiation of GPT-3, is one of the most recent examples of what the model can achieve, where by providing a prompt or a starting sentence, the model can continue generating text that is consistent with the provided context and undertake complex problems.

On the other hand, **T5**, or Text-to-Text Transfer Transformer, introduced by Raffel et al. [70] in 2019, is another powerful Transformer-based model. T5 takes a different approach by casting various NLP tasks, including text generation, as text-to-text problems. It is trained on a massive corpus of diverse text and can be fine-tuned for specific tasks such as translation, summarization, question answering, and more. T5's versatility and flexibility make it a highly effective model for a wide range of text-generation tasks.

In summary, Transformers have revolutionized the field of text generation in NLP. Their attention mechanisms and parallel processing capabilities allow for the generation of coherent and contextually relevant text, revolutionizing tasks such as creative writing, machine translation, and more. Continued advancements in Transformer models hold the potential to further enhance our ability to generate human-like text and create intelligent systems that understand and produce language in a nuanced and natural manner.

3.1.2 Text classification

Text classification is a fundamental task in Natural Language Processing (NLP) that involves assigning predefined categories or labels to text documents based on their content. It plays a crucial role in various applications, such as sentiment analysis, spam detection, topic categorization, and document classification. The goal of text classification is to develop models that can automatically analyze and classify text data accurately. This task typically involves a two-step process: training a classification model on labeled data and then using the trained model to predict the class of new, unseen text documents.

Several techniques have been employed for text classification, with machine learning algorithms being widely used. Traditional machine learning methods such as Naive Bayes, Support Vector Machines (SVM), and Decision Trees have been successfully applied to this task. These algorithms rely on handcrafted features extracted from the text, such as bag-of-words representations or TF-IDF (Term Frequency-Inverse Document Frequency) vectors. With advancements in deep learning, neural networks have also emerged as powerful models for text classification. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) networks, have shown great success in capturing the sequential and contextual information present in the text.

In recent years, pre-trained language models, such as BERT (Bidirectional Encoder Representations from Transformers) [12] and RoBERTa [46], have significantly advanced the field of text classification. These models are trained on large-scale text corpora and can effectively capture complex linguistic patterns and contextual information. By fine-tuning these pre-trained models on specific classification tasks, they can achieve state-of-the-art results in various domains.

Concerning the evaluation of text classification models, it is typically done using metrics like accuracy, precision, recall, and F1 score. These metrics measure the model's ability to correctly classify text documents across different categories.

Text classification finds applications in a wide range of fields, including sentiment analysis for social media monitoring, spam filtering for email systems, topic categorization for content organization, and sentiment-based recommendation systems, among others. The ability to automatically classify text documents based on their content allows for efficient information retrieval, data organization, and decision-making processes.

3.1.3 Sentiment Analysis

Sentiment analysis, also referred to as opinion mining, is a prominent research area within Natural Language Processing (NLP) that focuses on extracting subjective information, attitudes, and opinions expressed mainly in text data. It involves the use of computational methods to automatically classify text into sentiment categories such as positive, negative, or neutral. The analysis of sentiment has gained significant attention due to its practical applications in understanding public opinion, social media monitoring, brand reputation management, market research, and customer feedback analysis.

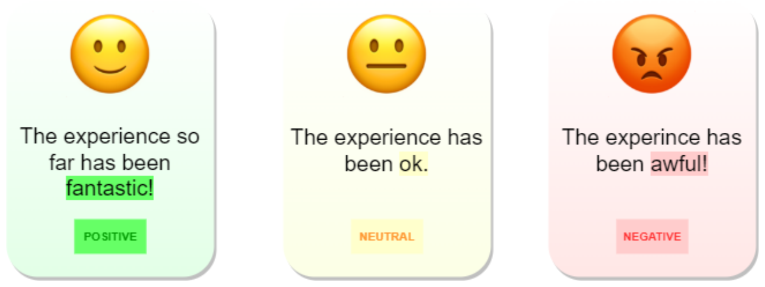

The experience so far has been fantastic!
POSITIVE
The experience has been ok.
NEUTRAL
The experince has been awful!
NEGATIVE

Figure 3.1.2: An example of how a text sentence can be annotated to have a positive, neutral, or negative main sentiment.

Traditional approaches to sentiment analysis relied on lexicon-based methods, where sentiment lexicons or dictionaries were used to assign sentiment scores to words or phrases, for example in [101]. These methods often suffered from limited coverage and failed to capture the complex nuances of sentiment expression.

However, with the advent of machine learning and deep learning techniques, sentiment analysis has witnessed substantial advancements. Machine learning algorithms, such as Support Vector Machines (SVM), Naive Bayes, and logistic regression, have been widely employed for sentiment classification tasks [63]. These algorithms learn to classify text based on features derived from the input, such as word frequencies, n-grams, or syntactic patterns. They can effectively model the relationship between textual features and sentiment labels, enabling accurate sentiment prediction.

On the other hand, deep learning models have shown remarkable success in sentiment analysis by leveraging their ability to capture intricate patterns and contextual information from text data, images, or even music [67] [68]. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), including variants like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU), have been extensively used for sentiment classification. These models can automatically learn hierarchical representations of text, enabling them to capture both local and global dependencies in sentiment expression. With the rise of deep language models, such as RoBERTa, more difficult data domains can be processed, e.g., news texts where authors usually do not express their opinion/sentiment [96].

Evaluating the performance of sentiment analysis models is essential for assessing their effectiveness. Commonly used evaluation metrics include accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC). Additionally, domain-specific evaluation measures, such as aspect-level sentiment analysis or sentiment intensity analysis, can be employed to capture more nuanced sentiment information.

However, sentiment analysis cannot perform accurately when specific language patterns are present in a text sentence. There are various such examples that contain ambiguations such as double negation e.g. "I do not dislike comedies.", sarcasm e.g. "I would really like to go out in this nice weather, I love rain on my clothes." and the use of negative characterisms to express positivity "The movie keeps you on the edge with plenty of unsettling plot twists.".

Researchers have also explored advanced techniques in sentiment analysis, such as aspect-based sentiment analysis, which aims to identify sentiment towards specific aspects or entities mentioned in the text. Furthermore, deep contextualized word representations, such as ELMo (Embeddings from Language Models) and GPT (Generative Pre-trained Transformer), have shown promise in capturing contextual information and improving sentiment analysis performance.

3.1.4 Part-of-speech tagging

Part-of-speech (POS) tagging, also called grammatical tagging, is a fundamental task in Natural Language Processing (NLP) that involves assigning grammatical labels to words in a sentence, indicating their syntactic role and function. The process includes marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition and its context. It plays a crucial role in various NLP applications, such as parsing, named entity recognition, and machine translation, providing essential linguistic information for language understanding and analysis.

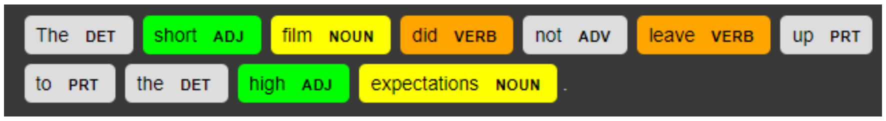


Figure 3.1.3: An example of the part-of-speech tagging of a sentence.

Traditional approaches to POS tagging have employed rule-based methods, stochastic models, and even human annotators. Rule-based methods utilize handcrafted linguistic rules and lexicons to determine appropriate POS tags based on contextual patterns. The first try to implement a major corpus for linguistic analysis was The Brown Corpus [21], where the researchers created a handmade rule list [27] and achieved a 70% accuracy. Stochastic models, such as Hidden Markov Models (HMMs) and Maximum Entropy Markov Models (MEMMs), leverage statistical patterns learned from annotated training data to make POS tag predictions.

In recent years, deep learning models, particularly Recurrent Neural Networks (RNNs) and Transformer-based architectures, have gained significant attention in POS tagging. These models excel at capturing sequential dependencies and contextual information, which are crucial for accurate POS tagging. For instance, the Bidirectional Long Short-Term Memory (BiLSTM) model processes input sequences in both forward and backward directions, allowing it to leverage information from both past and future contexts. It is important to mention that in 2014, a paper using the structure regularization method for part-of-speech tagging [82], achieved 97.36% accuracy on a standard benchmark dataset.

Deep learning models for POS tagging typically require extensive training on large annotated datasets, such as the Penn Treebank [54] or Universal Dependencies [61], which provide word sequences along with their corresponding POS tags. During training, the models learn to predict the most probable POS tags based on contextual information and distributed word representations.

Accurate POS tagging holds significant implications for downstream NLP tasks, facilitating more precise syntactic parsing, improved named entity recognition, and enhanced machine translation systems. It serves as a foundational step in NLP pipelines, contributing to a comprehensive understanding and analysis of natural language text.

3.2 Neural Networks

3.2.1 Basic concepts

Firstly, we explain the basic concepts of a neural network by presenting the architecture of a shallow neural network, which is essential in order to perceive more complex architectures.

In supervised learning, a model is trained using a fixed number of N samples from the training dataset $D = (x_1, y_1), ..., (x_n, y_n)$, where x_i represents the input and y_i represents the corresponding label. The model aims to compute a function $f : X ßY$ that maps the input X to the output Y, and its trainable parameters are often referred to as **weights**. The performance of f is evaluated using a loss function L, which measures the discrepancy between the predicted output and the true label. During training, the objective is to adjust the weights of f in a way that minimizes the loss function [23].

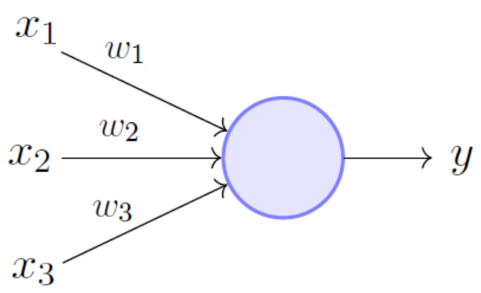


Figure 3.2.1: The shallow neural network of one neuron - Perceptron [74] [56]

In a neural network, the output of each neuron is not solely determined by the weighted sum of the inputs x_i. Instead, an activation function is employed to transform the weighted sum into an output. The choice of **activation function** is crucial as it significantly influences the network's performance. Activation functions can be linear or non-linear, and there are several commonly used examples, including sigmoid, tanh (Hyperbolic tangent), ReLU (Rectified Linear Unit), leaky ReLU, and softmax. The selection of an appropriate activation function depends on the specific problem and the desired behavior of the neural network. Careful consideration should be given to choosing an activation function that enables the network to capture complex patterns and exhibit desirable properties during training and inference. One of the most widely adopted and best-performing activation functions in neural networks is the Rectified Linear Unit (ReLU) and its variants. However, ReLU is not without its limitations, as it entails the "dying ReLU" problem, where some neurons become non-active, leading to decreased model capacity [99]. Variants of ReLU have been developed to address this issue.

The **loss function** or cost function, which was previously mentioned, is utilized to quantify the dissimilarity between the model's predictions and the true values. It maps the output space $Y \times Y$ to a non-negative real number, denoted as $L(y_i, f(x_i))$ for the ith sample. Neural models undergo training iteratively for a specific number of epochs, which can be understood as iterations until they achieve a predefined objective or reach the maximum allowed iterations [23]. Generally, the total loss of the model in each epoch is defined as a normalized average of the cost function computed for each data point in the training set. The goal of training is to discover the optimal parameters that minimize this function, leading to an effective model. The most notable loss functions are *Mean Squared Error (MSE)*, *Mean Absolute Error* or *L1*, and *Cross Entropy Loss*. It is important to add that for complex tasks, custom loss functions are created.

After defining the cost function, the next step is to minimize it through optimization. Gradient-based methods are commonly used for this purpose. These methods rely on the calculation of gradients, which provide information on how to adjust the model's parameters to approach the desired output. By iteratively computing the loss function and its gradients for all training samples, the parameters are updated in the opposite direction of the gradient.

One widely adopted gradient-based method is **Gradient Descent** [98]. It aims to minimize the objective function by updating the model's parameters according to the equation:

$$\theta' = \theta - \epsilon \bigtriangledown \theta L(\theta) \tag{3.2.1}$$

Here, θ represents the model's parameters, ϵ is the learning rate, and $\bigtriangledown \theta L(\theta)$ denotes the gradient of the loss function. The learning rate is a small positive constant chosen during training, and its selection is crucial for the model's performance.

There are several other gradient-based optimizers commonly used in practice, many of which are extensions or variations of Gradient Descent. Stochastic Gradient Descent (SGD) is one popular variant that computes gradients and updates parameters in batches, rather than considering the entire dataset at once. This approach provides a significant speed advantage. Other notable optimizers include *AdaGrad*, *RMSProp*, *AdaDelta*, and *Adam*.

Although the computation of analytical gradients is straightforward, the numerical evaluation can be computationally expensive. To address this, the **back-propagation** algorithm was introduced. It efficiently applies the chain rule to compute gradients in a specific order of operations, utilizing computational graphs and reusing previously computed values. The process of computing outputs and adjusting weights through back-propagation is typically repeated for multiple epochs until the loss function converges or reaches another predefined threshold. After training, the performance of the neural network can be evaluated using various metrics and unseen data that were not encountered during training.

3.2.2 Training

In this subsection, we discuss the key steps involved in **training neural networks**. The training of neural networks involves the process of optimizing the model's parameters in order to minimize a specified loss function.

The initial step of the training encompasses the parameter **initialization** of the neural network, including weights and biases, often initialized with small random values. Proper initialization is critical as it facilitates faster convergence and mitigates the issues of vanishing or exploding gradients during the training process. In fact, Goodfellow et al. even propose 'Parameter initialization strategies' to achieve nearly optimal initialization. After initializing, what follows is the **forward propagation** phase, where input data is fed through the neural network, and computations are executed to generate predictions or outputs. Each layer within the network applies a transformation to the input utilizing its parameters and an activation function. The outputs of one layer serve as inputs to the subsequent layer, effectively propagating information through the network.

Following the generation of predictions, the subsequent step involves the comparison of these predictions with the true labels or target values. This comparison is accomplished through the **calculation of a loss or cost function**, which quantifies the dissimilarity between the predictions and the ground truth. Commonly employed loss functions encompass mean squared error (MSE) for regression tasks and categorical cross-entropy for classification tasks. Then, **back propagation** stands as a fundamental step in training neural networks. It includes the computation of gradients of the loss function with respect to the parameters of the network, employing the chain rule of calculus. The gradients provide insights into the direction and magnitude of the parameter updates necessary for minimizing the loss function. The back-propagation process involves calculating gradients layer-by-layer, commencing from the output layer, and propagating backward through the network.

Once the gradients have been computed, the network's parameters can be updated through an optimization algorithm. Gradient Descent is the most widely adopted algorithm, whereby the parameters are adjusted in the direction opposite to the gradients, scaled by a learning rate. This iterative **parameter update** process is repeated multiple times, gradually reducing the loss and enhancing the model's performance. In this way, the training process typically entails multiple iterations or **epochs**. Each iteration involves passing the entire training dataset through the network and applying the parameter updates. An epoch signifies a complete pass through the entire training dataset. Multiple epochs are employed to facilitate comprehensive learning from the data.

Throughout the training process, it is vital to monitor the model's performance on unseen data to prevent overfitting. A distinct **validation** set is utilized to assess the model's generalization capability and make informed decisions regarding hyperparameter tuning or early stopping. Finally, the trained model undergoes evaluation on a separate **testing** set to gauge its performance on unseen data and ascertain its real-world applicability. By diligently following these steps, neural networks can be effectively trained to extract meaningful insights from data and provide accurate predictions. The training process entails searching for an optimal configuration of the model's parameters that minimizes the loss function and enables reliable generalization to unseen data.

3.2.3 Pre-trained Neural Networks

Pre-trained neural networks have become a valuable asset in the field of deep learning. These networks are pre-trained on large datasets and contain learned weights and parameters that capture knowledge about various features and patterns in the data. By leveraging the knowledge encoded in these pre-trained models, researchers and practitioners can benefit from the expertise of the original model developers and avoid the need to train models from scratch.

One of the key advantages of pre-trained neural networks is their ability to perform well on a wide range of tasks, even with limited training data. The pre-trained models are typically trained on massive datasets, often consisting of millions of examples, allowing them to learn rich representations of the input data. These learned representations can be transferred to new, related tasks, where the pre-trained models can be fine-tuned or used as feature extractors. For example, Dervakos et al. [9] present a method to enhance the pre-trained BERT embeddings using medical-related knowledge and thus show that it is possible to enhance BERT pre-trained models for domain-specific tasks.

A popular approach in utilizing pre-trained models **transfer learning**. In transfer learning, the pre-trained model is first trained on a source task, such as image classification on a large dataset like ImageNet. The knowledge acquired by the model in the source task is then transferred to a target task, which may have a smaller dataset or a different data distribution. By initializing the target task model with the pre-trained weights and fine-tuning the model on the target task data, it is possible to achieve better performance and faster convergence compared to training from scratch. For example, in [55] the authors fine-tune and stack dense convolutional networks pre-trained on ImageNet for the task of art style recognition.

When it comes to natural language processing and its many downstream tasks, pre-trained models have revolutionized the field by providing ready-to-use, robust and effective solutions for various tasks. Pretrained models in NLP are typically trained on large corpora of text data, such as Wikipedia articles, news articles, or web text. These models learn contextual representations of words, phrases, and sentences, capturing semantic and syntactic information. Several popular pre-trained NLP models are BERT, GPT, T5 etc.

One of the biggest advantages of pre-trained models is their availability, which has greatly facilitated research and development in deep learning. Accessing and using pre-trained neural networks typically involves using a **deep learning framework**, such as TensorFlow, PyTorch, or Keras. Then, we can load the model's architecture and weights into our programming environment. After **preprocessing** the input data based on the model's expected format, we can **fine tune** our model, i.e. train it on our target task by updating the weights of some or all of the layers. Otherwise, we can use the model for **feature extraction**, where essentially we remove the last few layers and use the output from earlier layers as input to our task-specific model. After these steps, we can use the model to perform **inference** in our input data, meaning to pass our input data through the model and obtain predictions for our task. Lastly, we can evaluate the performance of the pre-trained model by using metrics and visualizing predictions.

3.3 Explainable Artificial Intelligence (Explainable AI)

Explainable AI (XAI) is a rapidly growing field of AI that seeks to increase transparency and interpretability in machine learning models. It aims to provide a clear understanding of how models make decisions and provide insights into their inner workings. The motivation behind XAI is to increase trust and accountability in automated decision-making systems, particularly in high-stakes applications such as healthcare and finance. Moreover, interpretability helps ensure impartiality in the model's decision, meaning detecting and not promoting potential biases in the training dataset [3].

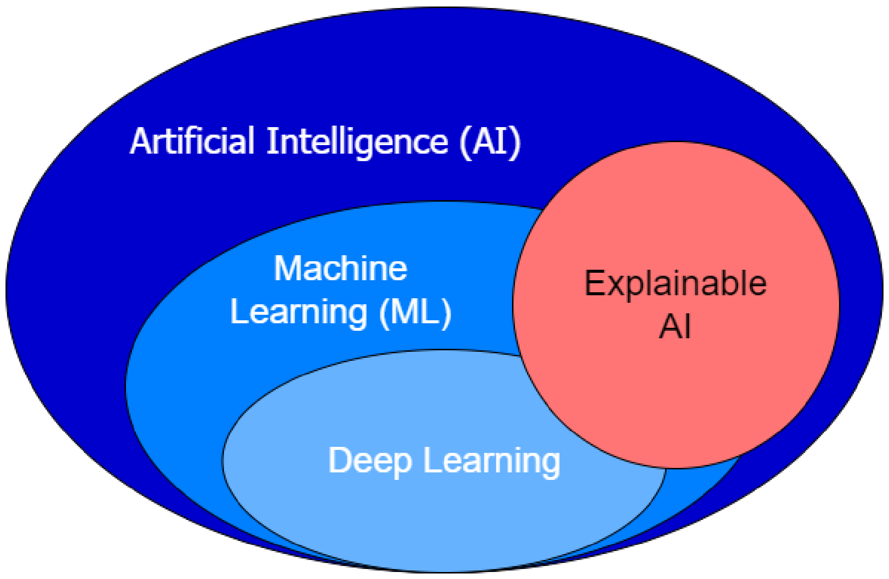


Figure 3.3.1: A Venn diagram that helps to better understand the position of Explainable AI in today's Artificial Intelligence scene.

There are various techniques and tools used in XAI, such as feature importance analysis, saliency maps, and decision trees, among others. These techniques aim to provide users with intuitive explanations of how models arrive at their predictions. They can also be used to identify potential sources of bias and evaluate the fairness of models.

One approach to XAI is to use **model-agnostic methods**, which can be applied to any machine learning model without modification. Examples of such methods include LIME (Local Interpretable Model-Agnostic Explanations) [71], which structures locally linear models around the predictions of an opaque model to explain it, [3] SHAP (Shapley Additive Explanations) [48], where its authors calculate an additive feature importance score for each particular prediction with a set of desirable properties that its antecedents lacked and Anchors which also utilizes local explanations [72]. Other methods, such as [49], propose query-agnostic evaluation methods that decompose and quantify the conceptual differences between ground truth and retrieved instances using adversarial queries. All these methods use perturbation techniques or surrogate models to generate explanations that are understandable to humans.

Another approach is to use **model-specific methods** that are tailored to the specific characteristics of the model. For example, decision trees and rule-based models are inherently interpretable and can provide clear explanations of their decision-making process. Deep learning models, on the other hand, can be more difficult to interpret due to their complex and non-linear nature. To address this, techniques such as Grad-CAM [79] and Integrated Gradients [83] have been proposed. Moreover, there are occasions where the researchers employ rule-based explanations for a machine learning model. For example, in [10] the authors compute rule-based explanations of ML classifiers using knowledge graphs.

Moreover, **counterfactual explanations** another approach to XAI, which is the focus of this thesis, are classified as *example-based approaches* for XAI [7]. Counterfactuals can be either model-agnostic or model-specific. Model-agnostic counterfactuals are designed to work with any machine learning model, without requiring access to the model's inner architecture or parameters. This type of counterfactual explanations offers us flexibility, transparency, and generalizability across models. On the other hand, model-specific counterfactual explanations are tailored to a specific machine learning model and leverage the model's structure, parameters, and internal mechanisms to generate counterfactual instances. These counterfactuals provide us with more detailed insights into the model's behavior but may need modifications to be directly applicable to other models. In the next chapter (4), we discuss counterfactual explanations in detail and provide the basic information needed to capture their importance in this work.

Finally, XAI is a crucial aspect of modern machine learning, particularly in high-stakes applications. The field is rapidly evolving, with new techniques and tools being developed to provide more transparent and interpretable models. The proper use of XAI can definitely lead to a better understanding and evaluation of the underlying data, models, and assumptions.

Chapter 4

Counterfactual Explanations

Explainable AI (XAI) is an emerging field that aims to provide transparency and interpretability to black-box models, which are increasingly being used in decision-making applications. The ability to understand how an AI model arrives at its predictions is crucial for building trust, avoiding bias, and ensuring that the model is making the right decisions for the right reasons[45]. While many methods for generating explanations have been proposed, such as feature importance, decision rules, and local surrogate models[57], counterfactual explanations represent a promising approach that goes beyond justifying a model's outputs[91].

In this thesis, we focus on Counterfactual Explanations, a technique that provides a deeper understanding of a model's decision-making process by constructing hypothetical scenarios in which an input feature or multiple features are altered while holding all other features constant[91]. By observing how the model responds to these changes, we can gain insight into how the model's reasoning is influenced by different inputs, and whether the model is robust and generalizable. Furthermore, counterfactual explanations can be used to test and improve the robustness and fairness of a model, by revealing any hidden biases or areas of uncertainty[91, 45]. As such, counterfactual explanations are an important tool for building trustworthy and responsible AI systems. In the following sections, we will discuss the motivation behind counterfactual explanations and how we can generate and evaluate counterfactual text sentences.

Contents

4.1 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 that contradicts our perception [58].

In order to be more precise and give a better understanding of what a counterfactual is, we can observe an example, which is the following hypothesis: "If I hadn't missed the bus, I would not have been late at work." In this case, we have an event(A), this person was late at work, and a cause(B) for that event, this person missed the bus. To grasp this in "counterfactual logic" we have to imagine a hypothetical case that is contrastive to the original facts, for example, a world in which this person had not missed the bus. Therefore, the name "counterfactuals".

To put the above example in the context of interpretable AI, it is necessary to note that the "event" we described in the example above is a model's prediction for a specific input, and the "causes" are some feature values for that same input that led that model to a specific prediction.

This brings us to the question that a counterfactual intends to answer: "What would have to change for something to be classified as X instead of Y" [19] or in other words, what changes are necessary to our input and consequently in its feature values in order that the model makes a different prediction.

Thus, another definition for a counterfactual can be the following: Given a classifier b that outputs the decision $y = b(x)$ for an instance x, a counterfactual explanation consists of an instance x' such that the decision for b on x' is different from y, i.e., $b(x') \neq y$, and such that the difference between x and x' is minimal. [28]

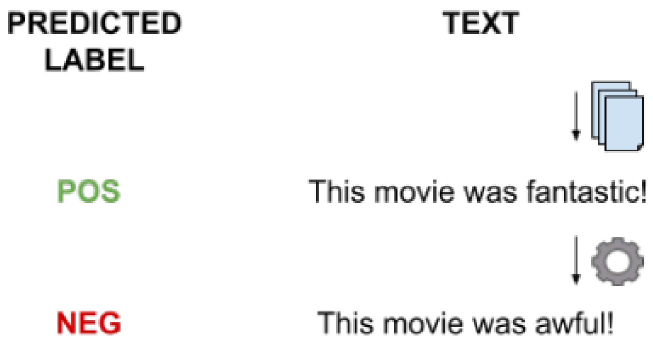


Figure 4.1.1: An example of a counterfactual, showing a change in the input sentence that leads to a different prediction. [20]

We also find importance in answering the question "What is a good counterfactual?". The basic characteristics of counterfactuals are **selectiveness and contrast**. This makes them friendly to the human eye as they target only a small fraction of the feature values to be altered. However, these prerequisites cause counterfactuals to have a broad range and sometimes become contradictory. In trying to solve this issue, different methods can either specify a criterion to select the best counterfactual based on that or select all possible outcomes and let the end user decide which counterfactuals suit them.

In general, the process of evaluating a counterfactual as a good one begins with the end user specifying a desired prediction. Hence, the principal requirement of a good counterfactual is that it generates that preset prediction. It should also provide a minimal explanation, meaning the closest possible to the original instance in terms of the similarity of feature values. Moreover, it is vital to generate numerous counterfactual explanations so that we ensure both diversity and a selection process over multiple options.

4.2 Motivation and Usage

Counterfactual explanations have become an increasingly important topic in natural language processing (NLP) due to their potential to provide greater transparency and **interpretability** for machine learning models [91]. In addition to addressing issues of bias in machine learning models, counterfactual explanations can also be used to improve the user experience of machine learning applications [38], as well as their interpretability and transparency [25].

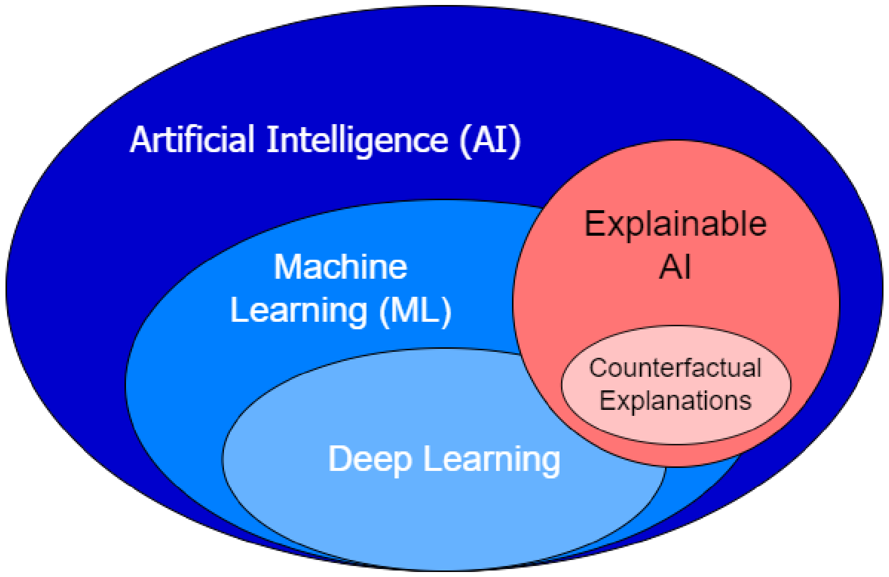


Figure 4.2.1: A Venn diagram showing the use of Counterfactual Explanations in Artificial Intelligence.

One of the key motivations for the use of counterfactual explanations in NLP is to address **issues of bias** in machine learning models. As noted by Wachter et al. [91], machine learning models can exhibit various forms of bias, including both explicit and implicit biases. Counterfactual explanations can help to identify these biases by enabling users to explore how the model would behave if the input were different.

For example, in a study by Kusner et al. in 2017 [40] the authors used counterfactual explanations to identify and mitigate indirect influence bias in several machine learning models. **Indirect influence bias** occurs when a model uses features that are correlated with sensitive attributes, such as race or gender, to make decisions. The study compares several models for predicting the future academic performance of law students. In this example, they successfully demonstrate that the model which uses race and sex as features is not fair. On the other hand, the two models proposed by the authors, which use counterfactual examples with no sensitive features, achieve fairness as they present a very weak causal link between sex and GPA.

Another motivation for the use of counterfactuals is to improve the user experience of machine learning applications. Kaur et al. [38] used counterfactual explanations to explain the reasoning behind the responses provided by a conversational chatbot. By offering alternative responses based on counterfactual examples, the chatbot was able to provide more personalized and effective responses to users.

A primary application of counterfactual explanations is in improving the interpretability and **transparency of machine learning models**. According to Goyal et al. [25], counterfactual explanations provide a way to explain how a machine learning model arrived at a particular decision. The lack of transparency in complex models like deep neural networks has led to their black-box nature, which has resulted in a lack of trust from end users. Thus, by generating counterfactual examples, users can explore how changes to the input affect the model's output, providing greater transparency and understanding of the model's underlying reasoning.

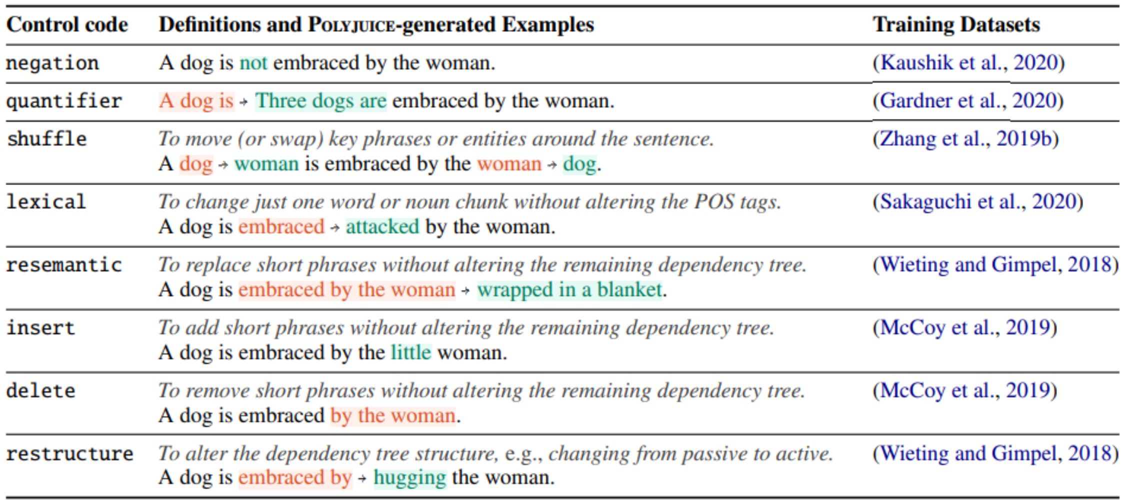
Counterfactuals have also been used in various NLP tasks such as sentiment analysis, text classification, and machine translation. In **sentiment analysis**, counterfactual explanations have been used to explain why a particular text was classified as positive or negative. For example, one approach is to generate counterfactual explanations by modifying the input text and seeing how the model's output changes.

In **text classification**, counterfactual explanations can be used to explain why a particular document was classified as belonging to a particular class. For example, in the framework GYC proposed by Madaan et al. [52] the authors use multiple models to generate counterfactuals that change the predicted class of a sentence to a specific "target" class given as input. Moreover, counterfactuals can also be applied in **machine translation** to explain why a particular translation was generated. For example, in a translation from English to Spanish, a counterfactual explanation can help explain why a particular word was translated in a particular way. This can be useful for language learners who want to understand the many nuances of a language.

Model: Topic Classifier
Source Topic: World, Target Topic: Sci-Fi
Input Sentence: The country is at war with terrorism.
Counterfactual Text Samples:
[1] The country is at war with piracy at international waters.
[2] The country is at war with its own beurocracy.
[3] The country is at war with piracy offenses.

Figure 4.2.2: An example of counterfactuals generated by GYC [52] for text classification

Counterfactuals can also be used in **data augmentation**, where they are generated to create new training examples that help improve the robustness and fairness of machine learning models. By generating counterfactual examples that represent plausible alternative scenarios, the model can learn to generalize better and make more fair and accurate predictions. This approach has been applied to various domains, including image recognition, natural language processing, and healthcare. It can also help mitigate issues related to data imbalance, where certain groups may be underrepresented in the training data.



Control code	Definitions and Polyjuice-generated Examples	Training Datasets
negation	A dog is not embraced by the woman.	(Kaushik et al., 2020)
quantifier	A dog is → Three dogs are embraced by the woman.	(Gardner et al., 2020)
shuffle	*To move (or swap) key phrases or entities around the sentence.* A dog → woman is embraced by the woman → dog.	(Zhang et al., 2019b)
lexical	*To change just one word or noun chunk without altering the POS tags.* A dog is embraced → attacked by the woman.	(Sakaguchi et al., 2020)
resemantic	*To replace short phrases without altering the remaining dependency tree.* A dog is embraced by the woman → wrapped in a blanket.	(Wieting and Gimpel, 2018)
insert	*To add short phrases without altering the remaining dependency tree.* A dog is embraced by the little woman.	(McCoy et al., 2019)
delete	*To remove short phrases without altering the remaining dependency tree.* A dog is embraced by the woman.	(McCoy et al., 2019)
restructure	*To alter the dependency tree structure, e.g., changing from passive to active.* A dog is embraced by → hugging the woman.	(Wieting and Gimpel, 2018)

Figure 4.2.3: Examples of different counterfactual examples using the different control codes of the counterfactual editor *Polyjuice*. The figure is obtained from the relevant paper. [102]

Although in this thesis we center around text counterfactual explanations, counterfactuals can also be used for image data. One approach for the generation of **image counterfactuals** is optimizing a perturbation to the input image that causes the model's output to change to a desired class. This approach has been used to generate counterfactual explanations for a variety of image-based machine learning applications, including image classification, object detection, and segmentation. Another method used recently is generative adversarial networks (GANs) to generate counterfactual images that are similar to the input image but with different features that lead to a different classification outcome. Moreover, a recent field of research is the evaluation of generative methods on images. For example, Lymperaiou et al. [50] propose a conceptual model-agnostic evaluation method that explains by design which concepts affect the images generated.

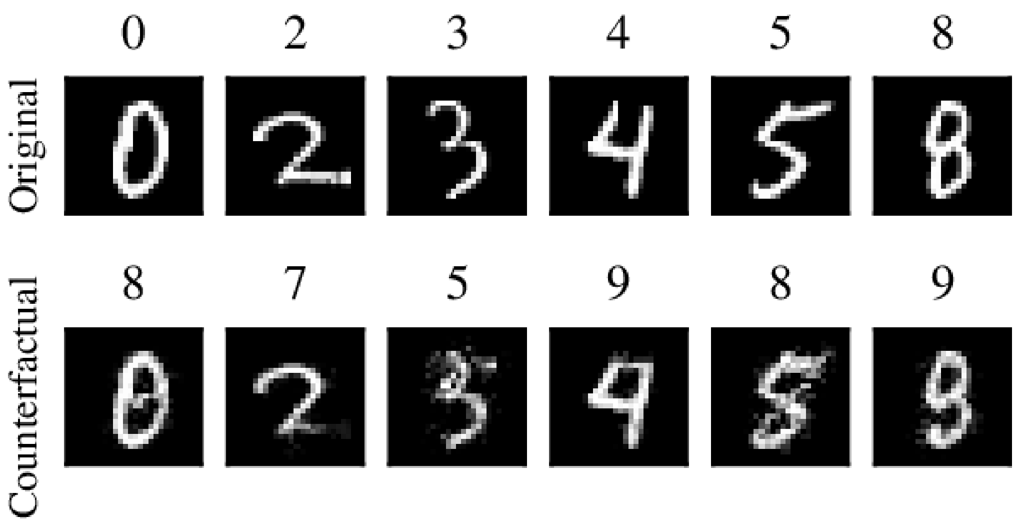


Figure 4.2.4: Examples of counterfactual images on the MNIST dataset of handwritten digits from [47] showing that the predicted class shifted from the original prediction.

Furthermore, counterfactual explanations show great interest when combines with graphs. For example, in [11] the authors propose an algorithm that provides counterfactual explanations in terms of knowledge graphs.

In summary, counterfactual explanations offer a powerful tool for improving the accuracy, interpretability, and trustworthiness of machine learning models in NLP applications. They have the potential to transform the way we use and interact with NLP applications by providing a clearer understanding of how machine learning models work and how they can be improved.

4.3 Counterfactual Editors

In this thesis, we explore systems that aim to minimally edit a given text instance in order to change the prediction of a classifier. From now on, we will refer to such systems as *counterfactual editors* or simply *editors*. In this section, we demonstrate a general overview of the counterfactual editors that we explore and the architecture they use, and we also categorize some counterfactual editors based on different criteria.

Firstly, we must underline that counterfactual editors use different methodologies to reach the desired outcome, as they usually have different use cases. For example, there are editors that attempt to generate edits that are directly dependent on the output of a specific predictor, f() by masking words in a text input and replacing the masks optimally in order that the output of f() changes. Such editors are **MiCE** [75] and DoCoGen [4].

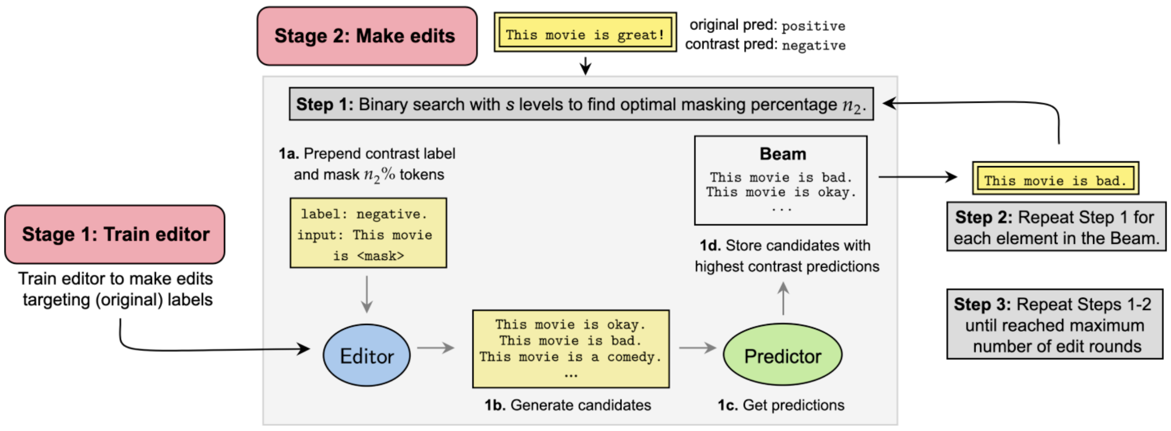

Stage 2: Make edits
This movie is great!
original pred: positive
contrast pred: negative
Step 1: Binary search with s levels to find optimal masking percentage n_2.
1a. Prepend contrast label and mask n_2% tokens
label: negative. input: This movie is <mask>
Beam
This movie is bad. This movie is okay. ...
This movie is bad.
Step 2: Repeat Step 1 for each element in the Beam.
Stage 1: Train editor
Train editor to make edits targeting (original) labels
1d. Store candidates with highest contrast predictions
Step 3: Repeat Steps 1-2 until reached maximum number of edit rounds
Editor
This movie is okay. This movie is bad. This movie is a comedy. ...
Predictor
1b. Generate candidates
1c. Get predictions

Figure 4.3.1: An overview of *MiCE*, where In Stage 1 the editor is trained to make edits targeting specific predictions from the predictor and in Stage 2 contrastive edits are generated by the editor model from Stage 1 such that the predictor changes its output to the contrast prediction. The figure is from the relevant paper [75].

On the other side, there are editors such as **Polyjuice** [102] that attempt to create generic text perturbations that alter the semantics of a sentence, without depending on a predictor. This type of counterfactuals can be multipurpose as they can be used for data-augmentation tasks or for conditionally generated counterfactuals to a specific dataset or task.

