Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Adversarial Attacks on the Natural Language Inference Task

Using Natural Language Explanations to Enhance Adversarial Robustness

---

Διπλωματικη Εργασια

του

ΑΛΕΞΑΝΔΡΟΥ Β. ΚΟΥΛΑΚΟΥ

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Adversarial Attacks on the Natural Language Inference Task

Using Natural Language Explanations to Enhance Adversarial Robustness

---

Διπλωματικη Εργασια

του

ΑΛΕΞΑΝΔΡΟΥ Β. ΚΟΥΛΑΚΟΥ

Επιβλέπων: Γεώργιος Στάμου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Ιουλίου 2024.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Στάμου	Αθανάσιος Βουλόδημος	Στέφανος Κόλλιας
Καθηγητής Ε.Μ.Π.	Επίκουρος Καθηγητής Ε.Μ.Π.	Ομότιμος Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2024

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνουμε, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμάς προσωπικά και αποκλειστικά και ότι, αναλαμβάνουμε πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μας ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Αλέξανδρος Κουλάκος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

16η Ιουλίου 2024

Περίληψη

Τα Νευρωνικά Δίκτυα Βαθιάς μάθησης έχουν καταφέρει τα τελευταία χρόνια να αντιμετωπίσουν με επιτυχία διάφορα προβλήματα που εντάσσονται στην Επεξεργασία Φυσικής Γλώσσας (π.χ. ταξινόμηση κειμένου, σύνοψη, μετάφραση, συμπερασμός φυσικής γλώσσας). Ωστόσο, ειδικά στο πρόβλημα του συμπερασμού φυσικής γλώσσας, έχει αποδειχθεί ότι τα σύγχρονα μοντέλα βαθιάς μάθησης, τα οποία εκπαιδεύονται στο σύνολο δεδομένων SNLI, είναι ευάλωτα σε ανταγωνιστικές επιθέσεις, οι οποίες αποσκοπούν στην εξαπάτηση του μοντέλου με την προσθήκη ανεπαίσθητων διαταραχών σε αυθεντικές εισόδους. Για την αντιμετώπιση αυτού του ζητήματος έχει προταθεί η μέθοδος της ανταγωνιστικής εκπαίδευσης, αλλά αποτυγχάνει να απομακρύνει τη μεροληψία, που υπάρχει εγγενώς στο σύνολο δεδομένων SNLI, από τη διαδικασία πρόβλεψης του μοντέλου. Με βάση την εργασία των Camburu et al., προτείνουμε την τροποποίηση του παραδοσιακού προβλήματος συμπερασμού φυσικής γλώσσας με την ενσωμάτωση εξηγήσεων φυσικής γλώσσας κατά τη διάρκεια της εκπαίδευσης και της εξαγωγής συμπερασμάτων και διεξάγουμε μια σειρά πειραμάτων προκειμένου να επαληθεύσουμε κατά πόσο οι εξηγήσεις φυσικής γλώσσας βελτιώνουν πράγματι την ανθεκτικότητα των μοντέλων. Χρησιμοποιούμε το TextFooler και το BERT-attack ως αλγορίθμους παραγωγής ανταγωνιστικών επιθέσεων και τα πειραματικά αποτελέσματα δείχνουν σταθερά ότι η ενσωμάτωση εξηγήσεων φυσικής γλώσσας στη διαδικασία εκπαίδευσης και εξαγωγής συμπερασμάτων ενισχύει την ανθεκτικότητα απέναντι σε ανταγωνιστικές επιθέσεις.

Λέξεις Κλειδιά

Εξηγήσεις Φυσικής Γλώσσας, Συμπερασμός Φυσικής Γλώσσας, Ανταγωνιστικές Επιθέσεις, Ανταγωνιστική Ανθεκτικότητα, Μετασχηματιστές, Επεξεργασία Φυσικής Γλώσσας

Abstract

DNNs have achieved remarkable success in various Natural Language Processing tasks (e.g., text classification, summarization, machine translation, natural language inference). However, especially in the natural language inference task, it has been shown that state-of-the-art DNN-based models, trained on SNLI dataset, are susceptible to adversarial attacks, which aim to fool the model by adding imperceptible perturbations into legitimate inputs. Adversarial training has been proposed in order to address this issue, but it fails in masking out the SNLI dataset bias from the model's decision-making process. Based on the work of Camburu et al., we propose the modification of the traditional natural language inference task by incorporating natural language explanations during training and inference and we conduct a range of experiments in order to verify whether natural language explanations actually improve adversarial robustness. We use TextFooler and BERT-attack as attack recipes and the experimental results consistently show that incorporating natural language explanations in training and inference process enhances robustness against adversarial attacks.

Keywords

Natural Language Explanations, Natural Language Inference, Adversarial Attacks, Adversarial Robustness, Transformers, Natural Language Processing

στην οικογένειά μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Στάμου για την ευκαιρία που μου έδωσε να εκπονήσω τη διπλωματική μου εργασία στο εργαστήριο Τεχνητής Νοημοσύνης και Συστημάτων Μάθησης (AILS) του ΕΜΠ. Επίσης, ευχαριστώ ιδιαίτερα τους υποψήφιους Διδάκτορες του εργαστηρίου, Μαρία Λυμπεραίου και Γιώργο Φιλανδριανό, για την καθοδήγησή τους και την άριστη συνεργασία που είχαμε καθόλη τη διάρκεια εκπόνησης της εργασίας μου. Θα ήθελα, ακόμη, να ευχαριστήσω τους γονείς μου, Βασίλη και Γεωργία, και την αδερφή μου, Βικτώρια, για την πολύτιμη ηθική υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια. Τέλος, ευχαριστώ τους φίλους μου, και πλέον συναδέλφους μου, Λευτέρη, Θοδωρή, Βασίλη, Δήμητρα και Βαγγέλη, για όλες τις πολύτιμες αναμνήσεις που δημιουργήσαμε εντός και εκτός σχολής, οι οποίες θα με συντροφεύουν για μια ζωή.

Αθήνα, Ιούλιος 2024

Αλέξανδρος Κουλάκος

Table of Contents

List of Figures

List of Images

List of Tables

Part I

Εκτεταμένη Περίληψη στα Ελληνικά

Chapter 1

Το μοντέλο Κωδικοποιητή - Αποκωδικοποιητή βασισμένο σε Μετασχηματιστές

1.1 Εισαγωγή

Το μοντέλο κωδικοποιητή - αποκωδικοποιητή (Encoder-Decoder) που βασίζεται στους μετασχηματιστές (transformers) εισήχθη από τους Vaswani et al. [5] μέσω της περίφημης δημοσίευσης *Attention is all you need* και αποτελεί σήμερα την de facto τυπική αρχιτεκτονική κωδικοποιητή - αποκωδικοποιητή στην Επεξεργασία Φυσικής Γλώσσας (ΕΦΓ) [6, 7]. Σε αυτό το κεφάλαιο, προσπαθούμε να διευκρινίσουμε πώς η αρχιτεκτονική κωδικοποιητή - αποκωδικοποιητή που βασίζεται σε μετασχηματιστές μπορεί να χρησιμοποιηθεί για τη μοντελοποίηση προβλημάτων ακολουθίας προς ακολουθία. Για το σκοπό αυτό, αναλύουμε το μοντέλο κωδικοποιητή - αποκωδικοποιητή βασισμένο σε μετασχηματιστή στο μέρος του κωδικοποιητή και του αποκωδικοποιητή και παρουσιάζουμε πώς το μοντέλο μπορεί να χρησιμοποιηθεί για εξαγωγή συμπερασμάτων με βάση το μαθηματικό του ορισμό.

Αυτό το κεφάλαιο χωρίζεται σε τρία μέρη:

- **Κωδικοποιητής - Αποκωδικοποιητής:** Παρουσιάζουμε το μοντέλο κωδικοποιητή - αποκωδικοποιητή βασισμένο σε μετασχηματιστή και εξηγούμε πώς το μοντέλο μπορεί να χρησιμοποιηθεί για εξαγωγή συμπερασμάτων σε εργασίες ακολουθίας προς ακολουθία.
- **Κωδικοποιητής:** Αναλύουμε το μέρος του κωδικοποιητή του μοντέλου.
- **Αποκωδικοποιητής:** Αναλύουμε το μέρος του αποκωδικοποιητή του μοντέλου.

1.2 Κωδικοποιητής - Αποκωδικοποιητής:

Το 2017, οι Vaswani et al. [5] εισήγαγαν την αρχιτεκτονική *Μετασχηματιστές* και έτσι γέννησαν τα μοντέλα κωδικοποιητή-αποκωδικοποιητή που βασίζονται σε *μετασχηματιστές*.

Τα μοντέλα κωδικοποιητή-αποκωδικοποιητή με βάση τον μετασχηματιστή αποτελούνται από έναν κωδικοποιητή και έναν αποκωδικοποιητή που είναι και οι δύο στοίβες από μπλοκ υπολειμματικής προσοχής. Η βασική καινοτομία των μοντέλων κωδικοποιητή-αποκωδικοποιητή με βάση το μετασχηματιστή είναι ότι αυτά τα μπλοκ υπολειμματικής προσοχής μπορούν να επεξεργαστούν μια ακολουθία εισόδου $\mathbf{X}_{1:n}$ μεταβλητού μήκους n χωρίς να παρουσιάζουν

μια αναδρομική δομή. Η μη εξάρτηση από μια αναδρομική δομή επιτρέπει στους βασισμένους σε μετασχηματιστές κωδικοποιητές-αποκωδικοποιητές να είναι εξαιρετικά παραλληλοποιήσιμοι, γεγονός που καθιστά το μοντέλο τάξεις μεγέθους πιο αποδοτικό από υπολογιστική άποψη από τα βασισμένα σε Αναδρομικά Νευρωνικά Δίκτυα (ΑΝΔ) μοντέλα κωδικοποιητών-αποκωδικοποιητών σε σύγχρονο υλικό.

Για να λύσουμε ένα πρόβλημα ακολουθίας προς ακολουθία, πρέπει να βρούμε μια απεικόνιση μιας ακολουθίας εισόδου $\mathbf{X}_{1:n}$ σε μια ακολουθία εξόδου $\mathbf{Y}_{1:m}$ μεταβλητού μήκους m. Ας δούμε πώς τα μοντέλα κωδικοποιητή-αποκωδικοποιητή που βασίζονται σε μετασχηματιστές χρησιμοποιούνται για την εύρεση μιας τέτοιας αντιστοίχισης.

Τα μοντέλα κωδικοποιητή - αποκωδικοποιητή με βάση το μετασχηματιστή ορίζουν μια υπό συνθήκη κατανομή των διανυσμάτων-στόχων $\mathbf{Y}_{1:m}$ δεδομένης μιας ακολουθίας εισόδου $\mathbf{X}_{1:n}$:

$$p_{\partial_{\text{enc}},\partial_{\text{dec}}}(\mathbf{Y}_{1:m}|\mathbf{X}_{1:n})$$

Το μέρος του κωδικοποιητή που βασίζεται στον μετασχηματιστή κωδικοποιεί την ακολουθία εισόδου $\mathbf{X}_{1:n}$ σε μια ακολουθία κρυφών καταστάσεων $\overline{\mathbf{X}}_{1:n}$, ορίζοντας έτσι την απεικόνιση:

$$f_{\partial_{\text{enc}}} : \mathbf{X}_{1:n} \rightarrow \overline{\mathbf{X}}_{1:n}$$

Το μέρος του αποκωδικοποιητή που βασίζεται σε μετασχηματιστές μοντελοποιεί στη συνέχεια την υπό συνθήκη κατανομή πιθανότητας της ακολουθίας διανυσμάτων-στόχων $\mathbf{Y}_{1:m}$ δεδομένης της ακολουθίας των κωδικοποιημένων κρυφών καταστάσεων $\overline{\mathbf{X}}_{1:n}$:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n}).$$

Με τον κανόνα του Bayes, η κατανομή αυτή μπορεί να παραγοντοποιηθεί σε ένα γινόμενο της υπό συνθήκη κατανομής πιθανότητας του διανύσματος στόχου $\mathbf{y}_i$ δεδομένων των κωδικοποιημένων κρυφών καταστάσεων $\overline{\mathbf{X}}_{1:n}$ και των προηγούμενων διανυσμάτων στόχου $\mathbf{Y}_{0:i-1}$:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$$

Ο αποκωδικοποιητής που βασίζεται στον μετασχηματιστή αντιστοιχίζει την ακολουθία των κωδικοποιημένων κρυφών καταστάσεων $\overline{\mathbf{X}}_{1:n}$ και όλα τα προηγούμενα διανύσματα στόχων $\mathbf{Y}_{0:i-1}$ στο διάνυσμα logit $\mathbf{l}_i$. Το διάνυσμα logit $\mathbf{l}_i$ επεξεργάζεται στη συνέχεια με την πράξη softmax για να οριστεί η υπό συνθήκη κατανομή πιθανότητας $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$, όπως ακριβώς γίνεται για τους αποκωδικοποιητές που βασίζονται σε ΑΝΔ. Ωστόσο, σε αντίθεση με τους αποκωδικοποιητές που βασίζονται σε ΑΝΔ, η κατανομή του διανύσματος στόχου $\mathbf{y}_i$ είναι *explicitly* (ή άμεσα) εξαρτημένη από όλα τα προηγούμενα διανύσματα στόχου $\mathbf{y}_0, \ldots, \mathbf{y}_{i-1}$ όπως θα δούμε αργότερα με περισσότερες λεπτομέρειες. Το υπ' αριθμόν 0 διάνυσμα-στόχος $\mathbf{y}_0$ αναπαρίσταται από ένα ειδικό διάνυσμα "αρχής της πρότασης" BOS.