Another implementation of an editor is **Tailor** [77]. The authors present a method for generating and manipulating counterfactuals with fine-grained control over their semantics. Tailor introduces a novel perturbation technique, called "control perturbation", which allows the user to modify specific aspects of the generated text while preserving its semantic properties. The perturbation process is guided by the same semantic controls used in text generation, which ensures that the modified text remains consistent with the desired semantics.

Moreover, a plethora of editors is intended for generating *adversarial examples*, which aim at identifying and exposing the vulnerabilities of a classifier. These models, also called *adversarial models*, usually do not aim to create fluent, grammatically correct text edits of the original input, so we might encounter noise, word repetition, or other structural issues in the text generated. A framework that encapsulates many such implementations is TextAttack [60], which includes many adversarial generators such as **TextFooler** [35], HotFlip [15] and Bert-Attack [43]. These are simpler methods than the previously mentioned counterfactual editors that use various techniques to generate examples, such as gradient descent word swap or counter-fitted word embedding swap.

Original
Perfect performance by the actor $\rightarrow$ Positive (99%)
Adversarial
Spotless performance by the actor $\rightarrow$ Negative (100%)

Table 4.1: Adversarial example generated using TextFooler [35] for a BERT-based sentiment classifier. Swapping "perfect" with its synonym "spotless" completely changes the model's prediction, even though the underlying meaning of the text has not changed. [60]

In addition, there are editors that, instead of attempting to generate random permutations of a sentence, use mechanisms to alter only the important features of the sentence. This "importance" can be evaluated in multiple ways, among which are: using the predictor's attention[75], and training a classifier to retrieve the correlation between each separate term and the task.[93], calculating the impact of feature deletion on the prediction of the classifier [35]. After that procedure, the masked-important terms can be infilled with synonyms, antonyms, and related words from other tasks or by using pre-trained seq2seq models. [75] [102] [18]

4.4 Evaluation of Counterfactual Editors

Despite the fact that there are no global methods established for evaluating the performance of a counterfactual editor, there are several ground truths and common metrics on which we can compare editors and how efficiently they operate.

One significant metric, which is used by editors that depend on a predictor for their output, is the *flip rate*. Flip rate also referred to as validity, fidelity, or attack success rate is a criterion for evaluating editors based on how often the output of a predictor is flipped to the desired class. A definition for the flip rate is:

$$flip\_rate = \frac{edits\ with\ successful\ flip\ to\ the\ desired\ class}{number\ of\ inputs\ to\ the\ editor}$$

This metric is used among others in MiCE, which depends on a predictor for the generated edits, in TextAttack and in our experiments which will be detailed in 7.

Another category of metrics, widely used in natural language processing, is distance metrics. Distance metrics are a rough, but reasonable, proxy for the overall performance of a counterfactual method. [39]. The word level Levenshtein edit distance [42] also referred to as *minimality* in MiCE, closeness in Polyjuice, and edit distance in CAT is a very common way to estimate how minimal are the changes that an editor makes to the input. The Levenshtein edit distance between two strings a, b (of length |a| and |b| respectively) is given by *lev(a, b)* and can be defined as:

$$lev(a,b) = \begin{cases} |a| & \text{if } b = 0 \\ |b| & \text{if } a = 0 \\ lev(\text{tail}(a), \text{tail}(b)) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} lev(\text{tail}(a), b) \\ lev(a, \text{tail}(b)) \\ lev(\text{tail}(a), \text{tail}(b)) \end{cases} & \text{otherwise} \end{cases}$$

, where the tail of some string x is a string of all but the first character of x, and x[n] is the nth character of the string x, counting from 0. [95]

In most papers and studies, the authors use the normalized format of the Levenshtein distance, which is on the scale [0,1] and can be defined as:

$$normalized\_lev(a,b) = \frac{lev(a,b)}{|a|}$$

We will use the normalized Levenshtein distance in our experiments in 7 in the form of the metric *minimality* and show its practical importance.

Moreover, there are several criteria that we observe in order to assess the quality of the generated text by an editor. One of them is the quality of the generated edits or in other words the fluency of the sequences that are generated. A fluent counterfactual can be defined as grammatically correct [60] and semantically meaningful (e.g. “Colorless green ideas sleep furiously” is not meaningful [6])[102].

In order to avoid human assessment, there are multiple metrics used in the bibliography to achieve that type of evaluation. For example, in MiCE, Polyjuice, and CAT [5] the metric used to assess the quality of the generated counterfactuals is *fluency*. Similar metrics include *grammaticality* and *semantics* as defined in TextAttack [60] and *perceptibility* in counterfactualGAN [73]. All of these metrics use some language model to compare masked language model loss between the original and edited text or to compute similarity in semantics between the original and the edited text.

However, identifying counterfactuals that are both fluent and within distribution can be a hard task because the real distribution of texts $\mathcal{L}$ is usually inaccessible, and thus fluency is difficult to evaluate systematically. A usual proxy used for fluency estimation is token-level *perplexity* of a large language model. This method has been applied in several NLP tasks [86] [36] [92] and it requires a language model $\mathcal{M}_{\mathcal{D}}$ trained on a large dataset $\mathcal{D}$ and the computation of the averaged perplexity over a given text sequence $x = x_1, x_2, ..., x_N$ as follows:

$$PPL(x) = exp\left\{ \frac{1}{T} \sum_{t=1}^{T} \log p_{\mathcal{M}_{\mathcal{D}}}(x_t | x_{1:t-1}) \right\}.$$

We must underline that for the fluency estimation, our language model $\mathcal{M}_{\mathcal{D}}$ is not fine-tuned on $\mathcal{L}$. If we assume that $\mathcal{L}$ is accessible, we can fine-tune $\mathcal{M}_{\mathcal{D}}$ on it, acquiring a new model $\mathcal{M}_{\mathcal{L}}$. We can then use $\mathcal{M}_{\mathcal{L}}$ to detect out-of-distribution cases [2] using the same PPL formula as shown in equation 3, and in this case, we can argue that the probability predicted over the tokens by $\mathcal{M}_{\mathcal{L}}$ represent the probability under the distribution of $\mathcal{L}$. This method of detecting and counting out-of-distribution cases out of the total ones generated is also employed by Keane et al. [39] in the form of *coverage*. In our experiments, we will employ both PPL over $\mathcal{M}_{\mathcal{D}}$ ($PPL_{\mathcal{D}}$) and PPL over $\mathcal{M}_{\mathcal{L}}$ ($PPL_{\mathcal{L}}$).

There are several other criteria that have been applied for the evaluation of counterfactuals, such as *diversity* employed in GYC [52] that ensures diversity in the generated counterfactual text samples by introducing diversity loss, *closeness* [90] or *relative distance* [39] to the training data which is calculated by comparing counterfactuals to instances from the original data and also *sparcity* [39] that refers to the number of features being changed.

Finally, more recent studies approach the problem of evaluating counterfactual explanations from a more "human" perspective by assessing the performance of editors based on how much the explanations can assist a human [84] [14] or a student model [66] to learn to imitate a black-box teacher model. These methods contribute to the evaluation of the informativeness of the explanations. For example, Treviso and Martins [84] introduce the *communication success rate* as a quantifiable measure of explainability.

In this thesis, we focus only on the automated metrics that do not depend on human input or external data and are most commonly used in editor evaluation, more significantly, minimality, flip-rate, and fluency.

Chapter 5

Proposal

In this section, we propose a new method of evaluating and analyzing counterfactual editors based on the work by Filandrianos et al. [20]. Our method is based on the concept of generating what we call *counterfactuals of counterfactuals*, which we further described in 5.2.1. Moreover, in this work, we evaluate editors to see how the vital parts of the architecture of each editor perform, in order to better understand if they operate efficiently.

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

Contents

5.1 Contributions

The contributions of this dissertation are multiple and can be summarized as follows:

- We generate counterfactual explanations using multiple counterfactual editors, that are very relevant in recent literature. Therefore, we explore the advantages and weaknesses that each one of them poses for future research. We cast our focus on evaluating and explaining the generated counterfactuals with automated metrics and qualitative criteria.
- We step upon a novel approach for generating counterfactuals, the method of Counterfactuals of counterfactuals, and the evaluation metric called *inconsistency*, both introduced by Filandrianos et al. [20]. Our work extends the proposed counterfactual generation framework significantly, as we experiment with more editor scenarios and generation methods, and utilize it for conducting a more thorough analysis.
- We introduce a simple yet powerful way to intervene in the semantics of a counterfactual edit by using part-of-speech tagging. We assess this method and withdraw valuable conclusions regarding constrained counterfactual generation.
- The comparative results extracted from this study help explain more aspects of the decisions of some methods and models that are widely used in counterfactual edits generation.

5.2 Proposed methods

In this section, we enumerate the proposed methods of evaluation and the approaches used for counterfactual generation. We describe in detail the method of Counterfactuals of counterfactuals, the inconsistency metric, the counterfactual generation based on part-of-speech tagging, and the modifications we can make in the structural components of the editors in order to identify which methods optimize their performance.

5.2.1 Counterfactuals of counterfactuals

Counterfactual editors typically generate edits by having the input perturbed in a one-step approach with the end-user or system utilizing the output as what we call a counterfactual. However, in this section, we discuss a method called *Counterfactuals of counterfactuals*, a **back-translation** inspired approach [20] as it is called by the authors, where the output of a counterfactual editor is resupplied as a new input to the editor.

To make the concept of Counterfactuals of counterfactuals understandable we first need to formalize the problem of counterfactual generation. Let us assume access to a classifier g such that g: $\mathcal{L} \rightarrow [0,1]^{\mathcal{C}}$, where $\mathcal{L}$ is the set of text for a specific language and $\mathcal{C}$ is the number of different classes. Then, we define the counterfactual editors for g as functions $f : \mathcal{L} \rightarrow \mathcal{L}$, and we assume that the editor f has a threefold goal [20]:

1. The generated text is classified into a different class:

$$arg\ max\ g(f(x)) \neq arg\ max\ g(x)$$

2. The edits are minimal with respect to some distance metric $d : f = arg\ min_{h \in \mathcal{F}} d(x, h(x))$, where $\mathcal{F}$ is the set of functions for which $arg\ max\ g(f(x)) \neq arg\ max\ g(x)$.
3. The edited text $f(x)$ is fluent and within the distribution of $\mathcal{L}$.

In order to evaluate the degree to which these conditions are valid, we examine the behavior of editors when they are iteratively fed back with their generated output, meaning that we are exploring the function $f(f(f(...f(x))))$, and assessing the three criteria we described above after n iterations of the editor.

To be more precise, we use a novel evaluation metric, **inconsistency**, to quantitatively evaluate the second criterion based on the back translation approach, and then we analyze how the other two criteria can be more carefully examined by utilizing automated performance metrics after n feedback iterations. This feedback technique and these evaluation methods are then used with all the different variations we examine for every editor.

Counterfactuals of counterfactuals prove to be a very significant tool for generating counterfactual edits, as opposed to the traditional one-step generation, it helps us evaluate the editor in **non-dataset dependent** content and monitor the editor's sensitivity to minor or major changes in an input sentence. Moreover, they help us unveil faulty patterns of the editors and assist us to explain them.

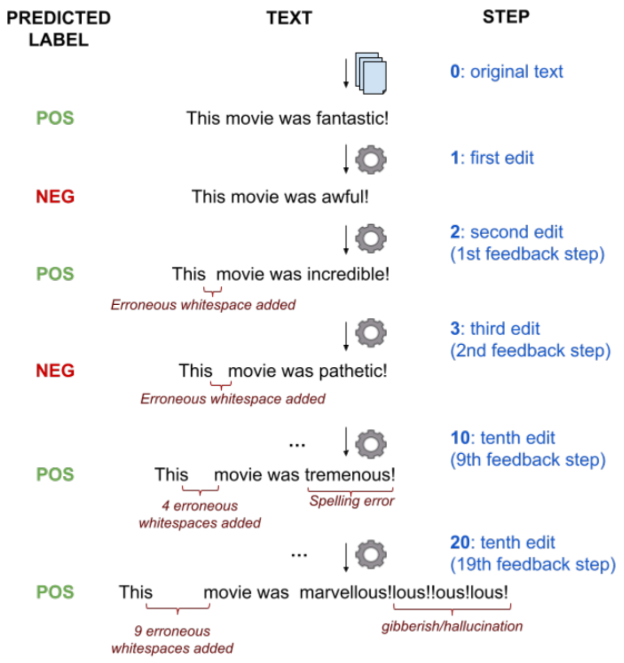

PREDICTED LABEL
TEXT
STEP
0: original text
POS
This movie was fantastic!
1: first edit
NEG
This movie was awful!
2: second edit (1st feedback step)
POS
This movie was incredible!
Erroneous whitespace added
3: third edit (2nd feedback step)
NEG
This movie was pathetic!
Erroneous whitespace added
...
10: tenth edit (9th feedback step)
POS
This   movie was tremenous!
4 erroneous whitespaces added
Spelling error
...
20: tenth edit (19th feedback step)
POS
This   movie was marvellous!lous!!ous!lous!
9 erroneous whitespaces added
gibberish/hallucination

Figure 5.2.1: Example of the back-translation framework we use to feed back the edited text to the editor. We see the progression of edits(centre) and the predicted labels(left) through the iteration steps(right). With the increase in feedback steps, we notice erroneous behavior in the edits. The figure is from [20].

5.2.2 Inconsistency of minimality

Inconsistency of minimality or simply *inconsistency* is a novel metric introduced in [20] which is tightly bound with the concept of Counterfactuals of counterfactuals. In our experiments, inconsistency constitutes the primary metric for the evaluation of the counterfactual editors.

In order to clarify what inconsistency measures we first need to explain some concepts about minimal edits which will then lead us to the definition of inconsistency. Assuming that the generated edits of an editor are minimal, if a sentence A is edited into a new sentence B and their distance is $d(A, B)$, then sentence B is fed back to the editor to generate another sentence C for which the distance $d(B, C) \leq d(A, B)$, because if that does not stand then C is not the result of a minimal edit. There are several requirements for the inequality mentioned above : (a) we know that A exists, (b) we assume all textual edits can be reversed, so A is reachable from B and (c) d is symmetric, meaning $d(A, B) = d(B, A)$ [20]. Therefore, A can be used as our basis, or in other words as ground truth, in order that we can compare it with C.

As a result, we can measure how consistent a counterfactual editor is with respect to a given distance metric d (e.g. Levenshtein distance, embedding cosine similarity, etc.) by iteratively giving us input the edited text to the editor and measuring the change in the value of d. In greater detail, given an editor $f : \mathcal{L} \to \mathcal{L}$, a text $x \in \mathcal{L}$ and a distance $d : \mathcal{L} \times \mathcal{L} \to \mathbb{R}_+$ we define the inconsistency of f with respect to d, for x as [20]:

$$inc(f, x) = relu[d(f(f(x)), f(x)) - d(f(x), x)]$$

The difference of the distances $d(f(f(x)), f(x)) - d(f(x), x)$ shows us how the distance d fluctuates between consecutive iterations on the editor, f and the $relu$ function is only affected by the increase of the distance.

As the authors state [20], when the value of inconsistency is positive it is certain that a better set of edits exists, specifically the one of the previous step. On the other hand, a negative value should be concerning, as it indicates probably a set of edits that is not good. For example, it may denote that there were not enough changes made by the editor or that a better, more minimal set of edits was found. Equation 5 only counts the difference in distance between consecutive steps, but we can continue to feed back the generated edits to the editor itself and compute the inconsistency at step n of the process to gain more information on the editor's inconsistency. Thus, the inconsistency at step n is measured as follows:

$$inc@n(f,x) = \frac{1}{n}\sum_{i=0}^{n-1} inc(f_{i+1}(x), f_i(x))$$

where $f_0(x) = x$ and $f_i(x) = f(f_{i-1}(x))$.

5.2.3 Generating counterfactuals with part-of-speech tagging

Generating counterfactuals is a task where we want to achieve minimality so that the generated edits are as close to the original input as possible. There are several ways to achieve that, but one way is to reduce the number of candidate tokens and phrases for modification. By generating counterfactuals with the help of part-of-speech (POS) tagging, we only target specific POS tags and therefore the search space for changes in the sentence is significantly reduced. Moreover, POS tagging can assist in generating more accurate and fluent counterfactual edits as the context and meaning of each masked word are more specific if we refer to a single part-of-speech tag.

The method we use for counterfactual generation in order to leverage the part-of-speech tagging task of NLP is to **target a specific POS tag** and then mask only the word tokens that belong in this specific POS tag group. Therefore, we implement a complementary component to the architecture of the editors that works as a filter and serves the purpose we describe above. Thus, after the tokenization of the sentence and after the filter process, the word tokens with the target POS tag are candidates to be masked by the masker that each editor uses. Then, each counterfactual editor works as a black box generating edits without us intervening in the generation process.

A formalization of the above problem is as follows. Let S be the input sentence and $TARGET$ be the targeted part-of-speech tag. Then, $tok(S)$ represents the tokenization process applied to an input sentence S that generates a list of tokens $T = [t_1, t_2, ..., t_n]$, where n is the number of tokens. We use a function $pos(t)$ that returns the part-of-speech tag of a token t, and generate a new list of tokens T' selected from T, which will contain only the tokens with the $TARGET$ POS tag, we formulate this as following:

$$T' = [t_i | t_i \in T, pos(t_i) = TARGET]$$

Then, $masker(T', S)$ is the function that replaces all the tokens in input sentence S that are in T' with a mask token to create a modified masked sentence M.

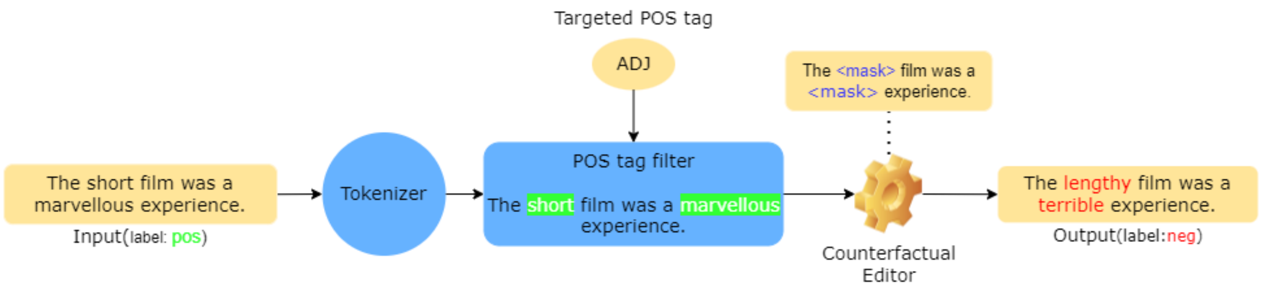

Targeted POS tag
ADJ
The <mask> film was a <mask> experience.
The short film was a marvellous experience.
Input(label: pos)
Tokenizer
POS tag filter
The short film was a marvellous experience.
Counterfactual Editor
The lengthy film was a terrible experience.
Output(label:neg)

Figure 5.2.2: An example explaining the process of counterfactual generation with a targeted part-of-speech tag. We notice that the targeted POS tag in the example is 'ADJ', i.e. adjective, and therefore the word tokens that are adjectives are targeted for modification. In the output of the editor, we observe that the counterfactual edit only contains changes for these word tokens.

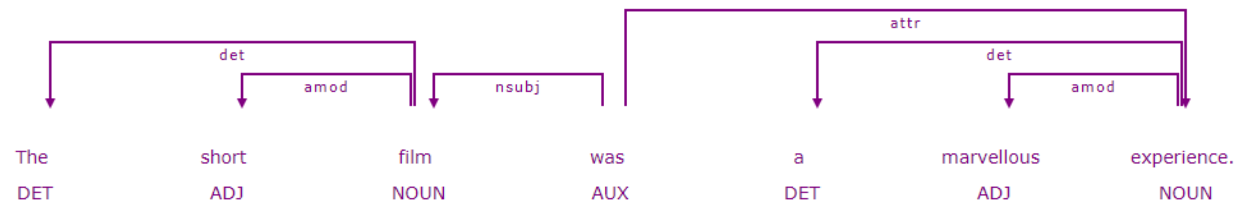


Figure 5.2.3: A visualization of the dependency parse of the input sentence in the example in figure 6.3.3. We observe that the POS tag of each word token is under the word, thus helping us identify which POS tags are candidates for modification.

5.2.4 Comparing editors based on different structural components

Counterfactual generation editors feature multiple structural parts which are combined in order to generate as accurate counterfactual edits as possible. In this thesis, we seek to evaluate and compare different editors that utilize different models and methods (e.g. for masking), in order to gain insights into their performance and see how these crucial components affect their efficacy in generating high-quality counterfactual text.

By using **various evaluation metrics**, which enable us to examine the problem from multiple perspectives, we aim to expose the capabilities and limitations of each editor and interpret the reasons behind their decisions. Some of the parts of the editor that we focus on are the underlying model architecture, the masking method employed as well as the text generation algorithm (e.g. beam-search versus multinomial sampling in 7.2.2). By systematically comparing the editors based on these components, we can assess their effectiveness in preserving the contextual integrity of the text, their ability to introduce plausible counterfactual variations, and their overall fluency.

This analysis can contribute largely to the community, as it helps us identify the strengths and weaknesses of multiple text generation methods, and thus facilitate future research and the development of more robust counterfactual editors based on the use case.

Chapter 6

Implementation of our experimental setup

In this section, we describe in detail the experimental setup that we use in order to conduct our experiments. Firstly, we present some preliminary information about the datasets, the editors, and other main components that are utilized. Then, we explain how we formed the experimental setup used in our experiments, including information about the implementation of the structural parts of the editors and how we modified some of them. Lastly, we describe some of the main technologies used in our work and how they contribute to it.

Contents

6.1 Technologies and systems used

In this segment, we overview the technical aspects needed to set up our counterfactual editing system, from programming languages and libraries to the systems used.

6.1.1 Python

The main programming language we use in all of the code studied, used, and written for the editors, data processing, and edits evaluation is *Python*. Python is a powerful and widely used computer language and is essential in machine learning and text generation tasks.

First, Python boasts extensive **Machine Learning** libraries including TensorFlow, **PyTorch**, and scikit-learn, which provide strong tools and frameworks for constructing and deploying machine learning models. These libraries include multiple methods, from classical models to deep learning architectures, making sophisticated machine learning pipelines for text creation jobs easy to deploy. Moreover, Python excels at **data manipulation and preprocessing tasks**, which are critical for machine learning and text production. *NumPy* and *pandas* libraries provide efficient data structures and operations for data manipulation, cleaning, and analysis. This enables us to efficiently process and prepare text data for modeling, including tasks like tokenization, stemming, and vectorization.

Furthermore, Python provides several powerful **NLP libraries** that greatly assist in text generation. Libraries such as *NLTK* (Natural Language Toolkit), *Spacy*, and *Gensim* offer a wide range of functionalities, including part-of-speech tagging, named entity recognition, topic modeling, and language modeling. These tools help us simplify the complex tasks involved in understanding and generating text, enhancing the efficiency and effectiveness of text-generation models. Finally, Python supports a variety of interactive development environments and **integrated development environments (IDEs)** that provide a smooth coding experience. IDEs like Jupyter Notebook and online IDEs like Kaggle Notebooks and Google Colaboratory offer features like code autocompletion, interactive debugging, and data visualization, making it easier to experiment, iterate, and debug machine learning and text generation models.

PyTorch

PyTorch is a popular deep learning framework [64], initially released by Facebook's AI Research (FAIR) lab in October 2016, that plays a significant role in enabling **text generation** with transformers. It is a library that provides a flexible and intuitive platform for building transformer-based models, as well as extensive support for various transformer models, such as GPT2. It also contains pre-trained model weights, tokenizers, and evaluation utilities, simplifying the process of integrating these models into text generation pipelines. PyTorch's efficiency and support for **GPU acceleration** also enable faster training and inference, crucial for complex text generation tasks.

Additionally, PyTorch provides powerful tools and libraries for efficient text preprocessing, tokenization, and batching, such as torchtext and transformers. These libraries handle tasks such as tokenizing text into subword units, building vocabularies, and batching data for training and inference. PyTorch also empowers text generation with transformers by providing a versatile development platform with efficient preprocessing, training, and fine-tuning capabilities, and also seamless **integration with language generation pipelines**. For instance, it can be combined with techniques such as beam search, nucleus sampling, or temperature scaling to enhance the diversity and quality of the generated text [34].

Spacy

Spacy is a powerful natural language processing library that seamlessly assists in **part-of-speech (POS) tagging**, a fundamental task in language analysis and our experiments. With its robust linguistic models and efficient processing capabilities, Spacy simplifies the process of automatically assigning the appropriate part-of-speech tags to words in a given text. Its pre-trained models are trained on extensive labeled data, enabling accurate and reliable POS tagging for a wide range of languages. Spacy's POS tagging capabilities allow us to access the basic yet important noun, verb, and adjective classifications. Additionally, Spacy's efficient tokenization and dependency parsing algorithms ensure accurate POS tagging by considering the contextual relationships between words [31]. All in all, with Spacy, we are able to effortlessly incorporate

POS tagging into our counterfactual generation pipeline, enabling deeper analysis and understanding of text data.

6.1.2 GPU accelerated environments

ARIS High-Performance Computer (ARIS HPC)

ARIS is the name of the Greek supercomputer, deployed and operated by GRNET S.A. (National Infrastructures for Research and Technology S.A.) in Athens. ARIS consists of 532 computational nodes separated into four "islands" of nodes. For our experiments, we used GPU-accelerated nodes that operated with the high-performance NVIDIA Tesla k40m GPU by NVIDIA.

The use of the hardware provided by the ARIS HPC was vital for this thesis and our experiments as it provided us with **multiple graphic processing units (GPUs)** and allowed parallelism, in the sense of executing different experiments simultaneously, each one in a dedicated GPU. ARIS also provided the necessary storage space that was necessary in order to store our models and parameters but also the results of our experiments.

Kaggle and Google Colaboratory

Kaggle and Google Colaboratory (Colab) provide invaluable assistance to machine learning tasks, particularly when it comes to **leveraging GPUs** (Graphics Processing Units) for accelerated computation.

Kaggle is a popular platform for data science and machine learning. It offers an environment where users can access a vast array of datasets, notebooks, and tools to solve real-world problems. Kaggle provides GPU resources that significantly speed up computationally intensive tasks. Using all of these tools, we were able to conduct our first experiments and test how the editors and models behave, obtain some first results, and generally use the platform to build the foundations of our code implementations [37]. Moreover, we were able to create multiple datasets with our counterfactual edits and results which are easily reusable for further analysis. Similarly, Google Colaboratory, commonly known as Colab, is a cloud-based Jupyter notebook environment that also provides free access to GPU resources. Colab notebooks also allowed us to conduct some first experiments by leveraging Colab's GPU support.

In general, both Kaggle and Colab offer user-friendly interfaces that simplify the setup, configuration, and use of GPU resources. Users in this way can easily have access to GPU-enabled environments, eliminating the need for local hardware upgrades or complex configurations. For us, it provided a necessary preparation step before moving to the resources of a hypercomputer.

6.2 Preliminaries

6.2.1 Datasets

For our experiments, we used two English-language datasets: IMDb [51] and 20 Newsgroups[41].

IMDd

The **IMDb** dataset is a widely used benchmark in natural language processing and machine learning research. The dataset covers a wide range of movies and genres ensuring diversity and has been carefully annotated by human annotators, ensuring labels that can be safely used for supervised learning tasks. It is also a very commonly used dataset in the task of Counterfactuals Generation as it is also used in the editors: MiCE, Polyjuice, TextAttack, and Textfooler among others. The dataset consists of 50,000 movie reviews, split into train and test sets, where each review is labeled as positive or negative depending on the sentiment expressed in the text. That being said, the dataset is widely utilized in the task of sentiment classification and namely the *binary sentiment classification* task.

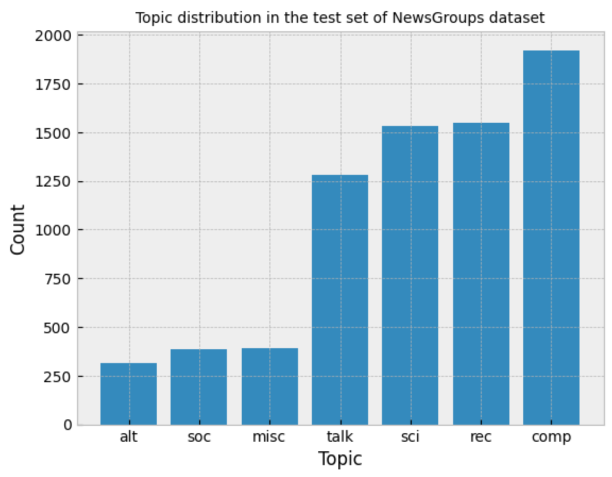
In the version of the dataset that we use, we have kept 500 randomly selected documents. In this sampled test set, the mean number of tokens and characters is 204 and 1000, with a standard deviation of 112 and 562 respectively. Moreover, the test set is balanced as 52% of the samples are classified as "positive" and 48% of them are classified as "negative". The mean number of characters and tokens for inputs that are classified with "positive" sentiment is 990 and 204, with a standard deviation of 530, and 108, respectively. The distribution for texts that are classified with "negative" sentiment is similar, where the mean number of characters and tokens is 1006 ± 589 and 204 ± 115, respectively.

20 Newsgroups

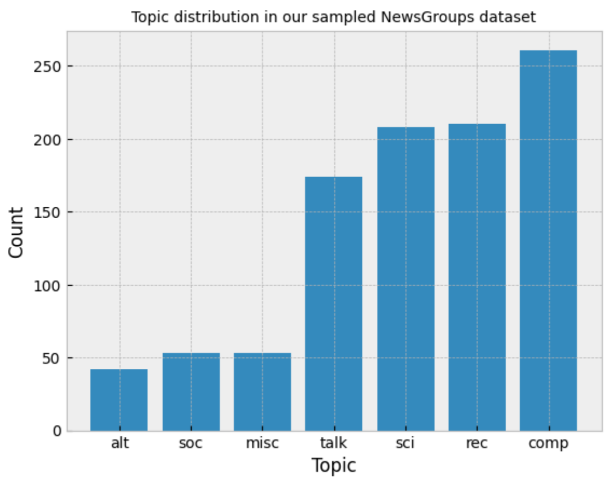
The **20 Newsgroups** dataset is a collection of approximately 20.000 newsgroups posts across 20 different topics, including topics like politics religion, science, and sports. The dataset is used as a benchmark for text classification and topic classification, where the goal is to predict the topic of an article or document in general based on its content. It is a diverse and complex dataset as it covers a wide range of topics, but this also makes it more challenging than simpler text datasets. It is worth noting that we gain access to the 20 newsgroups dataset from the scikit-learn datasets library [65]. Regarding counterfactual editors, the 20 Newsgroups dataset is used in MiCE and CAT editors among others.

In our experiments, we use a sample of 1,000 documents that are randomly derived from the test partition of the dataset which includes 7,000 documents. We use samples only from the test set due to the fact that the train set of the dataset has already been used for fine-tuning in some of the editors. Also, in order to experiment with a cleaner dataset we remove all headers, footers and quotes from the text samples, along with line changes.

During the creation of our test set, we constrained the created test set to follow the same distribution over the 7 different classes of the dataset as the original test set. The mean number of characters and tokens for our samples is 263 and 58, with a standard deviation of 134 and 31, respectively. Our test set's distribution over the 7 dataset classes is as follows: 'rec': 210 samples, 'comp': 261 samples, 'alt': 42 samples, 'talk': 174 samples, 'sci': 208 samples, 'soc': 53 samples, 'misc': 53 samples. The list of the 20 classes present in the original dataset is: [comp.graphics, comp.os.mswindows.misc, comp.sys.ibm.pc.hardware, comp.sys.mac.hardware, comp.windows.x, rec.autos, rec.motorcycles, rec.sport.baseball, rec.sport.hockey sci.crypt, sci.electronics, sci.med, sci.space, misc.forsale, talk.politics.misc, talk.politics.guns, talk.politics.mideast, talk.religion.misc, alt.atheism, soc.religion.christian]. However, we work on a 7-class version of the dataset comprising of the 7 main topics of the dataset, comprising of the following: [comp, rec, sci, misc, talk, alt, soc].



(a) The original NewsGroups test set with 7500 samples



(b) Our sampled test set with 1000 samples

Figure 6.2.1: Comparison between the topic distribution in the original NewsGroups test set and ours

Test set sizes

Regarding the sizes of the test sets used for our experiments, we underline that t-tests for different sample sizes, feedback steps, and the two datasets in the work of Filandrianos et al. [20] show that for test sizes greater than **200** the results converge for both datasets. Based on these findings, and in order to reduce the computational load we decide to utilize the test sets as described above which both fulfill the requirement of the 200 samples and above.

6.2.2 Counterfactual Editors

In this subsection, we will review the main characteristics of the counterfactual editors that are utilized for the generation and evaluation processes of our counterfactuals and how we integrated them into our experiments.

MiCE

MiCE is an editor that involves two main steps in order to generate counterfactual edits. In the first step, the editor is trained to infill masked spans of text in a targeted manner [75]. Specifically, a pre-trained model, the T5 model (Text-To-Text Transfer Transformer), is fine-tuned to infill masked spans given masked text and a target classification label as input. The model is fine-tuned for each dataset to better fit each dataset's distribution.

In the second step, the editor is used to generate edits using multinomial sampling, "a method which randomly selects the next token based on the probability distribution over the entire vocabulary given by the model" [34]. The input text can be masked either randomly or using the predictor's attention in a white-box manner and then the fine-tuned model fills these blanks. After that, a binary search of 4 levels is conducted which attempts to find edits after masking a percentage of the sentence which ranges from 0% to 55%. In this step, the editor's aim is to identify the most minimal edits, i.e. that have the lower minimality score, that will alter the classifier's prediction to the targeted one.

In the experiments of the authors of MiCE, the fine-tuning of the model is done by using the predictor's outputs (and not the ground-truth label), and for selecting the masks' locations, the classifier's attention is used as described above. Therefore, we used MiCE as a white box. We also used the pre-trained T5 model that the authors provided. This model was fine-tuned on the same data as the predictor.

For the edits generation, we used the code provided by the authors [76] and we generated edits based on the default arguments used by the authors for each one of the datasets. The authors use a mix of top-k [17] and top-p (nucleus) sampling [30] while early stopping is also used.

However, aside from the default arguments used, we conduct different experiments for both random and attention-based masking. Moreover, we introduce some new code in the original codebase in order to implement edit generation based on part-of-speech tagging. Another modification we made to the code is to serve the integration of our data as an input, and not the whole datasets, in order to generate counterfactuals from the data needed at each step. Finally, we experiment with different generation methods in order to evaluate their effect on the edits using beam-search with different numbers of beams, namely 1, 5, 15, 30, and 60 beams as well as multinomial sampling.

Polyjuice

In contrast to MiCE, which is an editor which generates counterfactuals based on a target prediction, Polyjuice is a general-purpose counterfactual generator. Polyjuice employs GPT-2, a large-scale language model used for the task of text generation, after fine-tuning it on various datasets consisting of pairs in the format of (input, counterfactual), including the IMDb dataset. A main characteristic of Polyjuice is that it utilizes multiple control codes in order to allow the user to specify what type of perturbation they aim for. These control codes are namely: [negation, quantifier, shuffle, lexical, resemantic, insert, delete, restructure] which are also shown in figure 4.2.3.

Polyjuice uses random masking and then for the edits generation uses multiple "application-agnostic relationships" [102], to create perturbations and afterward uses "application-specific selection methods" in order to find counterfactuals that best apply to specific control codes. Thus, Polyjuice is an editor that falls under the wider category of conditional text generation.

Polyjuice does not use a predictor during the counterfactual generation so for our experiments, we used the editor in a black-box manner combined with the same predictor we use with MiCE. Specifically, in our experiments, we first predict the original label of the input sentence and then select from the editor's output only the counterfactuals that flip the original label to the contrast one. After that, for the sake of the minimality of the counterfactuals, we only select the most minimal edit amongst them.

For Polyjuice, we used the code provided through this module [103]. For the generation of edits, we searched in all the control codes available, and we produce as many perturbations as possible for each instance by setting `num_pertubations = 1000`. However, Polyjuice did not generate this many perturbations in any of our experiments.

TextFooler

TextFooler differs a lot from the editors of MiCE and Polyjuice since it does not employ a model or neural network in order to generate counterfactual edits. TextFooler, therefore, utilizes a different technique to generate what we call adversarial examples and then use them as inputs for a black-box classifier.

In order to create counterfactuals, TextFooler identifies the word tokens that are most influential to the predicted class and replaces them with some fixed methods. In contrast to MiCE, where the model's gradients are used to solve this problem, in TextFooler the classifier is used as a black box and therefore these are not available. The authors score the importance of each word by monitoring the predictor's output before and after the removal of this specific word. [44]. Moreover, the replacements of the word tokens are identified based on the closest match in an embedding space and are consequently independent of the context of the word and the predictor. Thus, TextFooler opts for word swaps that are synonyms with the removed word and to maintain the grammar of the input sentence constrains these synonyms to have the same part of speech.

In our experiments, we use TextFooler through the TextAttack framework. As TextFooler can attack different classifiers in a black-box manner, we use the same predictor as in MiCE and Polyjuice in order to compare the editors fairly. For the edit generation, we use the same parameters as defined by the authors in the paper. Specifically, we use constraints that prohibit the modification of stopwords and already modified words. We also use a threshold of the word embedding distance (the distance that two words are considered synonyms) equal to 0.5, we perform a search in a maximum of 250 candidate words within this embedding distance and we enforce replacements based on POS tagging, i.e. the generated word has the same part-of-speech tag as the original word.

It is worth noting that in TextFooler, we do not have any control in terms of selecting the most minimal edit as in Polyjuice and MiCE, but TextFooler already creates very minimal edits and so this does not influence our problem specifications.

6.2.3 Masking methods

In this subsection, we will describe in short the masking methods used by the counterfactual editors. Masking plays a significant role in counterfactual generation, as the masker of each editor is the component that decides which word tokens will be substituted with a mask token in order to then be replaced by a new word token.

Random Masking

Random masking can be considered the baseline for masking methods used by a counterfactual editor. Its implementation is simple but can deliver good results if it is combined with a clever search algorithm. Specifically, a random masker receives the tokenized input text sentence and randomly selects some indices of the word tokens list to be masked with a predefined mask token, e.g. <mask>. The number of selected indices is either pre-determined or defined by some search algorithm, which iterates until it finds the optimal number of masked tokens which satisfy the editor's goal. For example, in MiCE, binary search is used to find optimal masking percentages [75] for the input sentence. In contrast, in Polyjuice, where random masking is also used, the user provides a maximum number of masked tokens, and all the randomly masked sentences are generated at once without the use of a search algorithm.

In general, random masking is computationally worse than other masking methods, which also makes it require more time in order to generate a counterfactual. However, it is considered to be a good starting point and a strong baseline for counterfactual generation experiments.

Attention Masking

Attention masking is the masking method that uses the predictor's attention mechanism to find which word tokens affect the predictor's prediction the most. Since our task is sentiment analysis and our goal is to flip the predictor's outcome in the most minimal way possible, we want to mask the tokens that are most strongly associated with our target class. One way of achieving that is the method of *gradient masking*, which is used in MiCE. The authors suggest the use of gradient attribution [81] to select the tokens to be masked. For each text sentence, they use the gradient of the predicted logit for the target label with respect to the embedding layers of the predictor and take the l_1 norm across the embedding dimension[75]. Then the gradient norms are sorted from highest to lowest and the masker selects a percentage of them based on a masking fraction which is the result of a binary search.

Intuitively, using attention masking allows our masker to have some "knowledge" of what tokens should be masked and this makes attention masking methods perform better than random masking methods.

Word Importance Ranking

Word Importance Ranking is a masking method used in TextFooler. Since TextFooler attacks models in a black-box setting, it does not have access to the model's architecture or parameters and therefore cannot use a masking method like gradient masking. Thus, the authors of Textfooler introduced a "selection mechanism" which measures the influence of a word towards the prediction of the model by calculating the prediction change before and after deleting each word[35]. This process is followed for all words, and then they are ranked by their importance score.

Using this masking process, the semantic similarity of the sentence is maintained as much as possible and the editor can result in successful contrastive edits faster.

6.2.4 Models

T5 Transformer

The T5 model [70] is an encoder-decoder model pre-trained on both supervised and unsupervised tasks, for which each task is formatted into a text-to-text format. The model achieves state-of-the-art results on many benchmarks including text classification, text generation, and more. For our task, which is counterfactuals generation, we use the version of T5 which addresses the task of Text-to-Text generation. The T5 model comes in five different sizes depending on the number of parameters and computational requirements, these are from smaller to larger: t5-small, t5-base, t5-large, t5-3b, and t5-11b. For our experiments, we use the standard version, t5-base, which uses approximately 220 million parameters. The model is pre-trained on the Colossal Clean Crawled Corpus (C4) dataset [70] which is a cleaned version of the Crawled Corpus dataset [8] that includes approximately 750 gigabytes of English text from the web.

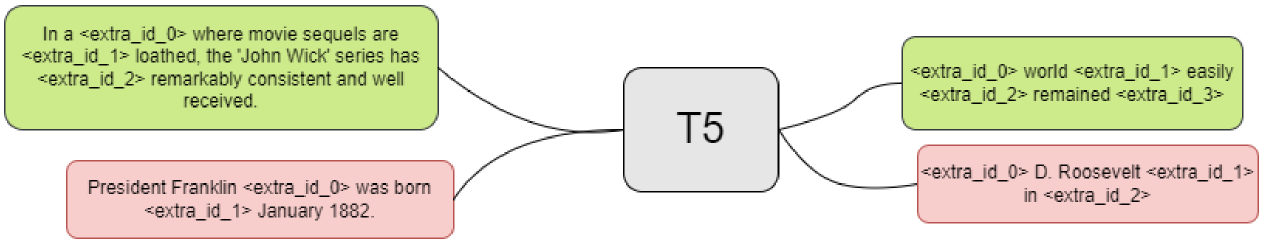

In a <extra_id_0> where movie sequels are <extra_id_1> loathed, the 'John Wick' series has <extra_id_2> remarkably consistent and well received.
President Franklin <extra_id_0> was born <extra_id_1> January 1882.
T5
<extra_id_0> world <extra_id_1> easily <extra_id_2> remained <extra_id_3>
<extra_id_0> D. Roosevelt <extra_id_1> in <extra_id_2>

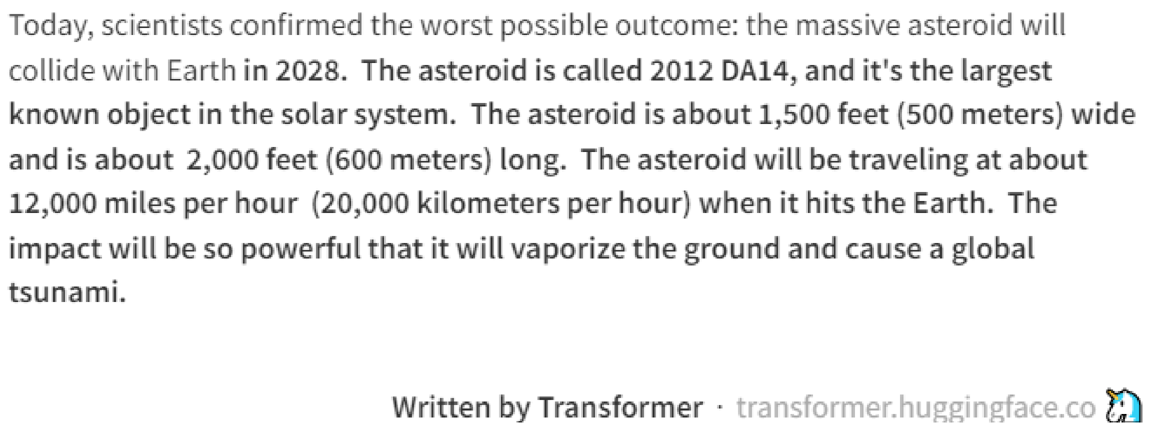
Figure 6.2.2: Examples of the results that the T5 text-to-text model generates for two sentences with some masked tokens. The mask token is formatted as "<extra_id_i>" where i is the number of occurrence of each masked token starting from zero.

In our experiments, we use T5 through MiCE where the authors utilize a fine-tuned version of the t5-base version. The generation method used by the authors is multinomial sampling, but we also experimented with beam search and different numbers of beams. Also, the most important parameters that are fine-tuned are:

- num_return_sequences equal to 15
- top_p equal to 0.95
- top_k equal to 30
- use of early stopping
- length penalty equal to 0.5
- no_repeat_ngram_size equal to 2

GPT2

GPT2 or Generative Pre-trained Transformer 2 is an open-source large language model created by OpenAI [69][100] which translates, summarizes, and generates text. GPT-2 has a pre-trained transformer architecture and uses attention mechanisms in order to predict text in the most suitable manner. The model was trained on a large dataset named WebText [69] that was published in the same paper as the model and contains text from millions of web pages. It is worth noting that the model uses approximately 1.5 billion parameters, and it is the predecessor to the widely known GPT-3 model which uses 175 billion parameters.


Today, scientists confirmed the worst possible outcome: the massive asteroid will collide with Earth **in 2028. The asteroid is called 2012 DA14, and it's the largest known object in the solar system. The asteroid is about 1,500 feet (500 meters) wide and is about 2,000 feet (600 meters) long. The asteroid will be traveling at about 12,000 miles per hour (20,000 kilometers per hour) when it hits the Earth. The impact will be so powerful that it will vaporize the ground and cause a global tsunami.**

Written by Transformer · transformer.huggingface.co

Figure 6.2.3: Example of the text that the GPT-2 model generates (in bold) given a random initial sentence. The web app Write With Transformer [33] used to generate the example is created by HuggingFace and uses gpt2-large and other models to generate text.

In our experiments, Polyjuice uses a fine-tuned version of GPT-2 for the task of Text-to-Text generation. To achieve that, the authors of Polyjuice incorporate fill-in-the-blank structures [102] [13] in the fine-tuned model to define which word tokens will be perturbed. The finetuned version of GPT-2 for Polyjuice also uses specific control codes, as referenced in 4.2.3. It is also of importance that the authors of Polyjuice fine-tuned the model using 7 different datasets, which do not include neither IMDb nor NewsGroups.

Moreover, the fine-tuned model uses the default parameters as they are stated in the GPT-2 codebase, but uses fewer hidden layers than all three of the open-source versions of the model available, namely: gpt2-medium, gpt2-large, and gpt2-xl. We must also note that the version of the model that Polyjuice uses cannot accept inputs larger than 512 tokens and consequently, we truncate each input larger than this threshold.

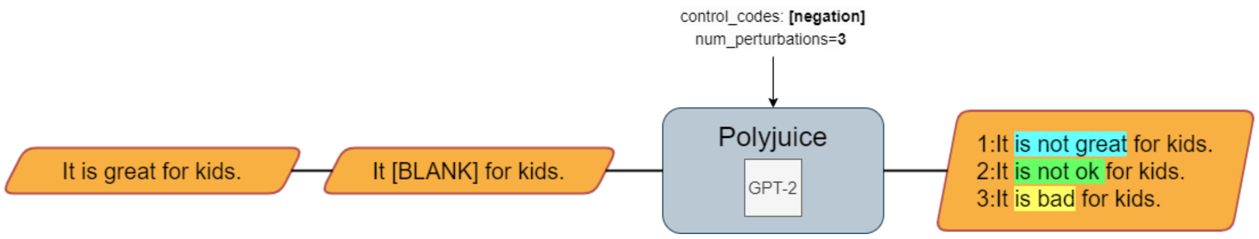


Figure 6.2.4: A diagram that shows the operation of Polyjuice on text input. The word or words that we want to mask are swapped with the mask token [BLANK], the desired control code is negation and the desired number of perturbations is 3.

6.2.5 Predictors

In this thesis, where the problem at hand is contrastive counterfactual generation we need classifiers that can support both binary classification, for the IMDb dataset where the task is sentiment analysis, and multiclass classification, for the NewsGroups dataset where the task is topic classification. In the scope of counterfactual editing, we refer to these classifiers as ***predictors***.

In our experimental setup, we use the pre-trained predictors used by the authors of MiCE. Those predictors, each for a different dataset, have been built on RoBERTa-Large [46] and have a maximum sequence length of 512 tokens. The test accuracies they achieve are 95.9% for the IMDb predictor and 85.3% for the NewsGroups predictor [75]. Concerning the training of the predictors, the authors of MiCE fine-tune both predictors for 5 epochs with batch size 8 using Adam with an initial learning rate of $2e-05$ [75]. The training is done with *AllenNLP* [22].

As far as how each predictor works, the IMDb predictor given an input sentence predicts the probabilities of two labels in the range of $[0, 1]$, which represent **positive and negative** sentiment. When we use this predictor, the target label for our counterfactual editors is always the contrast one.

The NewsGroups predictor, however, calculates the probability for each one of the **seven different classes** listed as: [comp, rec, sci, misc, talk, alt, soc] and labels the input with the class of the highest probability. The target label when using this predictor in MiCE and Polyjuice is the second highest probability calculated by the predictor. In TextFooler, we do not use a target prediction but rather check if the original prediction was changed.

6.2.6 Metrics

In this subsection, we enumerate the metrics that we use in order to evaluate the generated counterfactuals. In general, the symbolism used for the metrics used is $metric@n$, i.e. metric at step n of the iteration process.

Minimality

Minimality is the metric that calculates the word-level Levenshtein distance between the original text and the edited text. In our results, we symbolize minimality as **min@n**.

We calculate minimality with the help of the `score_minimality` python function from the MiCE module [76]. Also, this function was utilized in order to optimize edits' minimality in our experiments in MiCE and Polyjuice.

Inconsistency

Inconsistency calculates, as it is called, the inconsistency of the word-level Levenshtein distance. In our results, inconsistency is labeled as **inc@n**. In order to calculate inconsistency for our results we use equation 5.2.2 and use the Python implementation by the authors of [20] for that cause.

Flip rate

Flip rate is the metric that calculates the ratio $\frac{n_{flipped}}{n_{all}}$, where n_{all} is the size of the input dataset, and $n_{flipped}$ is the number of samples for which the predictor's output changed for the counterfactual edit. In our results, we label flip rate as **flip rate@n**.

In order to calculate the flip rate at step n, we count the edits for which the predictor's output was flipped in step n and then divide by the number of all input sentences at step n.

Base perplexity

The metric that calculates the language model perplexity of GPT-2, a large, general-domain language model. We label base perplexity as **ppl-base@n**. To calculate the base perplexity values in our experiments we use the implementation from the module `disentanglement-vae` [88] which supports the related paper of the authors [87].

Fine perplexity

Fine perplexity calculates the language model perplexity of GPT-2, fine-tuned on the IMDb [94] and on the 20 Newsgroups [26] datasets. This metric differs from Base Perplexity, as it is used to identify if the generated text contains "surprising" edits compared to the datasets' content. We label fine perplexity as **ppl-fine@n**. To calculate the fine perplexity values in our experiments, we use the implementation from the module `disentanglement-vae`.

We must note that both base perplexity and fine perplexity require a GPU in order to complete calculations in a reasonable amount of time, as they use a language model (GPT-2) and perform computationally intensive calculations.

6.3 Implementing our counterfactual generation and evaluation system

In this section, we describe in detail how we implement our counterfactual and generation system. To facilitate the understanding of the processes, we present the system as a pipeline and then explain each separate component and method of the system separately.

6.3.1 Overview of the system's architecture

For this thesis, we create a pipeline with distinct components which facilitate the generation and analysis of our counterfactual edits. The first step of our pipeline is the proper formatting of the two datasets used, IMDb and NewsGroups, in order to create the input for the counterfactual editors. The second step is the setup of the system we use for counterfactual generation. In this component, we encapsulate both the editors and the predictor and run experiments with each editor separately. The last step of our flow line is the evaluation process of all the generated edits. In order to, evaluate the edits on different aspects we use 5 different metrics along with qualitative analysis and examples.

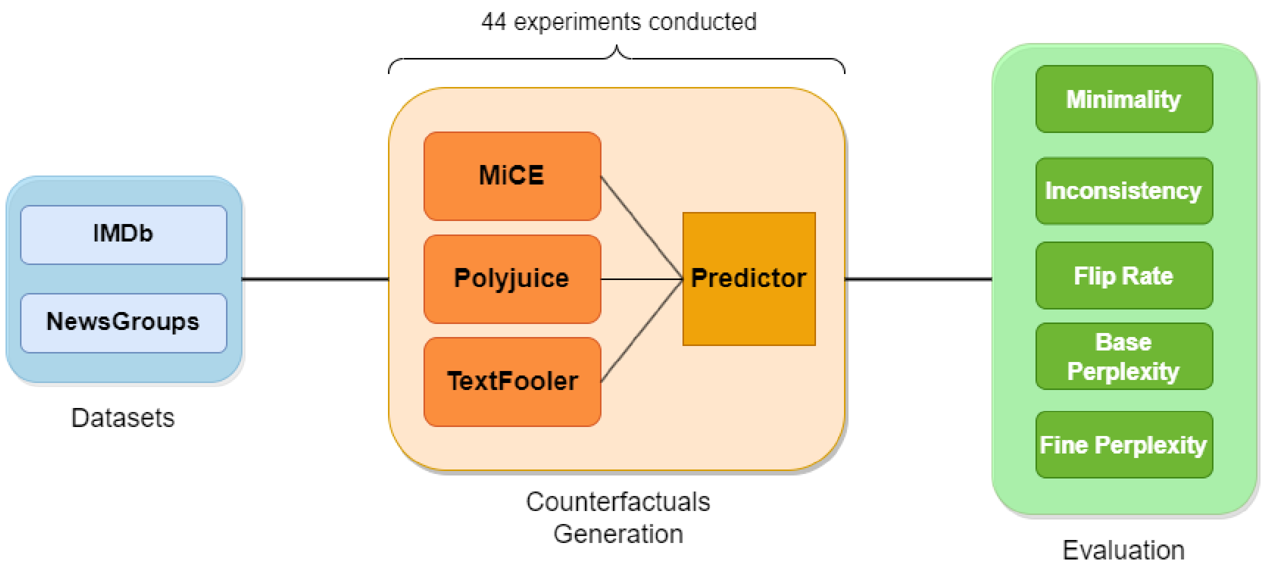


Figure 6.3.1: A pipeline diagram showing how we structure our system. We utilize two datasets, which are then passed as input to the editors in the correct format. We then conduct 44 separate experiments on the editors, which are wrapped with the predictor as a single component. After all the experiments are conducted, we format the output suitably and evaluate the generated edits based on the 5 metrics shown, which are also detailed in 6.2.6 and 4.4.

6.3.2 Counterfactual generation: Implementation

In the following segment, we present the architecture that we used for the counterfactual editors, how we integrated the use of part-of-speech tags, and the concept of Counterfactuals of counterfactuals in this system. We elucidate the processes followed and concepts used with the use of pipeline diagrams and their explanation in detail.

The editor's architecture

In this subsection, we provide a detailed overview of the architecture we use for our experiments. First, in order to use a counterfactual editor to generate counterfactual edits we need to train the editor, ideally on more than one dataset. For our tasks, we use pre-trained models for the editors of MiCE and Polyjuice, whereas in TextFooler, an editor is not needed.

Secondly, we focus on the counterfactuals edits generation. This process consists of three significant components, the editor, the predictor, and finally the evaluation of the edits. For each editor, a different masking algorithm is used, specifically in MiCE we either use random or attention-based masking, in Polyjuice we use random masking and in TextFooler we use an importance ranking algorithm as provided by the authors. After the masking, each editor utilizes different methods and editor models for generating edits, as described in 6.2.2.

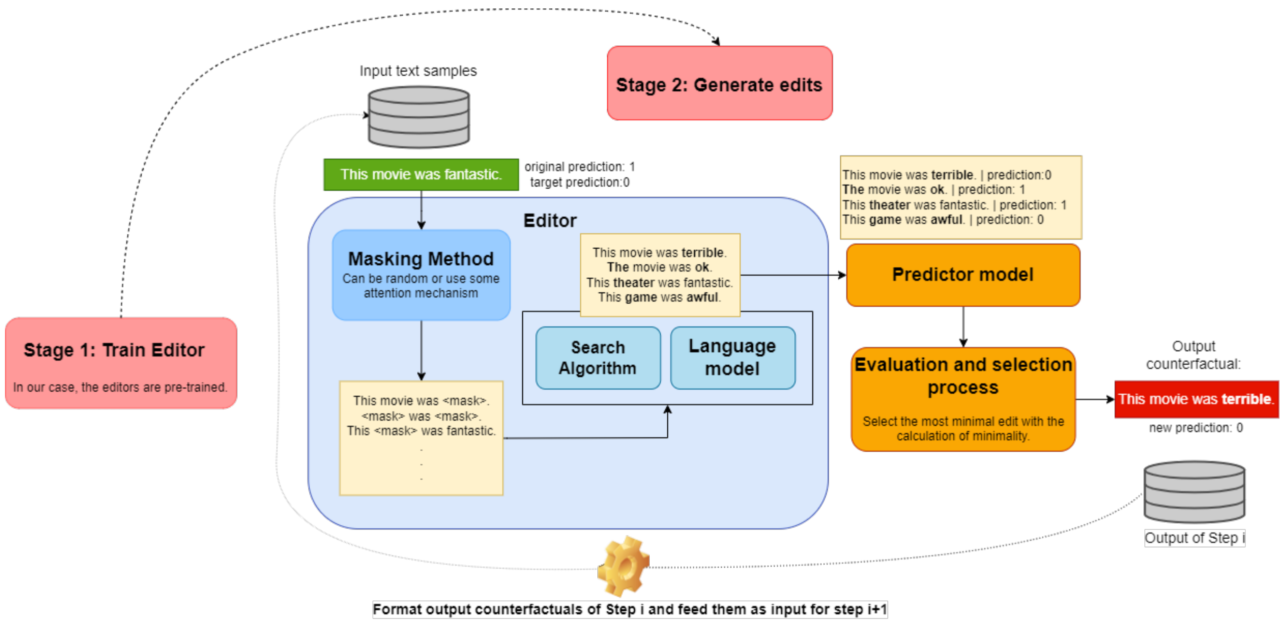

Input text samples
Stage 2: Generate edits
This movie was fantastic.
original prediction: 1
target prediction:0
This movie was terrible. | prediction:0
The movie was ok. | prediction: 1
This theater was fantastic. | prediction: 1
This game was awful. | prediction: 0
Editor
Masking Method
Can be random or use some attention mechanism
This movie was terrible.
The movie was ok.
This theater was fantastic.
This game was awful.
Predictor model
Stage 1: Train Editor
In our case, the editors are pre-trained.
Search Algorithm
Language model
Evaluation and selection process
Select the most minimal edit with the calculation of minimality.
Output counterfactual:
This movie was terrible.
new prediction: 0
This movie was <mask>.
<mask> was <mask>.
This <mask> was fantastic.
Output of Step i
Format output counterfactuals of Step i and feed them as input for step i+1

Figure 6.3.2: An overview of the counterfactual editor's architecture and how we wrap it with the predictor to conduct our experiments. In the figure, we can also see how we use the output of the counterfactual editor to create a new input

The next component we use for the counterfactuals generation is the predictor. We use two different predictor models for the IMDb and the NewsGroups datasets accordingly, which are pre-trained AllenNLP predictors, that are also used by the authors of MiCE. In order to integrate the predictor with all three of our editors, we used different methods for each editor as described in 6.2.2. Since our counterfactual generation method focuses on sentiment analysis, and we have a target prediction class for each edit, we need at least one counterfactual which satisfies the target prediction class. If more than one edit is classified in the contrast label class, we utilize a third step in the edits generation which evaluates the edits based on minimality 6.2.6 and selects the counterfactual with the lowest minimality score. This allows us to create edits that not only flip the predictor's class prediction but are also minimal.

After that, we format appropriately the output generated after these three main steps, in order to use it as input for the next step of the iteration process. This is the concept of Counterfactuals of Counterfactuals referred to in 5.2.1. In this way, we create a 10-step automated pipeline that is used for each one of our experiments.

Part-of-speech tags integration and implementation

The majority of experiments carried out in this thesis generate counterfactuals based on a targeted part-of-speech (POS) tag. In order to achieve this successfully, we need to integrate POS tagging in each one of the editors, in a way that does not influence the other components of the editor. Thus, we integrate POS tagging in the masking phase of each editor, adding in that way a restriction for the words that the masker can mask. The library we use to retrieve the POS tags of each word is *Spacy* [31], as it exposes the part of speech of each word in an effortless way.

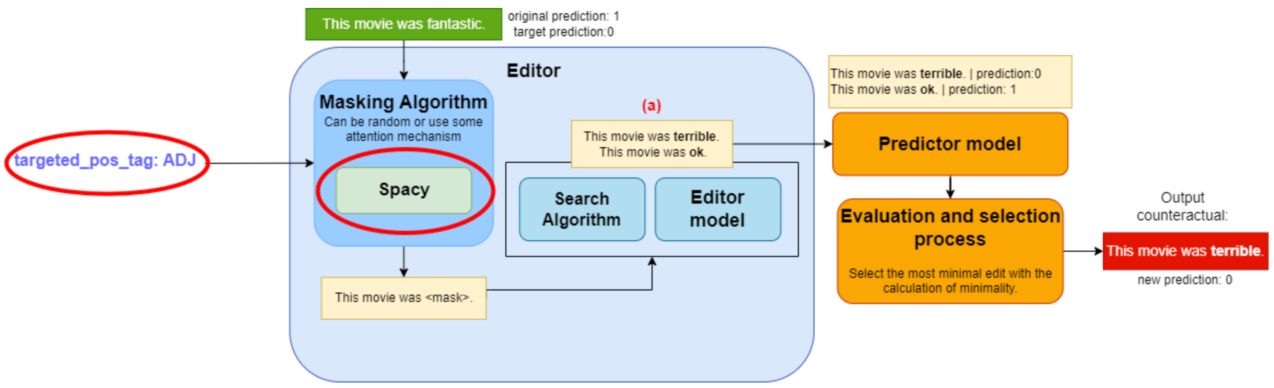


Figure 6.3.3: The diagram which shows the counterfactual generation system that we use, also shown in 6.3.2, but with the needed changes (in the red ovals) to highlight how we integrate the use of part-of-speech tags into the process. One can notice that the generated edits (see (a) in the diagram) only contain changes in the adjective of the input.

However, each counterfactual editor we use needs an appropriate implementation to support edit generation using a specific POS tag. So, we had to implement different methods and functions to fit each editor.

In **MiCE**, we intervene in the `Masker` class of the editor and introduce a new Python function called `pos_masker()` [37] that takes as arguments the input sentence, the tokenized sentence, and the targeted POS tag. As the MiCE editor uses a different tokenizer, the T5Tokenizer [32], than what Spacy uses, we create a mapping that ensures compatibility between the two tokenizers. Moreover, we noticed that the T5 tokenizer sometimes splits a token into two separate tokens, e.g. 'apple' to 'app'+'le'. As this erroneous behavior could influence our implementation, we locate these wrongly split tokens and join them back together in order to ensure the integrity of our tokens. After these necessary steps, we filter the input tokens based on the targeted POS tag and return them to the masker. After this process, either if we use random or gradient masking, the masker only operates on the tokens with the targeted POS tag.

In **Polyjuice**, we use a function written by the authors of Polyjuice which generates randomly masked sentences but allows us to pass as an argument the indexes of the words to be masked. Therefore, we identify the words with the desired POS tag through Spacy and pass their indexes as an argument to the function. In this way, we use a variation of random masking based on the targeted POS tag. Moreover, Polyjuice normally substitutes tokens, inserts tokens, or modifies the word's subtree(next and previous token of a word). To accommodate the need of masking specific words with a certain POS tag, we force Polyjuice to work only with single token substitutions. However, Polyjuice's language model, GPT-2, despite this refinement, still acts unpredictably by deleting or inserting large parts of text, rendering Polyjuice unable to qualitatively compare with the other editors. These implementations of Polyjuice can be found in our codebase [37].

Finally, in **TextFooler**, in order to generate counterfactuals based on a targeted POS tag, we create a new `PreTransformationConstraint` child class [37] in the TextAttack module, which enforces the editor's algorithms to only perturb tokens with the targeted part-of-speech, as it is given by the user. Then, we add this constraint method to the attack process along with the other methods and constraints in the TextAttack module.

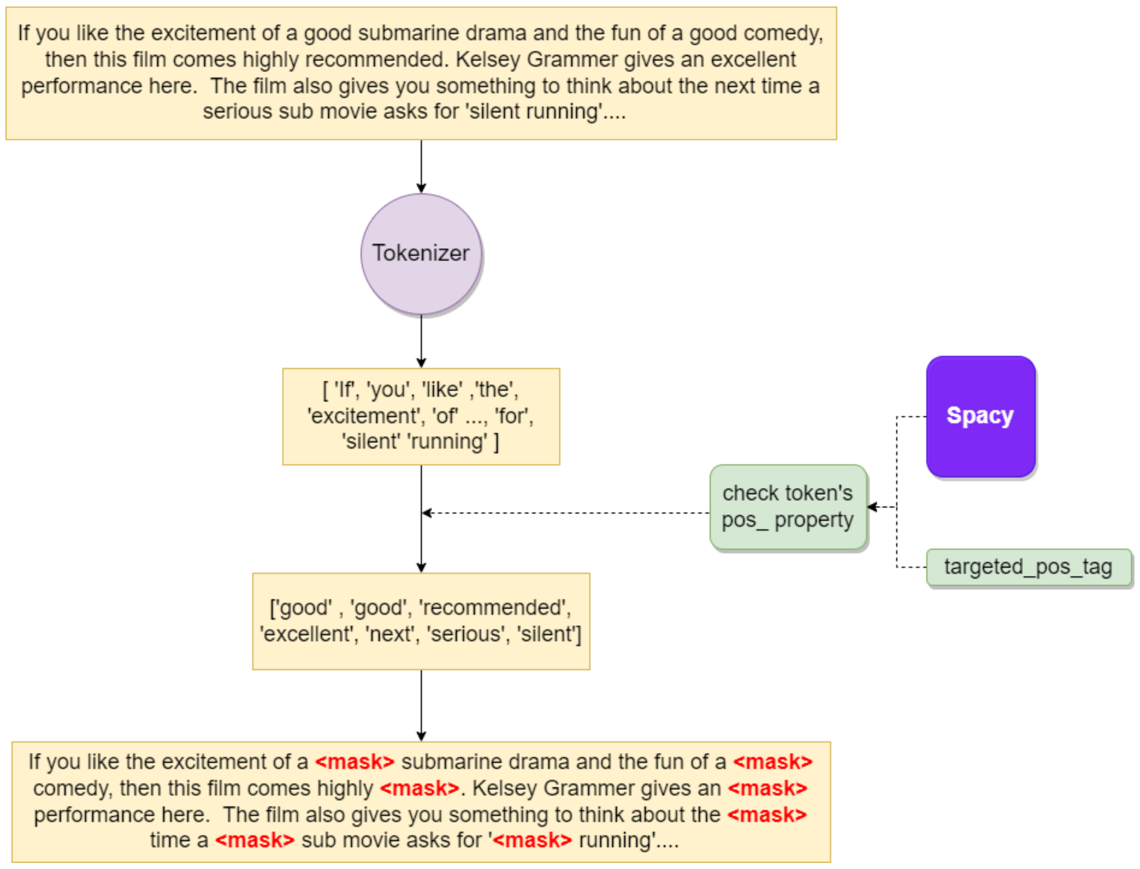

If you like the excitement of a good submarine drama and the fun of a good comedy, then this film comes highly recommended. Kelsey Grammer gives an excellent performance here. The film also gives you something to think about the next time a serious sub movie asks for 'silent running'....
Tokenizer
['If', 'you', 'like' ,'the', 'excitement', 'of' ..., 'for', 'silent' 'running']
Spacy
check token's pos_ property
targeted_pos_tag
['good' , 'good', 'recommended', 'excellent', 'next', 'serious', 'silent']
If you like the excitement of a <mask> submarine drama and the fun of a <mask> comedy, then this film comes highly <mask>. Kelsey Grammer gives an <mask> performance here. The film also gives you something to think about the <mask> time a <mask> sub movie asks for '<mask> running'....

Figure 6.3.4: An example of how we mask sentences based on POS tag by using the Spacy library. The targeted POS tag in the example is the adjective.

Counterfactuals of counterfactuals implementation

In order to implement the feedback process of Counterfactuals of counterfactuals, we carry out specific output and input formatting. After the editor finishes one step of edits generation and produces a CSV file with the edits, we save the file, and for all the edits that were successfully flipped we create separate text files with a counterfactual in each of them. Then the next round of edits reads its input from the folder where the text files with the edits of the previous round are saved. For each editor, we formulated the process in the same way for the sake of totality.

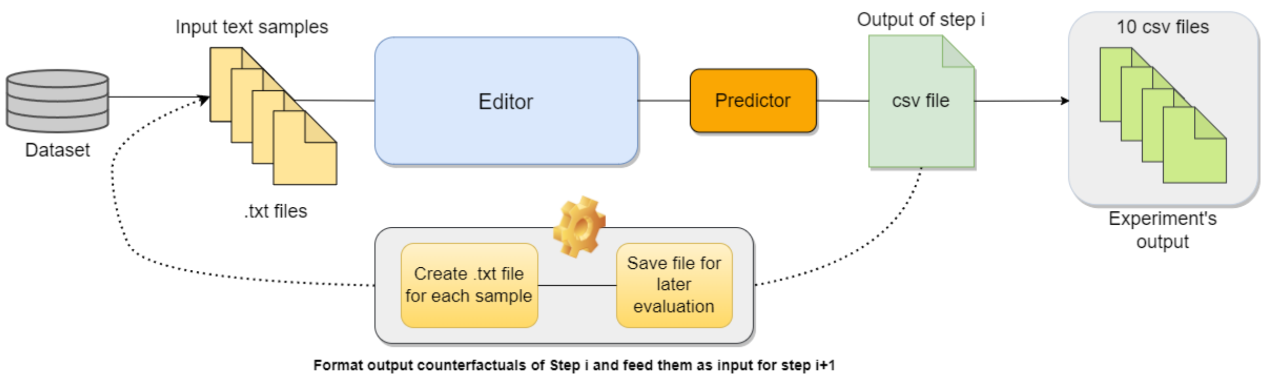

Input text samples
Dataset
.txt files
Editor
Predictor
Output of step i
csv file
10 csv files
Experiment's output
Create .txt file for each sample
Save file for later evaluation
Format output counterfactuals of Step i and feed them as input for step i+1

Figure 6.3.5: A flow diagram showing how we implement Counterfactuals of counterfactuals in our system.

6.3.3 Counterfactuals evaluation: Implementation

As far as evaluation is concerned, we conduct it as a separate process of our experiments after each experiment. In order to optimize the process and structure it in an organized way we use object files (pickle files) to store the edits and the results of each metric. The code used to generate each metric's results is largely borrowed from the works of Filandrianos et al. [20] for inconsistency and minimality, and from Vasilakes et al.[87] for base perplexity and fine perplexity. We present Figure 6.3.6 which conveys the pipelining of our counterfactual evaluation process.

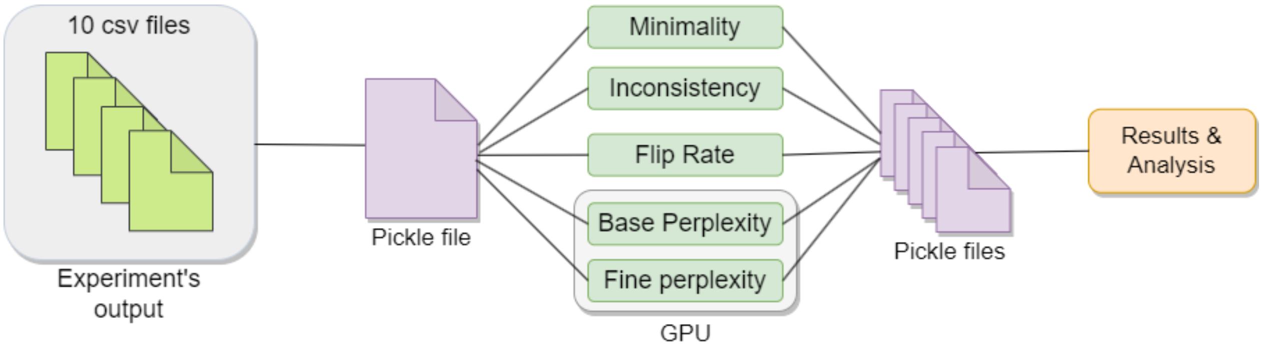


Figure 6.3.6: A flow diagram showing how we evaluate our edits as a separate process.

After having the results of all metrics, we use Jupyter Notebooks to present the results in an understandable way and to generate the tables and diagrams that help us structure and also support our conclusions.

Chapter 7

Experiments

In this section, we will present our experiments, their results and discuss the outcomes of the evaluation process. Besides the quantitative results, we will also provide qualitative results which will assist in a better understanding of the generation and evaluation methodologies that we use.

Contents

7.1 An outline of the experiments

First, we find importance in providing an outline of the experiments conducted, in order to clarify around which axis we perform the experiments and how we combine the methodologies discussed in the earlier chapters. In total, we use three editors, MiCE, Polyjuice, and TextFooler. Specifically for MiCE, we utilize two ways of masking, random masking and gradient masking. From now on, we will refer to MiCE with the use of random masking as *MiCERandom* and to MiCE with the use of gradient masking, simply as MiCE. In addition, we conduct all of our experiments in both datasets, IMDb and NewsGroups. In all the experiments that were carried out, we implement the concept of Counterfactuals of counterfactuals (5.2.1) for 10 steps. Moreover, we use all three editors, and MiCERandom, to generate counterfactual edits with the method of targeted part-of-speech tags (5.2.3). Finally, we execute some experiments to inspect the influence of the number of beams on MiCE.

To sum up all the following, we present the table of experiments below, which is an organized way of outlining all of our experiments.

Editors	**Experiment Types**				
	Out-of-the-box	**ADJ**	**NOUN**	**VERB**	**Beam - search***
MiCE	✓	✓	✓	✓	✓
MiCERandom	✓	✓	✓	✓	–
Polyjuice	✓	✓	✓	✓	–
TextFooler	✓	✓	✓	✓	–

* Beam-search experiments are conducted with multiple numbers of beams, namely 1,5,15,30,60 and 120

Table 7.1: An outline of our experiments. Out-of-the-box signifies the use of editors without any intervention from us, ADJ, NOUN, and VERB are the targeted part-of-speech tags and Beam-Search is an attempt on MiCE. All experiments are conducted for both IMDb and NewsGroups datasets.

7.2 Results

7.2.1 Interpreting the metrics results

In the following paragraphs, we evaluate the quantitative results with the help of the metrics used (6.2.6) and describe what each metric's results hint for the edits and how they can help us explain the editors' and models' decisions.

In this evaluation process, we present results both in table form and in various diagrams, which assist largely in understanding the results and help us focus on the point to be made. In the tables in the section, the optimal value of each metric in each step of the generation process is highlighted in bold.

In order to present a first example of how a counterfactual edit shapes after 10 steps of the feedback process and also have a reference point, we show the example below.

Table 7.2: An example of 10 steps of feeding back counterfactuals on MiCE on an IMDb sample.

Step	**Counterfactual**	**Prediction**
1	What a script, what a story, what a **mess**!	0
2	What a script, what a story, what a **movie**!	1
3	What a script, what a story, what a **disappointment**!	0
4	What a script, what a story, what a **movie**!	1
5	What a **script**, what a story, what a **waste**!	0
6	What a **film**, what a story, what a **movie**!	1
7	What a film, what a story, what a **disappointment**!	0
8	What a film, what a story, what a **delight**!	1
9	What a film, what a story, what a **disappointment**!	0
10	What a film, what a story, what a **movie**!	1

7.2.1.1 Minimality

Comparing our results from minimality's point of view is essential, as our target during counterfactuals generation was **minimal** contrastive counterfactuals. First, we compare the editors with one another as they are used out-of-the-box, i.e. without us intervening in any way.

Table 7.3: Minimality results from each editor without targetting some part of speech.

	IMDb			
	MiCE	MiCERandom	Polyjuice	TextFooler
min@1↑	41.78	87.31	167.86	**33.4**
min@2↑	24.91	54.43	90.95	**23.23**
min@3↑	22.11	43.48	62.83	**18.26**
min@5↑	17.42	33.83	39.41	**13.13**
min@9↑	13.65	27.76	23.12	**8.62**
	NewsGroups			
min@1↑	11.78	26.48	38.71	**7.03**
min@2↑	8.22	19.07	32.5	**4.04**
min@3↑	7.57	16.79	28.5	**3.26**
min@5↑	6.51	14.39	23.56	**2.47**
min@9↑	5.71	13.11	18.88	**1.9**

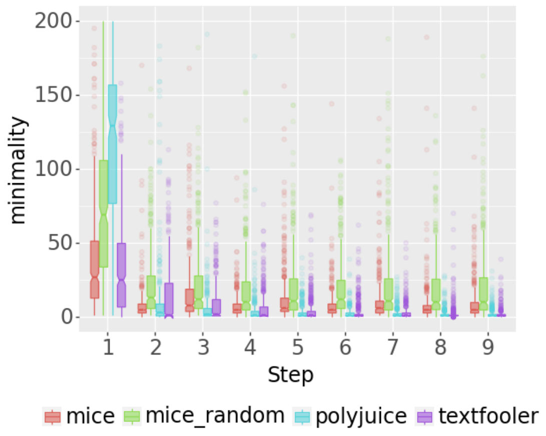


Figure 7.2.1: Boxplot of minimality for the IMDb dataset.

The main observation is that the editor which generates the most minimal edits in both datasets is **TextFooler**. As we mention in 6.2.2, TextFooler does not use a text generation model to create counterfactuals, but a more deterministic approach using multiple constraints. This 'constrained' logic of TextFooler enables it to make much fewer changes in an input sentence than the other editors, as it chooses specific tokens and replaces them with specific words that satisfy all of its constraints. On the other hand, a text generation model, like the ones utilized in the other three editors, is trained on big text datasets and therefore can modify a sentence in a more complex way. Also, TextFooler has a mechanism that ranks the tokens, based on how 'important' they are for the editor's prediction, which allows it to generate impactful edits without many modifications.

Another aspect of the results is that the editor of MiCE which uses gradient masking, as expected, performs better than the editors that use random masking (MiCERandom and Polyjuice). This can be attributed to the fact that gradient masking finds and masks the most influential tokens for the predictor, and therefore can flip the original class of the sentence with fewer modifications and more impact. A random masker, as is normal, cannot have the same efficacy as the randomness in the selection of tokens to mask leads it to also choosing tokens with no influence on the predictor.

In addition, we see a noticeable decrease in the minimality of the edits as the feedback steps increase, which hints that the editors tend to perform fewer edits after each feedback step [20].

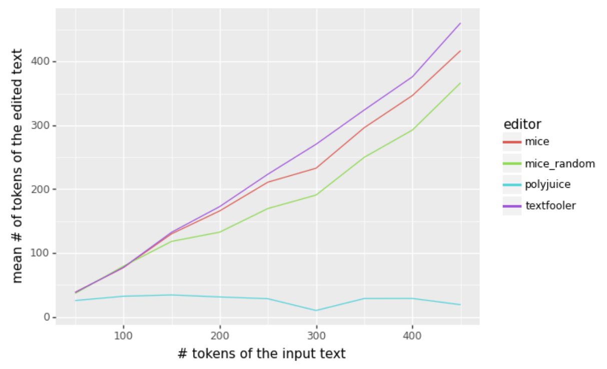
Targetting part-of-speech tags

We then evaluate the editors with the use of targetted POS tags method. In the table below, we can see once more that the editor which performs better is TextFooler. However, what must be noted is that **all the editors create significantly more minimal edits** on both datasets than they did without targetting part-of-speech tags. One of the factors that contribute towards that, is that our part-of-speech constraint causes less text to be masked and therefore less text to be modified. However, our method most of the time enforces the editor's model to make more aggressive edits in order to flip the predictor's outcome more minimally, as we show in the qualitative results in 7.2.3. Concerning part-of-speech tags, in IMDb, we also notice a pattern where the modification of adjectives produces more minimal counterfactuals than with verbs or nouns. In NewsGroups, we notice almost the same pattern, but with nouns contributing to more minimal edits than verbs. As we also explain in B, this is largely due to the part-of-speech tags distribution on the text, but also due to each dataset's task. Finally, it must be noted that all the conclusions we reached above for the editors, seem to be validated with the use of part-of-speech tags.

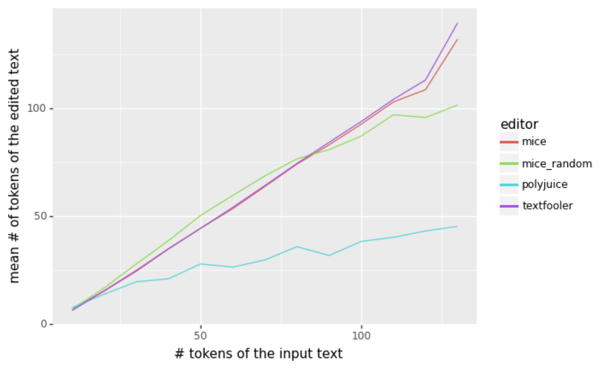
	IMDb											
	MiCE			MiCERandom			Polyjuice			TextFooler		
	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB
min@1↑	12.16	24.45	19.54	12.43	27.57	22.21	44.29	52.86	46.59	**6.33**	**10.28**	**9.5**
min@2↑	7.61	15.02	12.33	8.07	17.11	13.5	28.46	30.31	27.45	**3.69**	**5.7**	**5.25**
min@3↑	6.37	11.94	9.59	6.51	13.49	10.47	23.15	22.84	21.21	**3.18**	**4.48**	**4.26**
min@5↑	4.85	8.76	7.0	4.83	10.0	7.49	16.33	15.37	14.89	**2.46**	**3.26**	**3.21**
min@9↑	3.76	6.32	5.22	3.27	7.46	5.22	10.65	10.1	9.78	**1.92**	**2.53**	**2.45**
	NewsGroups											
min@1↑	3.27	4.59	5.07	3.0	4.95	5.18	30.55	20.51	23.55	**1.81**	**2.84**	**2.58**
min@2↑	2.54	3.45	3.49	2.22	3.55	3.66	20.95	13.9	15.96	**1.46**	**1.94**	**1.82**
min@3↑	2.14	3.08	2.84	1.88	3.05	2.98	16.79	11.25	12.74	**1.39**	**1.78**	**1.66**
min@5↑	1.69	2.56	2.15	1.45	2.42	2.17	13.18	8.59	9.67	**1.27**	**1.53**	**1.47**
min@9↑	1.21	2.05	1.6	**1.06**	1.84	1.53	10.19	6.36	7.13	1.16	**1.35**	**1.3**

Table 7.4: Minimality results from each editor with all part-of-speech tags.