Έχοντας ορίσει την υπό συνθήκη κατανομή $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$, μπορούμε τώρα να παράγουμε αυτόματα την έξοδο και έτσι να ορίσουμε μια απεικόνιση μιας ακολουθίας εισόδου $\mathbf{X}_{1:n}$ σε μια ακολουθία εξόδου $\mathbf{Y}_{1:m}$ κατά την εξαγωγή συμπερασμάτων.

Συνοψίζοντας:

- Ο κωδικοποιητής που βασίζεται σε μετασχηματιστή ορίζει μια αντιστοίχιση από την ακολουθία εισόδου $\mathbf{X}_{1:n}$ σε μια ακολουθία κωδικοποίησης με βάση το πλαίσιο $\overline{\mathbf{X}}_{1:n}$.

- Ο αποκωδικοποιητής που βασίζεται σε μετασχηματιστή ορίζει την κατανομή πιθανότητας (υπό συνθήκη) $p_{\theta_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$.

- Δεδομένου ενός κατάλληλου μηχανισμού αποκωδικοποίησης, η ακολουθία εξόδου $\mathbf{Y}_{1:m}$ μπορεί να δειγματοληπτηθεί αυτο-παλινδρομικά από $p_{\theta_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n}), \forall i \in \{1, \ldots, m\}$.

Τώρα που δόθηκε μια γενική επισκόπηση του τρόπου λειτουργίας του κωδικοποιητή - αποκωδικοποιητή που βασίζεται σε μετασχηματιστές, μπορούμε να εμβαθύνουμε περισσότερο τόσο στο μέρος του κωδικοποιητή όσο και στο μέρος του αποκωδικοποιητή του μοντέλου. Πιο συγκεκριμένα, θα δούμε ακριβώς πώς ο κωδικοποιητής κάνει χρήση του στρώματος αυτο-προσοχής (self-attention) για να δώσει μια ακολουθία κωδικοποιήσεων διανυσμάτων που εξαρτώνται από το πλαίσιο και πώς τα στρώματα αυτο-προσοχής επιτρέπουν τον αποδοτικό παραλληλισμό.

Στη συνέχεια, θα εξηγήσουμε λεπτομερώς πώς λειτουργεί το στρώμα αυτο-προσοχής στο μοντέλο του αποκωδικοποιητή και πώς ο αποκωδικοποιητής εξαρτάται από την έξοδο του κωδικοποιητή με στρώματα *διασταυρούμενης προσοχής (cross-attention)* για να ορίσει την υπό συνθήκη κατανομή $p_{\theta_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$.

Στην πορεία θα γίνει φανερό πώς τα μοντέλα κωδικοποιητή-αποκωδικοποιητή που βασίζονται σε μετασχηματιστές επιλύουν το πρόβλημα των εξαρτήσεων μεγάλης εμβέλειας των μοντέλων κωδικοποιητή-αποκωδικοποιητή που βασίζονται σε ΑΝΔ.

1.3 Κωδικοποιητής

Όπως αναφέρθηκε στην προηγούμενη ενότητα, ο κωδικοποιητής που βασίζεται σε *μετασχηματιστή* αντιστοιχίζει την ακολουθία εισόδου σε μια ακολουθία κωδικοποίησης με βάση τα συμφραζόμενα:

$$f_{\theta_{\text{enc}}} : \mathbf{X}_{1:n} \to \overline{\mathbf{X}}_{1:n}$$

Ρίχνοντας μια πιο προσεκτική ματιά στην αρχιτεκτονική, ο κωδικοποιητής που βασίζεται σε μετασχηματιστές είναι μια στοίβα από μπλοκ υπολειμματικού κωδικοποιητή. Κάθε μπλοκ κωδικοποιητή αποτελείται από ένα *διπλής κατεύθυνσης* στρώμα αυτο-προσοχής, ακολουθούμενο από δύο στρώματα τροφοδότησης προς τα εμπρός. Για λόγους απλότητας, δεν λαμβάνουμε υπόψη τα στρώματα κανονικοποίησης. Επίσης, δεν θα συζητήσουμε περαιτέρω το ρόλο των δύο στρωμάτων πρόωσης, αλλά απλά θα το δούμε ως μια τελική αντιστοίχιση διανύσματος σε διάνυσμα που απαιτείται σε κάθε μπλοκ κωδικοποιητή[1].

Το στρώμα αυτο-προσοχής διπλής κατεύθυνσης θέτει κάθε διάνυσμα εισόδου $\mathbf{x'}_j, \forall j \in \{1, \ldots, n\}$ σε σχέση με όλα τα διανύσματα εισόδου $\mathbf{x'}_1, \ldots, \mathbf{x'}_n$ και με τον τρόπο αυτό μετασχη-

[1] Οι Yun et. al. [8] υποστηρίζουν ότι τα στρώματα τροφοδότησης προς τα εμπρός είναι ζωτικής σημασίας για την αντιστοίχιση κάθε διανύσματος πλαισίου $\mathbf{x'}_i$ ξεχωριστά στον επιθυμητό χώρο εξόδου, κάτι που το στρώμα *αυτό-προσοχής* δεν καταφέρνει να κάνει από μόνο του. Θα πρέπει να σημειωθεί εδώ, ότι κάθε συμβολισμός εξόδου $\mathbf{x'}$ επεξεργάζεται από το ίδιο στρώμα προώθησης.

ματίζει το διάνυσμα εισόδου $\mathbf{x'}_j$ σε μια πιο "εκλεπτυσμένη" αναπαράσταση του εαυτού του, που ορίζεται ως $\mathbf{x''}_j$.

Με τον τρόπο αυτό, το πρώτο μπλοκ κωδικοποιητή μετασχηματίζει κάθε διάνυσμα εισόδου της ακολουθίας εισόδου $\mathbf{X}_{1:n}$ (που φαίνεται με ανοιχτό πράσινο χρώμα παρακάτω) από μια *ανεξάρτητη από το πλαίσιο* διανυσματική αναπαράσταση σε μια *εξαρτώμενη από το πλαίσιο* διανυσματική αναπαράσταση, και τα επόμενα μπλοκ κωδικοποιητή βελτιώνουν περαιτέρω αυτή τη διανυσματική αναπαράσταση μέχρι το τελευταίο μπλοκ κωδικοποιητή να εξάγει την τελική διανυσματική κωδικοποίηση $\overline{\mathbf{X}}_{1:n}$ (που φαίνεται με πιο σκούρο πράσινο χρώμα παρακάτω).

Ας ρίξουμε μια βαθύτερη ματιά στο πώς λειτουργεί η αυτο-προσοχή διπλής κατεύθυνσης. Κάθε διάνυσμα εισόδου $\mathbf{x'}_i$ μιας ακολουθίας εισόδου $\mathbf{X'}_{1:n}$ ενός μπλοκ κωδικοποιητή προβάλλεται σε ένα διάνυσμα κλειδί $\mathbf{k}_i$, ένα διάνυσμα τιμής $\mathbf{v}_i$ και ένα διάνυσμα ερώτησης $\mathbf{q}_i$ (που φαίνονται με πορτοκαλί, μπλε και μοβ χρώμα αντίστοιχα παρακάτω) μέσω τριών εκπαιδεύσιμων πινάκων βαρών $\mathbf{W}_q, \mathbf{W}_v, \mathbf{W}_k$:

$$\mathbf{q}_i = \mathbf{W}_q \mathbf{x'}_i,$$

$$\mathbf{v}_i = \mathbf{W}_v \mathbf{x'}_i,$$

$$\mathbf{k}_i = \mathbf{W}_k \mathbf{x'}_i,$$

$$\forall i \in \{1, \ldots n\}.$$

Σημειώνουμε εμφατικά ότι οι **ίδιοι** πίνακες βαρών εφαρμόζονται σε κάθε διάνυσμα εισόδου $\mathbf{x}_i, \forall i \; in \{i, \ldots, n\}$. Μετά την προβολή κάθε διανύσματος εισόδου $\mathbf{x}_i$ σε ένα διάνυσμα ερωτήματος, κλειδιού και τιμής, κάθε διάνυσμα ερωτήματος $\mathbf{q}_j, \forall j \in \{1, \ldots, n\}$ συγκρίνεται με όλα τα διανύσματα κλειδιού $\mathbf{k}_1, \ldots, \mathbf{k}_n$. Όσο πιο όμοιο είναι ένα από τα διανύσματα κλειδιά $\mathbf{k}_1, \ldots \mathbf{k}_n$ με ένα διάνυσμα ερώτησης $\mathbf{q}_j$, τόσο πιο σημαντικό είναι το αντίστοιχο διάνυσμα τιμών $\mathbf{v}_j$ για το διάνυσμα εξόδου $\mathbf{x''}_j$. Πιο συγκεκριμένα, ένα διάνυσμα εξόδου $\mathbf{x''}_j$ ορίζεται ως το σταθμισμένο άθροισμα όλων των διανυσμάτων τιμών $\mathbf{v}_1, \ldots, \mathbf{v}_n$ συν το διάνυσμα εισόδου $\mathbf{x'}_j$. Με τον τρόπο αυτό, τα βάρη είναι ανάλογα με την ομοιότητα συνημίτονου μεταξύ του $\mathbf{q}_j$ και των αντίστοιχων διανυσμάτων κλειδιών $\mathbf{k}_1, \ldots, \mathbf{k}_n$, η οποία εκφράζεται μαθηματικά από τη σχέση $\mathbf{Softmax}(\mathbf{K}_{1:n}^\intercal \cdot \mathbf{q}_j)$ όπως απεικονίζεται στην παρακάτω εξίσωση.

Για να κατανοήσουμε περαιτέρω τις συνέπειες του επιπέδου αυτο-προσοχής διπλής κατεύθυνσης, ας υποθέσουμε ότι επεξεργάζεται την ακόλουθη πρόταση: "The house is beautiful and well located in the middle of the city where it is easily accessible by public transport". Η λέξη "it" αναφέρεται στο "house", το οποίο βρίσκεται 12 "θέσεις μακριά". Στους κωδικοποιητές που βασίζονται σε μετασχηματιστές, το στρώμα αυτο-προσοχής διπλής κατεύθυνσης εκτελεί μια μόνο μαθηματική πράξη για να θέσει το διάνυσμα εισόδου του "house" σε σχέση με το διάνυσμα εισόδου του "it". Αντίθετα, σε έναν κωδικοποιητή που βασίζεται σε ΑΝΔ, μια λέξη που βρίσκεται 12 "θέσεις μακριά", θα απαιτούσε τουλάχιστον 12 μαθηματικές πράξεις, πράγμα που σημαίνει ότι σε έναν κωδικοποιητή που βασίζεται σε ΑΝΔ απαιτείται γραμμικός αριθμός μαθηματικών πράξεων. Αυτό καθιστά πολύ πιο δύσκολο για έναν κωδικοποιητή με βάση το ΑΝΔ να μοντελοποιήσει τις μακράς εμβέλειας αναπαραστάσεις πλαισίου.

Επίσης, καθίσταται σαφές ότι ένας κωδικοποιητής που βασίζεται σε μετασχηματιστή είναι πολύ λιγότερο επιρρεπής στο να χάσει σημαντικές πληροφορίες από ένα μοντέλο κωδικοποιητή-αποκωδικοποιητή που βασίζεται σε ΑΝΔ, επειδή το μήκος ακολουθίας της κωδικοποίησης διατηρείται το ίδιο, δηλ. $\textbf{len}(\mathbf{X}_{1:n}) = \textbf{len}(\overline{\mathbf{X}}_{1:n}) = n$, ενώ ένα ΑΝΔ συμπιέζει το μήκος από $\textbf{len}((\mathbf{X}_{1:n}) = n$ σε μόλις $\textbf{len}(\mathbf{c}) = 1$, γεγονός που καθιστά πολύ δύσκολο για τα ΑΝΔ να κωδικοποιήσουν αποτελεσματικά τις εξαρτήσεις μεγάλου εύρους μεταξύ των λέξεων εισόδου.