Another important observation for minimality, in general, is that MiCE generates much less minimal edits than Polyjuice. This can be explained up to a point by the fact that Polyjuice uses a random method in order to find where the changes should be made, and specifically for larger inputs this leads to exponentially larger search space. However, per Filandrianos et al. [20], and as we noticed in our experiments, **Polyjuice deletes over 70% of the original input** on the IMDb dataset and 50% of the original text on the NewsGroups dataset on the first two steps of edits. A factor that contributes to that behavior is our evaluation procedure. We opt for edits that have a different label than the original text, and this restriction combined with the "task-agnostic nature of Polyjuice" [20] forces it to delete large parts of the text in order to make more effective edits towards the predictor's flip. Another important factor, however, is the inner model used by Polyjuice, GPT-2, which is not constrained by any means, compared to T5 of MiCE which is a text-to-text model that can be fine-tuned for specific tasks like in our case. All of the above seem to add to the robustness issues identified for Polyjuice [20] [53]. In Figure 7.2.2 shown below, we present the behavior described by showing the mean number of the edited text's tokens in relation to the mean number of the input text's tokens. The figure idea derives from [20] and is adjusted to our results and editors.



(a) Mean number of tokens of the edited text compared to the mean number of tokens of the input on the IMDb dataset.



(b) Mean number of tokens of the edited text compared to the mean number of tokens of the input on the NewsGroups dataset.

Figure 7.2.2: Mean number of tokens of the edited text compared to the number of tokens of the input for the four editors.

7.2.1.2 Inconsistency

In Tables 7.5 and 7.6 we show the results of the inconsistency metric on our edits. Before, expressing our conclusions, we should refer to the practical meaning of this metric. Inconsistency of minimality expresses the mean number of words that the editor modifies on top of the words that would suffice in order to produce a valid counterfactual. This means that the minimum value of inconsistency a set of edits can have is 0.00, and corresponds to optimal edits made by the editor.

Observing Table 7.5, we can see that there are noticeable differences between the editors, which can be attributed to the methods and models each editor uses. **TextFooler is the most consistent** out of the editors, having very low inconsistency values, which hint that minimality rarely increases between steps. These low values help us explain once again that a largely constrained editor like TextFooler, can generate more consistent edits than MiCE and Polyjuice which seem more inconsistent editors. What seems to be the most important factor that explains this performance is the use of language models which have been proven to be more sensitive to input deviations [59].

Moreover, we should make a reference to the high value of inconsistency for Polyjuice in the first step of edits with the IMDd dataset. As we demonstrated above, Polyjuice erases a big part of the input text in the first steps, and therefore it is deemed inconsistent in the first step. However, this does not apply to the NewsGroups dataset, where the input contains text with almost 4 times fewer tokens (58 instead of 204 on IMDb). This signals that the Polyjuice editor is consistent for shorter inputs, which also explains its low inconsistency values in NewsGroups and all the steps in IMDb after the first, where the number of tokens is significantly decreased.

Table 7.5: Inconsistency results from each editor without targetting some part of speech.

	IMDb			
	MiCE	MiCERandom	Polyjuice	TextFooler
inc@1↑	0.86	2.42	6.21	**0.01**
inc@2↑	5.95	5.81	4.65	**0.33**
inc@3↑	4.65	6.37	3.98	**0.36**
inc@5↑	4.87	7.58	2.9	**0.47**
inc@9↑	4.73	8.11	2.22	**0.49**
	NewsGroups			
inc@1↑	1.23	2.66	0.53	**0.04**
inc@2↑	2.53	4.3	1.27	**0.36**
inc@3↑	2.44	4.37	1.28	**0.27**
inc@5↑	2.46	4.62	1.24	**0.27**
inc@9↑	2.42	4.94	1.2	**0.25**

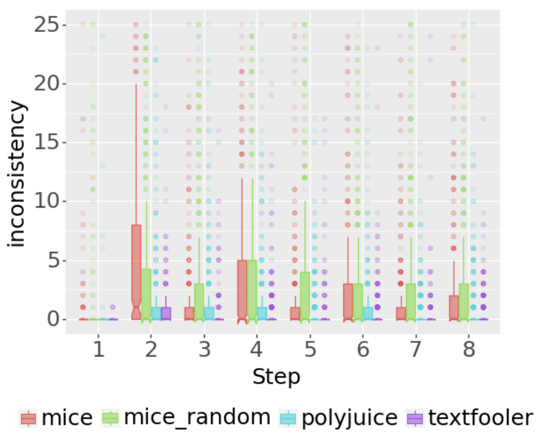


Figure 7.2.3: Boxplot of inconsistency for the IMDb dataset.

Another point worth discussing is the high value of inconsistency for MiCE on even numbers of steps. As it is also discussed in [20] even steps in our contrastive edits' system are the steps where we move from the original class of an input to a different one, and thus higher inconsistency in these steps indicates that the editor struggles to do so. On the other hand, **transitioning back to the original class is easier for the editor** as usually parts of the original input which contribute to the original class prediction still remain in the text sentence, leading to fewer edits needed to return to the starting label. This MiCE's tendency can be observed in the boxplot in Figure 7.2.3. In the qualitative results (7.2.3) we notice multiple occasions where there are noticeable text remnants of the input sentence that usually contribute towards the original class.

What is also worth mentioning is the increase in the values of inconsistency the more we continue on the feed-forward process for MiCERandom in both datasets. In this way, the inconsistency metric explains to us the potential existence of counterfactuals with lower minimality that the editor could but did not explore. These **unexplored states** by the editor, therefore, seem to increase as the steps increase because with random masking the editor does not have any knowledge as to what parts of the sentence are influential for the editor, and thus the editor makes multiple unnecessary changes that pile up after multiple steps of the feedback resulting in a significant deviation from the most minimal edit. However, we notice that this pattern does not repeat in Polyjuice which also uses random masking. This is probably due to the fact that after the first step in Polyjuice, the number of input tokens is significantly reduced, and therefore the editor stays more consistent with shorter sentences.

In general, **editors turn more consistent as the feedback steps increase**. After generating edits in the first steps, it seems that there are fewer modifications required in the next steps as there are multiple occasions where there is "text residue" from previous edits which assist in the predictor's outcome.

Targeting part-of-speech tags

After conducting our experiments with targeted POS tags, we notice that we enforce **all the editors to become far more consistent** than they are when they are not limited to a specific POS tag. We can characterize the edits with this method to be much closer to the most minimal edit possible, as almost all values are below 1, signaling a word-level distance of less than one token from the optimal edit. By this, we understand that the majority of the edits were indeed the most minimal edits possible, i.e. at most as minimal as the one of the previous step, but there were also some edits with higher inconsistencies. Especially for TextFooler, which achieves lower values than the other editors, we find that the percentage of sentences with a value of inconsistency of 0 is over 90% for odd steps and over 60% for even steps. This differentiation seems to also correlate with the high inconsistency values in the even steps of the feedback for MiCE. We analyze this behavior on all the editors and part-of-speech tags in the appendix. Finally, from the results in 7.6 the adjective is the part-of-speech tag with which the editors are generally the most consistent, followed by the verb and noun.

	IMDb											
	MiCE			MiCERandom			Polyjuice			TextFooler		
	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB
inc@1↑	0.13	0.75	1.0	0.47	0.58	0.69	4.7	1.39	1.33	**0.05**	**0.09**	**0.0**
inc@2↑	1.16	2.03	2.22	0.9	2.02	1.5	5.76	2.88	2.69	**0.58**	**0.56**	**0.62**
inc@3↑	0.99	1.89	1.95	0.88	2.21	1.52	4.76	2.39	2.15	**0.41**	**0.39**	**0.43**
inc@5↑	1.05	2.08	1.79	0.85	2.13	1.44	3.68	2.25	2.08	**0.38**	**0.42**	**0.5**
inc@9↑	1.06	1.99	1.73	0.73	2.15	1.3	2.88	1.9	1.79	**0.35**	**0.55**	**0.54**
	NewsGroups											
inc@1↑	0.3	0.5	0.26	0.22	0.47	0.32	2.19	1.11	0.91	**0.04**	**0.03**	**0.02**
inc@2↑	0.29	0.76	0.4	0.28	0.7	0.44	1.79	1.33	0.9	**0.13**	**0.23**	**0.18**
inc@3↑	0.26	0.69	0.37	0.24	0.67	0.41	1.65	1.26	0.8	**0.1**	**0.16**	**0.13**
inc@5↑	0.22	0.65	0.35	0.21	0.62	0.39	1.34	1.18	0.76	**0.08**	**0.16**	**0.14**
inc@9↑	0.17	0.6	0.31	0.18	0.55	0.34	1.19	1.03	0.69	**0.06**	**0.15**	**0.12**

Table 7.6: Inconsistency results from each editor with all part-of-speech tags.

To offer a comprehensive overview, through the inconsistency of minimality metric, we illustrate the importance of the feed-forward process on counterfactual editors. Through this iteration process, and with the use of inconsistency, we can notice the full range of capabilities of the editors as the one-step approach (represented by the @1 rows in our results) presents a "limited, dataset-dependent aspect" [20] of the editors' performance. Also, as we conducted our experiments with a goal of **minimal** contrastive counterfactual edits, inconsistency assists in revealing characteristics of the editor that minimality cannot and even depict the weaknesses of the editors in relation to minimality which is the studied metric. Taking a step forward, we prove that there are methods we can use to effectively "combat" some of these weaknesses, as we show that the editors become more consistent with the use of targeted part-of-speech tags.

7.2.1.3 Flip Rate

Another metric that acts as a useful tool in the explanations of the editors' inner decisions and results is the flip rate. As discussed in subsection 6.2.6, the flip rate indicates the percentage of successfully flipped inputs to a different class. Combined with the method of counterfactuals of counterfactuals, the flip rate reveals editors' imperfections and repeated patterns, which help us reach more stable conclusions.

Table 7.7: Flip rate results from each editor without targetting some part of speech.

	IMDb			
	MiCE	MiCERandom	Polyjuice	TextFooler
Flip Rate@1↑	**1.0**	0.9953	0.8747	0.6241
Flip Rate@2↑	0.8422	0.8419	**0.9107**	0.6984
Flip Rate@3↑	0.891	0.7163	**0.9392**	0.7193
Flip Rate@5↑	0.8677	0.6279	**0.9592**	0.7517
Flip Rate@9↑	0.8561	0.5674	**0.9668**	0.7865
	NewsGroups			
Flip Rate@1↑	0.89	0.79	0.726	**0.941**
Flip Rate@2↑	0.9188	0.715	0.9131	**1.0**
Flip Rate@3↑	0.8806	0.6395	0.9074	**1.0**
Flip Rate@5↑	0.8574	0.5972	0.9237	**1.0**
Flip Rate@9↑	0.8322	0.5444	0.9659	**1.0**

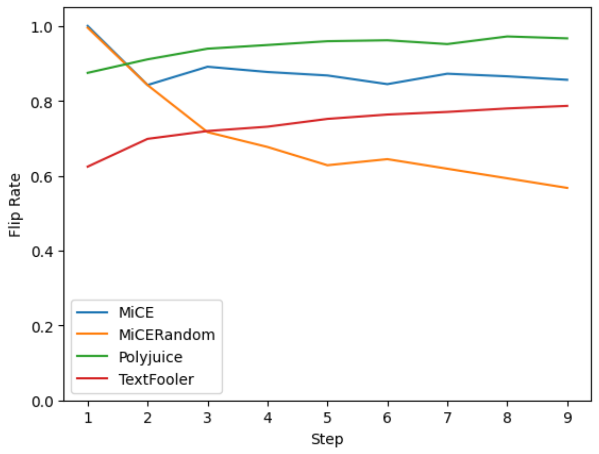


Figure 7.2.4: Flip rate for the IMDb dataset on the four editors.

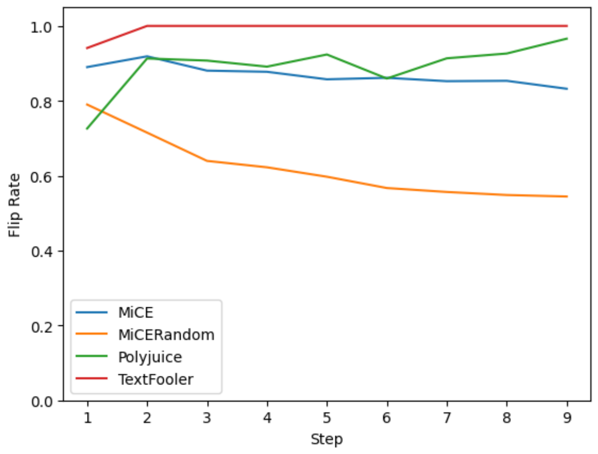


Figure 7.2.5: Flip rate for the NewsGroups dataset on the four editors.

For example, if we had generated edits with MiCE using a one-step approach, we would conclude that the editor always achieves the prediction flip on the IMDb dataset. However, by running the editor for 10 feedback steps, we see that this result varies depending on the input content or dataset. MiCE seems to not perform as perfectly after the first step where the input is perturbed by little or by a lot, confirming the sensitivity that language models are characterized with. Its flip rate continuously decreases until the last step of the feed-forward process, an observation that once again enforces the importance of analyzing editors' behavior further than one step of edits. In addition, the flip rate of step 2, which corresponds to the transition from the counterfactual to the original class, validates the difficulty of MiCE to return to the original class.

Trying to explain the results, we notice that **MiCE and MiCERandom flip fewer edits as the steps increase, while Polyjuice and TextFooler become more effective**. A potential explanation for the effectiveness of Polyjuice as the steps increase lies in the input size, which decreases largely after the first step. In this way, the language model used by Polyjuice, GPT2, has a smaller search space which enables the editor to reach successful states easier, and thus the desired contrastive counterfactual is achieved easier. On the other hand, another editor which uses a random way of masking input, MiCERandom, performs worse as the input is more and more perturbed, maybe because it uses a more constrained language model in T5 than GPT2.

In order to examine the reasons for these behaviors, we present Figure 7.2.6 which shows how the target class probability is affected as we move along the feedback steps. The flip of the sample is achieved when the target class probability has a value above 0.5. In this boxplot, we notice visible differences in the even and odd steps as we did in the inconsistency analysis, and also detect how TextFooler's target class probability improves towards the last steps, contrary to MiCERandom's where random masking and substitutions in the edits seem to create noise which progressively causes a decrease in the target class probability of the samples.

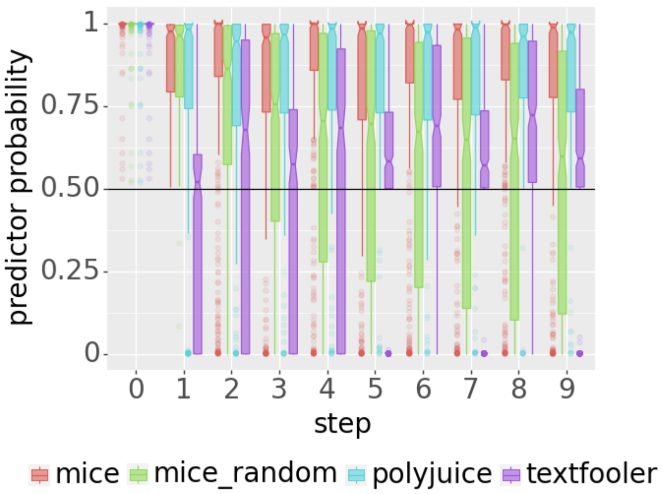


Figure 7.2.6: Probability of the target class on the IMDb dataset.

Targeting part-of-speech tags

Observing the flip rate results in Table 7.8 when we use our method of targeted part-of-speech tags, we see a noticeable **decrease in the flip rate of the first steps**. This is due to the fact that the editors are enforced by our method, to flip the predictor's outcome by performing perturbations only in a fraction of the tokens. As a result, many of the influential tokens for the predictor are not available for modification, and thus the flip is not accomplished.

It is also important to notice which part-of-speech tag performs better in terms of flip rate in each dataset. We observe that the prevalent part-of-speech tag in IMDb is the adjective, while in NewsGroups it is the noun. This difference in performance among the POS tags seems to be task-related as for example, IMDb is a dataset where the task is sentiment analysis, and the adjectives usually "carry" more importance toward the sentiment of a sentence, while verbs and nouns are more neutral. On the other hand, the NewsGroups dataset is used for the task of topic classification, and thus nouns are the words that can shift the meaning of a sentence from one topic to another. Hence, we notice that targeting adjectives in the NewsGroups dataset is a rather ineffective approach, as they evidently do not contribute much to different topic classifications. We notice many examples which highlight this explanation in the qualitative results section (7.2.3) and in A.3.

Table 7.8: Flip rate results from each editor wuth the use of part-of-speech tags.

	IMDb											
	MiCE			MiCERandom			Polyjuice			TextFooler		
	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB
Flip Rate@1↑	0.4419	0.4	0.393	0.4628	0.4442	0.4023	**0.8814**	**0.6977**	**0.807**	0.2581	0.2233	0.2395
Flip Rate@2↑	0.7917	0.7011	0.6608	0.801	0.6528	0.6571	0.9657	0.93	0.9251	**1.0**	**1.0**	**1.0**
Flip Rate@3↑	0.7241	0.6412	0.5549	0.601	0.4635	0.5523	**0.9809**	0.9785	0.9751	0.973	**1.0**	**0.9806**
Flip Rate@5↑	0.5395	0.5122	0.461	0.4731	0.3403	0.3373	0.9744	0.9852	**0.9935**	**1.0**	**1.0**	0.9901
Flip Rate@9↑	0.4054	0.3248	0.3284	0.2035	0.2246	0.2687	0.9819	**1.0**	**0.9967**	**1.0**	**1.0**	0.9899
	NewsGroups											
Flip Rate@1↑	0.272	0.571	0.335	0.247	0.5246	0.312	**0.492**	0.524	**0.472**	0.186	**0.577**	0.376
Flip Rate@2↑	0.3692	0.6803	0.4868	0.4179	0.6089	0.4697	0.6382	0.6469	0.6144	**0.8602**	**0.9879**	**0.9601**
Flip Rate@3↑	0.3202	0.6294	0.482	0.3441	0.5229	0.3579	0.7739	0.823	0.8034	**0.9563**	**0.9807**	**0.9584**
Flip Rate@5↑	0.216	0.5667	0.4012	0.2328	0.4372	0.2842	0.8031	0.8745	0.872	**0.98**	**0.9892**	**0.9735**
Flip Rate@9↑	0.155	0.448	0.2975	0.1307	0.2812	0.1923	0.8803	0.9167	0.9524	**0.9862**	**0.9945**	**1.0**

Regarding, the editors' performance, it is interesting to notice how consistent TextFooler remains after "deciding" which sentences it can flip successfully in the first step. In addition, Polyjuice also performs well in regard to the flip rate. In order to integrate the part-of-speech method, we enforced Polyjuice to make only single token substitutions (see *Polyjuice* in 6.3.4). In this way, we managed to increase the flip rate of Polyjuice. However, although we can identify qualitative examples where our POS tag method performs as expected, the results of Polyjuice are not representative as its inner language model, GPT-2, deletes big parts of the input text and also inserts text, practices that lead to the prediction's flip but not in a way that the targeted POS tag controls.

Total flip rate of each editor

As each sentence cannot be flipped by the editors for all POS tags, due to its content, we also present a combined flip rate for the first step of each editor that depicts what percentage of the input sentences were flipped with at least one of the targeted POS tags. In this manner, we can see a more comprehensive aspect of how our method performs in terms of flip rate. For this reason, we present Table 7.9 where we show these results.

Table 7.9: Combined flip rate results from each editor for all part of speech tags.

	IMDb			
	MiCE	MiCERandom	Polyjuice	TextFooler
Combined Flip Rate@1↑	**0.7697**	0.66	0.9093*	0.3558
	NewsGroups			
Combined Flip Rate@1↑	**0.769**	0.687	0.63	0.642

* Polyjuice is not a representative editor for the POS tag method in terms of flip rate

We observe that Textfooler struggles to flip the sentences' prediction when we specify a targeted POS tag in the IMDb dataset. Because IMDb samples feature longer texts, TextFooler's deterministic practices come across a wall as they are not able to insert generated text or delete existing text which might lead to the flip of the prediction. In addition, we see that MiCE with gradient mask is the most preferred editor when considering the flip rate with our method, as it is more efficient than the others.

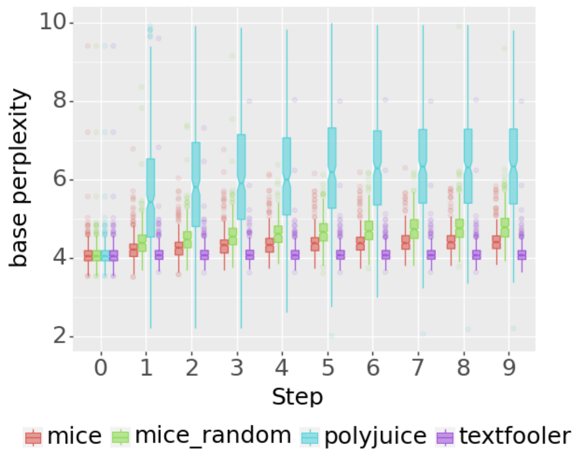
7.2.1.4 Base Perplexity and Fine Perplexity

In our experiments, we use two metrics to evaluate the fluency of the generated edits. The results for both of these metrics are shown in Table 7.10. For base perplexity, where lower values indicate more fluent text, we see that among the editors tested, TextFooler generates the most fluent text. In addition, specifically for TextFooler, we notice stability after multiple feedback steps, a result that corroborates highly with the consistency of the editor. As for MiCE, MiCERandom, and Polyjuice, we observe increased values of perplexity as the steps increase, and thus less fluent text. This result seems logical, as all three editors were much more inconsistent than TextFooler. The same results are observed in both IMDb and NewsGroups.

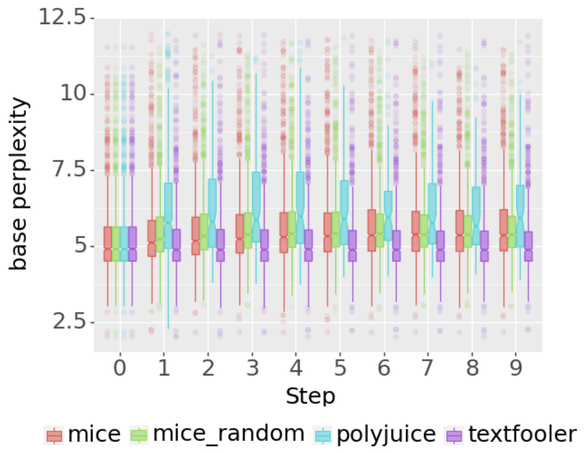
Moreover, we present Figure 7.2.7, where we show how base perplexity evolves for each editor for the feedback steps. What stands out in both figures is TextFooler's consistency and the deterioration of MiCE, MiCERandom, and Polyjuice in this order. TextFooler's behavior does not surprise us, as it is proved for this metric too, that the more constrained and deterministic an editor is the less it deviates from a specific distribution. On the other side, in a comparison between the two editors which utilize random masking, MiCERandom, and Polyjuice, we observe that the first performs better than the second as it uses a more specific way of text generation with the T5 Transformer. Polyjuice, as we saw, alters the input text largely on the first step, which certainly causes it to not be as efficient in terms of fluency.

Table 7.10: Base perplexity and fine perplexity computed for the four editors, on the IMDb and NewsGroups datasets, after the first step of the feedback (@1), and after 8 additional steps (@9).

	IMDb			
	MiCE	MiCERandom	Polyjuice	TextFooler
ppl-base@1↑	4.2546	4.4171	8.0101	**4.1178**
ppl-base@9↑	4.4512	4.8002	8.1586	**4.1161**
ppl-imdb@1↑	16.5315	**16.2096**	35.9845	18.0662
ppl-imdb@9↑	14.6069	**14.5973**	30.7309	17.9917
	NewsGroups			
ppl-base@1↑	5.6343	5.7294	7.226	**5.3049**
ppl-base@9↑	5.9288	5.7281	7.0663	**5.2383**
ppl-newsgroups@1↑	4.472	4.5179	6.4405	**4.2303**
ppl-newsgroups@9↑	4.6043	4.5173	5.1213	**4.2001**



(a) Base Perplexity on the IMDb dataset.

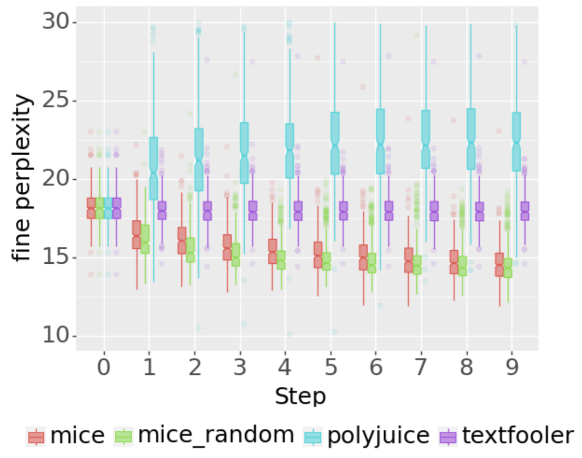


(b) Base Perplexity on the NewsGroups dataset.

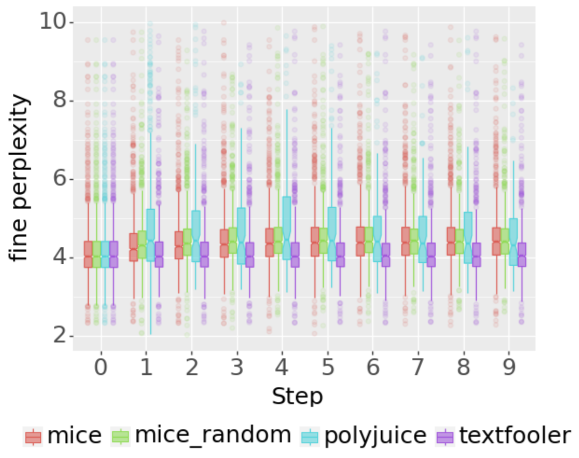
Figure 7.2.7: Base Perplexity

As to fine perplexity, as mentioned in 6.2.6, high values imply that the generated text is "diverse" when compared to the dataset used to fine-tune the language model whose perplexity we calculate. The quantitative results of *fine-ppl* are found in Table 7.10, but the main conclusions are mostly drawn based on Figure 7.2.8 where we show how fine perplexity shapes out as the steps increase.

In the figures, we notice that TextFooler once again is the most stable editor, but what stands out is the different patterns observed for the MiCE and MiCERandom editors. While the fluency for these editors seems to deteriorate when calculated with base perplexity, with fine perplexity the editors generate or seem to generate more fluent text. However, this behavior is only phenomenal as MiCE is trained exclusively on IMDb data, and thus the generated counterfactuals contain text that is closer to the distribution of the IMDb dataset. These results point to an "overfitting behavior" [20] of the MiCE and MiCERandom editors. On the other hand, Polyjuice which is trained on multiple datasets [102] generates text that is more diverse than that of MiCE.



(a) Fine Perplexity on the IMDb dataset.



(b) Fine Perplexity on the NewsGroups dataset.

Figure 7.2.8: Fine Perplexity

7.2.1.5 Targeting part-of-speech tags

When we use the editors with targeted POS tags, we notice that the editors produce **more fluent counterfactuals edits** than what they produced without their use. This happens because we constrain editors to substitute specific tokens, resulting in fewer modifications of the sentence, and hence the structure of the input sentence tends to be maintained in the counterfactual. This leads to lower values of base perplexity across all editors and part-of-speech tags, but also less diversity in the generated text. We present Table 7.11, which includes the results of base perplexity and fine perplexity, where the lower values of perplexity for each POS tag in each step are bolded.

Table 7.11: Base perplexity and fine perplexity computed for all editors with ADJ, NOUN and VERB part-of-speech tags, on the IMDb and NewsGroups datasets, after the first step of the feedback (@1), and after 8 additional steps (@9)

	IMDb											
	MiCE			MiCERandom			Polyjuice			TextFooler		
	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB	ADJ	NOUN	VERB
ppl-base@1↑	4.1748	4.278	4.2599	4.1968	4.3046	4.2639	5.5252	5.9737	6.0615	**4.091**	**4.1491**	**4.1166**
ppl-base@9↑	4.2493	4.3383	4.3154	4.2935	4.4345	4.3484	6.8178	6.9426	6.919	**4.1**	**4.1463**	**4.044**
ppl-imdb@1↑	**17.0192**	16.3556	16.6649	17.0942	**16.2741**	**16.6063**	22.2159	24.8189	24.9765	18.0654	18.1228	18.2363
ppl-imdb@9↑	16.1344	15.351	15.7454	**16.0439**	**14.7513**	**15.1843**	25.4604	25.6867	27.3699	18.0773	18.0512	18.0364
	NewsGroups											
ppl-base@1↑	5.331	5.4517	5.5203	**5.2863**	5.5137	5.3993	8.2656	7.4842	7.9413	5.3034	**5.2405**	**5.3797**
ppl-base@9↑	5.3465	5.4437	5.4864	5.2368	5.2812	5.294	9.2572	8.7526	8.5392	**5.0862**	**5.1495**	**5.2243**
ppl-news@1↑	4.2144	4.3176	4.3573	4.2003	4.3696	4.3367	7.1215	6.1762	6.5113	**4.1836**	**4.1996**	**4.2441**
ppl-news@9↑	4.2714	4.3539	4.3731	4.2053	4.2677	4.3365	7.7188	6.4435	6.4412	**4.0787**	**4.1495**	**4.1502**

Regarding the results, we notice that TextFooler is the editor that produces the most fluent edits with the use of part-of-speech tags. However, the editors of MiCE and MiCERandom, also seem to perform well and become more fluent and more consistent than their original versions. This result coincides with the lower inconsistency values of the editors when being constrained to a specific POS tag.

As to the editors' base perplexity results as the feedback steps increase, TextFooler seems to be very consistent whereas MiCE, MiCERandom, and Polyjuice become slightly less fluent. It is important to highlight the fact that the use of part-of-speech tags helps limit the overfitting behavior of the MiCE editors to the IMDb data, as we notice that the values of fine perplexity generally remain higher than those of the original versions of the MiCE editors. This advantage is gained by the substitutions of fewer tokens in the counterfactuals, especially with the adjective part-of-speech tag, whereby comparing Tables 7.11 and 7.10 we see that the fine perplexity values are much higher in the versions of the editors in which only adjectives are masked.

7.2.2 Experimenting on MiCE with beam-search

After experimenting with the four editors and the different part-of-speech tags, we conducted 12 more experiments on MiCE in both datasets to study what effect the model's search method has on the editor's results. The authors of MiCE [75] present their results using multinomial sampling as their search method. Contrary to greedy search (beam search with 1 beam) which always chooses a token with the highest probability as the next token, multinomial sampling is a search method that randomly selects the next token based on the probability distribution over the entire vocabulary given by the model [34]. In this way every token with a non-zero probability becomes a candidate token, reducing the risk of repetition.

In our experiments, we seek to evaluate if beam search, with a suitable number of beams, can outperform multinomial sampling for MiCE and with what consequences. In Tables 7.12 and 7.13, we notice that beam-search with 120 beams **outperforms** multinomial sampling in the IMDb dataset in terms of **both minimality and inconsistency**. In the NewsGroups dataset, we notice that it follows but it cannot perform better than the original version of MiCE. This discrepancy between the two datasets is mainly due to the different lengths of sentences in the two datasets. IMDb features more lengthy sentences than NewsGroups and so beam-search with a high number of beams is more effective, as it can capture more complex linguistic structures and patterns, leading to better coverage of the sentence space. On the other hand, the NewsGroups dataset contains shorter sentences, so multinomial sampling, which only explores a single token each time, performs better in keeping the structure of the sentence intact and allows for the possibility of more minimal edits.

It is interesting to observe how minimality and inconsistency values decrease as the number of beams increases. The algorithm explores a larger search space as the number of beams increases, thus, it considers more diverse possibilities for each decoding step, which can lead to a broader exploration of alternative phrases, structures, or wording. However, MiCE always sorts the candidate edits based on the minimality score and selects the most minimal, so the more we increase the number of beams, the more potential options there are for the editor to choose from, and the higher the possibility of selecting a more minimal edit than with a lower number of beams.

	IMDb						
	MiCE	**Greedy**	**5 beams**	**15 beams**	**30 beams**	**60 beams**	**120 beams**
min@1↑	41.78	76.61	71.32	65.75	63.32	53.77	**34.4**
min@2↑	24.91	48.55	43.8	39.11	38.58	32.09	**19.9**
min@3↑	22.11	43.98	38.08	33.41	34.33	28.02	**16.23**
min@5↑	17.42	34.95	28.8	25.52	26.14	20.96	**11.99**
min@9↑	13.65	26.66	21.8	19.21	19.16	15.18	**8.75**
	NewsGroups						
min@1↑	**11.78**	20.45	17.48	15.53	14.4	13.93	12.59
min@2↑	**8.22**	17.34	13.97	11.85	10.7	9.91	8.93
min@3↑	**7.57**	16.4	12.69	10.71	9.67	8.99	7.78
min@5↑	6.51	15.37	11.12	9.16	8.09	7.48	**6.33**
min@9↑	5.71	14.44	9.49	7.54	6.52	6.28	**5.05**

Table 7.12: Minimality results for all the variations of MiCE we experimented with.

	IMDb						
	MiCE	**Greedy**	**5 beams**	**15 beams**	**30 beams**	**60 beams**	**120 beams**
inc@1↑	0.86	3.5	3.59	1.91	3.35	2.27	**0.57**
inc@2↑	5.95	13.73	11.75	9.25	11.85	8.78	**3.29**
inc@3↑	4.65	11.62	10.09	8.2	9.91	7.5	**3.01**
inc@5↑	4.87	11.38	9.65	8.3	9.56	7.03	**3.04**
inc@9↑	4.73	10.51	9.0	8.19	8.34	6.43	**2.9**
	NewsGroups						
inc@1↑	**1.23**	4.32	3.64	3.04	2.74	2.36	2.13
inc@2↑	**2.53**	5.19	4.41	3.97	3.69	3.52	2.88
inc@3↑	**2.44**	5.18	4.31	3.83	3.43	3.21	2.67
inc@5↑	**2.46**	5.29	4.24	3.69	3.23	3.09	2.48
inc@9↑	2.42	5.33	3.99	3.38	2.98	2.9	**2.31**

Table 7.13: Inconsistency results for all the variations of MiCE we experimented with.

Regarding inconsistency, it is important to note that the increased inconsistency pattern in even steps is repeated in these experiments too, with Figure 7.2.9 illustrating this behavior in a noticeable way. This signifies that even if we change the generation method MiCE still struggles to transition to the contrast class.

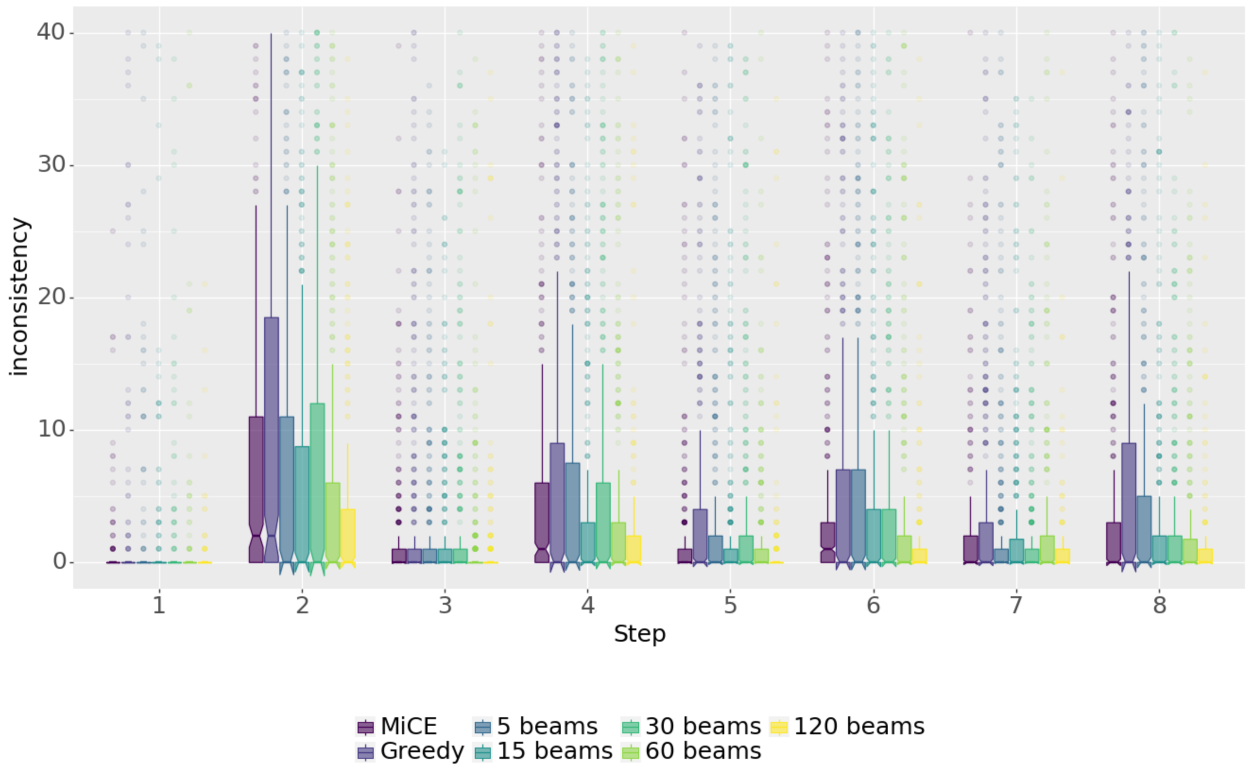


Figure 7.2.9: Inconsistency of minimality for different beams.

As far as **flip rate** is concerned, we notice that MiCE with multinomial sampling is the best option for the first step in IMDb but is outperformed by the 120 beams version of MiCE in later steps. The big number of possible candidate edits explored by the large search space of beam-search with 120 beams seems to enable the editor to detect counterfactuals that are not accessible to MiCE when using multinomial sampling.

A surprising result is that the implementation of greedy search achieves very high flip rates in both datasets, which denotes that selecting words with the highest probability leads to more aggressive edits which eventually flip the predictor's outcome, but with the cost of less minimal edits as the binary search algorithm of MiCE selects bigger fractions of the total tokens to be substituted. Another noticeable pattern in the results is that the flip rate in IMDb decreases as the number of beams increases.

Table 7.14: Flip rates using beam-search for the IMDb and NewsGroups datasets.

	IMDb						
	MiCE	**Greedy**	**5 beams**	**15 beams**	**30 beams**	**60 beams**	**120 beams**
Flip Rate@1↑	**1.0**	0.9977	0.8907	0.8767	0.8628	0.8605	0.9977
Flip Rate@2↑	0.8422	**1.0**	0.9186	0.893	0.8698	0.8837	0.958
Flip Rate@9↑	0.8561	0.8735	0.8465	0.807	0.7972	0.7786	**0.9568**
	NewsGroups						
Flip Rate@1↑	0.89	0.865	0.887	0.881	0.882	0.891	**0.9**
Flip Rate@2↑	**0.9188**	0.896	0.8811	0.8655	0.8927	0.8637	0.8758
Flip Rate@9↑	**0.8322**	0.7894	0.7368	0.6876	0.6875	0.701	0.7104

In order to evaluate the results of these experiments in terms of **fluency** we present Figures 7.2.10 and 7.2.11 where we show how the base and fine perplexity are influenced by the number of beams. In Figure 7.2.10 we see that the fluency of all editors deteriorates after feedback, with the original version of MiCE deteriorating the less, and thus being the most fluent version of the editor.

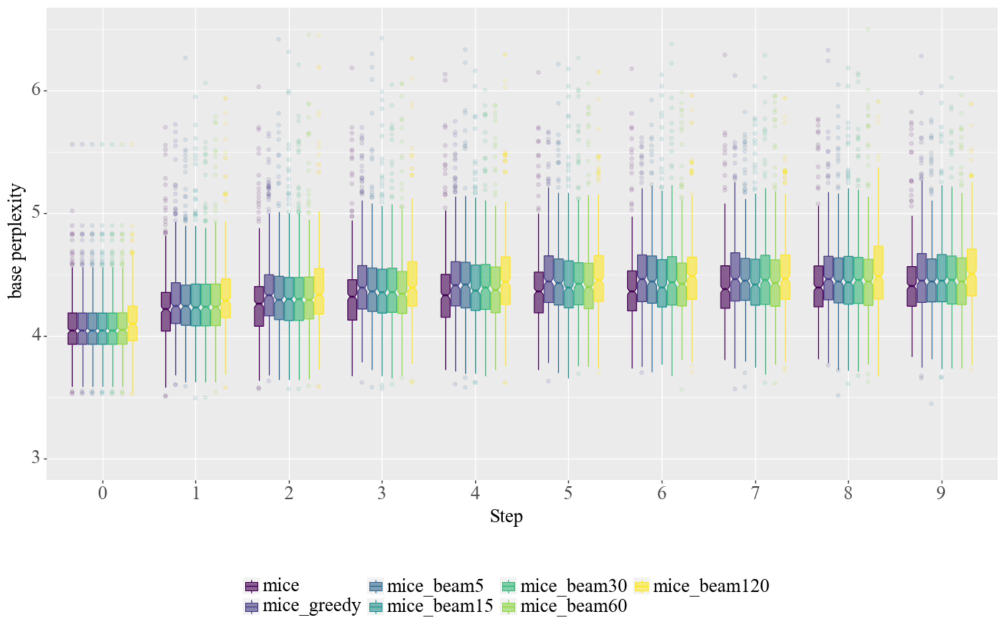


Figure 7.2.10: Base perplexity for the IMDb dataset.

In Figure 7.2.11 we notice that the versions of MiCE which use beam-search seem to also have an overfitting behavior to the IMDb data, but to approach the distribution of IMDb much faster than MiCE with multinomial sampling. We see that the less the number of beams is, the more aggressively this overfitting occurs, hinting that we move closer to the dataset's distribution when we perturb sentences with the sole criterion being the high probability of the next word or sequence of words.

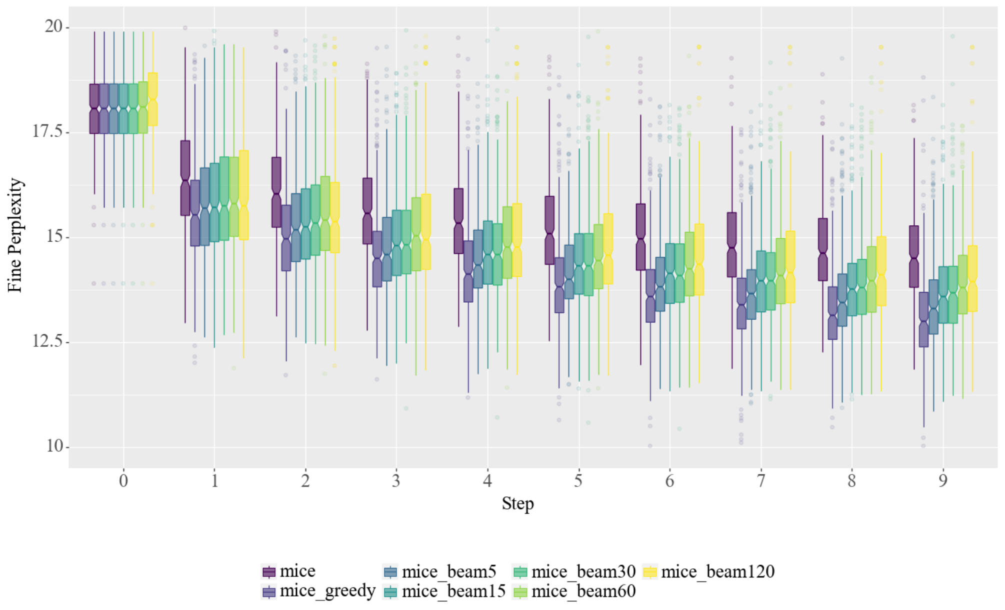


Figure 7.2.11: Fine perplexity for the IMDd dataset.

Finally, it is important to mention that in order to experiment with such high numbers of beams as 120, we have to accept some trade-offs such as the experiment's duration and the computational requirements which we discuss further in 7.2.4. To put the problem into perspective, MiCE requires approximately 15 and 21 hours in a GPU to complete a 10-step feed-forward experiment for the IMDb and NewsGroups datasets accordingly. For the version of MiCE with 120 beams, we need approximately 70 and 80 hours in a GPU accordingly for the same experiment.

7.2.3 Qualitative Results

In this part of the thesis, we present multiple qualitative results which help observe behaviors, patterns, and particularities in the generated text of each editor. Because of the nature of the results and the big number of different versions of editors, we try to showcase results that present the most interest to the conclusions drawn, but also provide a dedicated section in the appendix which features multiple qualitative results.

Also, through the following examples we try to explain the editors' characteristics with examples that are not lengthy, in order to focus on the points being made and not the specific nature of each example. However, most samples of the IMDb dataset are rather long in size, and so to ensure completeness we also present qualitative results in the appendix with lengthier sentences.

In the following tables which present the counterfactual edits generated from our experiments, the words which are altered in each edit are shown in bold.

Gradient vs random masking

In the example of Tables 7.15 and 7.16, we can see the main difference between the two editors and the "strong" effect it can have on our edits after multiple steps. Specifically, MiCE which uses gradient masking on the sentence, essentially "knows" which tokens are more important for the editor and substitutes just them to achieve the flip. In this way, we see why **MiCE is much more consistent** than MiCERandom. On the other hand, MiCERandom tries to flip the predictor's outcome by modifying random tokens. In this way, MiCERandom not only generates edits that are less minimal but also less fluent, as we can observe that the text gets more and more distorted.

Table 7.15: The generated counterfactuals for 10 steps for MiCE on an IMDb sample.

Step	**MiCE**
1	What a script, what a story, what a **mess**!
2	What a script, what a story, what a **filmmaker**!
3	What a script, what a story, what a **disappointment**!
4	What a script, what a story, what a **thriller**!
5	What a script, what a **script**, what a **suck**!
6	What a script, what a script, what a **director**!
7	What a script, what a script, what a **mess**!
8	What a script, what a script, what a **script**!
9	What a script, what a script, what a **disaster**!
10	What a script, what a script, what a **script**!

Table 7.16: The generated counterfactuals for 10 steps for MiCERandom on an IMDb sample.

Step	**MiCERandom**
1	What **a** script**, what a** story, **what** a mess!
2	What **great** script **&** story, **but** a mess!**!!!!!!**
3	What **fantastic** script & story, but **GREAT FUN**!!!!!!!**!!**
4	What fantastic script & story but **NO** FUN!!!!!!!!!
5	What fantastic script **. Unbelievable** story but **amazing story**!!!!!!!!!
6	What **unbelievable** script . Unbelievable **unbelievable** story!!!!!!!!!
7	What **incredible** script . Unbelievable unbelievable!!!!!!!!!**!!!!!!**
8	What incredible script . Unbelievable unbelievable **acting.**!!!!!!!!!!!!!!!
9	What incredible script . **Un believably** unbelievable acting.!!!!!!!!!!!!!!!
10	What incredible script . Un **believ** believably unbelievable acting.!!!!!!!!!!!!!!!

Another phenomenon we notice, which has also been noted by Filandrianos et al. [20] is the introduction of "erroneous whitespaces" which multiplies after some steps. This seems to be caused by the inner mechanisms or model of MiCE as it does not happen with Polyjuice or TextFooler. Also, another common faulty behavior that we notice is the splitting of tokens and hallucinations, like in steps 9 and 10 of the edits with MiCERandom and the word "Unbelievably".

After all 10 steps, the counterfactual edit of MiCERandom has shifted largely from the original input while MiCE has remained much more consistent, despite the repetition of the word "Script". This happens because the predictor consistently ranks the last token of the sentence as the most influential, leading MiCE to perturb the sentence mainly by substituting only this token.

Comparing the four editors

In Tables 7.17 and 7.18 we present an example of an input from the IMDb dataset and how the counterfactuals vary from editor to editor. Firstly, we observe that **MiCERandom and Polyjuice do not stay consistent** with the tokens they select to modify, hinting at their randomness. Polyjuice seems to make the most aggressive perturbation amongst the others in step 3, where the whole text is modified, generating a totally different sentence. This behavior of Polyjuice is common and validates the increased perplexity values as more diverse text is generated. On the other hand, TextFooler does not influence the sentence's structure at all, as it opts for very "strict" modifications. The counterfactuals of TextFooler in Table 7.18 feature an inconsistency value of 0 across all steps and validate the editor's consistency. For MiCE, the added whitespaces pattern, and also the splitting of words, take place again in the late steps of this edit.

Table 7.17: The generated edits for 10 steps from MiCE and MiCERandom editors.

Step	**MiCE**	**MiCERandom**
1	Read the book, **forget** the movie!	Read the book, forget the **movie**!
2	Read the book, **enjoy** the movie!	Read the book, forget the **computer**!
3	Read the book, **not** the movie!	Read the book **then** forget the computer!
4	Read the book, not the **screen**!	Read the book then forget the **TV**!
5	Read the book, not the **film**!	Read the book **,** forget the **movie**!
6	Read the book, not the **dictionary**!	Read the book , forget the **blah**!
7	Read the book, not the **movie**!	Read the book , forget the **duh**!
8	Read the book, not the **snob**!	Read the book , forget the **dash**!
9	Read the book, not the **Sn atchbox**!	**...**the book , forget the dash!
10	Read the **Book**, not the Sn **atchers**!	... **classic** book, forget the dash!

Table 7.18: The generated edits for 10 steps from Polyjuice and TextFooler editors.

Step	**Polyjuice**	**TextFooler**
1	**Read** the book, **forget** the movie!	Read the book, **forget** the movie!
2	**Watch** the book, **watch** the movie!	Read the book, **missed** the movie!
3	**Don't waste your time**, **or your money.**	Read the **accountancy**, missed the movie!
4	**You do**n't waste your time, or money.	**Dsl** the accountancy, missed the movie!
5	**Spend** your time, or money.	Dsl the accountancy, **forget** the movie!
6	Spend your time, **money,** or money.	Dsl the accountancy, **missed** the movie!
7	Spend your time, money or money **wisely**.	Dsl the accountancy, **forget** the movie!
8	Spend your time, money**, and** wisely.	Dsl the accountancy, **missed** the movie!
9	Spend your time, money, and **dollars** wisely.	Dsl the accountancy, **forget** the movie!
10	Spend your time, money and **time on this masterpiece**.	Dsl the accountancy, **missed** the movie!

At this point, we introduce Table 7.19 which presents the counterfactuals edits for 10 steps when we use beam-search on MiCE with 120 beams.

Table 7.19: The generated counterfactuals for 10 steps for MiCE that uses beam-search with 120 beams.

Step	**MiCE with 120 beams**
1	Read the book, **forget** the movie!
2	Read the book, **savor** the movie!
3	Read the book, **avoid** the movie!
4	Read the book, **LOVE** the movie!
5	Read the book, **avoid** the movie!
6	Read the book, **enjoy** the movie!
7	Read the book, **skip** the movie!
8	Read the book, **LOVE** the movie!
9	Read the book, **skip** the movie!
10	Read the book, **LOVE** the movie!

Although, both this version of the editor and MiCE use gradient masking, their difference lies in the fact that **this editor explores many more different substitutions** as it selects the token which generates the most probable sequence. Hence, we notice that in step 3 and onwards this editor is able to select verbs that carry the main sentiment of the sentence, and then easily flip the prediction of the sentence. In contrast, MiCE modifies the sentence with the word "not" in step 3, which seems to not be ideal and to affect the edits' generation later on. Moreover, in this example we observe a repetition of the words "LOVE", "avoid" and "skip". This demonstrates that when these words are selected by the editor, the predictor is strongly influenced toward the targeted label.

In addition, in Tables 7.20 and 7.21 we can observe that the edits created on a NewsGroups sample with MiCE and beam search of 120 beams are evidently more minimal than those of the original MiCE editor. Similar behavior is seen across multiple examples of both IMDb and NewsGroups, and it seems that the exponentially larger explored search space leads the editor that utilizes beam search to select edits that make the editor more minimal in general.

Table 7.20: The generated edits for 10 steps from MiCE editor for a NewsGroups sample.

Step	**MiCE**
1	Does **anyone** know of **any free X-servers** for **PCs**, **preferably** that run under **MS Windows**? **THANKS.**
2	Does **anybody** know of **MS-DOS v.3.1 games** for **Windows**, **Windows and DOS** that run under **DOSA**? **- SteveS.**
3	Does anybody know of MS-DOS v.3.1 games for Windows, Windows and DOS that run under **XWindow**? - SteveS.
4	Does anybody know of **eShop for the Sony 3.1.3.1 editions for Apple computers, TVs and other** games that run **on OS/2 or 3.2**? **E - mail reply.**
5	Does anybody know of **the available ROMs** for the Sony 3.1.3.1 **?** for **DOS** and other **computers? Depending** on **the version of the 3.3.1** or **3.2.3.1, please** E - mail.
6	**Where does a 3.1. 3.2 env. land**? Depending on the **state** of the **3. 2** or 3.2.3.1, **the** E - mail **address needs to be included**.
7	Where does a 3.1. 3.2 **.3.1 biker**. land? Depending on the state of the 3. 2 or 3.2.3.1, the E - mail address needs to be included.
8	**'unk' ' I am currently implementing version** 3.1. 3.2 **. The following is a common question: How does the word "alienation" work**? **To use** the **word p.3.1**, the **_alert_ name of umanish** needs to be **enunciated**.
9	**Hello.** I am **looking at p.** 3.1. 3.2 . The following is a common question: How does the **"president" of Israel mean**? To **translate** the word p.3.1, the **_ bereft_ symbol** of **Israel** needs to be enunciated.
10	Hello. I am looking at p. 3.1. 3.2 . The following is a common question: How does the **" pst_dft"** of **alt.atheism** mean? To translate the word p.3.1, the **_ft_ definition** of **alt** needs to be **en negated**.

Table 7.21: The generated edits for 10 steps from MiCE editor when it uses beam search with 120 beams for a NewsGroups sample.

Step	**MiCE with 120 beams**
1	Does anyone know of any free **X**-servers for PCs, preferably that run under MS Windows? THANKS.
2	Does anyone know of any free **DOS**-servers for PCs, preferably that run under MS Windows? THANKS.
3	Does anyone know of any free **graphic**-servers for PCs, preferably that run under MS Windows? THANKS.
4	Does anyone know of any free **DOS**-servers for PCs, preferably that run under MS Windows? THANKS.
5	Does anyone know of any free **graphic**-servers for PCs, preferably that run under MS Windows? THANKS.
6	Does anyone know of any free **DOS**-servers for PCs, preferably that run under MS Windows? THANKS.
7	Does anyone know of any free **graphic**-servers for PCs, preferably that run under MS Windows? THANKS.
8	Does anyone know of any free **DOS**-servers for PCs, preferably that run under MS Windows? THANKS.
9	Does anyone know of any free **graphic**-servers for PCs, preferably that run under MS Windows? THANKS.
10	Does anyone know of any free **DOS**-servers for PCs, preferably that run under MS Windows? THANKS.

For the **NewsGroups** dataset, we present the examples in Tables 7.22, 7.23, 7.24 and 7.25.

What we notice at first is that both the input and generated text are more diverse than that of the IMDb dataset. As NewsGroups is a text dataset with text samples classified into multiple topics, this is an expected difference between the two datasets.

Regarding the editors, it is important to notice the consistency of MiCE and TextFooler, contrary to MiCERandom and Polyjuice which perturb the sentences much more intensely. Moreover, in this example, we can see that TextFooler essentially "fools" the predictor by mainly substituting words with their synonyms. The use of embeddings and counterfactual examples like this one seem to **unveil one of the predictor's vulnerabilities**, which is its sensitivity to very small changes in a sentence. The fact that the modification "arrived" → "occurred" results in a change of class mainly means that the predictor needs more specific training for each class, so it cannot be "fooled" by such methods. In general, we see the same erroneous patterns as in IMDb, e.g. added whitespaces in MiCE editors.

Table 7.22: The generated edits for 10 steps from MiCE editor for a NewsGroups sample.

Step	**MiCE**
1	I heard he had a **strained abdominal muscle** or something like that.
2	I heard he had a **car** or something like that.
3	I heard he had a car or something like **a satellite**.
4	I heard he had a car or something like a **Porsche**.
5	I heard he had a car or something like a **rocket**.
6	I heard he had a car or something like a **Honda**.
7	I heard he had a **bible** or something like a **bible**.
8	I heard he had a **scroll** or something like a bible.
9	I heard he had a **bible** or something like a bible.
10	I heard he had a bible or something like a **bibleshelf**.

Table 7.23: The generated edits for 10 steps from MiCERandom editor for a NewsGroups sample.

Step	**MiCERandom**
1	I heard **he** had a **strained abdominal muscle** or something like that.
2	I heard **that Honda** had a **factorytrained brake guy** or something like that.
3	I heard that **DRAM makes** a **trained sound** like that.
4	I heard that DRAM makes a trained **GPU** like that.
5	I heard **AMD** makes **new processor chips.........................!**
6	I heard AMD makes new **x86** chips.........................
7	I heard AMD makes new x86 chips.........................
8	I heard AMD makes **386P** chips.........................
9	I heard AMD makes 386P chips.........................
10	I heard AMD makes 386P chips.........................

Table 7.24: The generated edits for 10 steps from Polyjuice editor for a NewsGroups sample.

Step	**Polyjuice**
1	**I** heard he had a **strained abdominal muscle** or something like that.
2	**well. . . i** heard he had a **hamstring** or something like that.
3	well. . . i heard he had a hamstring or something like **pinprick** .
4	well. . . i heard he had a hamstring or something **of an injury**.
5	well. . . i heard he had a **little problem** or something of an injury.
6	well. . . i heard he had a little problem or something **with** an **old stick** .
7	well. . . i heard he had a **slight** problem or something with an old stick.
8	well. . . i heard he had a slight problem **during the run** an old stick .
9	well. . . i heard he had a slight problem during **his** run**,** an old **lung complaint** .
10	well**,** i heard he **suffered** a slight problem **prior to** his **move on**, **a** lung complaint .

Table 7.25: The generated edits for 10 steps from TextFooler editor for a NewsGroups sample.

Step	**TextFooler**
1	**I** heard he had a **strained abdominal muscle** or something like that.
2	**me** heard he had a **jerked underside biceps** or something like that.
3	me heard he had a **tensed** underside biceps or something like that.
4	me heard he **brought** a tensed underside biceps or something like that.
5	me heard he **occurred** a tensed underside biceps or something like that.
6	me heard he occurred a **glared** underside biceps or something like that.
7	me heard he occurred a **trembled** underside biceps or something like that.
8	me heard he **arrived** a trembled underside biceps or something like that.
9	me heard he **occurred** a trembled underside biceps or something like that.
10	me heard he **arrived** a trembled underside biceps or something like that.

Polyjuice truncates large parts of the input text

In Tables 7.18 and 7.27 and 7.28, we see three examples from the IMDb dataset where we notice Polyjuice removes the majority of the input text, changing dramatically the structure and semantics of the sentence. This validates the behavior of Polyjuice we investigate in the Minimality subsection of 7.2.1, that Polyjuice cuts an average of more than 70% of the input text.

Table 7.26: An example from IMDb where Polyjuice truncates most of the input text.

Step	Polyjuice
1	This movie was **horrible** and corny. James Agee is rolling in his grave.**This movie was nothing at all like the book and made a mockery of it. No one should see this movie unless they want to gag.**
2	This movie was **fun** and corny. James Agee is rolling in his grave.
3	This movie was **boring** and **depressing**. James Agee is rolling in his grave.
4	This movie was **good** and **moving**. James Agee is **at** his **best**.

Table 7.27: A second example where Polyjuice erases most of the input text.

Step	Polyjuice
1	**Wonderful songs, sprightly animation and authentic live action make** this **a classic adaptation of a classic tale. A nice British feel which sets it apart and above from the standard, saccharine sweet Disney cartoons.**
2	**Don't bother to watch** this **movie with all the friends**.
3	**Do** bother to watch this movie with all the friends.
4	**Not** bother to watch this movie with all the friends.

Table 7.28: An example from the NewsGroups dataset where Polyjuice truncates most of the input text.

Step	Polyjuice
1	**On all 1.44Mb drives (both Mac and PC), the disk spins at a constant RPM. On 800k Mac disk drives, the spin rate of the disk is varied so that the tracks pass under the head at a constant speed; a slower rotation for the outer tracks, and a faster rotation for the inner tracks.** A PC needs **special controller hardware** to make this happen.
2	A PC needs **to be able** to make this **game** happen.
3	A PC needs to be **installed** to **run** this game happen.
4	A PC needs to be installed to **play** this game **correctly** .

Evaluating targeted part-of-speech tags

In Tables 7.29 and 7.30, the targeted part-of-speech tag is the noun in a NewsGroups sample. As the NewsGroups dataset is used for topic classification, nouns play a significant role in the predictor's outcome. From the displayed sequences of edits, we notice exactly that. The examples demonstrate how effective the editors are when we target a specific POS tag, and also depict the significant decrease of minimality and subsequent increase of consistency for the editors.

For the MiCE editor, we notice that although it presents very consistent behavior until the 6th step, in the 7th step it perturbs most of the sentence's text, altering the meaning largely. However, this also potentially shows that a non-constrained editor generates more diverse text. On the other hand, MiCE NOUN and TextFooler NOUN both remain consistent, but potentially create less diverse edits.

Table 7.29: An example from the NewsGroups dataset for MiCE and MiCE when we target nouns for modification.

Step	MiCE	MiCE NOUN
1	This is very curious being that they are both built by **Mercury** in the very same factory. Steve	This is very curious being that they are both built by Mercury in the very same **factory**. Steve
2	This is very curious being that they are both built by **Apple** in the very same factory. Steve	This is very curious being that they are both built by Mercury in the very same **company**. Steve
3	This is very curious being that they are both built by **NASA** in the very same factory. Steve	This is very curious being that they are both built by Mercury in the very same **era**. Steve
4	This is very curious being that they are both built by **Apple** in the very same factory. Steve	This is very curious being that they are both built by Mercury in the very same **company**. Steve
5	This is very curious being that they are both built by **Tesla** in the very same factory. Steve	This is very curious being that they are both built by Mercury in the very same **spacecraft**. Steve
6	This is very curious being that they are both built by **Compaq** in the very same factory. Steve	This is very curious being that they are both built by Mercury in the very same **company**. Steve
7	**The only difference between Microsoft Windows and DOS** is **the fact** being that they are both **MS-Windows Comp and** the very same **OS**. Steve	This is very curious being that they are both built by Mercury in the very same **year**. Steve
8	The only difference between **Canon** and **Sony** is the fact being that they are both **Canon** and the **CMOS are the** same. Steve	This is very curious being that they are both built by Mercury in the very same **company**. Steve
9	The only difference between Canon and **CMOS** is the fact being that they are both Canon and the CMOS are the same. Steve	This is very curious being that they are both built by Mercury in the very same **year**. Steve
10	The only difference between **NES** and **NT** is the fact being that they are both **PS/2 compatible** and the **motherboards** are the same. Steve	This is very curious being that they are both built by Mercury in the very same **company**. Steve

Table 7.30: An example from the NewsGroups dataset for TextFooler with the targetted part-of-speech being the noun.

Step	TextFooler NOUN
1	This is very curious **being** that they are both built by Mercury in the very same factory. Steve
2	This is very curious **continual** that they are both built by Mercury in the very same factory. Steve
3	This is very curious continual that they are both built by Mercury in the very same **factories**. Steve
4	This is very curious **indefatigable** that they are both built by Mercury in the very same factories. Steve
5	This is very curious indefatigable that they are both built by Mercury in the very same **fabrication**. Steve
6	This is very curious indefatigable that they are both built by Mercury in the very same **manufacture**. Steve
7	This is very curious indefatigable that they are both built by Mercury in the very same **fabrication**. Steve
8	This is very curious indefatigable that they are both built by Mercury in the very same **manufacture**. Steve
9	This is very curious indefatigable that they are both built by Mercury in the very same **fabrication**. Steve
10	This is very curious indefatigable that they are both built by Mercury in the very same **manufacture**. Steve

In Tables 7.31 and 7.32, we present an example for MiCE and TextFooler when targeting the 'ADJ' POS tag, each one with 10 steps of counterfactual edits on the same sample.

Table 7.31: An example from the IMDb dataset for MiCE with the targetted part-of-speech being the adjective.

Step	MiCE ADJ
1	The **biggest** heroes, is one of the **greatest** movies ever. A good story, great actors and a brilliant ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
2	The **great carrier,** heroes, is one of the **worst horror** movies ever. A good story, great actors and a brilliant ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
3	The great carrier, heroes, is one of the **best** horror movies ever. A good story, great actors and a **surprisingly satisfying** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
4	The **original** carrier, heroes, is one of the best horror movies ever. A good story, great actors and a surprisingly **predictable** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
5	The original carrier, heroes, is one of the best horror movies ever. A good story, great actors and a surprisingly **satisfying** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
6	The **liminal** carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's **romplimlim** carrier.
7	The **romplimpig** carrier, heroes, is one of the **best** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's **romplimpig** carrier.
8	The romplimpig carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.
9	The romplimpig carrier, heroes, is one of the **greatest** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.
10	The romplimpig carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.

Table 7.32: An example from the IMDb dataset for TextFooler with the targetted part-of-speech being the adjective.

Step	TextFooler ADJ
1	The **biggest** heroes, is one of the **greatest** movies ever. A **good** story, **great** actors and a **brilliant** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
2	The **longer** heroes, is one of the **more** movies ever. A **ok** story, **dramatic** actors and a **creditable** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
3	The longer heroes, is one of the more movies ever. A **okay** story, dramatic actors and a creditable ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
4	The longer heroes, is one of the more movies ever. A okay story, dramatic actors and a creditable ending is what makes this film the jumping start of the director Thomas Vinterberg's **unbelievable** carrier.
5	The longer heroes, is one of the more movies ever. A okay story, dramatic actors and a **laudable** ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
6	The longer heroes, is one of the **further** movies ever. A okay story, dramatic actors and a laudable ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
7	The longer heroes, is one of the further movies ever. A okay story, dramatic actors and a **praiseworthy** ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
8	The longer heroes, is one of the further movies ever. A okay story, **disastrous** actors and a praiseworthy ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
9	The longer heroes, is one of the further movies ever. A okay story, **catastrophic** actors and a praiseworthy ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.
10	The **plus** heroes, is one of the **alternatively** movies ever. A okay story, catastrophic actors and a praiseworthy ending is what makes this film the jumping start of the director Thomas Vinterberg's unbelievable carrier.

In the examples above, we observe how the targeted part-of-speech tag method greatly decreases the minimality of the counterfactual edits and thus leads the editors to be more consistent. In addition, we notice that even though the sample is not short, the editors achieve to flip the prediction by only modifying one or two tokens. Hence, the constraints that we enforce on the editors make them perturb the input sentence more aggressively in order to achieve the target label. It is also important to note that TextFooler's changes seem minor to the human eye in terms of altering the meaning of the sentence. For example, from Step 5 onwards in Table 7.30 the swapping of the words "fabrication" and "manufacture", which are close in embedding distance and practically synonyms, achieves the flip of the predictor's outcome. In this way, TextFooler reveals the vulnerability of the predictor to small changes.

Moreover, in Tables 7.33 and 7.34 we present some examples from the NewsGroups dataset where we observe the differences in the generated edits depending on the targeted part-of-speech tag. In this way, we notice that the targeted part-of-speech tags provide us a level of control over the generated counterfactual edits and at the same time ensure high consistency of the edits. In addition, these perturbations show us the flexibility of the edits that the method ensures by modifying different POS tags each time.

Furthermore, with the targeted part-of-speech tag method we validate both in this and the examples above that the fluency of the generated text generally remains stable as we perturb the sentence, so the linguistic quality of the counterfactual edits is in most cases adequate.

Table 7.33: An example from the NewsGroups dataset for MiCE with the targetted part-of-speech being the adjective.

Step	MiCE ADJ
1	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democracy better than [for example] a **totalitarian** regim?"
2	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democracy better than [for example] a **libertarian** regim?"
3	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democracy better than [for example] a **dictator** regim?"
4	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democracy **more representative** than [for example] a dictator regim?"
5	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democracy more **evil** than [for example] a dictator regim?"
6	Ultimately it rests with **your** opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democracy **a democracy** than [for example] a dictator regim?"
7	Ultimately it rests with your opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be **reasonable** to ask, "What makes a democracy a democracy than [for example] a dictator regim?"
8	Ultimately it rests with your opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be **nicer** to ask, "What makes a democracy a democracy than [for example] a dictator regim?"
9	Ultimately it rests with your opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be **reasonable** to ask, "What makes a democracy a democracy than [for example] a dictator regim?"
10	Ultimately it rests with your opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be **nicer** to ask, "What makes a democracy a democracy than [for example] a dictator regim?"

Table 7.34: An example from the NewsGroups dataset for MiCE with the targetted part-of-speech being the noun.

Step	MiCE NOUN
1	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a **democracy** better than [for example] a totalitarian regim?"
2	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a **government** better than [for example] a totalitarian regim?"
3	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a **democratic regime** better than [for **the most part**] a totalitarian regim?"
4	Ultimately it rests with personal opinion...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democratic **communism** better than [for the most part] a totalitarian regim?"
5	Ultimately it rests with personal **experience**...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democratic **dictatorship** better than [for the most part] a totalitarian regim?"
6	Ultimately it rests with personal experience...in my opinion. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democratic **wreckship** better than [for the most part] a totalitarian regim?"
7	Ultimately it rests with personal **preference**...in my **case, the CIA**. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democratic wreckship better than [for the most part] a totalitarian regim?"
8	Ultimately it rests with personal **bias**...in my case, the CIA. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democratic wreckship better than [for the most part] a totalitarian regim?"
9	Ultimately it rests with personal **preference**...in my case, the CIA. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democratic wreckship better than [for the most part] a totalitarian regim?"
10	Ultimately it rests with personal **experience**...in my case, the CIA. :-) The question doesn't make sense to me. Maybe it would be better to ask, "What makes a democratic wreckship better than [for the most part] a totalitarian regim?"

7.2.4 Duration: A noteworthy limitation

Although this work provides us with valuable results and explanations for the decisions models take, there is an important limitation that must be highlighted, and this is the duration of the experiments. Amongst 44 experiments, the mean time to execute a 10-step iteration of generating counterfactual edits was an astonishing **38 hours**.

Therefore, the duration of the experiments, which is often an expected limitation in machine learning tasks, seems to be a factor in our study that can only be bypassed and not entirely tackled. In order to address this limitation we only experiment with a fraction of the test sets of the datasets, a method that certainly confines us from generating a bigger number of edits and having a more wide picture, but which does not influence our quantitative results, as proved in [20].

To provide some numerical results, the average time needed to generate a counterfactual edit is 20 and 9.5 seconds approximately for IMDb and NewsGroups accordingly. The total GPU hours (consecutive hours that one GPU needs to operate) needed for all of our experiments were **1670** (!) translating to 69.5 days. These numbers allow us to understand how resource-intensive counterfactual generation is, and that available resources are a factor that must be reckoned with.

Moreover, in Figures 7.2.12, 7.2.13a and 7.2.13, we show how the needed time was distributed among the different categories of experiments that we conducted and how some methods significantly influence the experiments' duration. For example, we point out that MiCE with 120 beams needs approximately a quadruple amount of runtime than MiCE with multinomial sampling. On the other hand, we show that the method of targeted POS tags reduces significantly the needed runtime, presenting in this way another benefit.

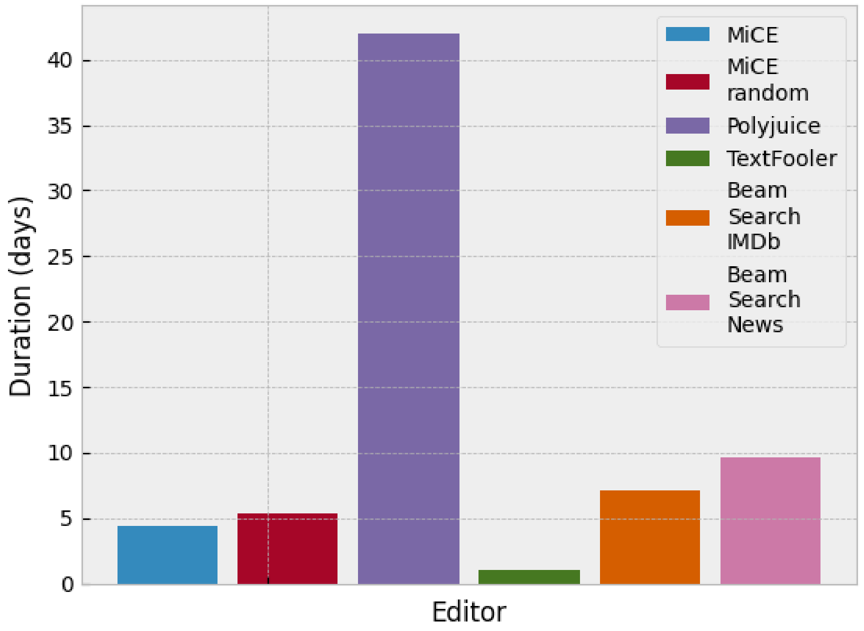
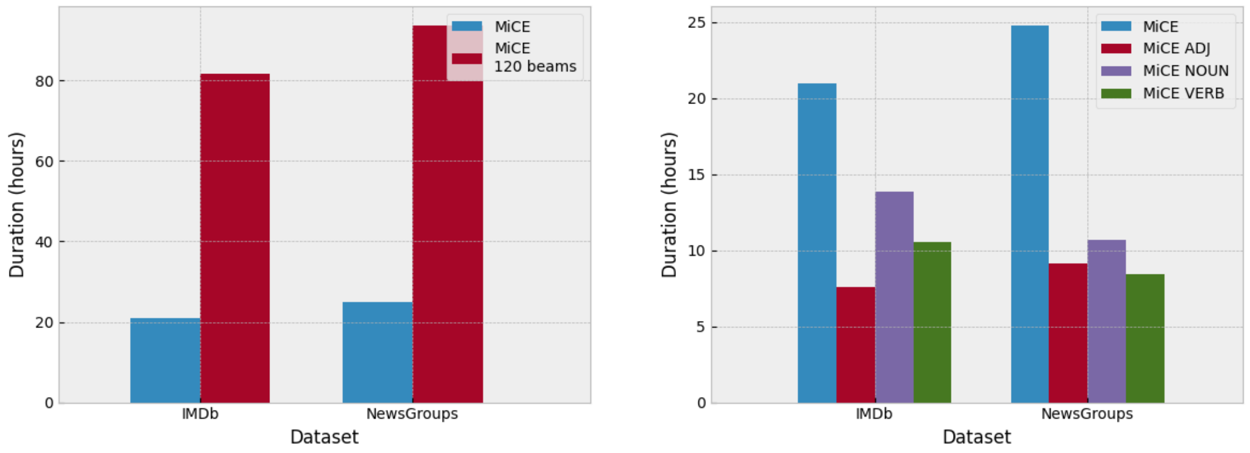


Figure 7.2.12: Distribution of total runtime among 6 big categories of experiments: the experiments with the 4 editors and the beam search experiments on MiCE for both datasets.



(a) Duration comparison between MiCE and MiCE with 120 beams for both datasets.

(b) Duration diagram for the method of targeted POS tag on the MiCE editor for both datasets.

Figure 7.2.13: Duration diagrams.

Chapter 8

Conclusion

8.1 Discussion

In this work, we explored the use of different counterfactual editors for the task of Contrastive Counterfactual Explanations Generation in order to evaluate the different components and methods of these editors using multiple metrics and qualitative analysis. To conduct our experiments, we employed the novel method of Counterfactuals of counterfactuals [20], and for the evaluation, we explored the results of the recently introduced metric named inconsistency. Moreover, motivated by the subtask of generating minimal counterfactual edits, we introduce a variation of counterfactual explanations where the editors are constrained to generate counterfactuals while targeting a specific part-of-speech tag. In our work, we combined the methods outlined above to create and utilize a pipeline-like counterfactual generation system that generated the counterfactual edits that we use to evaluate the basic components of the editors.

We experimented with four different editors, each one with different characteristics, in order to explain their decisions and which of their structural components affect these decisions and how. In the results, we noticed that each counterfactual editor presents multiple weaknesses and advantages which are explained by their performance on one or multiple metrics, but also on the qualitative analysis. Specifically, we saw that the masking method used displays a significant impact on the generated counterfactual edits, with the editors that use a masking method that has any knowledge of the predictor's tendencies (either its gradients or its predictions when a specific word is missing) performing better in terms of all metrics and the quality of the generated text. On the other hand, random masking proved to provide us with an adequate baseline of results, but is not a reliable option to generate quality edits. Moreover, we observed that more deterministic approaches in counterfactual generation, such as those used by TextFooler, can lead to more minimal, more consistent editors than text generation models. In addition, MiCE and MiCERandom, which use T5, a language model which appears to have more controlled generation, perform better than Polyjuice across all metrics. Polyjuice employs GPT2, a language model which proves to generate more diverse text than MiCE and TextFooler which translates to worse performance regarding the metrics examined. In general, the conclusion that we encounter on multiple occasions is that adding constraints to the editor and models clearly leads to more optimal results. Nevertheless, the efficacy of these models is not always guaranteed and the diversity of results is a loss for us that seems inevitable.

In terms of part-of-speech tags, we employed counterfactual generation targeting three POS tags: ADJ, NOUN, and VERB with all four editors. Our experiments showed that the editors in the POS tag versions become much more consistent, but at the cost of failing to flip some of the edits. However, aside from the flip rate, the edits present fluency levels comparable and in some cases even higher than edits with no targeted POS tags. Moreover, through this method, we collected valuable information regarding the importance of each part-of-speech tag combined with the task of each dataset. This linguistic analysis showed that for sentiment analysis and the IMDb dataset, perturbing the adjectives is crucial to successful counterfactuals, as adjectives are more influential than verbs and nouns. On the other hand, the NewsGroups dataset, which specifies in the task of topic classification, proves to attribute more importance to nouns that can influence the main topic of the text more effectively.

As to the novel methods utilized, through the results of this work, it is demonstrated how feeding the editor's output back as a new input can provide us with an abundance of vulnerabilities and patterns an editor entails. Moreover, we can see how an editor performs on non-dataset-dependent content and what quality of edits it generates after multiple steps. Thus, the method of counterfactuals of counterfactuals proved to be invaluable for the research community as it explores states previously unseen by traditional counterfactual generation methods, unveiling editors' weaknesses and providing valuable insights on how they can be combatted. Regarding inconsistency, we saw that it is a metric that largely complements the method of Counterfactuals of counterfactuals and in our case provides a clear image regarding which methods and models contribute to more consistent but also easily explainable editors. The metric leads us to conclusions about the editors that other metrics cannot provide, such as the stability of the editor and how optimally it performs in terms of minimality. Of course, in this work, we use inconsistency of minimality but the logic behind the metric shows the premise for use with other evaluation metrics too.

To conclude, we can affirm that with suitable evaluation methods and the use of counterfactual explanations, we can provide adequate explanations for the performance of various editors, models, and generation methods. Through our experiments, we generated thousands of counterfactual explanations, we were able to leverage part-of-speech tags to introduce a new efficient method of text counterfactual generation, and utilized novel metrics and methods of counterfactual generation. However, most importantly we managed to explain the behavior of counterfactual editors and text generation models and algorithms, demonstrating that valuable conclusions can be drawn from their performance, but also their sensitivity and vulnerabilities. This evaluation process in general allowed us to gain interpretable knowledge of the counterfactual editors we worked with and not resort to simple comparisons between them.

To move a step further though, it is important to contemplate what these conclusions offer for future work. The results of our work provide further motivation for future research in the same scope, with additional metrics, editors, and tasks. To put one of our conclusions in perspective and form a potential research question, we found that constraining models can lead by proof to better quantitative results for multiple metrics. Nevertheless, what is the "cost" regarding the quality of the results? Is this quantitative optimization sufficient? To generalize that properly: Can we use the explanation of existing models to eventually create optimal editors/models, or should we use it to decide the tradeoffs depending on the task at hand?

8.2 Limitations

During the course of this thesis, we encountered several limitations, which mainly stemmed from the large computational requirements of our task. As we describe in 7.2.4, the conducted experiments required a considerable period of time and sometimes parallel execution in multiple GPUs. This limitation directly affected the size of the datasets which we experimented on, which fortunately did not impede our results and conclusions. Moreover, based on this constraint, we opted to not exhaust all available counterfactual editors as this would require significantly more computational requirements and time. Finally, our experiments resort to the language of English, as we did not experiment with datasets from other languages. However, we expect that the conclusions generated throughout this work hold across languages and that the methods used in this work can function without modifications to other languages too.