Εκτός από το ότι οι εξαρτήσεις μεγάλης εμβέλειας γίνονται πιο εύκολα μαθήσιμες, μπορούμε να δούμε ότι η αρχιτεκτονική του μετασχηματιστή είναι σε θέση να επεξεργάζεται κείμενο παράλληλα. μαθηματικά, αυτό μπορεί εύκολα να αποδειχθεί γράφοντας τον τύπο αυτο-προσοχής ως γινόμενο των πινάκων ερωτήματος, κλειδιού και τιμής:

$$\mathbf{X''}_{1:n} = \mathbf{V}_{1:n}\text{Softmax}(\mathbf{Q}^{\intercal}_{1:n}\mathbf{K}_{1:n}) + \mathbf{X'}_{1:n}$$

Η έξοδος $\mathbf{X''}_{1:n} = \{\mathbf{x''}_1, \ldots, \mathbf{x''}_n\}$ υπολογίζεται μέσω μιας σειράς πολλαπλασιασμών πινάκων και μιας πράξης softmax, η οποία μπορεί να παραλληλιστεί αποτελεσματικά. Σημειώστε, ότι σε ένα μοντέλο κωδικοποιητή που βασίζεται σε ΑΝΔ, ο υπολογισμός της κρυφής κατάστασης $\mathbf{c}$ πρέπει να γίνει διαδοχικά: Υπολογισμός της κρυφής κατάστασης του πρώτου διανύσματος εισόδου $\mathbf{x}_1$, στη συνέχεια υπολογισμός της κρυφής κατάστασης του δεύτερου διανύσματος εισόδου που εξαρτάται από την κρυφή κατάσταση του πρώτου κρυφού διανύσματος κ.λπ. Η διαδοχική φύση των ΑΝΔ εμποδίζει τον αποτελεσματικό παραλληλισμό και τα καθιστά πολύ πιο αναποτελεσματικά σε σύγκριση με τα μοντέλα κωδικοποιητών που βασίζονται σε μετασχηματιστές στο σύγχρονο υλικό GPU.

Τώρα θα πρέπει να υπάρχει καλύτερη κατανόηση του τρόπου με τον οποίο τα μοντέλα κωδικοποιητή που βασίζονται σε μετασχηματιστές μοντελοποιούν αποτελεσματικά τις μακράς εμβέλειας πλαισιακές αναπαραστάσεις και πώς επεξεργάζονται αποτελεσματικά εκτεταμένες ακολουθίες διανυσμάτων εισόδου.

1.4 Αποκωδικοποιητής

Όπως αναφέρθηκε στην ενότητα 1.2 (Κωδικοποιητής - αποκωδικοποιητής), ο αποκωδικοποιητής που βασίζεται σε μετασχηματιστή ορίζει την υπό συνθήκη κατανομή πιθανότητας μιας ακολουθίας στόχου δεδομένης της ακολουθίας κωδικοποίησης με βάση το πλαίσιο:

$$p_{theta_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n})$$

η οποία με τον κανόνα του Bayes μπορεί να αναλυθεί σε ένα γινόμενο των υπό συνθήκη κατανομών του επόμενου διανύσματος στόχου δεδομένης της ακολουθίας κωδικοποίησης με βάση το πλαίσιο και όλων των προηγούμενων διανυσμάτων στόχου:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{dec}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$$

Ας καταλάβουμε πρώτα πώς ο αποκωδικοποιητής που βασίζεται σε μετασχηματιστή ορίζει

μια κατανομή πιθανότητας. Ο αποκωδικοποιητής που βασίζεται σε μετασχηματιστή είναι μια στοίβα από μπλοκ αποκωδικοποίησης που ακολουθείται από ένα πυκνό στρώμα, την "κεφαλή LM". Η στοίβα των μπλοκ αποκωδικοποιητή αντιστοιχίζει την κωδικοποιημένη ακολουθία κωδικοποίησης $\overline{\mathbf{X}}_{1:n}$ και μια ακολουθία διανυσμάτων-στόχων που προτάσσεται από το διάνυσμα BOS και κόβεται στο τελευταίο διάνυσμα-στόχο, δηλαδή $\mathbf{Y}_{0:i-1}$, σε μια κωδικοποιημένη ακολουθία διανυσμάτων-στόχων $\overline{\mathbf{Y}}_{0:i-1}$. Στη συνέχεια, η "κεφαλή LM" αντιστοιχίζει την κωδικοποιημένη ακολουθία διανυσμάτων στόχου $\overline{\mathbf{Y}}_{0:i-1}$ σε μια ακολουθία διανυσμάτων logit $\mathbf{L}_{1:n} = \mathbf{l}_1, \ldots, \mathbf{l}_n$, ενώ η διαστατικότητα κάθε διανύσματος logit $\mathbf{l}_i$ αντιστοιχεί στο μέγεθος του λεξιλογίου. Με αυτόν τον τρόπο, για κάθε $i \in \{1, \ldots, n\}$ μπορεί να προκύψει μια κατανομή πιθανότητας σε ολόκληρο το λεξιλόγιο εφαρμόζοντας μια πράξη softmax στο $\mathbf{l}_i$. Αυτές οι κατανομές ορίζουν την υπό συνθήκη κατανομή:

$$p_{\partial_{dec}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n}), \forall i \in \{1, \ldots, n\},$$

αντίστοιχα. Η "κεφαλή LM" συχνά συνδέεται με την αντιμετάθεση του πίνακα ενσωμάτωσης λέξεων, δηλαδή $\mathbf{W}_{\text{emb}}^{\intercal} = \left[\mathbf{y}^1, \ldots, \mathbf{y}^{\text{vocab}}\right]^{\intercal}$. Διαισθητικά αυτό σημαίνει ότι για όλα τα $i \in \{0, \ldots, n-1\}$ το στρώμα "LM Head" συγκρίνει το κωδικοποιημένο διάνυσμα εξόδου $\overline{\mathbf{y}}_i$ με όλες τις ενσωματώσεις λέξεων στο λεξιλόγιο $\mathbf{y}^1, \ldots, \mathbf{y}^{\text{vocab}}$ έτσι ώστε το διάνυσμα logit $\mathbf{l}_{i+1}$ να αντιπροσωπεύει τα σκορ ομοιότητας μεταξύ του κωδικοποιημένου διανύσματος εξόδου και κάθε ενσωμάτωσης λέξης. Η λειτουργία softmax απλώς μετατρέπει τις βαθμολογίες ομοιότητας σε μια κατανομή πιθανότητας. Για κάθε $i \in \{1, \ldots, n\}$, ισχύουν οι ακόλουθες εξισώσεις:

$$\begin{aligned} & p_{\partial_{dec}}(\mathbf{y}|\overline{\mathbf{X}}_{1:n}, \mathbf{Y}_{0:i-1}) \\ &= \text{Softmax}(f_{\partial_{\text{dec}}}(\overline{\mathbf{X}}_{1:n}, \mathbf{Y}_{0:i-1})) \\ &= \text{Softmax}(\mathbf{W}_{\text{emb}}^{\intercal}\overline{\mathbf{y}}_{i-1}) \\ &= \text{Softmax}(\mathbf{l}_i). \end{aligned}$$

Αν τα βάλουμε όλα μαζί, προκειμένου να μοντελοποιήσουμε την υπό συνθήκη κατανομή μιας ακολουθίας διανυσμάτων στόχων $\mathbf{Y}_{1:m}$, τα διανύσματα στόχων $\mathbf{Y}_{1:m-1}$ που προστίθενται από το ειδικό διάνυσμα BOS, δηλ. $\mathbf{y}_0$, αντιστοιχίζονται πρώτα μαζί με την ακολουθία κωδικοποίησης με συμφραζόμενα $\overline{\mathbf{X}}_{1:n}$ στην ακολουθία διανυσμάτων logit $\mathbf{L}_{1:m}$. Κατά συνέπεια, κάθε διάνυσμα-στόχος logit $\mathbf{l}_i$ μετατρέπεται σε μια υπό συνθήκη κατανομή πιθανότητας του διανύσματος-στόχου $\mathbf{y}_i$ χρησιμοποιώντας την πράξη softmax. Τέλος, οι υπό συνθήκη πιθανότητες όλων των διανυσμάτων στόχων $\mathbf{y}_1, \ldots, \mathbf{y}_m$ πολλαπλασιάζονται μεταξύ τους για να προκύψει η υπό συνθήκη πιθανότητα της πλήρους ακολουθίας διανυσμάτων στόχων:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{dec}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n}).$$

Σε αντίθεση με τους κωδικοποιητές που βασίζονται σε μετασχηματιστές, στους αποκωδικοποιητές που βασίζονται σε μετασχηματιστές, το κωδικοποιημένο διάνυσμα εξόδου $\overline{\mathbf{y}}_i$ θα πρέπει να είναι μια καλή αναπαράσταση του επόμενου διανύσματος-στόχου $\mathbf{y}_{i+1}$ και όχι του ίδιου του διανύσματος εισόδου. Επιπλέον, το κωδικοποιημένο διάνυσμα εξόδου $\overline{\mathbf{y}}_i$ θα

πρέπει να εξαρτάται από όλες τις πλαισιωμένες ακολουθίες κωδικοποίησης $\overline{\mathbf{X}}_{1:n}$. Για την ικανοποίηση αυτών των απαιτήσεων, κάθε μπλοκ αποκωδικοποιητή αποτελείται από ένα στρώμα αυτο-προσοχής *μονής-κατεύθυνσης*, ακολουθούμενο από ένα στρώμα *διασταυρούμενης προσοχής* και δύο στρώματα τροφοδότησης προς τα εμπρός. Το *μονής-κατεύθυνσης* στρώμα αυτο-προσοχής θέτει κάθε ένα από τα διανύσματα εισόδου του $\mathbf{y'}_j$ μόνο σε σχέση με όλα τα προηγούμενα διανύσματα εισόδου $\mathbf{y'}_i$, με $i \leq q$ για όλα τα $j \in \{1, \ldots, n\}$ για να μοντελοποιήσει την κατανομή πιθανότητας των επόμενων διανυσμάτων στόχου. Το στρώμα *διασταυρούμενης προσοχής* θέτει κάθε ένα από τα διανύσματα εισόδου του $\mathbf{y''}_j$ σε σχέση με όλα τα διανύσματα κωδικοποίησης με πλαίσιο $\overline{\mathbf{X}}_{1:n}$ για να εξαρτήσει την κατανομή πιθανότητας των επόμενων διανυσμάτων στόχου και από την είσοδο του κωδικοποιητή.

Ας επικεντρωθούμε τώρα στην αυτο-προσοχή *μονής κατεύθυνσης*.

Όπως και στην αυτο-προσοχή διπλής κατεύθυνσης, στην αυτο-προσοχή *μονής κατεύθυνσης*, τα διανύσματα ερωτήσεων $\mathbf{q}_0, \ldots, \mathbf{q}_{m-1}$ (φαίνονται με μωβ χρώμα παρακάτω), τα διανύσματα κλειδιών $\mathbf{k}_0, \ldots, \mathbf{k}_{m-1}$ (φαίνονται με πορτοκαλί χρώμα παρακάτω), και τα διανύσματα τιμών $\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}$ (εμφανίζονται με μπλε χρώμα παρακάτω) προβάλλονται από τα αντίστοιχα διανύσματα εισόδου $\mathbf{y'}_0, \ldots, \mathbf{y}_{m-1}$ (εμφανίζονται με ανοιχτό κόκκινο χρώμα παρακάτω). Ωστόσο, στην *uni-directional* αυτο-προσοχή, κάθε διάνυσμα ερωτήματος $\mathbf{q}_i$ συγκρίνεται μόνο με το αντίστοιχο διάνυσμα κλειδί και όλα τα προηγούμενα, δηλαδή $\mathbf{k}_0, \ldots, \mathbf{k}_i$ για να προκύψουν τα αντίστοιχα βάρη προσοχής. Αυτό εμποδίζει ένα διάνυσμα εξόδου $\mathbf{y''}_j$ (που εμφανίζεται με σκούρο κόκκινο χρώμα παρακάτω) να περιλαμβάνει οποιαδήποτε πληροφορία για το ακόλουθο διάνυσμα εισόδου $\mathbf{y}_i$, με $i > 1$ για όλα τα j $in\{0, \ldots, m-1\}$. Όπως συμβαίνει στην αυτο-προσοχή διπλής κατεύθυνσης, τα βάρη προσοχής πολλαπλασιάζονται στη συνέχεια με τα αντίστοιχα διανύσματα τιμών τους και αθροίζονται.

Μπορούμε να συνοψίσουμε την *αυτο-προσοχή μονής κατεύθυνσης* ως εξής:

$$\mathbf{y''}_i = \mathbf{V}_{0:i} \cdot \mathbf{Softmax}(\mathbf{K}_{0:i}^{\intercal} \mathbf{q}_i) + \mathbf{y'}_i.$$

Σημειώστε ότι το εύρος των δεικτών των διανυσμάτων κλειδιού και τιμής είναι $0 : i$ αντί για $0 : m - 1$ που θα ήταν το εύρος των διανυσμάτων κλειδιού στην αυτο-προσοχή διπλής-κατεύθυνσης.

Γιατί λοιπόν είναι σημαντικό να χρησιμοποιούμε *αυτο-προσοχή μονής κατεύθυνσης* στον αποκωδικοποιητή αντί για αυτο-προσοχή διπλής-κατεύθυνσης; Όπως αναφέρθηκε παραπάνω, ένας αποκωδικοποιητής βασισμένος σε μετασχηματιστές ορίζει μια αντιστοίχιση από μια ακολουθία διανυσμάτων εισόδου $\mathbf{Y}_{0:m-1}$ στα logit που αντιστοιχούν στα επόμενα διανύσματα εισόδου του αποκωδικοποιητή, δηλαδή $\mathbf{L}_{1:m}$.

Αυτό είναι προφανώς μειονεκτικό, καθώς ο αποκωδικοποιητής που βασίζεται στον μετασχηματιστή δεν θα μάθει ποτέ να προβλέπει την επόμενη λέξη με βάση όλες τις προηγούμενες λέξεις, αλλά απλώς θα αντιγράψει το διάνυσμα-στόχο $\mathbf{y}_i$ μέσω του δικτύου σε $\overline{\mathbf{y}}_{i-1}$ για όλα τα $i \in \{1, \ldots, m\}$. Προκειμένου να οριστεί μια υπό συνθήκη κατανομή του επόμενου διανύσματος στόχου, η κατανομή δεν μπορεί να εξαρτάται από το ίδιο το επόμενο διάνυσμα στόχου. Δεν έχει νόημα να προβλέψουμε το $\mathbf{y}_i$ από το $p(\mathbf{y}|\mathbf{Y}_{0:i}, \overline{\mathbf{X}})$ επειδή η κατανομή εξαρτάται από το διάνυσμα-στόχο που υποτίθεται ότι μοντελοποιεί. Επομένως, η αρχιτεκτονική αυτο-προσοχής *μονής κατεύθυνσης* μας επιτρέπει να ορίσουμε μια αιτιώδη κατανομή πι-

θανότητας, η οποία είναι απαραίτητη για να μοντελοποιήσουμε αποτελεσματικά μια υπό συνθήκη κατανομή του επόμενου διανύσματος στόχου.

Τώρα μπορούμε να περάσουμε στο επίπεδο που συνδέει τον κωδικοποιητή και τον αποκωδικοποιητή - τον μηχανισμό *διασταυρούμενης προσοχής*.

Το στρώμα *διασταυρωμένης προσοχής* δέχεται δύο διανυσματικές ακολουθίες ως εισόδους: τις εξόδους του στρώματος αυτο-προσοχής *μονής κατεύθυνσης*, δηλαδή $\mathbf{Y''}_{0:m-1}$ και τα διανύσματα κωδικοποίησης με βάση το περιεχόμενο $\overline{\mathbf{X}}_{1:n}$. Όπως και στο στρώμα αυτο-προσοχής, τα διανύσματα ερωτήματος $\mathbf{q}_0, \ldots, \mathbf{q}_{m-1}$ είναι προβολές των διανυσμάτων εξόδου του προηγούμενου στρώματος, δηλαδή $\mathbf{Y''}_{0:m-1}$. Ωστόσο, τα διανύσματα κλειδιού και τιμής $\mathbf{k}_0, \ldots, \mathbf{k}_{m-1}$ και $\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}$ είναι προβολές των διανυσμάτων κωδικοποίησης με πλαίσιο $\overline{\mathbf{X}}_{1:n}$. Έχοντας ορίσει τα διανύσματα κλειδί, τιμή και ερώτημα, ένα διάνυσμα ερώτησης $\mathbf{q}_i$ συγκρίνεται στη συνέχεια με όλα τα διανύσματα κλειδί και η αντίστοιχη βαθμολογία χρησιμοποιείται για τη στάθμιση των αντίστοιχων διανυσμάτων τιμής, όπως ακριβώς συμβαίνει στην περίπτωση της αυτο-προσοχής διπλής κατεύθυνσης, ώστε να προκύψει το διάνυσμα εξόδου $\mathbf{y'''}_i$ για όλα τα $i \in \{0, \ldots, m-1\}$. Το *Cross-attention* μπορεί να συνοψιστεί ως εξής:

$$\mathbf{y'''}_i = \mathbf{V}_{1:n}\mathbf{Softmax}(\mathbf{K}_{1:n}^{\intercal}\mathbf{q}_i) + \mathbf{y''}_i$$

Σημειώστε ότι το εύρος των δεικτών των διανυσμάτων κλειδιού και τιμής είναι $1:n$ που αντιστοιχεί στον αριθμό των διανυσμάτων κωδικοποίησης με πλαίσιο.

Αυτό που συμβαίνει εδώ είναι ότι κάθε διάνυσμα εξόδου $\mathbf{y'}_i$ είναι ένα σταθμισμένο άθροισμα όλων των προβολών των τιμών των εισόδων του κωδικοποιητή $\mathbf{v}_1, \ldots, \mathbf{v}_7$ συν το ίδιο το διάνυσμα εισόδου $\mathbf{y}_i$ (βλ. τον παραπάνω τύπο). Ο βασικός μηχανισμός που πρέπει να κατανοήσουμε είναι ο ακόλουθος: Ανάλογα με το πόσο παρόμοια είναι μια προβολή ερωτήματος του διανύσματος εισόδου του αποκωδικοποιητή $\mathbf{q}_i$ με μια βασική προβολή του διανύσματος εισόδου του κωδικοποιητή $\mathbf{k}_j$, τόσο πιο σημαντική είναι η προβολή τιμής του διανύσματος εισόδου του κωδικοποιητή $\mathbf{v}_j$. Σε αδρές γραμμές αυτό σημαίνει ότι, όσο πιο "συγγενής" είναι μια αναπαράσταση εισόδου αποκωδικοποιητή με μια αναπαράσταση εισόδου κωδικοποιητή, τόσο περισσότερο επηρεάζει η αναπαράσταση εισόδου την αναπαράσταση εξόδου αποκωδικοποιητή.

Τώρα μπορούμε να δούμε πώς αυτή η αρχιτεκτονική εξαρτά όμορφα κάθε διάνυσμα εξόδου $\mathbf{y'''}_i$ από την αλληλεπίδραση μεταξύ των διανυσμάτων εισόδου του κωδικοποιητή $\overline{\mathbf{X}}_{1:n}$ και του διανύσματος εισόδου $\mathbf{y''}_i$. Μια άλλη σημαντική παρατήρηση σε αυτό το σημείο είναι ότι η αρχιτεκτονική είναι εντελώς ανεξάρτητη από τον αριθμό n των πλαισιωμένων διανυσμάτων κωδικοποίησης $\overline{\mathbf{X}}_{1:n}$ στα οποία εξαρτάται το διάνυσμα εξόδου $\mathbf{y'''}_i$. Όλοι οι πίνακες προβολής $\mathbf{W}_k^{\text{cross}}$ και $\mathbf{W}_v^{\text{cross}}$ για την εξαγωγή των διανυσμάτων κλειδιών $\mathbf{k}_1, \ldots, \mathbf{k}_n$ και των διανυσμάτων τιμών $\mathbf{v}_1, \ldots, \mathbf{v}_n$ αντίστοιχα μοιράζονται σε όλες τις θέσεις $1, \ldots, n$ και όλα τα διανύσματα τιμών $\mathbf{v}_1, \ldots, \mathbf{v}_n$ αθροίζονται σε ένα ενιαίο σταθμισμένο μέσο διάνυσμα. Τώρα γίνεται επίσης φανερό, γιατί ο αποκωδικοποιητής που βασίζεται σε μετασχηματιστή δεν υποφέρει από το πρόβλημα της εξάρτησης μεγάλης εμβέλειας, από το οποίο υποφέρει ο αποκωδικοποιητής που βασίζεται σε ΑΝΔ. Επειδή κάθε διάνυσμα λογαρίθμου αποκωδικοποιητή εξαρτάται άμεσα από κάθε μεμονωμένο κωδικοποιημένο διάνυσμα εξόδου, ο αριθμός των μαθηματικών πράξεων για τη σύγκριση του πρώτου κωδικοποιημένου διανύσματος εξόδου και

του τελευταίου διανύσματος λογαρίθμου αποκωδικοποιητή ανέρχεται ουσιαστικά σε μόλις μία.

Συμπερασματικά, το στρώμα αυτο-προσοχής *μονής κατεύθυνσης* είναι υπεύθυνο για την προσαρμογή κάθε διανύσματος εξόδου σε όλα τα προηγούμενα διανύσματα εισόδου του αποκωδικοποιητή και το τρέχον διάνυσμα εισόδου και το στρώμα *διασταυρούμενης προσοχής* είναι υπεύθυνο για την περαιτέρω προσαρμογή κάθε διανύσματος εξόδου σε όλα τα κωδικοποιημένα διανύσματα εισόδου.

Chapter 2

Ανταγωνιστικές Επιθέσεις

2.1 Εισαγωγή

Ο όρος "ανταγωνιστική επίθεση" σε ένα μοντέλο βαθιάς μάθησης αναφέρεται στη συστηματική διαδικασία παραγωγής *ανταγωνιστικών παραδειγμάτων*, δηλαδή προσεκτικά διαμορφωμένων εισόδων που αποσκοπούν στην εξαπάτηση του μοντέλου-στόχου. Συγκεκριμένα, τα ανταγωνιστικά παραδείγματα έχουν το βασικό χαρακτηριστικό ότι δημιουργούνται με την προσθήκη **ανεπαίσθητης διαταραχής (θορύβου)** στην αρχική είσοδο [9] και για το λόγο αυτό οι όροι "ανταγωνιστικά παραδείγματα" και *"ανταγωνιστικές διαταραχές"* χρησιμοποιούνται ως συνώνυμοι στην παρούσα εργασία. Αυτές οι ελαφρώς διαταραγμένες είσοδοι μπορεί να φαίνονται αθώες, αλλά συχνά οδηγούν το μοντέλο σε λανθασμένες προβλέψεις με υψηλό βαθμό εμπιστοσύνης [1]. Γίνεται σαφές ότι οι ανταγωνιστικές επιθέσεις είναι κακόβουλες επιθέσεις που μπορούν να μειώσουν αποτελεσματικά την απόδοση και να εκθέσουν κρίσιμες αδυναμίες διάφορων μοντέλων μηχανικής μάθησης που χρησιμοποιούνται ευρέως σε ποικίλους τομείς [10, 11, 12, 13, 14, 15, 16, 17]. Με άλλα λόγια, οι ανταγωνιστικές επιθέσεις αποτελούν σημαντική απειλή για κρίσιμες πτυχές των συστημάτων μηχανικής μάθησης, όπως η ασφάλεια, η διαφάνεια και η αξιοπιστία, και ως εκ τούτου η ανάγκη θέσπισης αντιμέτρων κατά των ανταγωνιστικά διαμορφωμένων εισόδων είναι μεγαλύτερη από ποτέ.

2.2 Παραδείγματα Ανταγωνιστικών Επιθέσεων

Πριν προχωρήσουμε σε περισσότερες λεπτομέρειες σχετικά με τις ανταγωνιστικές επιθέσεις, θα παρουσιάσουμε μερικά παραδείγματα επιθέσεων στους δύο πιο γνωστούς τομείς μηχανικής μάθησης: Όραση Υπολογιστών, όπου τα δεδομένα είναι εικόνες, και Επεξεργασία Φυσικής Γλώσσας (ΕΦΓ), όπου τα δεδομένα είναι κείμενα σε φυσική γλώσσα. Τα παραδείγματα αυτά έχουν ως στόχο να δώσουν μια βασική διαίσθηση των ανταγωνιστικών επιθέσεων και να καταδείξουν το γεγονός ότι η προστιθέμενη διαταραχή είναι πρακτικά ανεπαίσθητη από τον άνθρωπο, ωστόσο μπορεί να οδηγήσει το μοντέλο στη δημιουργία λανθασμένων προβλέψεων με υψηλό σκορ εμπιστοσύνης.

2.2.1 Όραση Υπολογιστών

Η εικόνα 2.1 παρακάτω απεικονίζει μια περίπτωση ανταγωνιστικής επίθεσης σε έναν ταξινομητή εικόνων. Συγκεκριμένα, το μοντέλο-στόχος είναι το GoogLeNet το οποίο παράγει κορυφαία αποτελέσματα στον διαγωνισμό ταξινόμησης ImageNet [18, 19]. Παρατηρούμε ότι η υπέρθεση μιας μικρής (αλλά σκόπιμης) ποσότητας θορύβου προκαλεί το μοντέλο να ταξινομήσει λανθασμένα το πάντα ως γίββωνα με εξαιρετικά υψηλό σκορ εμπιστοσύνης (99.3%). Μπορούμε επίσης να επαληθεύσουμε ότι η διαταραγμένη εικόνα είναι πρακτικά πανομοιότυπη με την αρχική.