8.3 Future Work

In closing this thesis, we would like to propose several avenues for further improvement or alternative approaches that could inspire future research. Firstly, as in this work the datasets we worked with were specified in the tasks of sentiment analysis and topic classification, it would be interesting to investigate how the editors perform combined with other tasks such as Named Entity Recognition (NER), where the interest would be in perturbing only named entities, or aspect-based classification where the sentiment is categorized with respect to specific target entities or specific aspects. Moreover, another interesting approach would be to experiment with different predictors in order to capture how they perform in a combination of datasets, seeking to unveil any existing prejudice or bias. Finally, what could be also explored is the inconsistency of other metrics aside from minimality. As our results show, the back-translation approach of the authors of [20] combined with the concept of inconsistency allow us to reveal the potential limitations of a metric, and thus using a similar approach with other literature-related metrics could prove to be beneficial.

Another idea for future research would also be the inverse problem, meaning to attempt to use the explanations and limitations of each editor in order to try to improve existing implementations or even create a new editor from scratch based on which methods and models perform better. For example, in order to create a counterfactual editor which is more consistent with the modifications it makes, we could explore the implementation of an editor fine-tuned with the feedback approach and the inconsistency of minimality, which seems that could provide promising results and generated edits. Finally, the thousands of generating counterfactual edits could be leveraged in order to potentially create a dataset for further evaluation of counterfactual explanations or to use them in other tasks such as data augmentation.

Chapter 9

Bibliography

[1] "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". In: *Annals of Mathematical Statistics* 41.1 (1970), pp. 164–171. DOI: 10.1214/AOMS/1177697196.

[2] Arora, S. et al. "Pretrained Transformers Improve Out-of-Distribution Robustness". In: *arXiv preprint arXiv:2106.03880* (2021).

[3] Arrieta, A. B. et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information fusion* 58 (2020), pp. 82–115.

[4] Calderon, N. et al. *DoCoGen: Domain Counterfactual Generation for Low Resource Domain Adaptation.* 2022. arXiv: 2202.12350 [cs.CL].

[5] Chemmengath, S. et al. "Let the CAT out of the bag: Contrastive Attributed explanations for Text". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing.* Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 7190–7206. URL:

[6] Chomsky, N. "On nature and language". In: *Linguistic Inquiry* 33.1 (2002), pp. 1–45.

[7] Chou, Y.-L. et al. "Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications". In: *Information Fusion* 81 (2022), pp. 59–83.

[8] *Common Crawl Dataset.* Accessed: May 15, 2023.

[9] Dervakos, E. et al. "Semantic Enrichment of Pretrained Embedding Output for Unsupervised IR". In: *CEUR Workshop Proc.* 2846 (2021).

[10] Dervakos, E. et al. "Computing Rule-Based Explanations of Machine Learning Classifiers using Knowledge Graphs". In: *arXiv preprint arXiv:2202.03971* (2022).

[11] Dervakos, E. et al. *Choose your Data Wisely: A Framework for Semantic Counterfactuals.* 2023. arXiv: 2305.17667 [cs.AI].

[12] Devlin, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019. arXiv: 1810.04805 [cs.CL].

[13] Donahue, C., Lee, M., and Liang, P. "Enabling Language Models to Fill in the Blanks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Online: Association for Computational Linguistics, July 2020, pp. 2492–2501. DOI: 10.18653/v1/2020.acl-main.225. URL:

[14] Doshi-Velez, F. and Kim, B. *Towards A Rigorous Science of Interpretable Machine Learning.* 2017. arXiv: 1702.08608 [stat.ML].

[15] Ebrahimi, J. et al. *HotFlip: White-Box Adversarial Examples for Text Classification.* 2018. arXiv: 1712.06751 [cs.CL].

[16] Elman, J. L. "Finding structure in time". In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL:

[17] Fan, A., Lewis, M., and Dauphin, Y. "Hierarchical neural story generation". In: *ACL.* 2018.

[18] Fern, X. and Pope, Q. "Text Counterfactuals via Latent Optimization and Shapley-Guided Search". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing.* Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 5578–5593. DOI: 10.18653/v1/2021.emnlp-main.452. URL:

[19] Filandrianos, G. et al. “Conceptual Edits as Counterfactual Explanations.” In: *AAAI Spring Symposium: MAKE*. 2022.

[20] Filandrianos, G. et al. *Counterfactuals of Counterfactuals: a back-translation-inspired approach to analyse counterfactual editors*. 2023. arXiv: 2305.17055 [cs.CL].

[21] Francis, W. N. and Kucera, H. “The Brown Corpus”. In: *Linguistic investigations of the written language*. Vol. 3. De Gruyter Mouton. 1964, pp. 1–51.

[22] Gardner, M. et al. *AllenNLP: A Deep Semantic Natural Language Processing Platform*. 2018. arXiv: 1803.07640 [cs.CL].

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

[24] Goodfellow, I. J. et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[25] Goyal, Y. et al. “Counterfactual explanations of machine learning models: A survey”. In: *arXiv preprint arXiv:2012.14420* (2020).

[26] *GPT2-News*. 2021.

[27] Greene, B. and Rubin, G. *Automatic Grammatical Tagging of English*. Department of Linguistics, Brown University, 1971. URL:

[28] Guidotti, R. “Counterfactual explanations and how to find them: literature review and benchmarking”. In: *Data Mining and Knowledge Discovery* (Apr. 2022), pp. 1–55. DOI: 10.1007/s10618-022-00831-6.

[29] Hochreiter, S. and Schmidhuber, J. “Long Short-term Memory”. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[30] Holtzman, A. et al. “The curious case of neural text degeneration”. In: *ICLR*. 2019.

[31] Honnibal, M. and Montani, I. *spaCy: Industrial-strength Natural Language Processing in Python*. 2022.

[32] HuggingFace contributors. *T5Tokenizer*. Accessed: May 18, 2023.

[33] HuggingFace contributors. *Write With Transformer*. Accessed: May 15, 2023.

[34] HuggingFace.co. *HuggingFace*. 2021. URL:

[35] Jin, D. et al. *Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment*. 2020. arXiv: 1907.11932 [cs.CL].

[36] John, A. et al. “A review of data augmentation techniques for natural language processing”. In: *2018 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE. 2018, pp. 1–5.

[37] Karavangelis, A. *thesis_ counterfactuals*. GitHub repository. 2023. URL:

[38] Kaur, H., Chakraborty, A., and Mukherjee, S. “Counterfactual evaluation of conversational recommendation systems”. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 1235–1238.

[39] Keane, M. T. et al. *If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques*. 2021. arXiv: 2103.01035 [cs.LG].

[40] Kusner, M. J. et al. “Counterfactual Fairness”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4066–4077.

[41] Lang, K. and Turney, P. “Unsupervised Learning of Synchronous Syntactic Structure for Hidden Markov Model Induction”. In: *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1995, pp. 531–537.

[42] Levenshtein, V. I. “Binary codes capable of correcting deletions, insertions, and reversals”. In: *Soviet physics doklady* 10.8 (1966), pp. 707–710.

[43] Li, L. et al. *BERT-ATTACK: Adversarial Attack Against BERT Using BERT*. 2020. arXiv: 2004.09984 [cs.CL].

[44] Liang, B. et al. “Deep Text Classification Can be Fooled”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, July 2018. DOI: 10.24963/ijcai.2018/585. URL:

[45] Lipton, Z. C. “The mythos of model interpretability”. In: *arXiv preprint arXiv:1606.03490* (2018).

[46] Liu, Y. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].

[47] Looveren, A. V. and Klaise, J. *Interpretable Counterfactual Explanations Guided by Prototypes*. 2020. arXiv: 1907.02584 [cs.LG].

[48] Lundberg, S. M. and Lee, S.-I. “A unified approach to interpreting model predictions”. In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 4765–4774.

[49] Lymperaiou, M. et al. "Towards Explainable Evaluation of Language Models on the Semantic Similarity of Visual Concepts". In: *ArXiv* abs/2209.03723 (2022).

[50] Lymperaiou, M. et al. "Counterfactual Edits for Generative Evaluation". In: *ArXiv* abs/2303.01555 (2023).

[51] Maas, A. L. et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. URL:

[52] Madaan, N. et al. *Generate Your Counterfactuals: Towards Controlled Counterfactual Generation for Text*. 2021. arXiv: 2012.04698 [cs.CL].

[53] Madsen, A., Reddy, S., and Chandar, S. "Post-hoc Interpretability for Neural NLP: A Survey". In: *ACM Computing Surveys* 55.8 (Dec. 2022), pp. 1–42. DOI: 10.1145/3546577. URL:

[54] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. "Building a Large Annotated Corpus of English: The Penn Treebank". In: *Computational Linguistics* 19.2 (1993), pp. 313–330. URL:

[55] Menis-Mastromichalakis, O., Sofou, N., and Stamou, G. "Deep Ensemble Art Style Recognition". In: *2020 International Joint Conference on Neural Networks (IJCNN)* (2020), pp. 1–8.

[56] Minsky, M. and Papert, S. *Perceptrons; an Introduction to Computational Geometry*. MIT Press, 1969. ISBN: 9780262630221. URL:

[57] Molnar, C. "Interpretable Machine Learning: A Guide for Making Black Box Models Explainable". In: *https://christophm.github.io/interpretable-ml-book/* (2019).

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

[59] Moradi, M. and Samwald, M. *Evaluating the Robustness of Neural Language Models to Input Perturbations*. 2021. arXiv: 2108.12237 [cs.CL].

[60] Morris, J. X. et al. *TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP*. 2020. arXiv: 2005.05909 [cs.CL].

[61] Nivre, J. et al. *Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection*. 2020. arXiv: 2004.10643 [cs.CL].

[62] OpenAI. *ChatGPT: Language Models for Task-Oriented Conversations*. Available online at. Accessed: May 27, 2023.

[63] Pang, B., Lee, L., and Vaithyanathan, S. *Thumbs up? Sentiment Classification using Machine Learning Techniques*. 2002. arXiv: cs/0205070 [cs.CL].

[64] Paszke, A. et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[65] Pedregosa, F. et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[66] Pruthi, D. et al. "Evaluating Explanations: How Much Do Explanations from the Teacher Aid Students?" In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 359–375. DOI: 10.1162/tacl_a_00465. URL:

[67] Pyrovolakis, K., Tzouveli, P. K., and Stamou, G. "Mood detection analyzing lyrics and audio signal based on deep learning architectures". In: *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. IEEE, 2020, pp. 9363–9370. DOI: 10.1109/ICPR48806.2021.9412361. URL:

[68] Pyrovolakis, K., Tzouveli, P. K., and Stamou, G. "Multi-Modal Song Mood Detection with Deep Learning". In: *Sensors* 22.3 (2022), p. 1065. DOI: 10.3390/s22031065. URL:

[69] Radford, A. et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[70] Raffel, C. et al. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2020. arXiv: 1910.10683 [cs.LG].

[71] Ribeiro, M. T., Singh, S., and Guestrin, C. "" Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1135–1144.

[72] Ribeiro, M. T., Singh, S., and Guestrin, C. "Anchors: High-precision model-agnostic explanations". In: *AAAI* 18 (2018), pp. 1527–1535.

[73] Robeer, M., Bex, F., and Feelders, A. "Generating Realistic Natural Language Counterfactuals". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican

Republic: Association for Computational Linguistics, Nov. 2021, pp. 3611–3625. DOI: 10.18653/v1/2021.findings-emnlp.306. URL:

[74] Rosenblatt, F. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65 6 (1958), pp. 386–408.

[75] Ross, A., Marasovi'c, A., and Peters, M. E. "Explaining NLP Models via Minimal Contrastive Editing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 7077–7082.

[76] Ross, A., Marasović, A., and Peters, M. E. *Explaining NLP Models via Minimal Contrastive Editing (MiCE)*. 2021.

[77] Ross, A. et al. *Tailor: Generating and Perturbing Text with Semantic Controls*. 2022. arXiv: 2107.07150 [cs.CL].

[78] Samuel, A. L. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210.

[79] Selvaraju, R. R. et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[80] Shannon, C. E. "A mathematical theory of communication". In: *Bell Syst. Tech. J.* 27 (1948), pp. 623–656.

[81] Simonyan, K., Vedaldi, A., and Zisserman, A. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: 1312.6034 [cs.CV].

[82] Sun, X. *Structure Regularization for Structured Prediction: Theories and Experiments*. 2015. arXiv: 1411.6243 [cs.LG].

[83] Sundararajan, M., Taly, A., and Yan, Q. "Axiomatic attribution for deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), pp. 3319–3328.

[84] Treviso, M. and Martins, A. F. T. "The Explanation Game: Towards Prediction Explainability through Sparse Communication". In: *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. Online: Association for Computational Linguistics, Nov. 2020, pp. 107–118. DOI: 10.18653/v1/2020.blackboxnlp-1.10. URL:

[85] Tzouveli, P. K., Mylonas, P., and Kollias, S. D. "An intelligent e-learning system based on learner profiling and learning resources adaptation". In: *Comput. Educ.* 51 (2008), pp. 224–238.

[86] Vasilakes, J., Papadopoulos, S., and Karkaletsis, V. "Counterfactual Data Augmentation for Neural Machine Translation". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2022.

[87] Vasilakes, J. et al. "Learning Disentangled Representations of Negation and Uncertainty". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8380–8397. DOI: 10.18653/v1/2022.acl-long.574. URL:

[88] Vasilakes, J. et al. *Learning Disentangled Representations of Negation and Uncertainty*. 2022.

[89] Vaswani, A. et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[90] Verma, S. et al. "A survey of reinforcement learning algorithms for autonomous navigation of unmanned aerial vehicles". In: *Journal of Field Robotics* 37.4 (2020), pp. 618–648.

[91] Wachter, S., Mittelstadt, B., and Russell, C. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR". In: *Harvard Journal of Law & Technology* 31.1 (2018).

[92] Wang, B. and Cho, K. "Learning to ask unanswerable questions for machine reading comprehension". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3431–3441.

[93] Wang, Z., Shu, K., and Culotta, A. *Enhancing Model Robustness and Fairness with Causality: A Regularization Approach*. 2021. arXiv: 2110.00911 [cs.LG].

[94] Werra, L. von. *GPT2-IMDB*. 2021.

[95] Wikipedia. *Levenshtein distance — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-April-2023]. 2021. URL:

[96] Wikipedia. *Sentiment Analysis*. 2023.

[97] Wikipedia contributors. *Machine learning — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-September-2022]. 2022.

[98] Wikipedia contributors. *Gradient descent — Wikipedia, The Free Encyclopedia*. [Online; accessed 1-June-2023]. 2023.

[99] Wikipedia contributors. *Rectifier (neural networks) — Wikipedia, The Free Encyclopedia*. [Online; accessed 27-May-2023]. 2023.

[100] Wikipedia contributors. *GPT-2*. Accessed: May 15, 2023.

[101] Wilson, T., Wiebe, J., and Hoffmann, P. "Articles: Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis". In: *Computational Linguistics* 35.3 (Sept. 2009), pp. 399–433. DOI: 10.1162/coli.08-012-R1-06-90. URL:

[102] Wu, T. et al. *Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models*. 2021. arXiv: 2101.00288 [cs.CL].

[103] Wu, T. et al. *Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models*. 2021.

Chapter 10

Appendix

A Further Results and Analysis

In this section, we provide some further results from the metrics explained in 7 in order to add more insights and analyze them at a deeper level.

A.1 Minimality

First, we show one box plot for each one of the editors that depict how minimality is altered after several feedback steps. For **MiCE**, we notice increased minimality in the versions that use random masking. We also notice that **adjectives** provide more minimal results than verbs and nouns in that order. Regarding part-of-speech tags, we notice the same pattern in the other editors too. Moreover, it is noticeable that editors with a targeted part-of-speech tag perform better than the version that does not target POS tags.

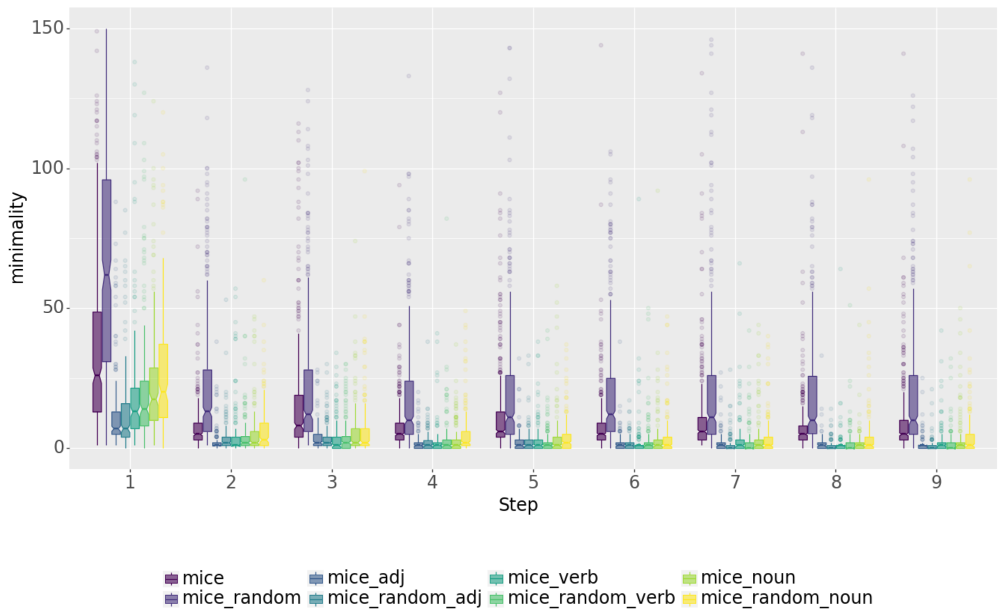


Figure A.1: Minimality on MiCE on the IMDb dataset.

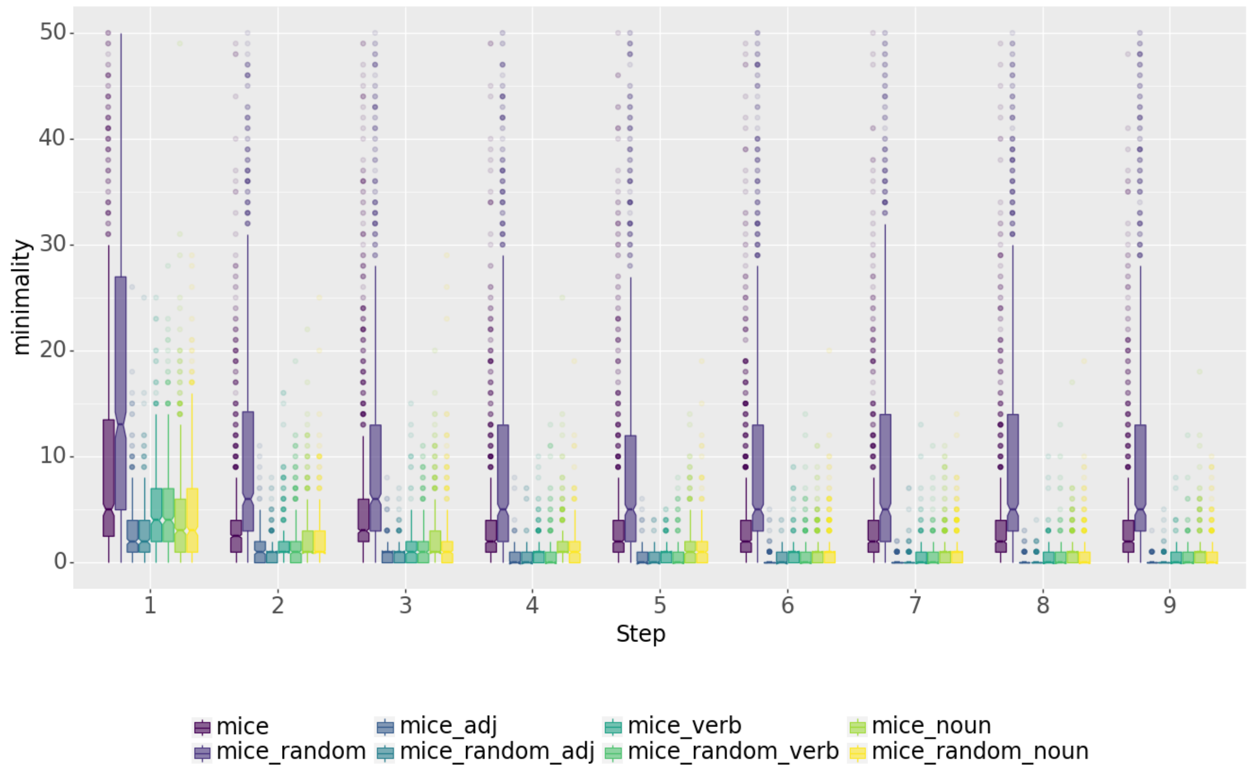


Figure A.2: Minimality on MiCE on the NewsGroups dataset.

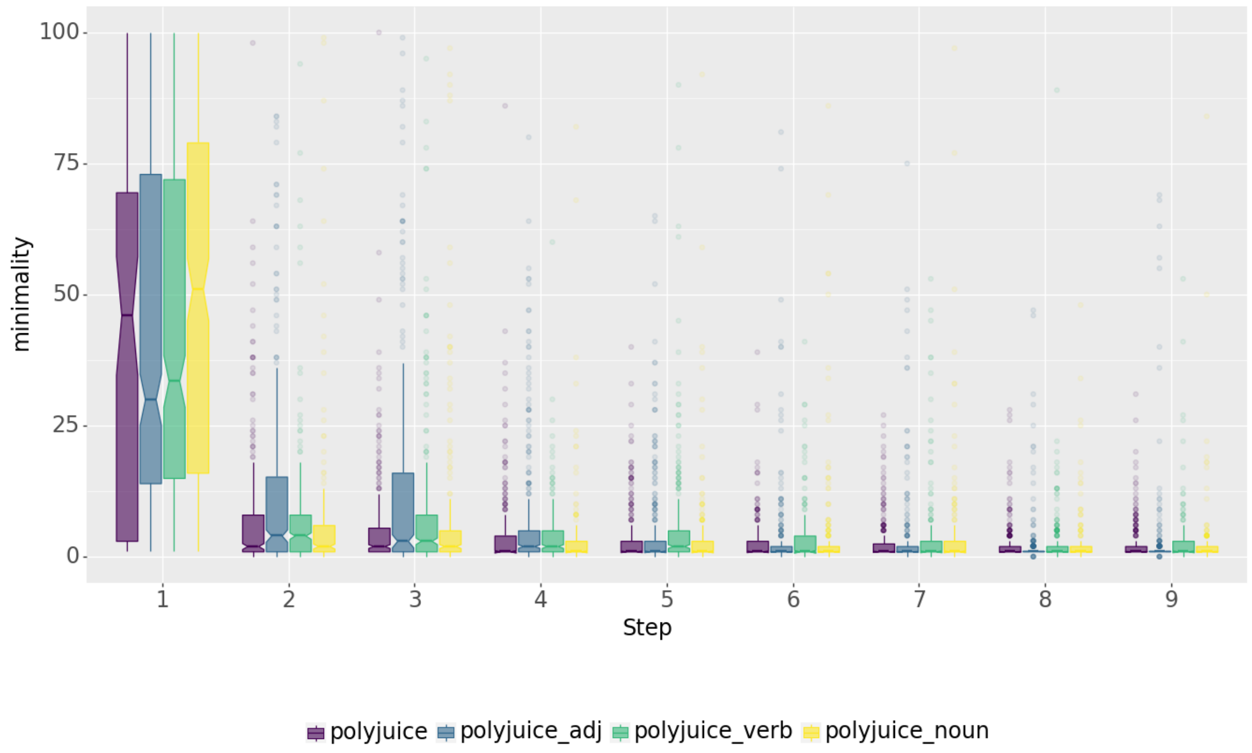
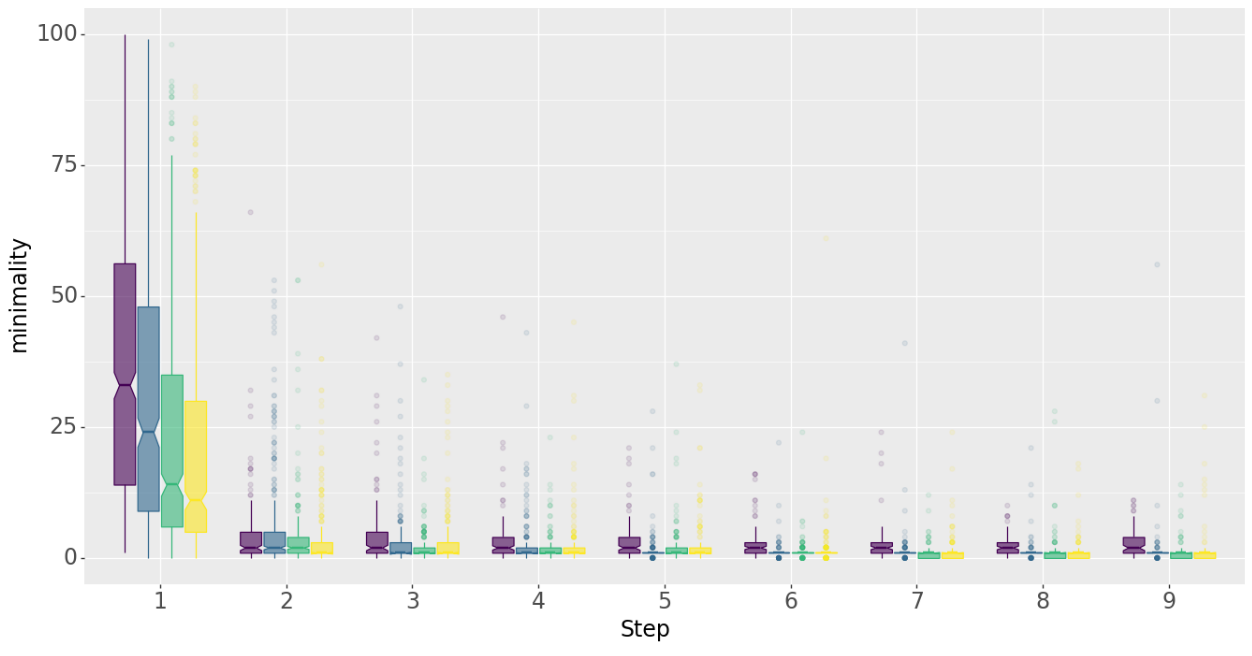


Figure A.3: Minimality on Polyjuice on the IMDb dataset.



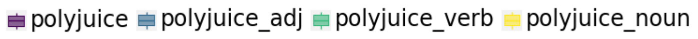


Figure A.4: Minimality on Polyjuice on the NewsGroups dataset.

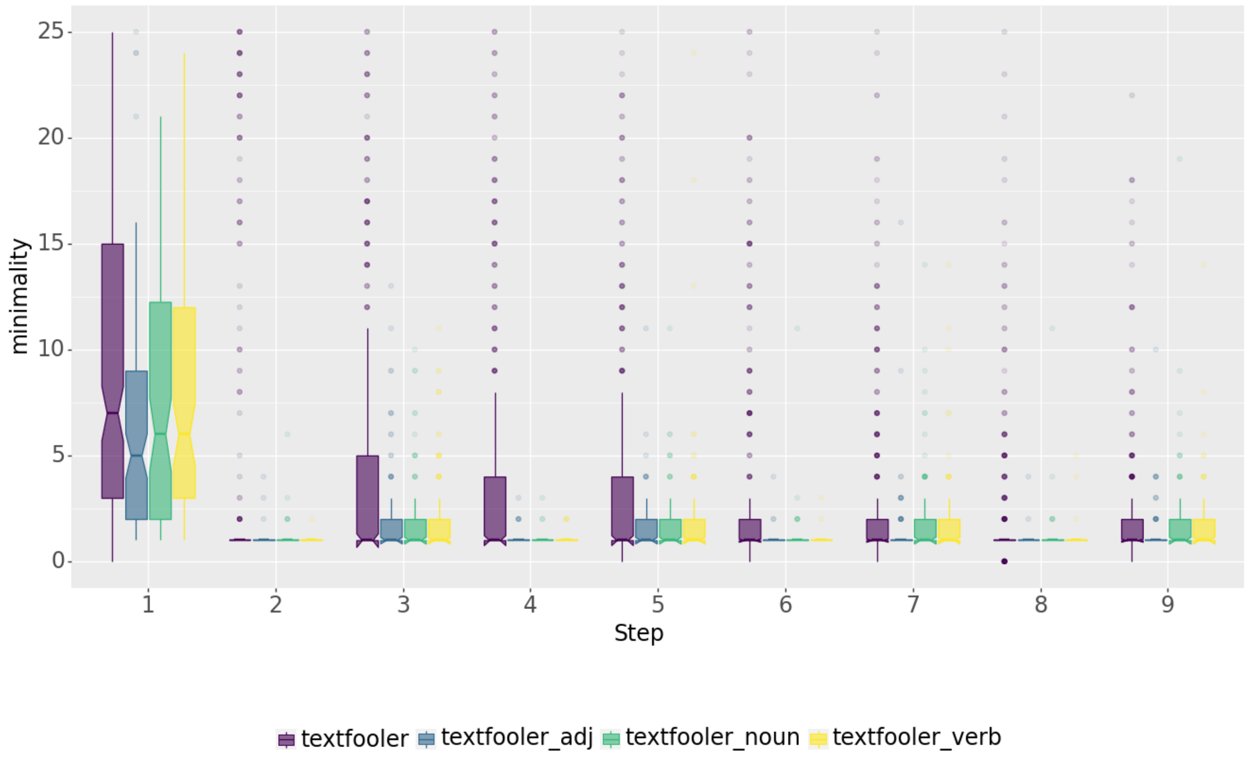


Figure A.5: Minimality on TextFooler on the IMDb dataset.

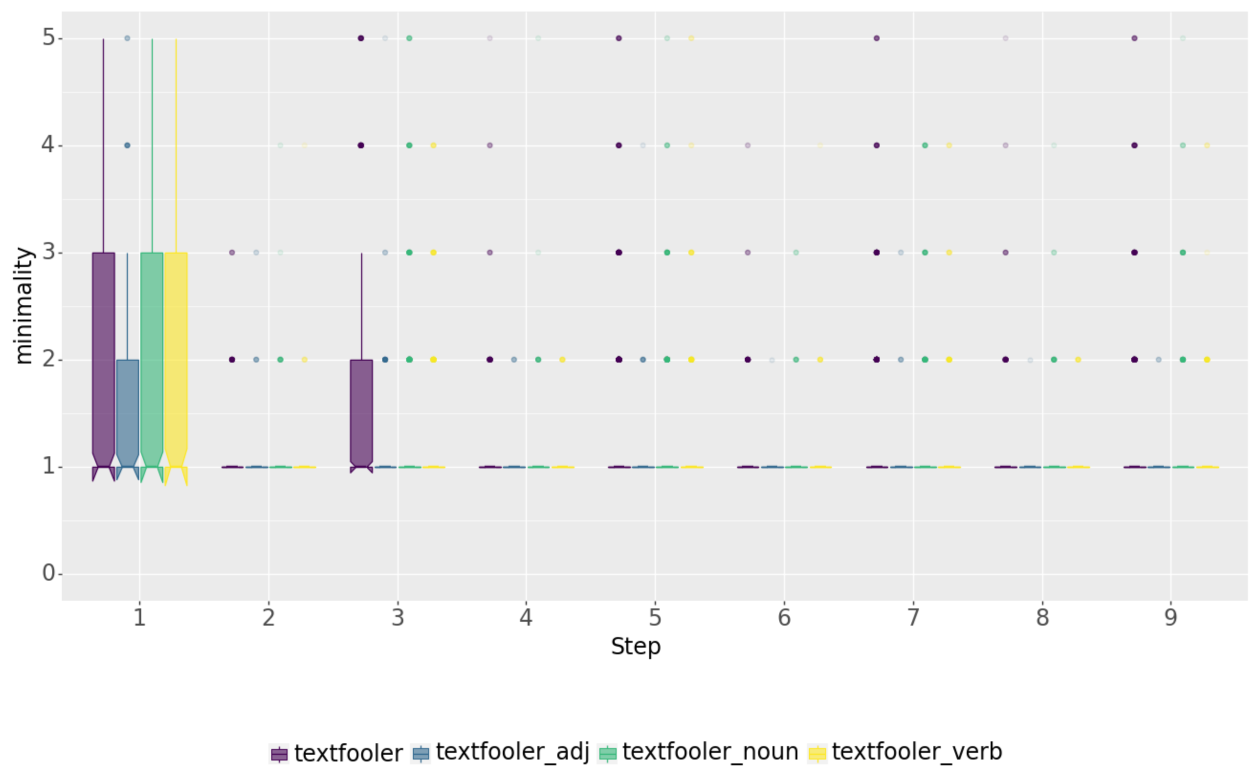


Figure A.6: Minimality on TextFooler on the NewsGroups dataset.

A.2 Inconsistency

Regarding inconsistency, we first notice that editors that target some part-of-speech tags present **very low** inconsistency values, as we also explained in 7. It is also interesting to observe the higher values of inconsistency in even steps in Figures A.7, A.9, and A.11.

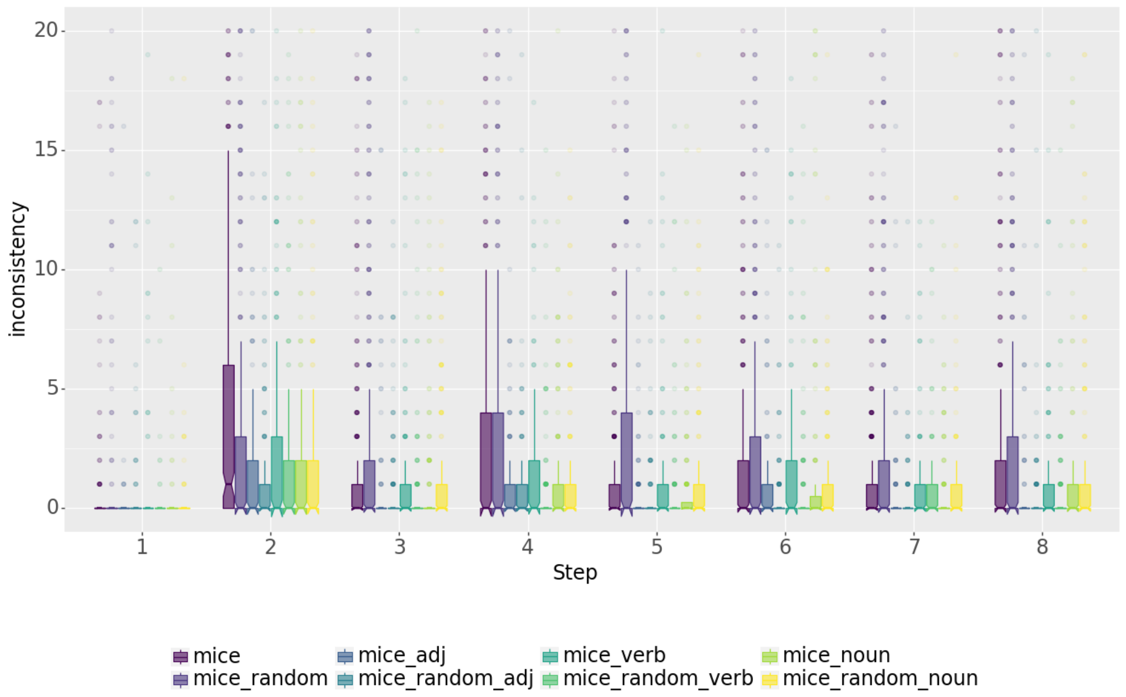


Figure A.7: Inconsistency on MiCE on the IMDb dataset.

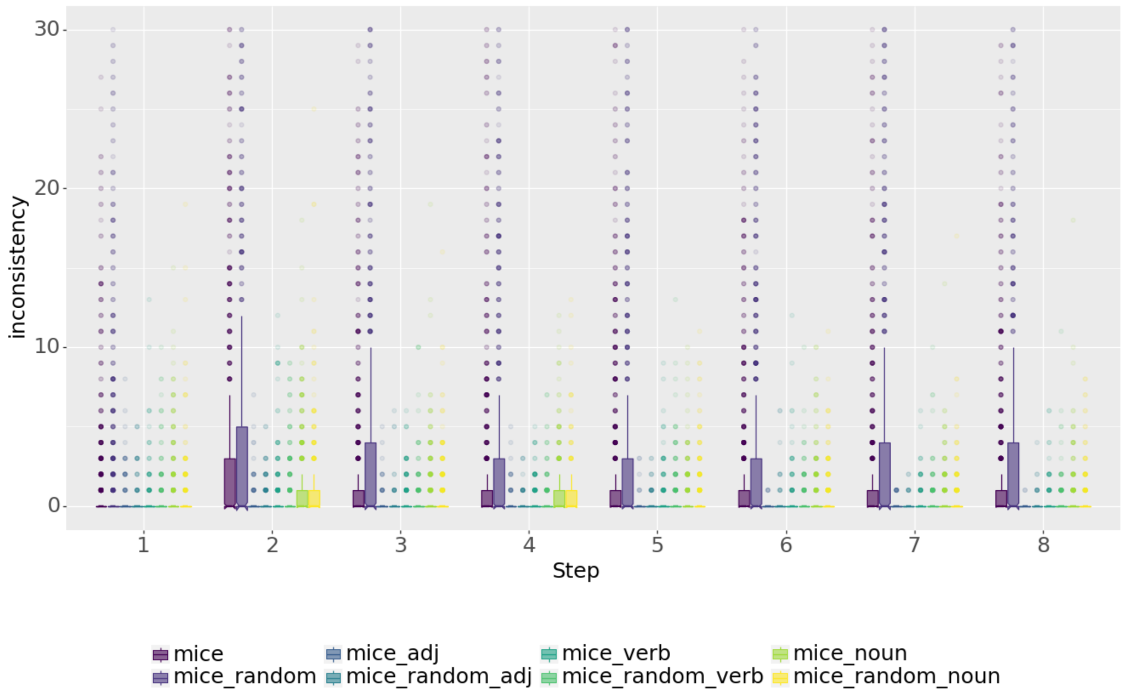


Figure A.8: Inconsistency on MiCE on the NewsGroups dataset.

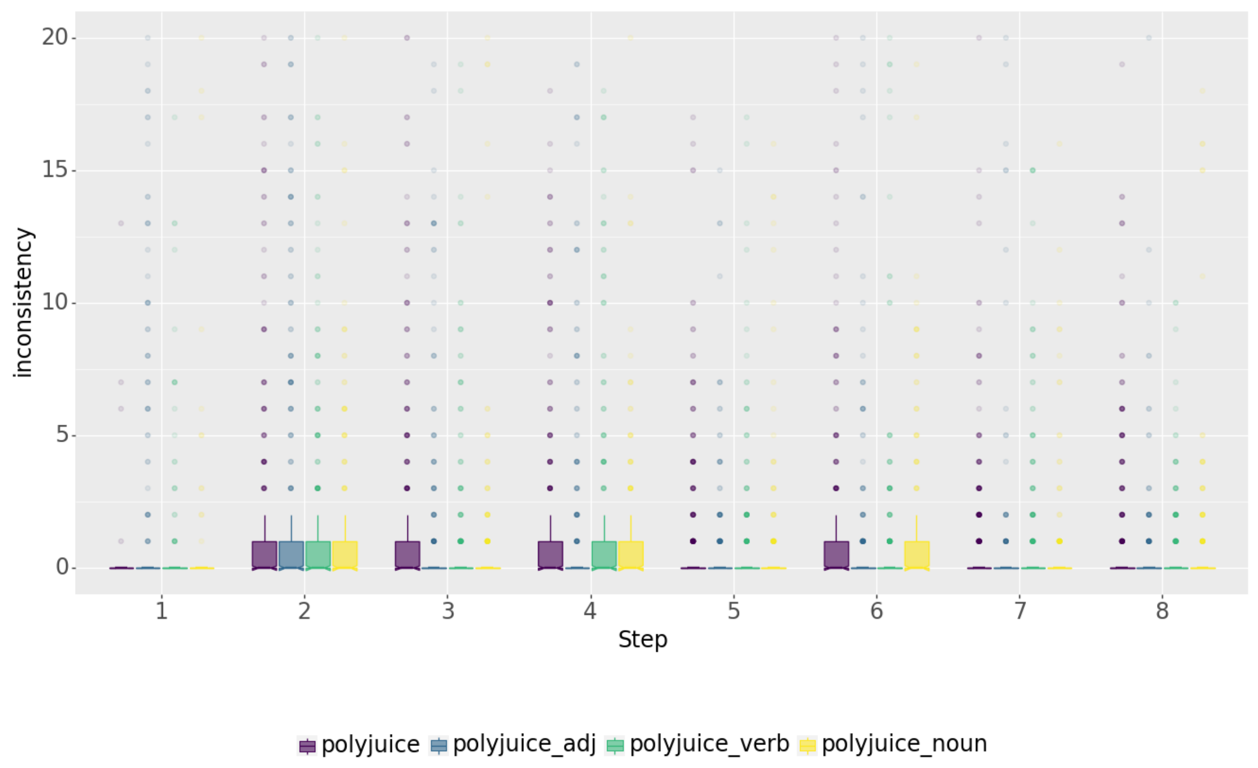


Figure A.9: Inconsistency on Polyjuice on the IMDb dataset.