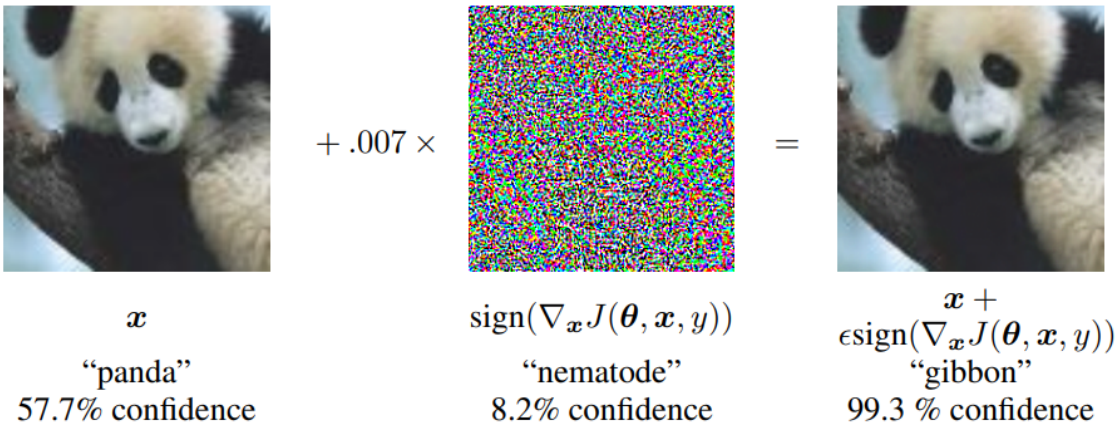

Figure 2.1. *Το πάντα ταξινομείται λανθασμένα ως γίββων, μετά την προσθήκη πρακτικά απαρατήρητης διαταραχής στην αρχική εικόνα (Image credit: Goodfellow et al. [1])*

2.2.2 Επεξεργασία Φυσικής Γλώσσας

Δεν υπάρχει φυσική αναλογία με τα ανταγωνιστικά παραδείγματα στην Όραση Υπολογιστών για τον τομέα της ΕΦΓ. Για την ακρίβεια, το παραπάνω παράδειγμα καταδεικνύει το γεγονός ότι η αρχική και η διαταραγμένη εικόνα είναι απολύτως ταυτόσημες στο ανθρώπινο μάτι, παρολαυτά ερμηνεύονται ως εντελώς διαφορετικές από το σύστημα μηχανικής μάθησης. Ωστόσο, πώς μπορούν δύο ακολουθίες κειμένου να είναι πανομοιότυπες για τον άνθρωπο χωρίς να είναι ίδιες;

Έχει ήδη αναφερθεί ότι το βασικό χαρακτηριστικό ενός έγκυρου ανταγωνιστικού παραδείγματος είναι η υψηλή *ομοιότητα* του με την αρχική είσοδο. Στον τομέα των εικόνων, είδαμε ότι είναι δυνατό να δημιουργηθούν ανταγωνιστικές εικόνες που μοιάζουν ακριβώς ίδιες με τις πρωτότυπες, οπότε δεν υπήρχε ανάγκη να εμβαθύνουμε στην έννοια της ομοιότητας, επειδή η ομοιότητα ήταν προφανώς οπτική. Όταν πρόκειται για τον τομέα του κειμένου, πρέπει να διακρίνουμε την έννοια της ομοιότητας σε δύο κατηγορίες:

- **Οπτική ομοιότητα** Η διαταραγμένη ακολουθία κειμένου *μοιάζει πολύ* με την αρχική είσοδο. Συνήθως, οι ανταγωνιστικές επιθέσεις που διατηρούν την οπτική ομοιότητα προσπαθούν να αλλάξουν όσο το δυνατόν λιγότερους χαρακτήρες ή να εισάγουν ανθρώπινα τυπογραφικά λάθη στην αρχική ακολουθία (βλ. πίνακα 7.1 παρακάτω).

- **Σημασιολογική ομοιότητα** Η διαταραγμένη ακολουθία κειμένου είναι *σημασιολογικά*

ταυτόσημη με την αρχική είσοδο. Συνήθως, οι ανταγωνιστικές επιθέσεις που διατηρούν τη σημασιολογική ομοιότητα προσπαθούν να παραφράσουν την αρχική είσοδο ή να αντικαταστήσουν "ευάλωτες" λέξεις με τις συνωνύμές τους (βλ. πίνακα 2.1 παρακάτω).

Παραδείγματα Ανταγωνιστικών Επιθέσεων στην ΕΦΓ	
WordBug (οπτική ομοιότητα)	
Original input (label: POS)	This film has a special place in my heart
Adversarial input (label: NEG)	This film has a special plcae in my herat
TextFooler (σημασιολογική ομοιότητα)	
Original input (label: NEG)	The characters are totally estranged from reality
Adversarial input (label: POS)	The characters are fully estranged from reality

Table 2.1. *Ο αλγόριθμος ανταγωνιστικών επιθέσεων WordBug (που στοχεύει στη διατήρηση της οπτικής ομοιότητας) καταφέρνει να αντιστρέψει την ετικέτα εξόδου του ταξινομητή εισάγοντας μόνο 2 τυπογραφικά λάθη στην αρχική ακολουθία [2]. Ομοίως, ο αλγόριθμος TextFooler (που στοχεύει στη διατήρηση της σημασιολογικής ομοιότητας) καταφέρνει να ξεγελάσει τον ταξινομητή αντικαθιστώντας τη λέξη "totally" με την ακριβώς συνώνυμη λέξη "fully" [3]. (POS: θετική κριτική, NEG: αρνητική κριτική)*

Στην παρούσα εργασία, εστιάζουμε στις ανταγωνιστικές επιθέσεις στον χώρο της ΕΦΓ και συγκεκριμένα σε αυτές που διατηρούν το νόημα της αρχικής ακολουθίας και παράγουν γραμματικά και συντακτικά έγκυρο κείμενο. Αυτός ο τύπος επιθέσεων είναι κατάλληλος για το πρόβλημα Συμπερασμού Φυσικής Γλώσσας (ΣΦΓ), επειδή μας ενδιαφέρει να αναγνωρίζουμε τη σημασιολογική συσχέτιση μόνο σε προτάσεις με νόημα και συνοχή.

2.3 Βασικές Έννοιες και Ορισμοί

Αυτή η ενότητα παρέχει τις θεμελιώδεις προκαταρκτικές γνώσεις σχετικά με τις ανταγωνιστικές επιθέσεις, συμπεριλαμβανομένων των περιγραφών των τύπων, της βασικής ταξινόμησης των ανταγωνιστικών επιθέσεων, καθώς και των μετρικών που ποσοτικοποιούν την ανθεκτικότητα ενός μοντέλου έναντι των ανταγωνιστικών επιθέσεων.

2.3.1 Μαθηματική Μοντελοποίηση

Η ενότητα 2.2 παρείχε μια βασική διαισθητική κατανόηση μιας ανταγωνιστικής επίθεσης, επομένως μπορούμε τώρα να εισαγάγουμε τον κατάλληλο συμβολισμό που περιγράφει επίσημα την έννοια των ανταγωνιστικών επιθέσεων.

Μια ανταγωνιστική επίθεση μπορεί να οριστεί πλήρως με τον καθορισμό των θεμελιωδών συστατικών της. Όπως αναφέρθηκε παραπάνω, μια ανταγωνιστική επίθεση σε ένα *μοντέλο βαθιάς μάθησης* αναφέρεται στη συστηματική διαδικασία δημιουργίας *ανταγωνιστικών παραδειγμάτων*, πράγμα που σημαίνει ότι το *μοντέλο βαθιάς μάθησης* και τα *ανταγωνιστικά παραδείγματα* μπορούν να αναγνωριστούν ως τα αναπόσπαστα συστατικά μιας ανταγωνιστική επίθεσης. Επιπλέον, η παρούσα εργασία εισάγει αντίμετρα και μηχανισμούς άμυνας

κατά των ανταγωνιστικών επιθέσεων σε μοντέλα μηχανικής μάθησης, επομένως πρέπει να ορίσουμε την έννοια της *ανθεκτικότητας*. Οι ορισμοί των μοντέλων βαθιάς μάθησης[1], των ανταγωνιστικών παραδειγμάτων και της ανθεκτικότητας δίνονται παρακάτω [9]:

Μοντέλο Βαθιάς Μάθησης: Ένα τυπικό μοντέλο βαθιάς μάθησης αναπαρίσταται ως μια συνάρτηση $F : X \to Y$, η οποία αντιστοιχίζει ένα σύνολο δεδομένων εισόδου X σε ένα σύνολο ετικετών εξόδου Y. Το Y είναι ένα σύνολο που περιλαμβάνει k κλάσεις, συνήθως $Y = \{0, 1, ..., k-1\}$. Ένα δείγμα $x \in X$ ταξινομείται σωστά από την F στην πραγματική ετικέτα y αν και μόνο αν $F(x) = y$.

Ανταγωνιστικά Παραδείγματα: Ο σκοπός ενός αλγόριθμου παραγωγής ανταγωνιστικών επιθέσεων είναι να εισάγει απειροελάχιστη διαταραχή (θόρυβο) ϵ στο δείγμα x προς δημιουργία της διαταραγμένης εισόδου $x' = x + \epsilon$, έτσι ώστε $F(x') = y' \neq y$, όπου y η πραγματική ετικέτα του x (το οποίο αρχικά ταξινομούνταν σωστά από το μοντέλο βαθιάς μάθησης). Πέρα από την απαίτηση το δείγμα x να μην ταξινομείται σωστά από το μοντέλο βαθιάς μάθησης, ο θόρυβος που υπερτίθεται στο x πρέπει να είναι ανεπαίσθητος για τον άνθρωπο. Για αυτό το σκοπό, η διαταραχή ϵ δεν πρέπει να ξεπερνά ένα προκαθορισμένο κατώφλι δ, ήτοι $\|\epsilon\| < \delta$. Επιπλέον, χρησιμοποιούνται διάφορες μετρικές που ποσοτικοποιούν την ένταση του υπερτιθέμενου θορύβου (βλ. 2.3.3).

Ανθεκτικότητα: Ένα ανθεκτικό μοντέλο βαθιάς μάθησης αντιστέκεται σε μια ανταγωνιστική επίθεση, ταξινομώντας σωστά τα ανταγωνιστικά παραδείγματα x' που δημιουργούνται από την επίθεση αυτή. Ως εκ τούτου, σε ένα ανθεκτικό μοντέλο βαθιάς μάθησης, η προβλεπόμενη ετικέτα που αντιστοιχεί στο x' θα πρέπει να είναι y και όχι y', δηλαδή $F(x') = y$, όπου y είναι η ετικέτα του x που είχε αρχικά προβλεφθεί από το μοντέλο. Οι αμυντικές μέθοδοι για την ενίσχυση της ανθεκτικότητας των μοντέλων θα πρέπει να αντιμετωπίζουν αποτελεσματικά ένα ευρύ φάσμα διαταραχών ϵ.

2.3.2 Ταξινόμηση Ανταγωνιστικών Επιθέσεων

Υπάρχουν τρία βασικά κριτήρια βάσει των οποίων μπορούν να κατηγοριοποιηθούν οι επιθέσεις. Αυτά τα κριτήρια παρατίθενται παρακάτω:

- **Στόχος του ανταγωνιστή[2]:** Ποιος είναι ο στόχος ή ο σκοπός του ανταγωνιστή; Θέλει να αλλοιώσει τη διαδικασία απόφασης του μοντέλου για ένα δείγμα ή να επηρεάσει τη συνολική απόδοση του μοντέλου; [10]

- **Γνώση του ανταγωνιστή:** Ποιες πληροφορίες έχει στη διάθεσή του ο επιτιθέμενος; Γνωρίζουν τη δομή του μοντέλου, τις παραμέτρους του ή το σύνολο εκπαίδευσης που χρησιμοποιήθηκε για την εκπαίδευσή του; [10]

- **Μονάδα διαταραχής:** Ποια είναι η ελάχιστη μονάδα στο δείγμα εισόδου (χαρακτήρας, λέξη, πρόταση) που διαταράσσεται; (*αφορά μόνο του τομέα του κειμένου*) [9]

[1] Στην εργασία μας, ασχολούμαστε με τις ανταγωνιστικές επιθέσεις σε μοντέλα ΣΦΓ. Ωστόσο, ο ΣΦΓ είναι μια υπο-κατηγορία του προβλήματος ταξινόμησης στη μηχανική μάθηση, επομένως ο επίσημος ορισμός που δίνεται για τα μοντέλα βαθιάς μάθησης πρακτικά αφορά τους ταξινομητές βαθιάς μάθησης.

[2] Ο όρος "ανταγωνιστής" αναφέρεται στο σύστημα που διεξάγει τις ανταγωνιστικές επιθέσεις.

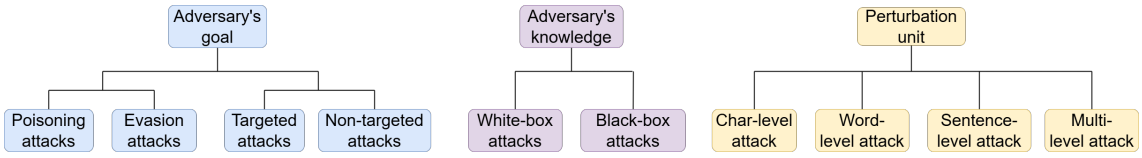

Figure 2.2. *Classification of adversarial attacks*

2.3.2.1 Στόχος του ανταγωνιστή

- **Επιθέσεις δηλητηρίασης έναντι επιθέσεων αποφυγής**

  Οι **επιθέσεις δηλητηρίασης** αναφέρονται στους αλγορίθμους επίθεσης που επιτρέπουν σε έναν ανταγωνιστή να εισάγει ψεύτικα δείγματα στη βάση δεδομένων εκπαίδευσης ενός αλγορίθμου μοντέλου βαθιάς μάθησης ή να τροποποιεί αυθεντικά δείγματα εκπαίδευσης. Αυτά τα ψεύτικα (*δηλητηριασμένα*) δείγματα μπορεί να έχουν σημαντικό αντίκτυπο στην απόδοση του εκπαιδευμένου ταξινομητή, π.χ. μπορεί να έχουν ως αποτέλεσμα τη χαμηλή ακρίβεια σε δείγματα ελέγχου. Αυτός ο τύπος επιθέσεων εμφανίζεται συχνά στην περίπτωση που ο ανταγωνιστής έχει πρόσβαση στη βάση δεδομένων εκπαίδευσης. Για παράδειγμα, οι διαδικτυακές αποθήκες συχνά συλλέγουν παραδείγματα κακόβουλου λογισμικού για εκπαίδευση, γεγονός που παρέχει την ευκαιρία στους ανταγωνιστές να δηλητηριάσουν τα δεδομένα [10].

  Στις **επιθέσεις αποφυγής**, οι ταξινομητές είναι σταθεροί και συνήθως έχουν καλή απόδοση σε καλοήθη δείγματα ελέγχου. Οι ανταγωνιστές δεν έχουν την εξουσία να αλλάξουν τον ταξινομητή ή τις παραμέτρους του, αλλά δημιουργούν κάποια ψεύτικα δείγματα που ο ταξινομητής δεν μπορεί να αναγνωρίσει. Με άλλα λόγια, οι ανταγωνιστές δημιουργούν κάποια δόλια παραδείγματα για να αποφύγουν την ορθή αναγνώριση από τον ταξινομητή [10]. Για παράδειγμα, στα οχήματα αυτόνομης οδήγησης, το κόλλημα μερικών κομματιών ταινίας στις πινακίδες στοπ μπορεί να μπερδέψει τον ταξινομητή οδικών σημάτων [20] του οχήματος.

- **Στοχευμένες επιθέσεις έναντι μη στοχευμένων επιθέσεων**

  Σε μια **στοχευμένη επίθεση**, όταν δίνεται το δείγμα-θύμα (x, y), όπου x είναι διάνυσμα χαρακτηριστικών και $y \in Y$ είναι η πραγματική ετικέτα του x, ο ανταγωνιστής στοχεύει στον "εξαναγκασμό" του ταξινομητή να παράξει μια συγκεκριμένη ετικέτα $t \in Y$ στο διαταραγμένο δείγμα x'. Για παράδειγμα, ένας απατεώνας είναι πιθανό να επιτεθεί στο μοντέλο αξιολόγησης της πιστοληπτικής ικανότητας μιας χρηματοπιστωτικής εταιρείας για να μεταμφιεστεί σε έναν εξαιρετικά αξιόπιστο πελάτη της εν λόγω εταιρείας [10].

  Σε μια **μη στοχευμένη επίθεση**, δεν υπάρχει καθορισμένη ετικέτα-στόχος t για το δείγμα-θύμα x και ο αντίπαλος στοχεύει απλώς στην εσφαλμένη πρόβλεψη του ταξινομητή [10].

2.3.2.2 Γνώση του ανταγωνιστή

- **Επιθέσεις "λευκού κουτιού"**

  Σε ένα περιβάλλον επίθεσης "λευκού κουτιού", ο ανταγωνιστής έχει πρόσβαση σε όλες τις πληροφορίες του μοντέλου-στόχου, συμπεριλαμβανομένης της αρχιτεκτονικής του,

των παραμέτρων κ.λπ. Ως εκ τούτου, ο ανταγωνιστής μπορεί να κάνει πλήρη χρήση των πληροφοριών του δικτύου για να δημιουργήσει με προσεκτικό τρόπο ανταγωνιστά παραδείγματα. Οι επιθέσεις "λευκού κουτιού" έχουν μελετηθεί εκτενώς, επειδή η αποκάλυψη της αρχιτεκτονικής και των παραμέτρων του μοντέλου βοηθά τους ανθρώπους να κατανοήσουν τις αδυναμίες των μοντέλων βαθιάς μάθησης. Σύμφωνα με τους Tramèr et al. [21], η ασφάλεια έναντι επιθέσεων "λευκού κουτιού είναι η ιδιότητα που επιθυμούμε να έχουν τα μοντέλα μηχανικής μάθησης. [10].

- **Επιθέσεις "μαύρου κουτιού"**
  Σε ένα περιβάλλον επίθεσης "μαύρου κουτιού", η πληροφορία της εσωτερικής διαμόρφωσης των μοντέλων βαθιάς μάθησης δεν είναι διαθέσιμη στους ανταγωνιστές. Οι ανταγωνιστές μπορούν μόνο να τροφοδοτούν το μοντέλο-στόχο με δεδομένα εισόδου και να ζητούν πληροφορίες για τις εξόδους που αυτό παράγει. Συνήθως επιτίθενται στα μοντέλα συνεχίζοντας να τροφοδοτούν με δείγματα το "μαύρο κουτί" και παρατηρώντας την έξοδο για να εκμεταλλευτούν τη σχέση εισόδου-εξόδου του μοντέλου. Αυτή η ρύθμιση μπορεί να αποκαλύψει τις αδυναμίες ενός μοντέλου, επειδή επιτρέπει στον αντίπαλο να εντοπίσει πιθανές ρηχές στατιστικές ιδιότητες μεταξύ εισόδου και εξόδου. Σε σύγκριση με τις επιθέσεις "λευκού κουτιού", οι επιθέσεις "μαύρου κουτιού" είναι πιο πρακτικές στις εφαρμογές, επειδή οι σχεδιαστές μοντέλων συνήθως δεν δημοσιοποιούν τις παραμέτρους του μοντέλου τους για λόγους ιδιοκτησίας [10].

2.3.2.3 Μονάδα διαταραχής

Στον τομέα του κειμένου, οι ανταγωνιστικές επιθέσεις μπορούν να ταξινομηθούν σε επίπεδο χαρακτήρα, σε επίπεδο λέξης, σε επίπεδο πρότασης και σε πολλαπλά επίπεδα ανάλογα με τις μονάδες διαταραχής κατά τη δημιουργία ανταγωνιστικών παραδειγμάτων.

- **Επιθέσεις σε επίπεδο χαρακτήρα**
  Οι επιθέσεις σε επίπεδο χαρακτήρα υποδηλώνουν ότι οι ανταγωνιστές τροποποιούν αρκετούς χαρακτήρες εντός λέξεων για να δημιουργήσουν ανταγωνιστικά παραδείγματα που μπορούν να ξεγελάσουν τους ταξινομητές. Συγκεκριμένα, οι τροποποιήσεις είναι ως επί το πλείστον ορθογραφικά λάθη και συνήθως περιλαμβάνουν την εισαγωγή, την ανταλλαγή, τη διαγραφή και την αναστροφή χαρακτήρων [9].

- **Επιθέσεις σε επίπεδο λέξης**
  Οι επιθέσεις σε επίπεδο λέξης περιλαμβάνουν τροποποιήσεις διάφορων λέξεων. Οι επιτιθέμενοι δημιουργούν ανταγωνιστικά παραδείγματα εισάγοντας, αντικαθιστώντας ή διαγράφοντας ορισμένες λέξεις με διάφορους τρόπους [9].

- **Επιθέσεις σε επίπεδο πρότασης**
  Οι επιθέσεις σε επίπεδο πρότασης συνήθως εισάγουν μια πρόταση σε ένα κείμενο ή ξαναγράφουν την πρόταση διατηρώντας τα νόημά της [9].

- **Επιθέσεις πολλαπλών επιπέδων**
  Οι επιθέσεις πολλαπλών επιπέδων ενσωματώνουν περισσότερες από μία από τις τρεις

προαναφερθείσες επιθέσεις για να επιτύχουν την ανεπαίσθητη και με υψηλό ποσοστό επιτυχίας ανταγωνιστική επίθεση [9].

2.3.3 Μετρικές μη αντιληπτότητας για δείγματα κειμένου

Τα ανταγωνιστικά παραδείγματα συντίθενται με την προσθήκη ανεπαίσθητων διαταραχών σε αυθεντικές εισόδους για να προκαλέσουν την παραγωγή λανθασμένων ετικετών εξόδου από το μοντέλο [22]. Στον τομέα της εικόνας, υιοθετούνται διάφορες μετρικές για τη μέτρηση της μη αντιληπτότητας των ανταγωνιστικών παραδειγμάτων. Η νόρμα L_p είναι η πιο συχνά χρησιμοποιούμενη μέθοδος που ορίζεται ως εξής

$$\|\Delta x\|_p = \sqrt[p]{\sum_{i=1}^{n} |x'_i - x_i|^p} \tag{2.1}$$

όπου το:

- p αντιπροσωπεύει τον τύπο της απόστασης που χρησιμοποιείται για τη μέτρηση της ομοιότητας του αρχικού δείγματος x και του διαταραγμένου δείγματος x' (π.χ. $p = 2$ υποδηλώνει τη γνωστή ευκλείδεια απόσταση)
- $\|\Delta x\|_p$ αντιπροσωπεύει την ένταση της διαταραχής
- n αντιπροσωπεύει τις διαστάσεις των x και x', ήτοι x και x' έχουν n χαρακτηριστικά
- x_i αντιπροσωπεύει το υπ' αριθμόν i χαρακτηριστικό του αρχικού δείγματος x, $i = 1, 2, ..., n$
- x'_i αντιπροσωπεύει το υπ' αριθμόν i χαρακτηριστικό του διαταραγμένου δείγματος x', $i = 1, 2, ..., n$

Ωστόσο, δεν είναι δυνατόν να χρησιμοποιηθεί η νόρμα L_p για την ακριβή μέτρηση της μη αντιληπτότητας σε εισόδους κειμένου. Στο πεδίο της εικόνας, μπορεί κανείς να εισαγάγει μια διαταραχή σε επίπεδο εικονοστοιχείου χωρίς να αλλάξει πραγματικά το πώς εμφανίζεται η εικόνα στο ανθρώπινο μάτι (βλ. σχήμα 2.1). Αυτό, όμως, δεν ισχύει στον τομέα του κειμένου, επειδή κάθε διαταραχή είναι ορατή στον άνθρωπο, λόγω των εισαγόμενων γραμματικών και ορθογραφικών λαθών. Για τους σκοπούς της εργασίας μας, μια ανταγωνιστική επίθεση στο πεδίο του κειμένου είναι επιτυχής (από την άποψη της μη αντιληπτότητας) εάν:

- Δεν παρατηρούνται προφανή γραμματικά και συντακτικά λάθη [9].
- Το ανταγωνιστικό παράδειγμα διατηρεί το νόημα του αρχικού δείγματος [9].
- Η έξοδος του μοντέλου στο ανταγωνιστικό κείμενο είναι διαφορετική από αυτή της αρχικής εισόδου, πράγμα που σημαίνει ότι έχει προκύψει λανθασμένη έξοδος [9].

Έτσι, η πλειονότητα των μετρικών που έχουν υιοθετηθεί για τις εικόνες δεν μπορούν να εφαρμοστούν άμεσα για την αξιολόγηση της ποιότητας των ακολουθιών κειμένου που

αποτελούν προϊόντα ανταγωνιστικής επίθεσης, λόγω των εγγενώς διαφορετικών μηχανισμών διαταραχής σε αυτούς τους δύο τύπους δεδομένων. Στη συνέχεια, παρουσιάζουμε τις μετρικές που χρησιμοποιούνται για τη μέτρηση της μη αντιληπτότητας σε ανταγωνιστικά δείγματα που έχουν μορφή κειμένου [9].