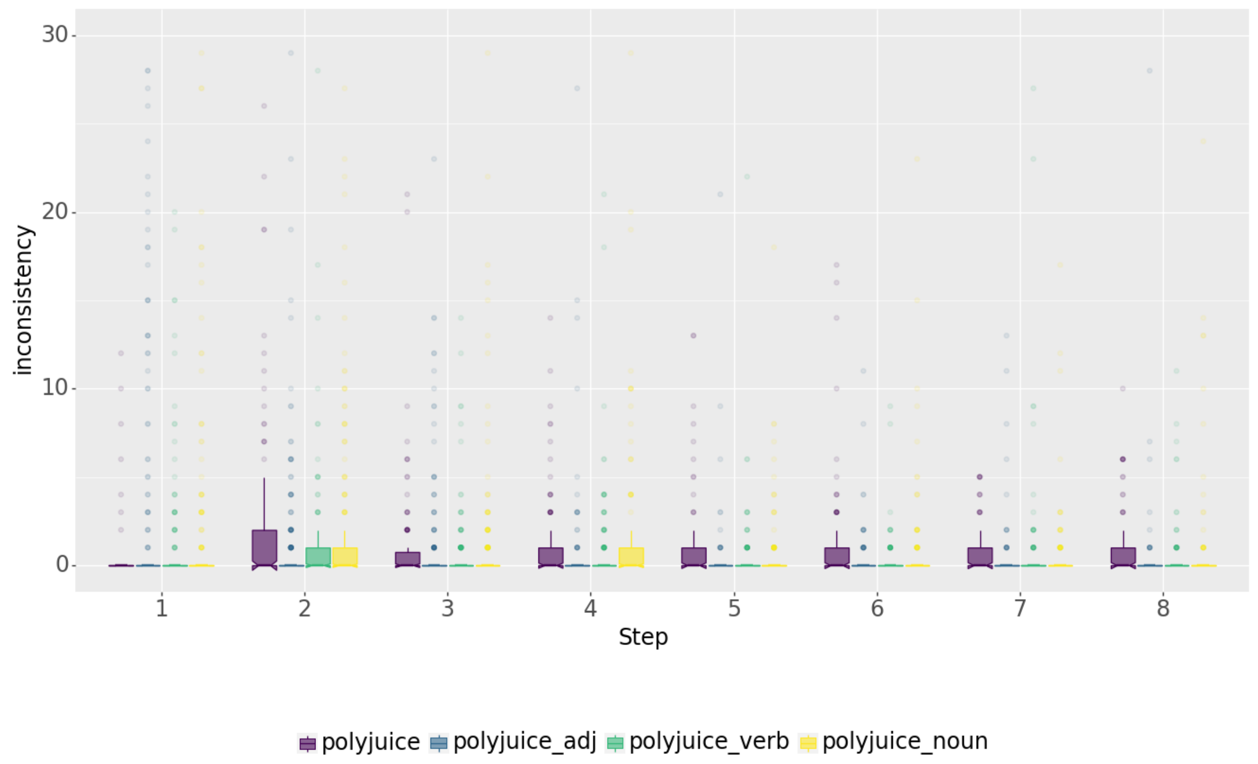


Figure A.10: Inconsistency on Polyjuice on the NewsGroups dataset.

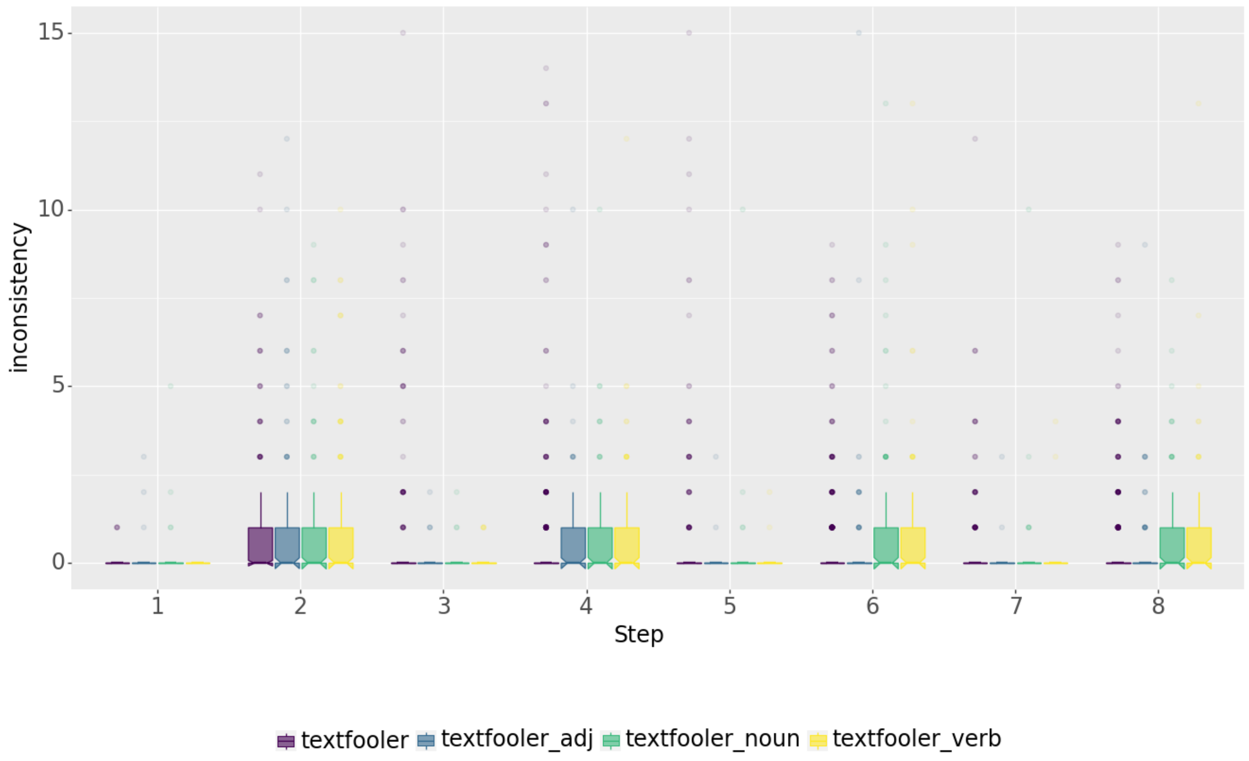


Figure A.11: Inconsistency on TextFooler on the IMDb dataset.

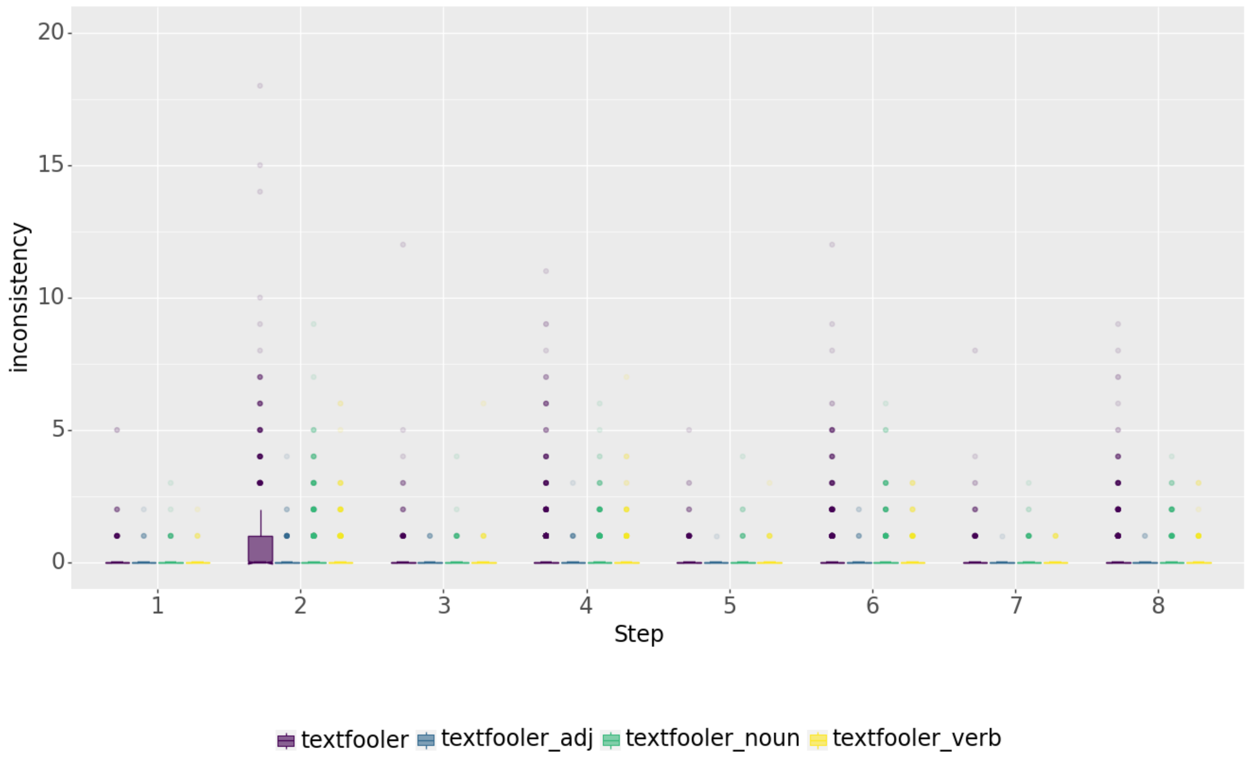


Figure A.12: Inconsistency on TextFooler on the NewsGroups dataset.

A.3 Flip Rate

In Figures A.13 and A.14, we notice how the flip rates shape for each editor and part-of-speech tag in the first step of our edits. An important takeaway from these two figures is that the prevalent POS tag in the IMDb dataset is the adjective, while in NewsGroups, it is the noun. This hints that in IMDb, where the task is sentiment analysis, adjectives play a more significant role than other POS tags. On the other hand, in NewsGroups, nouns seem to flip more samples as they have the ability to alter a sentence's topic more drastically.

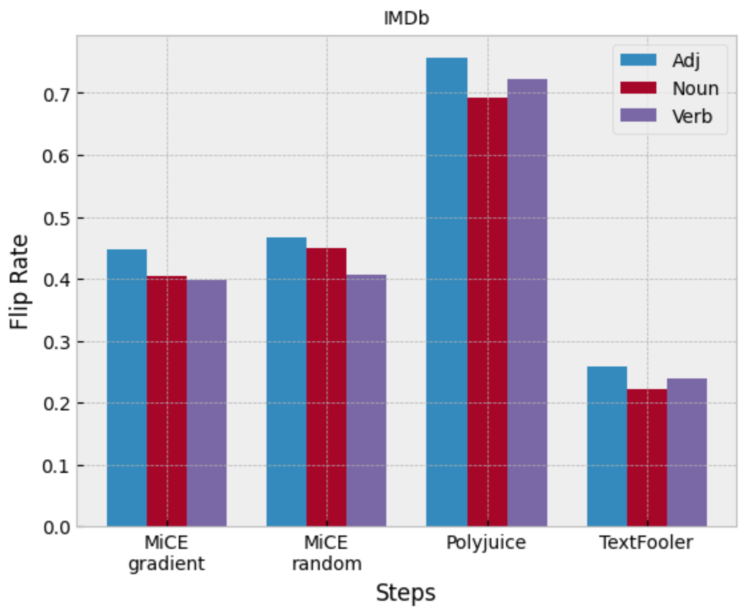


Figure A.13: Flip Rates on the first step of the IMDb dataset.

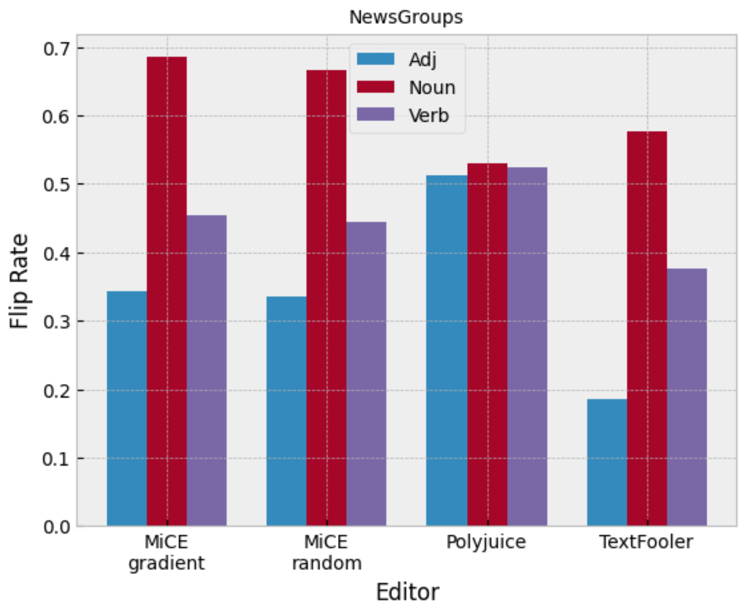


Figure A.14: Flip Rate on the first step of the NewsGroups dataset.

A.4 Base Perplexity

As we explain in the Experiments chapter (7), increased base perplexity values indicate more diverse edits, while lower values show more predictable text has been generated. The figures below indicate that part-of-speech tags present more stable, predictable edits in almost all the editors, and with all possible part-of-speech tags. This is due to the fact that the part-of-speech constraint does not "allow" the editors to "improvise" as it usually produces modifications that retain the sentence's structure and semantics.

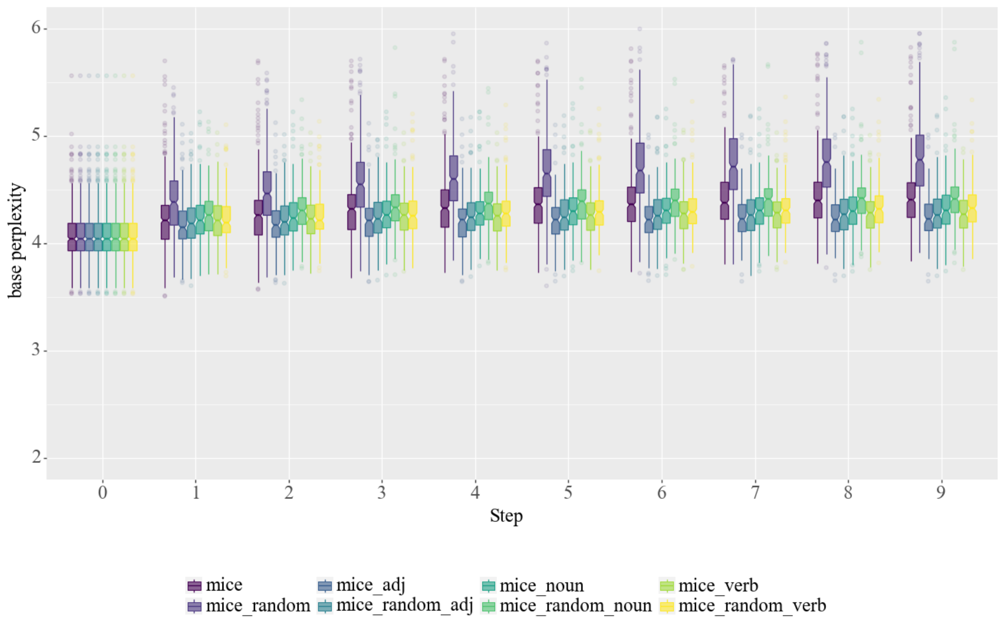


Figure A.15: Base Perplexity on MiCE on the IMDb dataset.

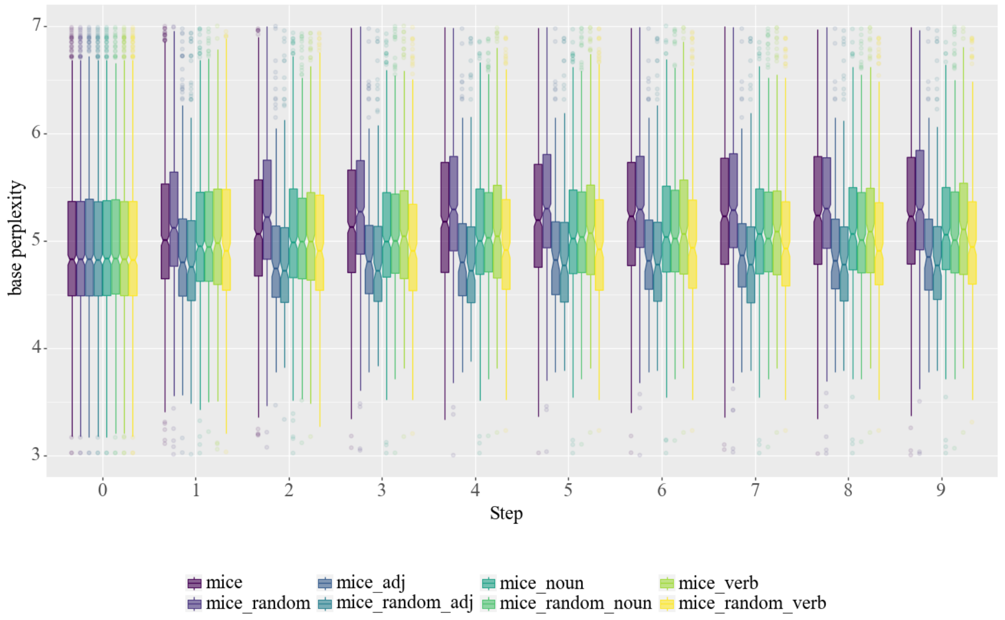


Figure A.16: Base Perplexity on MiCE on the NewsGroups dataset.

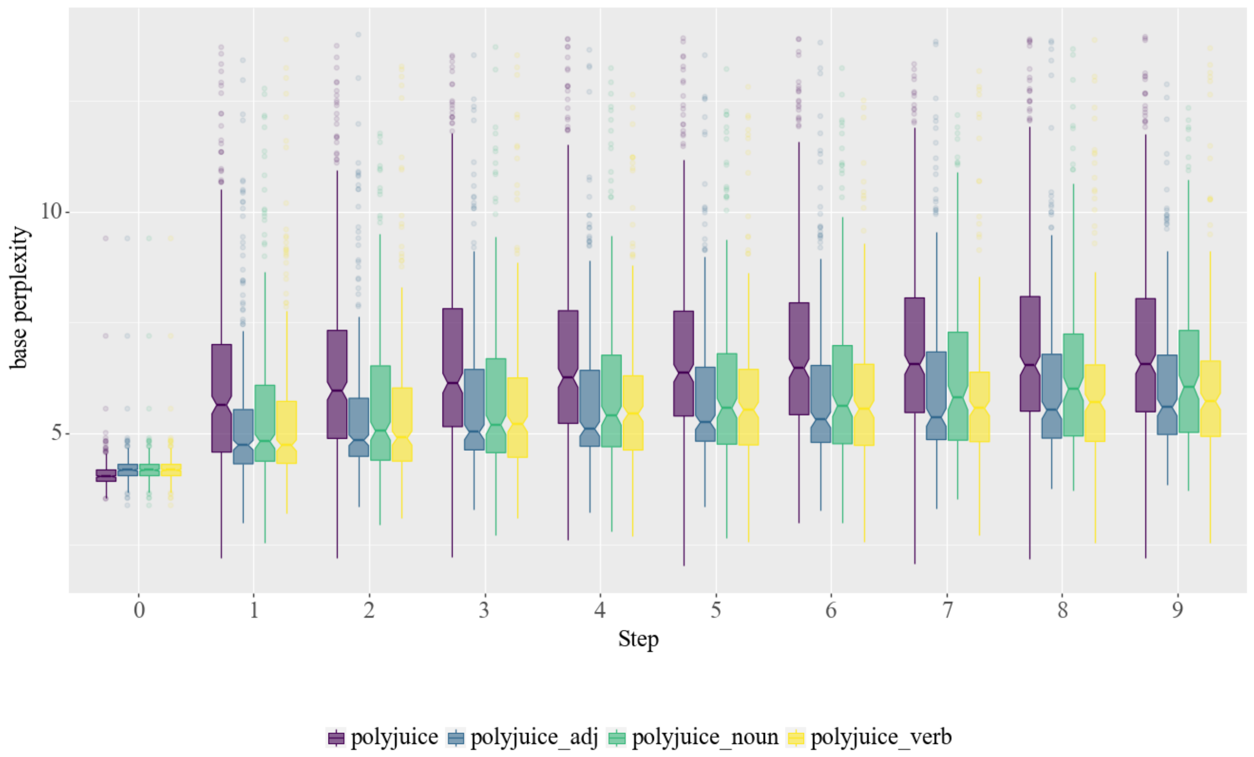


Figure A.17: Base Perplexity on Polyjuice on the IMDb dataset.

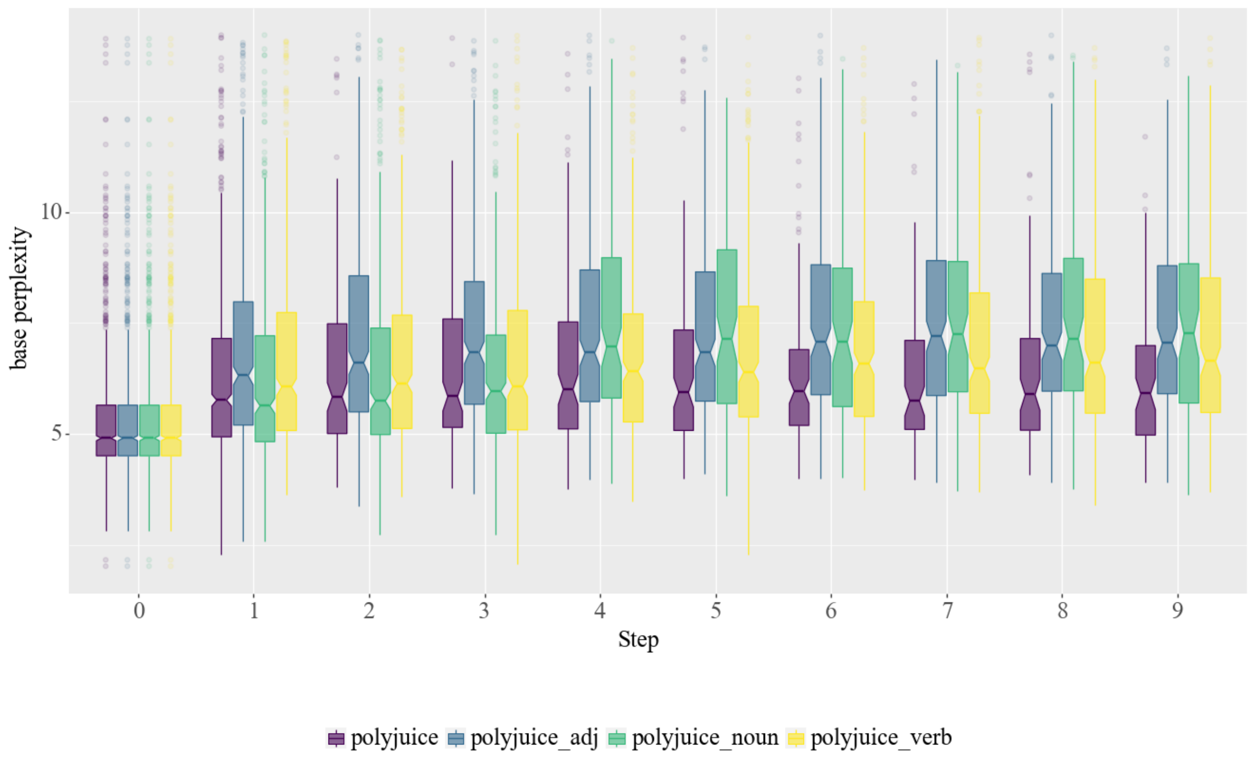


Figure A.18: Base Perplexity on Polyjuice on the NewsGroups dataset.

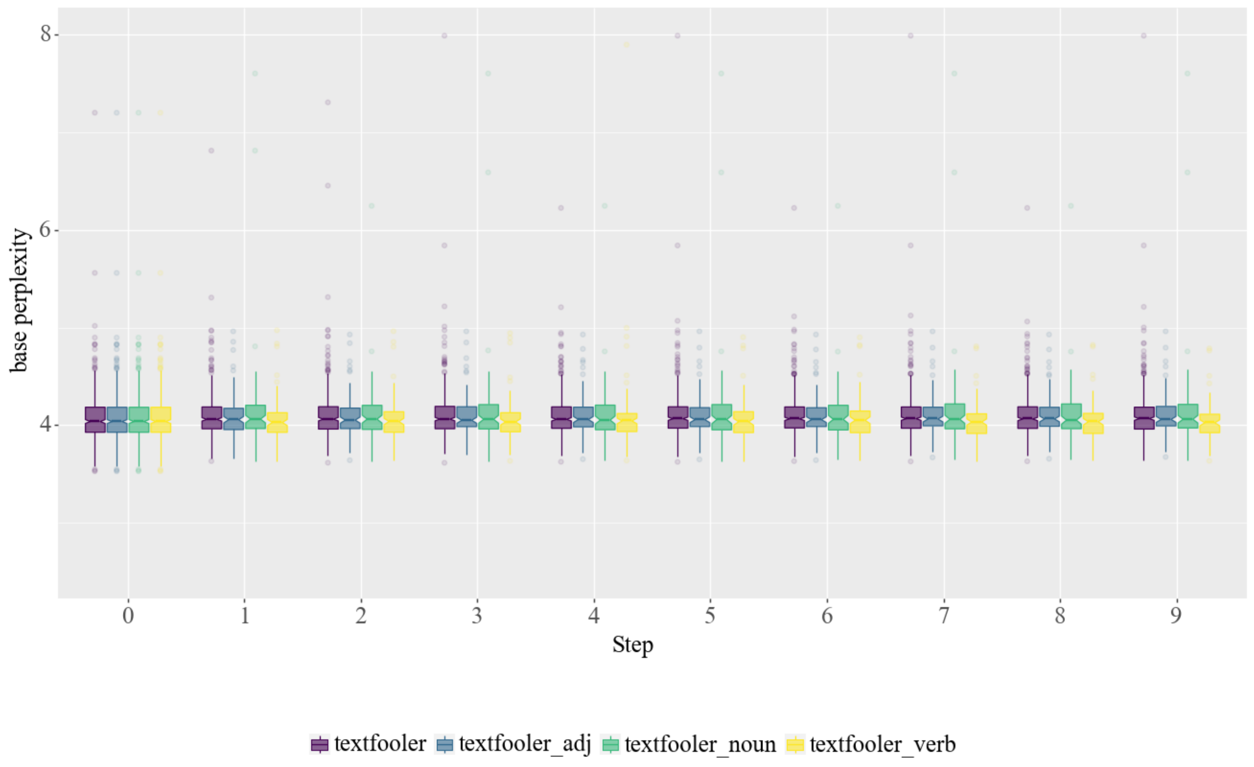


Figure A.19: Base Perplexity on TextFooler on the IMDb dataset.

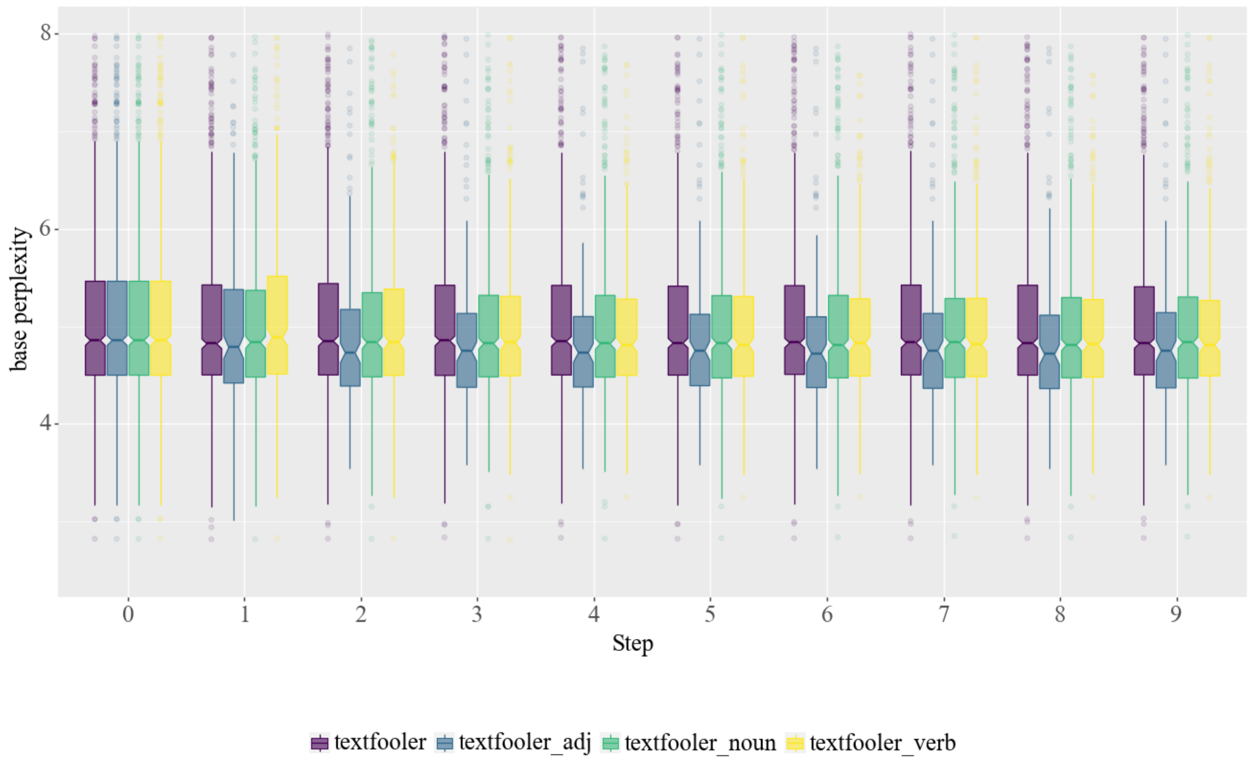


Figure A.20: Base Perplexity on TextFooler on the NewsGroups dataset.

A.5 Fine Perplexity

As for Fine Perplexity, we observe that POS tags generally help the editors generate more predictable edits, i.e. present values closer to the values of step 0. Especially for MiCE on IMDb, we see that the part-of-speech tags, and specifically the adjectives, reduce the overfitting phenomenon we pointed out in 7.2.1.4.

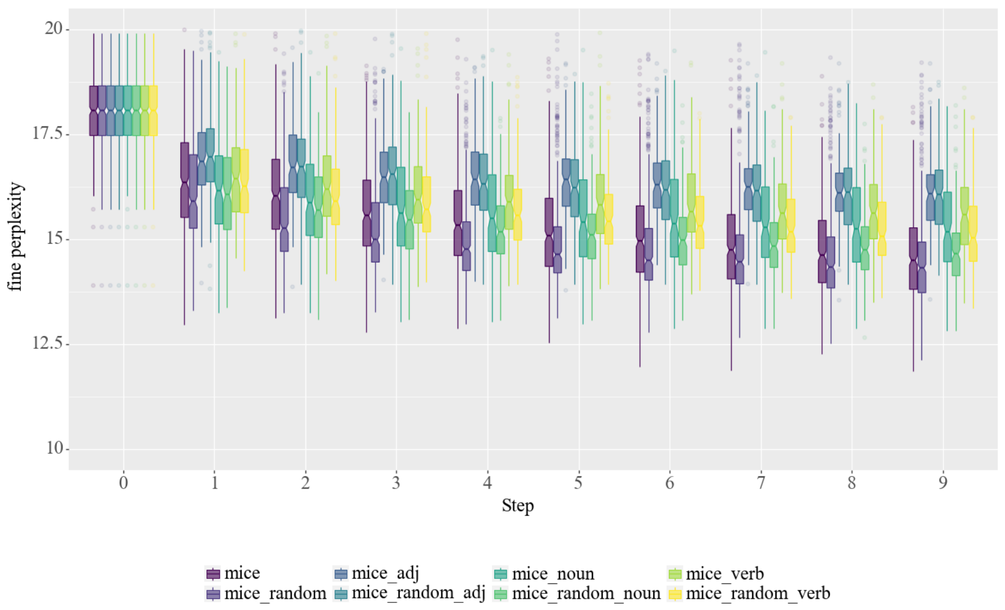


Figure A.21: Fine Perplexity on MiCE on the IMDb dataset.

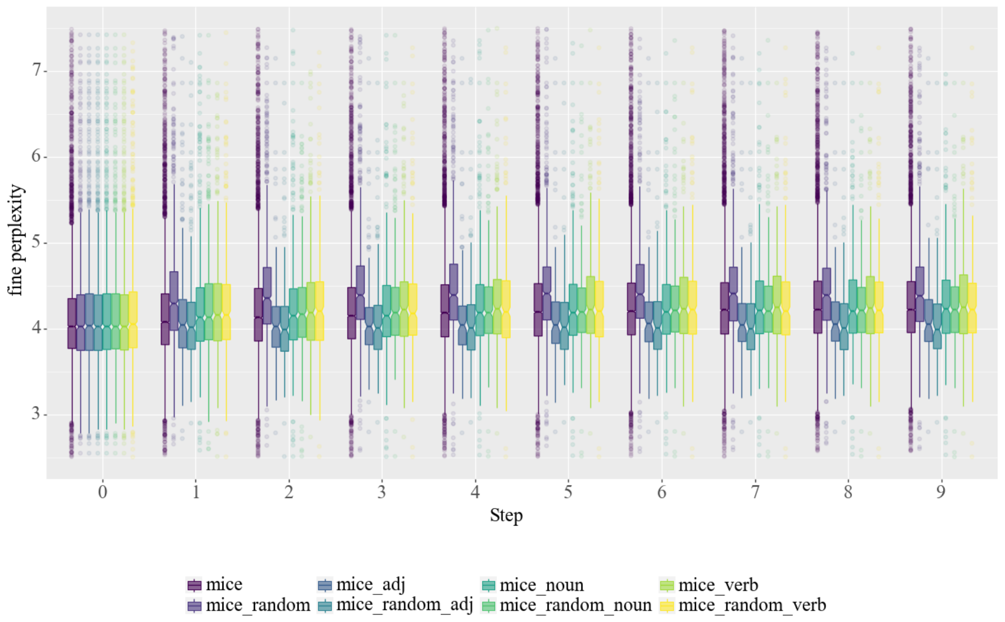


Figure A.22: Fine Perplexity on MiCE on the NewsGroups dataset.

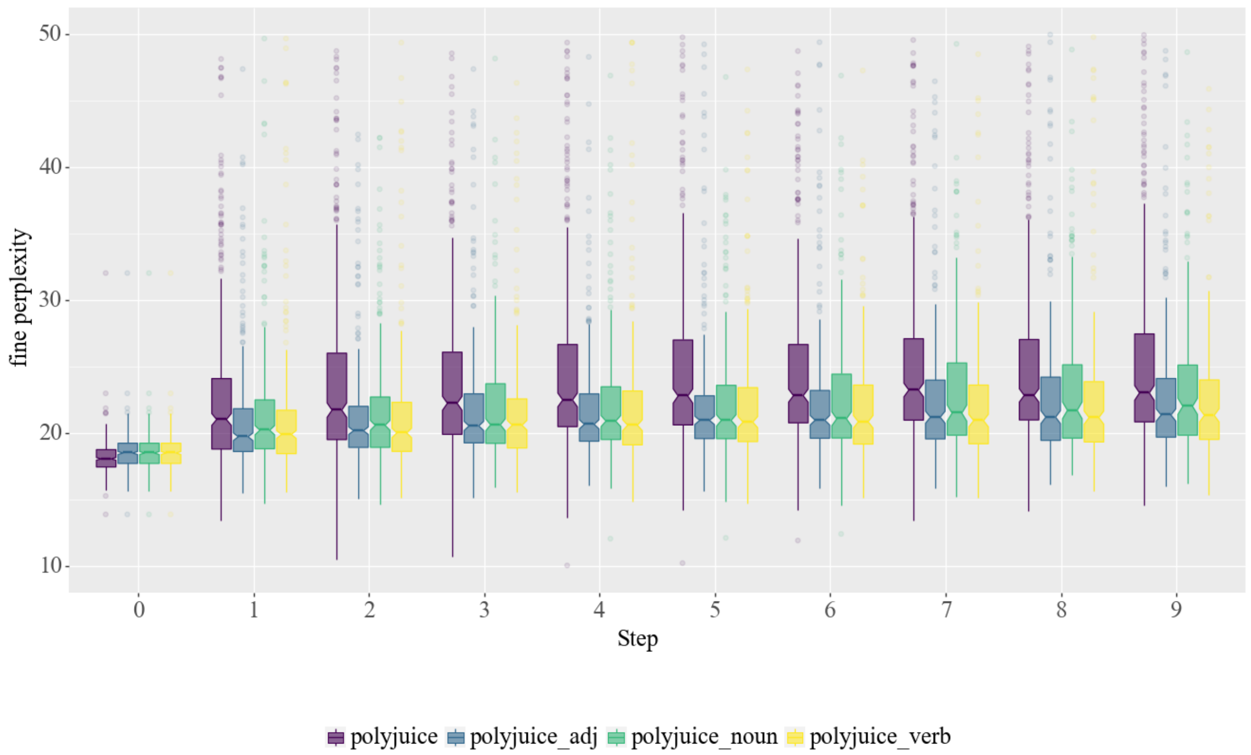


Figure A.23: Fine Perplexity on Polyjuice on the IMDb dataset.

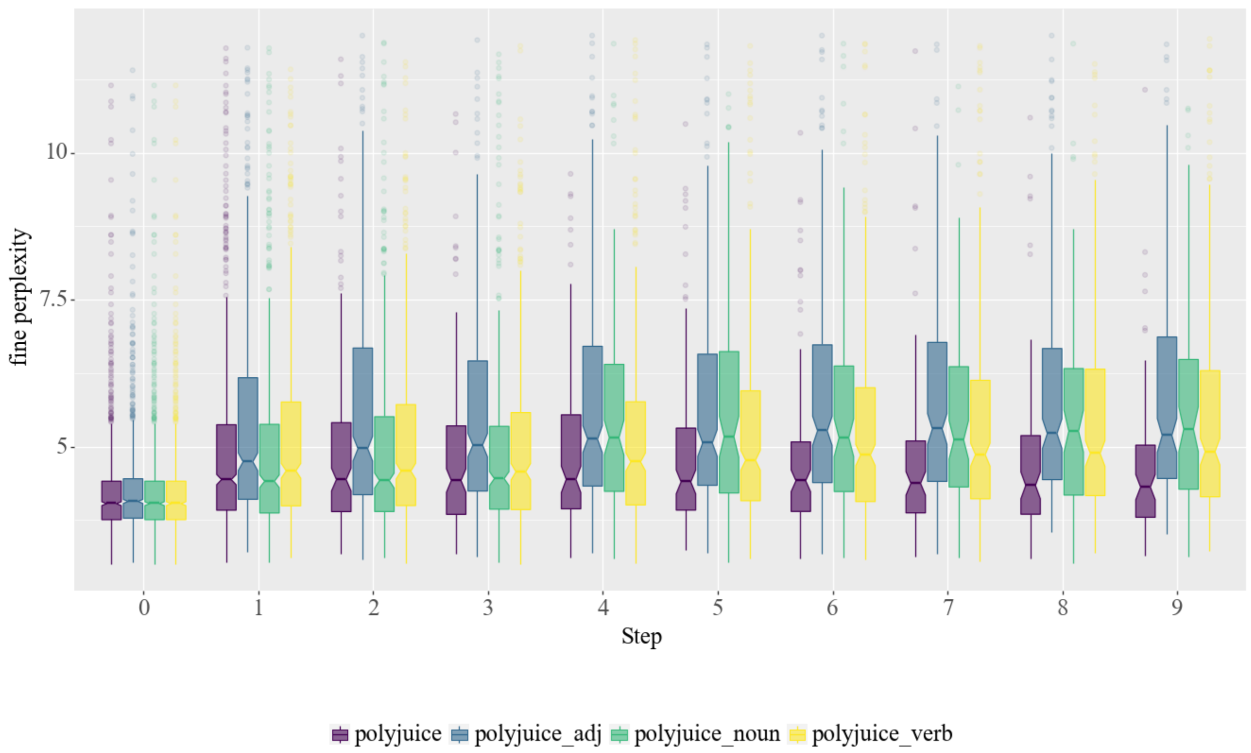


Figure A.24: Fine Perplexity on Polyjuice on the NewsGroups dataset.