- **Ευκλείδεια απόσταση** [23]. Η ευκλείδεια απόσταση αναφέρεται στην ομοιότητα μεταξύ δύο **διανυσμάτων** μετρώντας την απόστασή τους στον ευκλείδειο χώρο. Δοθέντων δύο διανυσμάτων-λέξεων $\overrightarrow{u}, \overrightarrow{v}$ διαστατικότητας n, ήτοι $\overrightarrow{u} = (u_1, u_2, ..., u_n)$ και $\overrightarrow{v} = (v_1, v_2, ..., v_n)$, η Ευκλείδεια απόσταση ED αυτών των διανυσμάτων ορίζεται ως:

$$ED = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + ... + (u_n - v_n)^2} \tag{2.2}$$

  Όσο μικρότερη είναι η απόσταση, τόσο πιο παρόμοια είναι τα διανύσματα.

- **Απόσταση επεξεργασίας** [24]. Η απόσταση επεξεργασίας αναφέρεται στην ομοιότητα μεταξύ δύο **συμβολοσειρών** υπολογίζοντας τον αριθμό των πράξεων επεξεργασίας που απαιτούνται για τη μετατροπή της μιας συμβολοσειράς στην άλλη. Η *απόσταση Levenshtein* [25] είναι μια ευρέως χρησιμοποιούμενη απόσταση επεξεργασίας. Για δύο συμβολοσειρές a και b, η απόσταση Levenshtein lev υπολογίζεται ως εξής

$$lev(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} lev(i-1,j) + 1 \\ lev(i,j-1) + 1 \\ lev(i-1,j-1) + 1_{a_i \neq b_i} \end{cases} & \text{otherwise} \end{cases} \tag{2.3}$$

  όπου $lev(i,j)$ είναι η απόσταση μεταξύ των πρώτων χαρακτήρων i στο a και των πρώτων χαρακτήρων j στο b. Όσο μικρότερη είναι η απόσταση Levenshtein, τόσο πιο παρόμοιες είναι οι δύο συμβολοσειρές.

- **Απόσταση συνημιτόνου** [3]. Η απόσταση συνημιτόνου αναφέρεται στην ομοιότητα μεταξύ δύο **διανυσμάτων** μετρώντας το συνημίτονο της μεταξύ τους γωνίας. Συγκεκριμένα, δοθέντων δύο διανυσμάτων λέξεων $\overrightarrow{u}, \overrightarrow{v}$ διαστατικότητας n, δηλαδή $\overrightarrow{u} = (u_1, u_2, ..., u_n)$ και $\overrightarrow{v} = (v_1, v_2, ..., v_n)$, το συνημίτονο ομοιότητας CD υπολογίζεται ως εξής:

$$CD = \frac{\overrightarrow{u} \cdot \overrightarrow{v}}{\|\overrightarrow{u}\| \cdot \|\overrightarrow{v}\|} = \frac{\sum_{i=1}^{n} u_i \times v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}} \tag{2.4}$$

  Σε σύγκριση με την ευκλείδεια απόσταση, η απόσταση συνημίτονου δίνει μεγαλύτερη βαρύτητα στη διαφορά μεταξύ των κατευθύνσεων των δύο διανυσμάτων. Όσο πιο συνεπείς είναι οι κατευθύνσεις τους, τόσο πιο παρόμοια είναι τα διανύσματα.

- **Συντελεστής ομοιότητας Jaccard** [26]. Ο συντελεστής ομοιότητας Jaccard χρησιμοποιείται για τη σύγκριση της ομοιότητας μεταξύ δύο αντικειμένων, τα οποία αναπαρίστανται ως σύνολα. Για δύο δεδομένα σύνολα A και B, ο συντελεστής ομοιότητας Jaccard $J(A,B)$ υπολογίζεται ως εξής

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.5}$$

όπου $0 \leq J(A,B) \leq 1$. Όσο πιο κοντά στο 1 είναι η τιμή του $J(A,B)$, τόσο πιο παρόμοια είναι τα δύο σύνολα. Στον τομέα των κειμένων, η τομή $A \cap B$ είναι το σύνολο που περιέχει λέξεις που υπάρχουν και στις δύο ακολουθίες κειμένου, ενώ η ένωση $A \cup B$ είναι το σύνολο που περιέχει λέξεις που υπάρχουν σε κάθε ακολουθία κειμένου (χωρίς επανάληψη).

Chapter 3

Προτεινόμενη Μέθοδος

3.1 Διατύπωση προβλήματος

Τα μοντέλα μηχανικής μάθησης που εκπαιδεύονται για να επιτύχουν κορυφαία ακρίβεια σε προκαθορισμένα σύνολα δεδομένων, συχνά μαθαίνουν να βασίζουν τις προβλέψεις τους σε επιφανειακές στατιστικές ιδιότητες ανάμεσα σε είσοδο και έξοδο που συνάγονται κατά την εκπαίδευση [4]. Ως εκ τούτου, μια μικρή διαταραχή στην είσοδο μπορεί να επηρεάσει σημαντικά τις προβλέψεις τέτοιων μοντέλων, καθιστώντας τα έτσι ευάλωτα σε ανταγωνιστικές επιθέσεις. Για παράδειγμα, οι Jin et. al. [3] έδειξαν ότι η επίθεση στον κορυφαίο ταξινομητή ΣΦΓ μειώνει την ακρίβεια από 89,4% σε μόλις 4%.

3.2 Κίνητρο

Το 2018, οι Camburu et. al. [4] εισήγαγαν το σύνολο δεδομένων e-SNLI, το οποίο επεκτείνει το σύνολο δεδομένων SNLI [27] με την ενσωμάτωση επεξηγήσεων φυσικής γλώσσας ελεύθερης μορφής με ανθρώπινο σχολιασμό, οι οποίες αιτιολογούν γιατί ένα συγκεκριμένο ζεύγος προτάσεων (premise και hypothesis) συνδέεται με τη σημασιολογική σχέση συνεπαγωγής, αντίφασης ή ουδετερότητας.

Οι Camburu et. al. [4] υποστηρίζουν ότι το σύνολο δεδομένων e-SNLI μπορεί να αξιοποιηθεί προς την κατεύθυνση της εκπαίδευσης ερμηνεύσιμων μοντέλων που παρέχουν ισχυρές εξηγήσεις για τις προβλέψεις τους. Ωστόσο, δεν έχουν διεξαχθεί συστηματικές έρευνες ή πειράματα προκειμένου να επαληθευτεί ή να απορριφθεί ο ισχυρισμός αυτός. Στην παρούσα εργασία, διερευνούμε κατά πόσον ο παραπάνω ισχυρισμός ισχύει από την άποψη της ανθεκτικότητας απέναντι σε ανταγωνιστικές επιθέσεις. Συγκεκριμένα, διεξάγουμε μια σειρά από πειράματα προκειμένου να ελέγξουμε αν τα μοντέλα ΣΦΓ που εκπαιδεύονται με εξηγήσεις φυσικής γλώσσας είναι πιο ανθεκτικά απέναντι σε ανταγωνιστικές επιθέσεις.

3.3 Πρόταση

Τα πειράματά μας βασίζονται σε μεγάλο βαθμό στη διάταξη *ExplainThenPredict* που εισήγαγαν οι Camburu et. al. [4]. Σύμφωνα με αυτό τη διάταξη, η προβλεπόμενη ετικέτα δεν εξαρτάται άμεσα από τις προτάσεις premise και hypothesis στην είσοδο. Αντ' αυτού, εξαρτάται από μια εξήγηση που προσπαθεί να περιγράψει τη σημασιολογική σχέση μεταξύ

premise και hypothesis. Η ενδιάμεση εξήγηση αποτελεί την καινοτομία της εργασίας μας και διερευνούμε κατά πόσον μπορεί να φιλτράρει τον θόρυβο που υπερτίθεται στις προτάσεις εισόδου, βελτιώνοντας έτσι την ανταγωνιστική ανθεκτικότητα.

Chapter 4

Πειραματικό Μέρος

4.1 Πειραματική διάταξη

4.1.1 Διάταξη βασισμένη σε εξηγήσεις

Αυτή η διάταξη εκμεταλλεύεται τις εξηγήσεις φυσικής γλώσσας και στην πραγματικότητα αντιστοιχεί στο ExplainThenPredict (βλέπε εικόνα 4.1) που εισήγαγαν οι Camburu et. al. [4]. Σύμφωνα με το ExplainThenPredict, η εργασία Συμπερασμού Φυσικής Γλώσσας (ΣΦΓ) τροποποιείται ως εξής:

1. Δεδομένου ενός ζεύγους premise και hypothesis, ένα παραγωγικό μοντέλο (*Seq2Seq* ως κωδική ονομασία) παράγει μια εξήγηση ελεύθερης μορφής σε φυσική γλώσσα που δικαιολογεί τη σημασιολογική σχέση μεταξύ του premise και του hypothesis (συνεπαγωγή, αντίφαση ή ουδέτερη).

2. Αυτή η εξήγηση τροφοδοτείται σε έναν ταξινομητή *Expl2Label* ως κωδική ονομασία) ο οποίος προβλέπει την ετικέτα εξόδου (συνεπαγωγή, αντίφαση ή ουδέτερη).

Η εικόνα 4.2 απεικονίζει μια υποδειγματική διαδικασία εξαγωγής συμπερασμάτων που ακολουθεί την παραπάνω διάταξη. Για σκοπούς επίδειξης, επιλέγουμε ένα ζεύγος σύντομων και απλών προτάσεων από το σύνολο δεδομένων e-SNLI, δηλαδή η *'Ενα Land Rover διασχίζει ένα ποτάμι* χρησιμεύει ως premise και η *'Ενα όχημα διασχίζει ένα ποτάμι* χρησιμεύει ως hypothesis. Κατά τη διάρκεια της συμπερασματολογίας, το παραγωγικό μοντέλο παράγει την εξήγηση *Το Land Rover είναι όχημα*, η οποία αποτυπώνει σωστά τη σημασιολογική σχέση των δύο προτάσεων, και, τέλος, η εξήγηση αυτή τροφοδοτείται στον ταξινομητή, ο οποίος, με τη σειρά του, προβλέπει την ετικέτα *συμπέρασμα*. Αυτή η ετικέτα ταιριάζει με την πραγματική και είναι εννοιολογικά ευθυγραμμισμένη με την παραγόμενη εξήγηση.

Γίνεται σαφές ότι η προβλεπόμενη ετικέτα εξαρτάται τώρα από την ενδιάμεση εξήγηση που παράγεται από το παραγωγικό μοντέλο, αντί να εξαρτάται άμεσα από την προϋπόθεση και την υπόθεση εισόδου. Επιπλέον, παρέχοντας την ενδιάμεση εξήγηση ως είσοδο στον ταξινομητή εξήγησης, μπορούμε να θεωρήσουμε τον συνδυασμό των δύο μοντέλων ως ένα μαύρο κουτί που τροφοδοτείται με ένα ζεύγος premise και hypothesis και προβλέπει την αντίστοιχη ετικέτα. Επομένως, η τροποποιημένη εργασία ΣΦΓ καταλήγει στην παραδοσιακή εργασία ΣΦΓ.