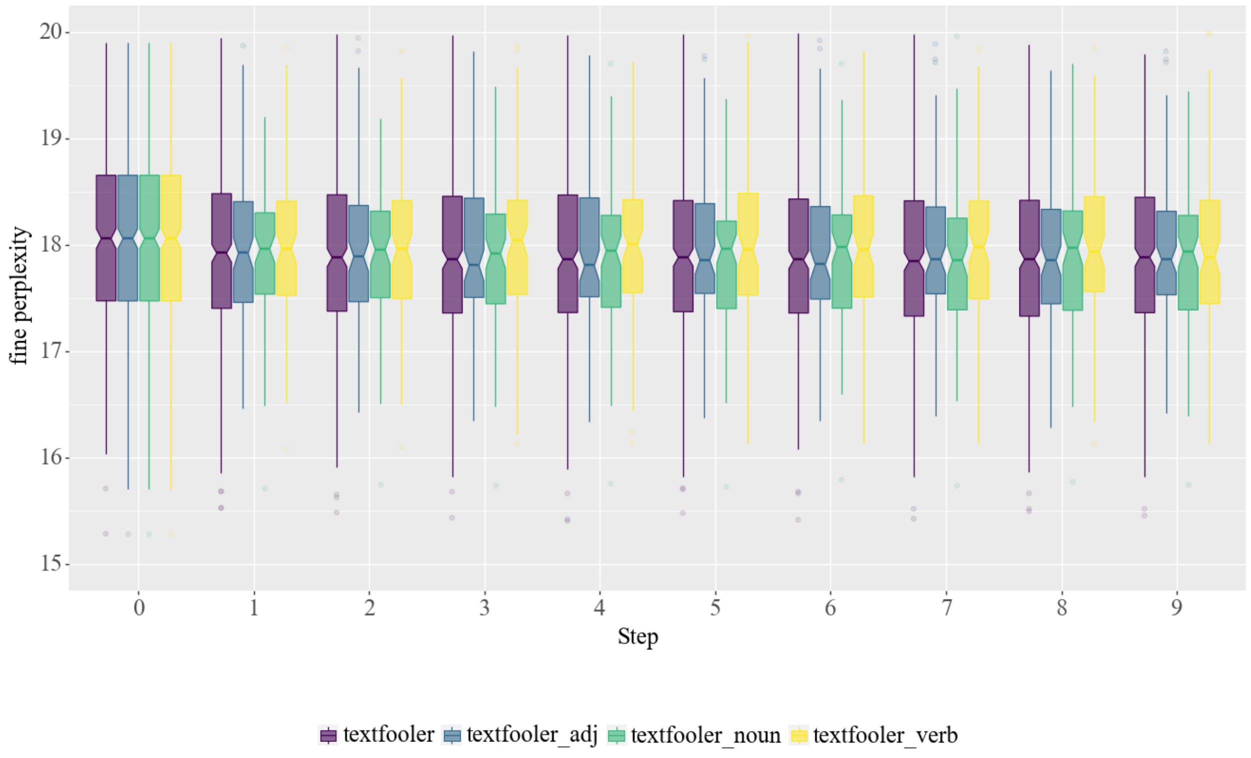


Figure A.25: Fine Perplexity on TextFooler on the IMDb dataset.

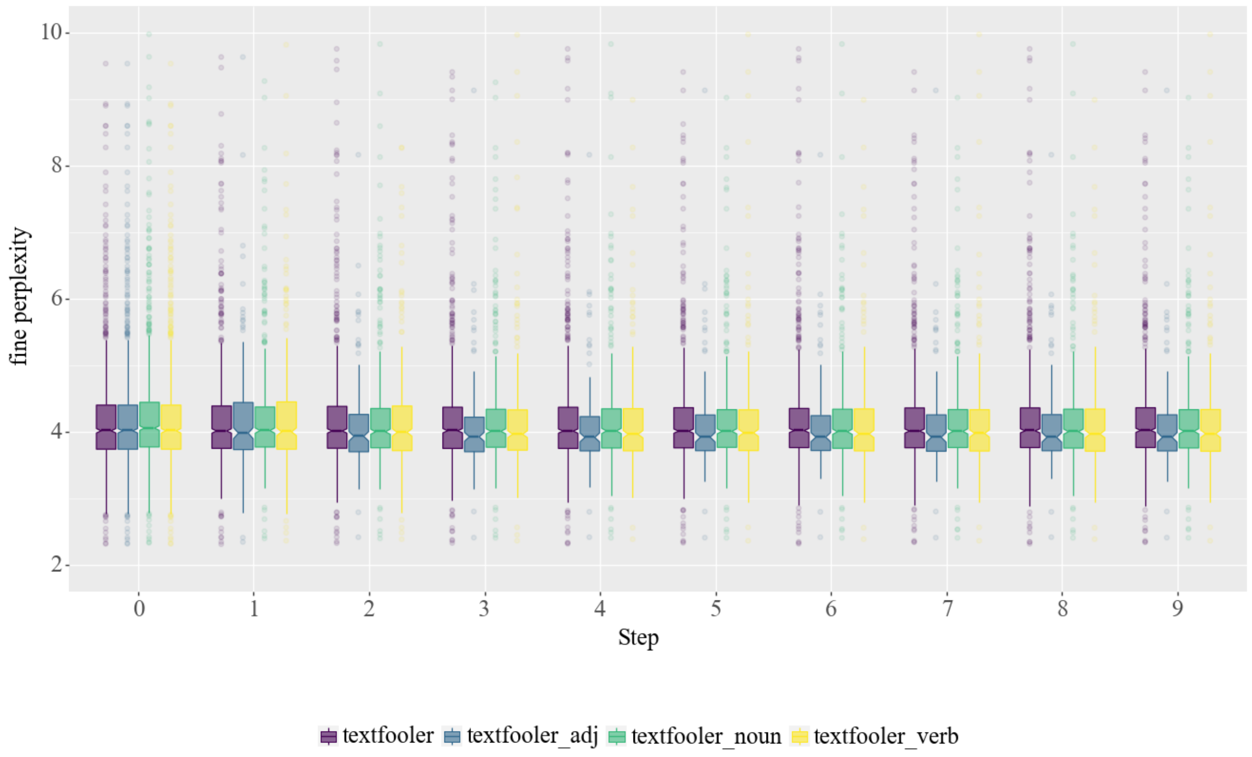


Figure A.26: Fine Perplexity on TextFooler on the NewsGroups dataset.

B Analysis on part-of-speech tags

An important aspect of our part-of-speech tag method presented in 5.2.3 is the distribution of part-of-speech tags in the text. We notice in Figure B.1 that in both datasets and as it stands in more texts, there are more nouns than verbs than adjectives.

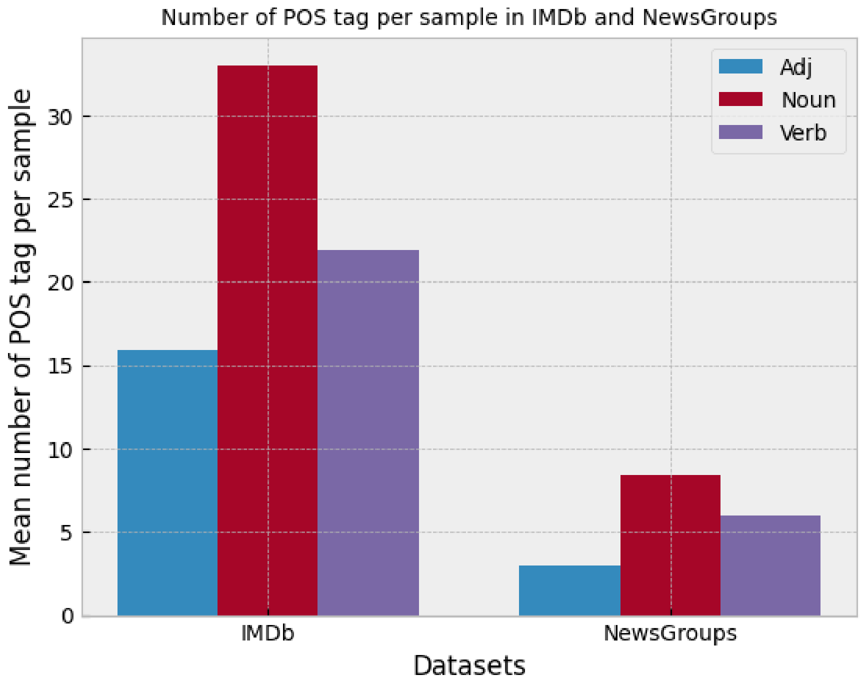


Figure B.1: Part-of-speech tags distribution in the two datasets.

Moreover, in order to interpret each editor's behavior when targeting a part of speech, it is important to calculate how many words of that particular part of speech it modifies. As a result, we present Figures B.2 and B.3 that show the mean number of the tokens of each POS tag in comparison with the mean number of modified tokens of each POS tag. We can notice the same distribution in the mean numbers of swapped tokens per POS tag across all editors. However, we must underline that in Polyjuice the number of modified tokens is much higher than in the other three methods. This behavior explains why Polyjuice does not generate as minimal edits as the other editors, but also why it flips more samples. As for MiCE with attention masking used in IMDb, we notice a higher number of modified tokens in the adjective POS tag. This is probably due to the fact that MiCE's masking mechanism unveils that the adjectives are more influential for the classifier for the task of sentiment analysis rather than other POS tags that usually carry more neutral information. In general, we notice that the methods that use random masking (Polyjuice and MiCE with random masking) have a higher number of swapped tokens per POS tag and thus create less minimal counterfactuals.

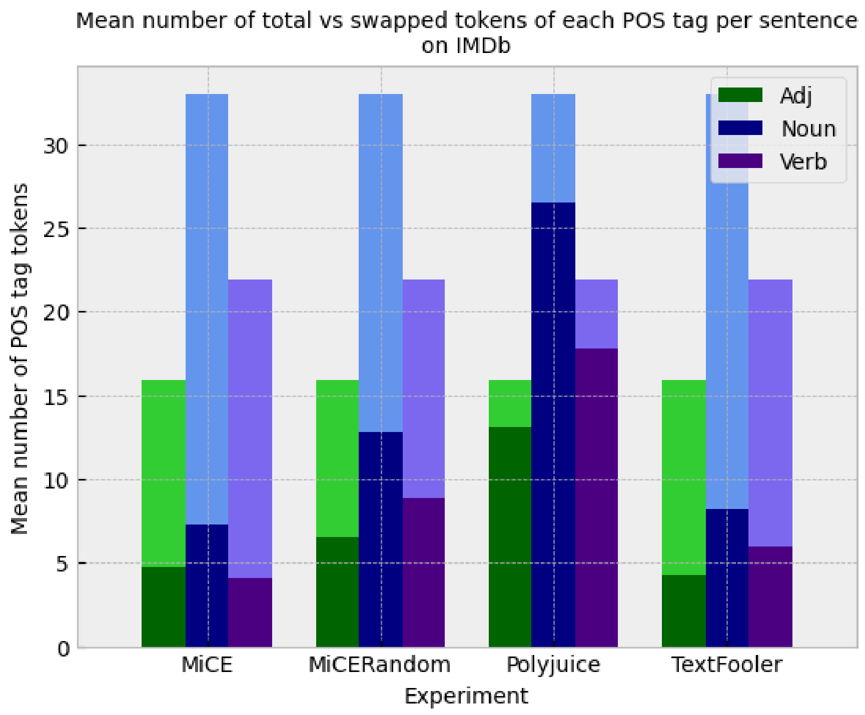


Figure B.2: Part-of-speech tags distribution in the two datasets.

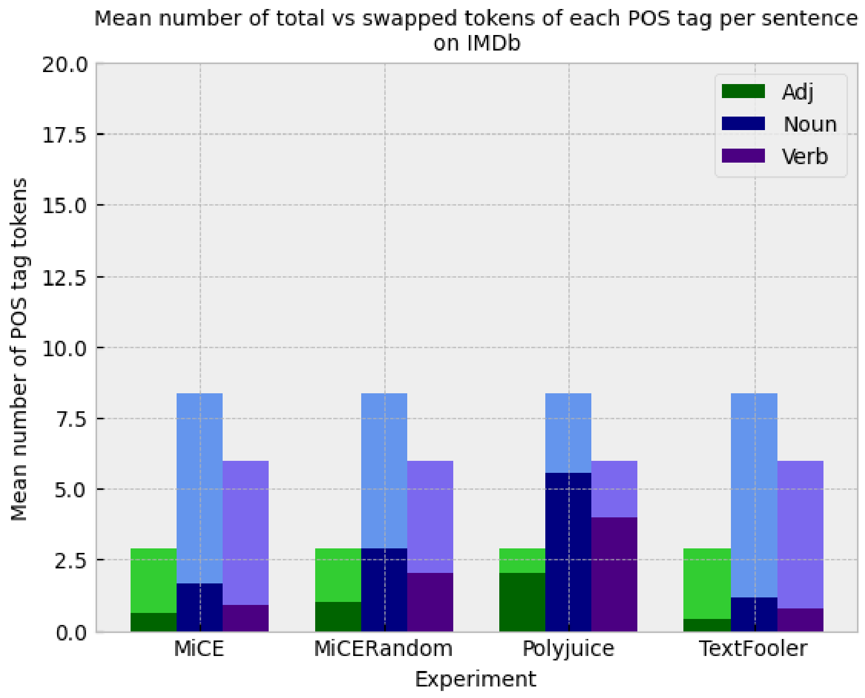


Figure B.3: Part-of-speech tags distribution in the two datasets.

C Duration

Concerning the duration of our experiments, we present some complementary figures for section C. In Figure C.1 , we observe how Polyjuice consumed the majority of the runtime for the experiments. On the other hand, we see that TextFooler only required 2% of the total runtime, which makes it more efficient than the others if we take into account its efficacy across all qualitative metrics.

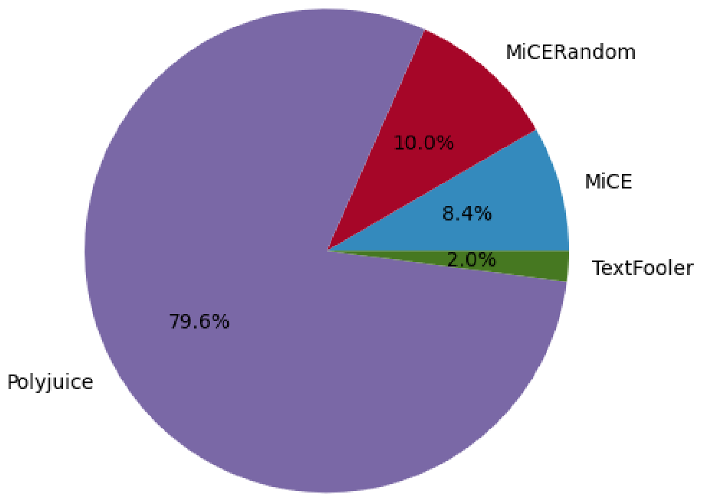


Figure C.1: Pie diagram for the duration of experiments per editor.

Finally, we also present Figure C.2 which shows the amount of time consumed for the experiments of each dataset. Despite the fact, that NewsGroups contains shorter sentences, its 1000 samples require more time than the 500 used in IMDb.

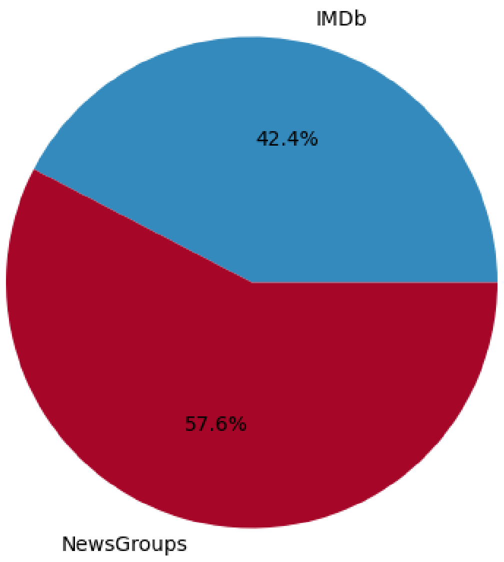


Figure C.2: Pie diagram for the duration of experiments per dataset.

D Extra Qualitative Results

A.1 IMDb Dataset

Example 1

Table 10.1

Step	MiCE
1	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the **jumping start** of the director Thomas Vinterberg's great **carrier**.
2	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the **absolute worst** of the director Thomas Vinterberg's great **masterpiece**.
3	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a **happy** ending is what makes this film the absolute **crowning** of the director Thomas Vinterberg's great masterpiece.
4	The biggest **stinker**, is one of the greatest movies ever. A good story, great actors and a happy ending is what makes this film the absolute **trashing** of the director Thomas Vinterberg's great masterpiece.
5	The biggest **Marx Brothers trashing**, is one of the **funniest** movies ever. A good story, great actors and a happy ending is what makes this film the **greatest** trashing of the director Thomas Vinterberg's great masterpiece.
6	The biggest **Swedish film** trashing, is one of the funniest movies ever. A good story, great actors and a happy ending is what makes this film the greatest **ripoff** of the director Thomas Vinterberg's great masterpiece.
7	The biggest Swedish film trashing, is one of the funniest movies ever. A good story, great actors and a happy ending is what makes this film the greatest **trash** of the director Thomas Vinterberg's great masterpiece.
8	The biggest Swedish film trashing, is one of the funniest movies ever. A good story, great actors and a happy ending is what makes this film the **ruin** of the director Thomas Vinterberg's great masterpiece.
9	The biggest Swedish film **ever**, is one of the funniest movies ever. A good story, great actors and a happy ending is what makes this film the **highlight** of the director Thomas Vinterberg's great **career**.
10	The biggest Swedish film ever, is one of the funniest movies ever. A good story, great actors and a happy ending is what makes this film the **worst** of the director Thomas Vinterberg's great career.

Table 10.2

Step	MiCE ADJ
1	The **biggest** heroes, is one of the **greatest** movies ever. A good story, great actors and a brilliant ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
2	The **great carrier,** heroes, is one of the **worst horror** movies ever. A good story, great actors and a brilliant ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
3	The great carrier, heroes, is one of the **best** horror movies ever. A good story, great actors and a **surprisingly satisfying** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
4	The **original** carrier, heroes, is one of the best horror movies ever. A good story, great actors and a surprisingly **predictable** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
5	The original carrier, heroes, is one of the best horror movies ever. A good story, great actors and a surprisingly **satisfying** ending is what makes this film the jumping start of the director Thomas Vinterberg's great carrier.
6	The **liminal** carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's **romplimlim** carrier.
7	The **romplimpig** carrier, heroes, is one of the **best** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's **romplimpig** carrier.
8	The romplimpig carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.
9	The romplimpig carrier, heroes, is one of the **greatest** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.
10	The romplimpig carrier, heroes, is one of the **worst** horror movies ever. A good story, great actors and a surprisingly satisfying ending is what makes this film the jumping start of the director Thomas Vinterberg's romplimpig carrier.

Table 10.3

Step	**MiCERandom**
1	The biggest heroes, **is one** of **the greatest** movies ever. **A good** story, **great** actors and a brilliant **ending** is what **makes this film the jumping start of** the director **Thomas** Vinterberg's great carrier**.**
2	The biggest heroes, **heroes** of **slasher** movies ever. **Stupid** story, **bad** actors and a brilliant **script** is what **saved** the director **Wolfgang** Vinterberg's great carrier **-operatio!**
3	The **original** heroes, heroes of **slash** movies ever **created** **-** actors and **actresses ! This** is **legendary** director **Üne** Vinterberg's great carrier **-opera duo..**
4	The original heroes, heroes **&** slash movies ever created **! WORST** actors and actresses ! This is legendary director Üne Vinterberg's great carrier -opera duo..
5	The original heroes, heroes & slash movies ever created ! WORST actors and actresses ! This is legendary director Üne Vinterberg's great carrier -opera **du c**..
6	**!** original heroes**!! Original** heroes **!!! Cheaper** slash movies **.. WORST story** ! WORST actors **&** actresses ! This **remake of** legendary director **Werner** Vinterberg's great carrier -opera **movie screamed great**..
7	! **Original** heroes!! Original heroes !!! Cheaper slash movies **!! WORST horror.** WORST story ! WORST actors & actresses ! **Complete** remake of legendary director Werner Vinterberg**' s** great carrier**.. Mockopera** movie screamed great..
8	**Classic Movies**! Original heroes!! Original heroes ! **Exceptional** movies !! **Original animation! PERFECT acting**. **TERT** story ! **GERT** actors / actresses ! Complete **resemblance to** legendary **legend** **Wer niet Vinter va** carrier.. Mockopera movie **s emble ly** great..
9	**Script idiots**! Original heroes!! Original heroes ! **OOPERATIVE acting**! Original animation**. COMPLETEERFECT !!** TERT **animation** ! GERT actors / **actress -educating idiots !!**! Complete **re semblance** to legendary legend **män** niet Vinter va carrier**. SUBSTANT** Mockopera movie s **tremble een nowlsy** great..
10	Script idiots! Original heroes!! Original heroes ! OOPERATIVE acting! Original animation. COMPLETEERFECT !! TERT **TERRIFICATIONS** ! GERT actors / actress -educating **idiot s** !!! Complete re semblance to legendary legend män niet **einter** va carrier. SUBSTANT Mockopera movie s tremble **ien** nowlsy great..

Table 10.4

Step	MiCE 120 beams
1	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the **jumping** start of the director Thomas Vinterberg's great **carrier**.
2	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the **weakest** start of the director Thomas Vinterberg's great **career**.
3	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the **coolest** start of the director Thomas Vinterberg's great career.
4	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the **embarrassing** start of the director Thomas Vinterberg's great career.
5	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the **absolute** start of the director Thomas Vinterberg's great career.
6	**Unforgivable**, **but** one of the greatest **thrillers** ever. A good story, great actors and a brilliant ending is what makes this film the **terrible** start of the director Thomas Vinterberg's great career.
7	**Unforg ivable**, but one of the greatest thrillers ever. A good story, great actors and a brilliant ending is what makes this film the **embarrassing** start of the director Thomas Vinterberg's great career.
8	Unforg ivable, but one of the greatest thrillers ever. A good story, great actors and a brilliant ending is what makes this film the **absolute** start of the director Thomas Vinterberg's great career.
9	**Absolutely abominable**, **as** one of the greatest thrillers ever. A good story, great actors and a brilliant ending is what makes this film the absolute start of the director Thomas Vinterberg's great career.
10	Absolutely abominable, as one of the greatest thrillers ever. A good story, great actors and a brilliant ending is what makes this film the absolute start of the director Thomas Vinterberg's great career.

Table 10.5

Step	Polyjuice
1	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what makes this film the jumping start of the director Thomas Vinterberg**'s great carrier**.
2	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what **doesn't** makes this film the jumping start of the director Thomas Vinterberg **EMPTY** .
3	The biggest heroes, is one of the greatest movies ever. A good story, great actors and a brilliant ending is what doesn't **EMPTY** makes this film the jumping start of the director Thomas Vinterberg EMPTY .
4	The biggest **problem with "Dan in Real Life" is that the rebellious**, **middle daughter** is **played so outrageously by actress Brittany Robertson**.
5	The biggest problem with "Dan in Real Life" is that the rebellious, middle daughter is **never** played so outrageously by actress Brittany Robertson.
6	The biggest problem with "Dan in Real Life" is that the rebellious, middle daughter is **actually** played so outrageously by actress Brittany Robertson.
7	The biggest **strength of** "Dan in Real Life" is that the rebellious, middle daughter is actually played so outrageously by actress Brittany Robertson.
8	The biggest **problem** of "Dan in Real Life" is that the rebellious, middle daughter is actually played so outrageously by actress Brittany Robertson.
9	The biggest **strength** of "Dan in Real Life" is that the rebellious, middle daughter is actually played so outrageously by actress Brittany Robertson.
10	The biggest strength of "Dan in Real Life" is that the rebellious, middle daughter is actually **EMPTY** played so outrageously by actress Brittany Robertson.

Table 10.6

Step	**TextFooler**
1	The biggest heroes, is one of the greatest movies ever. **A** good story, great actors and a brilliant ending is what **makes** this film the jumping **start** of the director Thomas Vinterberg's great carrier.
2	The biggest heroes, is one of the greatest movies ever. **another** good story, great actors and a brilliant ending is what **do** this film the jumping **beginner** of the director Thomas Vinterberg's great carrier.
3	The biggest heroes, is one of the greatest movies ever. another good story, great actors and a brilliant ending is what do this film the jumping **beginners** of the director Thomas Vinterberg's great carrier.
4	The biggest heroes, is one of the greatest movies ever. another good **historic**, great actors and a brilliant ending is what do this **teatro** the jumping beginners of the **directors** Thomas Vinterberg's great carrier.
5	The biggest heroes, is one of the greatest movies ever. another good historic, great actors and a brilliant ending is what do this teatro the jumping **novices** of the directors Thomas Vinterberg's great carrier.
6	The biggest heroes, is one of the greatest movies ever. another good historic, great **officer** and a **shiny** ending is what do this teatro the jumping novices of the directors Thomas Vinterberg's great carrier.
7	The biggest heroes, is one of the greatest movies ever. another good historic, great officer and a shiny ending is what do this teatro the jumping **newbies** of the directors Thomas Vinterberg's great carrier.
8	The biggest heroes, is one of the greatest movies ever. another good historic, great officer and a shiny **end** is what do this teatro the jumping newbies of the directors Thomas Vinterberg's great carrier.
9	The biggest heroes, is one of the greatest movies ever. another good historic, great officer and a shiny end is what do this teatro the jumping **starters** of the directors Thomas Vinterberg's great carrier.
10	The biggest heroes, is one of the greatest movies ever. another good historic, great officer and a shiny end is what do this teatro the jumping starters of the directors Thomas Vinterberg's **grand** carrier.

Example 2

Table 10.7

Step	**MiCE**
1	This movie was **horrible** and **corny**. **James** Agee is rolling in his grave.This movie was nothing at all like the book and made a **mockery** of it. No one should see this movie unless they want to gag.
2	This movie was **hilarious** and **totally amazing**. **Jean Du** Agee is rolling in his grave.This movie was nothing at all like the book and made a **lot** of it. No one should see this movie unless they want to gag.
3	This movie was **horrible** and totally amazing. Jean Du Agee is rolling in his grave.This movie was nothing at all like the book and made a lot of it. No one should see this movie unless they want to **laugh**.
4	This movie was **simply excellent** and totally **hilarious**. Jean Du Agee is **perfect** in his **role**.This movie was nothing at all like the book and made a lot of it. No one should see this movie unless they want to laugh.
5	This movie was simply excellent and totally **embarrassing**. Jean Du Agee is perfect in his role.This movie was nothing at all like the book and made a lot of it. No one should see this movie unless they want to laugh.
6	This movie was simply **hilarious** and totally **unique**. Jean Du Agee is perfect in his role.This movie was nothing at all like the book and made a lot of it. No one should see this movie unless they want to laugh.
7	This movie was simply **terrible** and totally unique. Jean Du Agee is perfect in his role.This movie was nothing at all like the **original** and made a lot of it. No one should see this movie unless they want to laugh.
8	This movie was simply **amazing** and totally unique. Jean Du Agee is perfect in his role.This movie was nothing at all like the original and made a lot of it. No one should see this movie unless they **like** to laugh.
9	This movie was simply **terrible** and totally unique. Jean Du Agee is perfect in his role.This movie was nothing at all like the original and made a lot of it. No one should see this movie unless they like to laugh.
10	This movie was simply **wonderful** and totally unique. Jean Du Agee is perfect in his role.This movie was nothing at all like the original and made a lot of it. No one should see this movie unless they like to laugh.

Table 10.8

Step	**MiCERandom**
1	This movie was **horrible** and corny. **James Agee** is **rolling** in his **grave**.This movie **was** nothing at all like the book and made a **mockery** of it. No one should **see this movie unless** they want **to** gag.
2	This movie was **silly** and corny. **Teen Age Joe** is **really good** in his **roles**.This movie **portrayed** nothing at all like the book and made a **joke** of it. No one should **think** they want **a** gag.
3	This movie was **extremely** silly and corny. Teen Age Joe **sucked** really good in his roles.This movie portrayed nothing at all like the book and made a joke of it. No one should think they want a gag.
4	This movie was extremely silly and corny. Teen Age Joe sucked really good in his roles.This movie portrayed nothing at all like the book and made a joke of it.No one should think they **wrote** a gag.
5	This movie was extremely silly and corny. Teen **ager** Joe sucked really good in his roles.This movie portrayed nothing at all like the book and made a joke of it. No one should think **he** wrote a gag.
6	This movie was extremely **boring** and corny. Teen ager Joe **s warts** really good in his roles.This movie portrayed nothing at all like the book and made a joke of it. No one should think he wrote a gag.
7	This movie was extremely **silly** and corny. Teen ager Joe s **wart looks** really good in **their** roles. This movie **looked** nothing at all like the book and made a joke **about** it. No one should think he wrote a gag.
8	This movie was extremely silly and corny. Teen ager Joe s wart looks really good in their roles. This movie **read** nothing at all like the book and made a joke about it. No one should think **Eugene** wrote a gag.
9	This movie was extremely silly and corny. Teen ager Joe s wart looks really **pathetic** in their roles. This movie read nothing at all like the book and made a joke about it. No one should think Eugene wrote a gag.
10	This movie **is** extremely silly / corny. Teen ager Joe **Bog** wart looks really **approximathetic** in their roles. This movie read nothing at all like the book and made **hilarious** joke **. I loved** it**!** No one should think **Brooke** wrote a gag.

Table 10.9

Step	**MiCE 120 beams**
1	This movie was **horrible** and **corny. James** Agee is **rolling** in his **grave**.This movie was nothing at all like the book and made a **mockery** of it. No one should see this movie unless they want to **gag**.
2	This movie was **great** and **John Le** Agee is **great** in his **role**.This movie was nothing at all like the book and made a **lot** of it. No one should see this movie unless they want to **laugh**.
3	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **mockery** of it. No one should see this movie unless they want to laugh.
4	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **lot** of it. No one should see this movie unless they want to laugh.
5	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **mockery** of it. No one should see this movie unless they want to laugh.
6	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **lot** of it. No one should see this movie unless they want to laugh.
7	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **mockery** of it. No one should see this movie unless they want to laugh.
8	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **lot** of it. No one should see this movie unless they want to laugh.
9	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **mockery** of it. No one should see this movie unless they want to laugh.
10	This movie was great and John Le Agee is great in his role.This movie was nothing at all like the book and made a **lot** of it. No one should see this movie unless they want to laugh.

Table 10.10

Step	**Polyjuice**
1	This movie was **horrible and corny. James Agee is rolling in his grave.This movie was nothing at all like the book** and made a mockery of it. **No one should see this movie unless they want to gag.**
2	This movie was **great** and made a mockery of it. **EMPTY**
3	This movie was great and made a mockery of it.
4	This movie was great and made a **star** of it.
5	This movie was **not** great and made a star of it.
6	This movie was **EMPTY** great and made a star of it.
7	This movie **lacked** EMPTY great and made a star of it.
8	This movie **made** EMPTY great and made a star of it.
9	This movie made EMPTY great and made a **mockery** of it.
10	This movie made EMPTY great and made a **difference** of it.

Table 10.11

Step	**TextFooler**
1	This movie was **horrible** and corny. James **Agee** is rolling in his grave.This movie was nothing at all **like** the book and **made** a mockery of it. **No** one **should** see this **movie** unless they want to gag.
2	This movie was **abysmal** and corny. James **Jefferies** is rolling in his grave.This movie was nothing at all **adore** the book and **tabled** a mockery of it. **Anything** one **ought** see this **film** unless they want to gag.
3	This movie was abysmal and corny. James Jefferies is rolling in his grave.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought **seeing** this film unless they want to gag.
4	This movie was abysmal and corny. James Jefferies is rolling in his **burying**.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought seeing this film unless they want to gag.
5	This movie was **terrible** and corny. James Jefferies is rolling in his burying.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought seeing this film unless they want to gag.
6	This movie was **abysmal** and corny. James Jefferies is rolling in his burying.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought seeing this film unless they want to gag.
7	This movie was **terrible** and corny. James Jefferies is rolling in his burying.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought seeing this film unless they want to gag.
8	This movie was **abysmal** and corny. James Jefferies is rolling in his burying.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought seeing this film unless they want to gag.
9	This movie was **terrible** and corny. James Jefferies is rolling in his burying.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought seeing this film unless they want to gag.
10	This movie was **abysmal** and corny. James Jefferies is rolling in his burying.This movie was nothing at all adore the book and tabled a mockery of it. Anything one ought seeing this film unless they want to gag.

A.2 NewsGroups Dataset

Example 1

Table 10.12

Step	**MiCE**
1	Anyone have a phone number for Applied **Engineering** so I can give them a call?
2	Anyone have a phone number for Applied **Microbiology** so I can give them a call?
3	Anyone have a phone number for Applied **X** so I can give them a call?
4	Anyone have a phone number for **PC Disc Makers** so I can give them a call?
5	Anyone have a phone number for PC **Key** Makers so I can give them a call?
6	Anyone have a phone number for PC **File** Makers so I can give them a call?
7	Anyone have a phone number for **SNES** so I can give them a call?
8	Anyone have a phone number for **Dell** so I can give them a call?
9	Anyone have a **good place** for **selling CDs?** I can give them a call?
10	Anyone have a good place for selling **sci.science**? I can give them a call?

Table 10.13

Step	**MiCE NOUN**
1	Anyone have a **phone** number for Applied Engineering so I can give them a call?
2	Anyone have a **telephone** number for Applied Engineering so I can give them a call?
3	Anyone have a **fax** number for Applied Engineering so I can give them a call?
4	Anyone have a **telephone** number for Applied Engineering so I can give them a call?
5	Anyone have a **phone** number for Applied Engineering so I can give them a call?
6	Anyone have a **telephone** number for Applied Engineering so I can give them a call?
7	Anyone have a **phone** number for Applied Engineering so I can give them a call?
8	Anyone have a **telephone** number for Applied Engineering so I can give them a call?
9	Anyone have a **phone** number for Applied Engineering so I can give them a call?
10	Anyone have a **telephone** number for Applied Engineering so I can give them a call?

Table 10.14

Step	**MiCERandom**
1	Anyone **have** a phone number for Applied Engineering so I can give them a call?
2	Anyone **got** a phone number for Applied Engineering so I can give them a call?
3	Anyone got a phone number for Applied Engineering so I can give them a call?
4	Anyone got a phone number for Applied Engineering so I can give them a call?
5	Anyone got a phone number for Applied Engineering so I can give them a call?
6	Anyone got a phone number for **GS** Engineering so I can give them a call?
7	Anyone got a phone number for GS **Software** so I can give them a call?
8	Anyone got a **coupon** for **theS NES** so I can **buy my Sony VHS card? Anyone** give **me** a call?
9	Anyone got a coupon for theS **ALESTORES** so I can buy **an SD** VHS card? Anyone give me a call?
10	Anyone **know of** coupon **codes** for theS **ALE DEALS** so I can buy an **AT&T** VHS card? Anyone give me a **mailing address! ———.**

Table 10.15

Step	**MiCE 120 beams**
1	Anyone have a phone number for Applied **Engineering** so I can give them a call?
2	Anyone have a phone number for Applied **Physics** so I can give them a call?
3	Anyone have a phone number for Applied **Visualisation** so I can give them a call?
4	Anyone have a phone number for **Dolby Audio** so I can give them a call?
5	Anyone have a phone number for **Spaceby** Audio so I can give them a call?
6	Anyone have a phone number for **Spacescape** Audio so I can give them a call?
7	Anyone have a phone number for **Sony** Audio so I can give them a call?
8	Anyone have a phone number for **Space** Audio so I can give them a call?
9	Anyone have a phone number for Space **Graphics** so I can give them a call?
10	Anyone have a phone number for Space **Research** so I can give them a call?

Table 10.16

Step	**Polyjuice**
1	Anyone have a phone number for Applied Engineering so I can give them **a** call?
2	Anyone have a phone number for Applied Engineering so I can give them **EMPTY** call?
3	
4	

Table 10.17

Step	**TextFooler**
1	Anyone have a phone **number** for Applied Engineering so I can give them a call?
2	Anyone have a phone **numbers** for Applied Engineering so I can give them a call?
3	Anyone have a phone numbers for Applied **Tech** so I can give them a call?
4	Anyone have a phone numbers for **Conducted** Tech so I can give them a call?
5	Anyone have a phone numbers **de** Conducted Tech so I can give them a call?
6	Anyone have a phone numbers de Conducted **Technology** so I can give them a call?
7	Anyone have a phone numbers de **Proceeded** Technology so I can give them a call?
8	Anyone have a phone **numero** de Proceeded Technology so I can give them a call?
9	Anyone have a phone numero de **Embarked** Technology so I can give them a call?
10	Anyone have a phone numero de Embarked **Technological** so I can give them a call?

Table 10.18

Step	**TextFooler NOUN**
1	Anyone have a phone **number** for Applied Engineering so I can give them a call?
2	Anyone have a phone **numbers** for Applied Engineering so I can give them a call?
3	Anyone have a phone numbers for Applied Engineering so I can give them a **contacting**?
4	Anyone have a phone **number** for Applied Engineering so I can give them a contacting?
5	Anyone have a **telephoning** number for Applied Engineering so I can give them a contacting?
6	Anyone have a telephoning **nombre** for Applied Engineering so I can give them a contacting?
7	Anyone have a telephoning **number** for Applied Engineering so I can give them a contacting?
8	Anyone have a telephoning **nombre** for Applied Engineering so I can give them a contacting?
9	Anyone have a telephoning **number** for Applied Engineering so I can give them a contacting?
10	Anyone have a telephoning **nombre** for Applied Engineering so I can give them a contacting?

Example 2

Table 10.19

Step	**MiCE**
1	**Static** test firings are now scheduled for this Saturday.....after many schedule changes.....
2	**Xserver** test firings are now scheduled for this Saturday.....after many schedule changes.....
3	**Programs** are now scheduled for this Saturday.....after many schedule changes.....
4	**Game** are now scheduled for this Saturday.....after many schedule changes.....
5	**DVD nights** are now scheduled for this Saturday.....after many schedule changes.....
6	**CPU** nights are now scheduled for this Saturday.....after many schedule changes.....
7	**Hockey** nights are now scheduled for this Saturday.....after many schedule changes.....
8	**DVD** nights are now scheduled for this Saturday.....after many schedule changes.....
9	**Mac** nights are now scheduled for this Saturday.....after many schedule changes.....
10	**DVD** nights are now scheduled for this Saturday.....after many schedule changes.....

Table 10.20

Step	**MiCERandom**
1	Static test firings are now scheduled for this Saturday**.....**after many schedule changes.....
2	Static test firings are now scheduled for this Saturday**,**after many schedule changes.....
3	Static **playoffs** are now scheduled for this **month**,after many schedule changes.....
4	Static **mailings** are now scheduled for this month, **with** many schedule changes.....
5	Static mailings are now scheduled for this **week**, with many schedule changes.....
6	Static mailings are now scheduled for this week, with many schedule changes.....
7	Static mailings are now scheduled for this week, with schedule changes.....
8	Static mailings are now scheduled for this week, with schedule changes.....
9	Static mailings are **up** for this week, with schedule changes.....
10	Static mailings are up for this week, with schedule changes.....

Table 10.21

Step	**MiCE 120 beams**
1	**Static** test firings are now scheduled for this Saturday.....after many schedule changes.....
2	**ATI** test firings are now scheduled for this Saturday.....after many schedule changes.....
3	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....
4	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....
5	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....
6	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....
7	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....
8	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....
9	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....
10	ATI test firings are now scheduled for this Saturday.....after many schedule changes.....

Table 10.22

Step	**Polyjuice**
1	Static test firings are now **scheduled** for this Saturday**.....**after many schedule changes**.....**
2	Static test firings are now **available** for this Saturday **,** after many schedule changes**.**
3	Static test firings are now available for this Saturday , after **testing several more** .
4	Static test firings are now available for this Saturday, **and** testing several **static** .
5	Static test firings are now available for **EMPTY** , and **there is** several static **testing noted** .
6	Static test **tests** are now available for EMPTY , and there is **a few** static **sketches** .
7	Static test tests are now available EMPTY , and there is a few static **methods** .
8	Static test tests are now **included** , and there is a few static methods **for simple targets** .
9	Static test tests are now included , and there **are several** static methods for simple **subinterval's** .
10	

Table 10.23

Step	**TextFooler**
1	**Static** test firings are now scheduled for this Saturday.....after many schedule changes.....
2	**Immobile** test firings are now scheduled for this Saturday.....after many schedule changes.....
3	**Stationary** test firings are now scheduled for this **Sabbath**.....after many schedule changes.....
4	Stationary test firings are now scheduled for this **Saturday**.....after many schedule changes.....
5	Stationary test firings are now scheduled for this **Shabbat**.....after many schedule changes.....
6	Stationary test firings are now scheduled for this **Saturday**.....after many schedule changes.....
7	Stationary test firings are now scheduled for this **Shabbat**.....after many schedule changes.....
8	Stationary test firings are now scheduled for this **Saturday**.....after many schedule changes.....
9	Stationary test firings are now scheduled for this **Shabbat**.....after many schedule changes.....
10	Stationary test firings are now scheduled for this **Saturday**.....after many schedule changes.....

Table 10.24

Step	**TextFooler NOUN**
1	Static test firings are now scheduled for this Saturday.....after many **schedule** changes.....
2	Static test firings are now scheduled for this Saturday.....after many **programming** changes.....
3	Static **experiment** firings are now scheduled for this Saturday.....after many programming changes.....
4	Static **experience** firings are now scheduled for this Saturday.....after many programming changes.....
5	Static experience firings are now scheduled for this Saturday.....after many **broadcasting** changes.....
6	Static experience firings are now scheduled for this Saturday.....after many **programming** changes.....
7	Static experience firings are now scheduled for this Saturday.....after many **broadcasting** changes.....
8	Static experience firings are now scheduled for this Saturday.....after many **programming** changes.....
9	Static experience firings are now scheduled for this Saturday.....after many **broadcasting** changes.....
10	Static experience firings are now scheduled for this Saturday.....after many **programming** changes.....