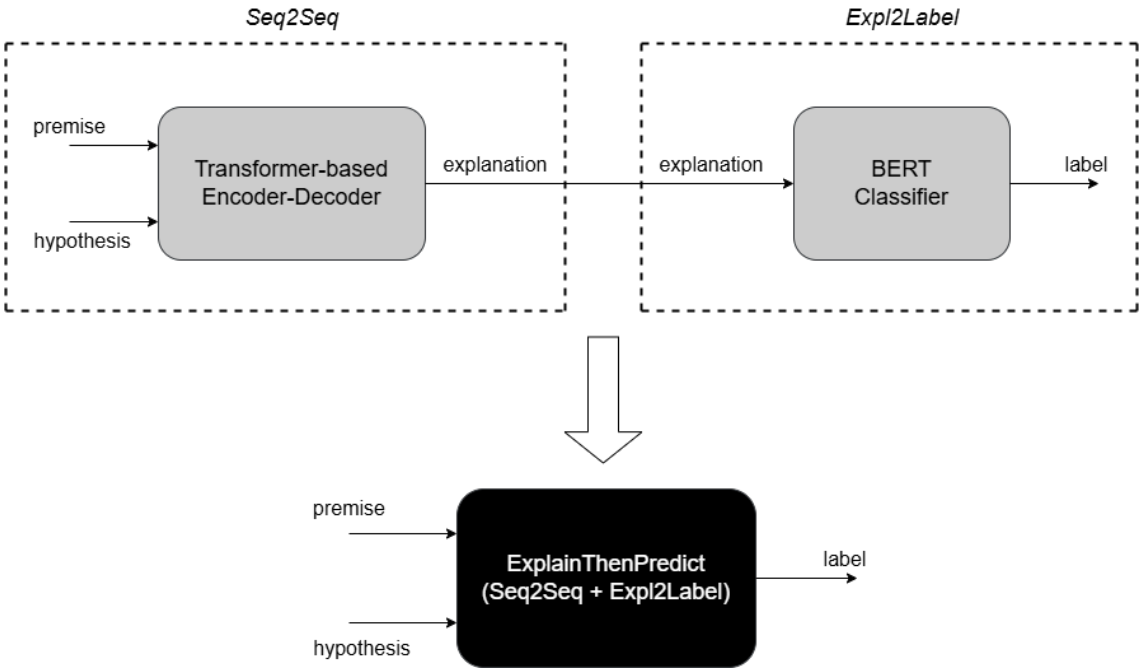

Figure 4.1. *ExplainThenPredict setup*

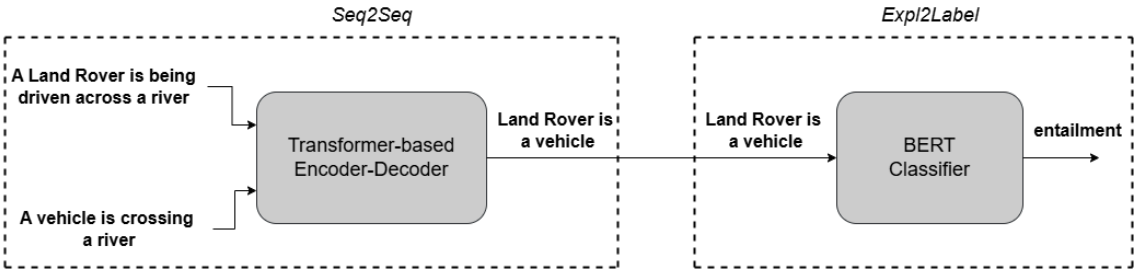

Figure 4.2. *Υποδειγματική διαδικασία συμπερασμού, σύμφωνα με τη διάταξη ExplainThenPredict.*

Σε όλα τα πειράματα που θα αναφερθούν παρακάτω, εξετάζουμε τις ακόλουθες αρχιτεκτονικές για τα μοντέλα μας:

- **Generative model:** Οι Camburu et. al. [4] χρησιμοποίησαν στα πειράματά τους παραγωγικά μοντέλα βασισμένα σε RNN. Εμείς επιλέγουμε να διαφοροποιηθούμε εξετάζοντας την αρχιτεκτονική των μετασχηματιστών (transformers), η οποία είναι πιο αποτελεσματική και χρησιμοποιείται ευρέως στις μέρες μας. Έτσι, το δικό μας παραγωγικό μοντέλο είναι ουσιαστικά ένα *transformers-based encoder-decoder* μοντέλο (βλ. κεφάλαιο 1 για λεπτομέρειες). Συγκεκριμένα, για το μέρος του αποκωδικοποιητή χρησιμοποιούμε το GPT-2 και για το μέρος του κωδικοποιητή χρησιμοποιούμε τις πιο διαδεδομένες παραλλαγές BERT, δηλαδή τις BERT, DistilBERT, ALBERT και RoBERTa. Για παράδειγμα, εάν ο κωδικοποιητής ακολουθεί την αρχιτεκτονική BERT, ονομάζουμε το μοντέλο μας *BERT2GPT*, εάν ο κωδικοποιητής ακολουθεί την αρχιτεκτονική RoBERTa, ονομάζουμε το μοντέλο *ROBERTA2GPT* κ.ο.κ.

- **Ταξινομητής επεξήγησης:** Χρησιμοποιούμε έναν ταξινομητή βασισμένο σε BERT,

δηλαδή έναν κωδικοποιητή BERT με μια κεφαλή ταξινόμησης πάνω από τις ενσωματώσεις. Πρόκειται για μια απλή, αλλά αποτελεσματική αρχιτεκτονική η οποία, όπως θα αποδειχθεί αργότερα, επιτυγχάνει εκπληκτικά υψηλή ακρίβεια.

Αξίζει να τονιστεί ότι η καινοτομία της εργασίας μας περιστρέφεται γύρω από την παραγωγή εξηγήσεων που αποδίδεται στο παραγωγικό μοντέλο. Κατά συνέπεια, η πειραματική μας ανάλυση επικεντρώνεται σε διάφορες παραλλαγές της αρχιτεκτονικής του παραγωγικού μοντέλου, διατηρώντας παράλληλα την απλή αρχιτεκτονική BERT ως βάση για τον ταξινομητή εξηγήσεων.

4.1.2 Διάταξη χωρίς εξηγήσεις

Αυτή η διάταξη αντιστοιχεί στην παραδοσιακή εργασία ΣΦΓ, όπου ένας ταξινομητής τροφοδοτείται με ένα ζεύγος premise και hypothesis και προβλέπει τη σημασιολογική τους σχέση (συνεπαγωγή, αντίφαση, ουδέτερη). Στην περίπτωση αυτή, η ετικέτα εξόδου εξαρτάται άμεσα από το premise και το hypothesis στην είσοδο. Αυτή η ρύθμιση δεν περιλαμβάνει εξηγήσεις και συνεπώς θα χρησιμεύσει ως σημείο αναφοράς (baseline). Ο ταξινομητής είναι ουσιαστικά ένας ταξινομητής βασισμένος σε BERT, δηλαδή ένας κωδικοποιητής BERT με μια κεφαλή ταξινόμησης πάνω από τις ενσωματώσεις.

4.2 Εκπαίδευση και Αξιολόγηση

4.2.1 Σημείο αναφοράς (baseline)

Η διαδικασία εκπαίδευσης του βασικού μοντέλου είναι αρκετά απλή. Υπενθυμίζουμε ότι το βασικό μοντέλο είναι ουσιαστικά ένας ταξινομητής βασισμένος στο BERT, δηλαδή ένας κωδικοποιητής BERT με μια κεφαλή ταξινόμησης πάνω από τις ενσωματώσεις. Κατά την εκπαίδευση, παρέχεται τόσο το ζεύγος (premise, hypothesis) όσο και η αυθεντική ετικέτα, ενώ κατά την εξαγωγή συμπερασμάτων ο ταξινομητής προβλέπει την ετικέτα με βάση το ζεύγος εισόδου premise και hypothesis. Φυσικά, η *ακρίβεια (accuracy)* χρησιμοποιείται ως μετρική επικύρωσης για τον ταξινομητή. Όλες οι απαραίτητες λεπτομέρειες για την εκπαίδευση του μοντέλου αναφοράς μπορούν να βρεθούν στον παρακάτω πίνακα 4.1.

Λεπτομέρειες σχετικά με την εκπαίδευση	
encoder checkpoint	bert-base-uncased
encoder max length	128
# epochs	5
batch size	32
# GPUs	1
GPU type	NVIDIA Volta V100
training time	~6 hours
test acc	90.13%

Table 4.1. *Λεπτομέρειες σχετικά με την εκπαίδευση του μοντέλου αναφοράς.*

4.2.2 Μοντέλο ExplainThenPredict

Στην εικόνα 4.2, το παραγωγικό μοντέλο και ο ταξινομητής εξήγησης φαίνεται να συνδέονται μεταξύ τους, εφόσον η έξοδος του πρώτου χρησιμεύει ως είσοδος για το δεύτερο. Ωστόσο, αυτό δεν είναι απολύτως αληθές, διότι τα δύο μοντέλα εκπαιδεύονται *χωριστά* και "συνδέονται" μόνο κατά τη στιγμή της εξαγωγής συμπερασμάτων. Αυτή είναι μια λογική προσέγγιση, καθώς τα δύο μοντέλα επιτελούν διαφορετική λειτουργία, ήτοι το παραγωγικό μοντέλο είναι παράγει κείμενο και ο ταξινομητής επεξήγησης ταξινομεί κείμενο.

Έχοντας εκπαιδεύσει το παραγωγικό μοντέλο και τον ταξινομητή εξηγήσεων για τις αντίστοιχες εργασίες τους, τα δύο μοντέλα μπορούν να διασυνδεθούν μεταξύ τους, όπως φαίνεται στην εικόνα 4.2. Επομένως, μπορούμε να θεωρήσουμε αυτόν τον συνδυασμό μοντέλων ως ένα μαύρο κουτί και να εκτελέσουμε συμπερασμό σε αυτό το μαύρο κουτί, όπου η είσοδος είναι τώρα ένα ζεύγος (premise, hypothesis) και η έξοδος είναι η ετικέτα που περιγράφει τη σημασιολογική τους σχέση (συνεπαγωγή, αντίφαση, ουδέτερη).

4.2.2.1 Seq2Seq

Για την εκπαίδευση του παραγωγικού μοντέλου που βασίζεται σε μετασχηματιστή, κατά την εκπαίδευση παρέχουμε τόσο το ζεύγος (premise, hypothesis) όσο και την αντίστοιχη εξήγηση, ενώ κατά την εξαγωγή συμπερασμάτων το μοντέλο προβλέπει μια εξήγηση με βάση μόνο το ζεύγος (premise, hypothesis).

Τώρα, είναι υψίστης σημασίας ο καθορισμός της μετρικής επικύρωσης που θα χρησιμοποιήσουμε για την επιλογή του βέλτιστου μοντέλου Seq2Seq κατά τη φάση της εκπαίδευσης. Για να μπορέσουμε να το κάνουμε αυτό, πρέπει να εξετάσουμε το έργο που προσπαθούν να επιλύσουν τα μοντέλα Seq2Seq. Το συγκεκριμένο έργο είναι η παραγωγή επεξηγήσεων, η οποία εμπίπτει στην κατηγορία *παραγωγή κειμένου*. Ορισμένες δημοφιλείς μετρικές παραγωγής κειμένου είναι οι BLEU [28], METEOR [29], ROUGE [30] και BERT-score [31]. Ωστόσο, καμία από αυτές τις μετρικές δεν εμπίπτει ακριβώς στο πρόβλημά μας, συνεπώς αναγκαζόμαστε να δανειστούμε μια μετρική από άλλο πρόβλημα. Μετά από εκτεταμένο πειραματισμό, καθώς και human evaluation, καταλήγουμε ότι η πιο κατάλληλη μετρική για το task μας είναι η METEOR, η οποία, σύμφωνα με το manual annotation, παράγει τις πιο ποιοτικές εξηγήσεις και έχει πράγματι το υψηλότερο correlation με το human judgement. Παραδείγμα από το manual annotation υπάρχουν στον πίνακα 4.2

premise	hypothesis	gold label	predicted explanation	Does the explanation fully justify the label?
A blond headed child in yellow boots and yellow jacket vest playing in the gravel with his pail, shovel and trucks	A blonde child is playing	entailment	A child playing is the same as a child playing	NO
A blond headed child in yellow boots and yellow jacket vest playing in the gravel with his pail, shovel and trucks	A blonde child is playing	entailment	A blond headed child is a type of blond child and playing in the gravel is a type of playing	**YES**
An elderly woman wearing a skirt is picking out vegetables at a local market	A young girl is blowing bubbles	contradiction	An elderly woman is not a young girl	NO
An elderly woman wearing a skirt is picking out vegetables at a local market	A young girl is blowing bubbles	contradiction	An elderly woman is not a young girl. Picking out vegetables is not the same as blowing bubbles	**YES**

Table 4.2. *Παραδείγματα manual annotation των εξηγήσεων που συλλέχθηκαν τυχαία.*

	BERT2GPT	**ALBERT2GPT**	**DISTILBERT2GPT**	**ROBERTA2GPT**
encoder checkpoint	bert-base-uncased	albert-base-v2	distilbert-base-uncased	roberta-base
encoder max length	128	128	128	128
decoder checkpoint	gpt2	gpt2	gpt2	gpt2
decoder max length	64	64	64	64
text generation strategy	greedy search	greedy search	greedy search	greedy search
# epochs	5	5	5	5
batch size	32	32	32	32
# GPUs	1	1	1	1
GPU type	NVIDIA Volta V100	NVIDIA Volta V100	NVIDIA Volta V100	NVIDIA Volta V100
training time (↓)	12 hrs & 20 mins	12 hrs & 16 mins	**9 hrs & 35 mins**	12 hrs & 29 mins
meteor (↑)	0.5332	**0.5591**	0.5393	0.5509
bert-score (↑)	0.8707	0.8742	0.8701	**0.8744**
rouge (↑)	0.5885	0.6005	0.5859	**0.6011**
bleu (↑)	0.3859	0.3911	0.3719	**0.3992**

Table 4.3. *Λεπτομέρειες σχετικά με την εκπαίδευση των παραλλάγων Seq2Seq που εξετάζουμε.*

Έχοντας καθιερώσει τη METEOR ως τη μετρική επικύρωσης για την εργασία μας, είμαστε σε θέση να προχωρήσουμε στη διαδικασία εκπαίδευσης των παραλλαγών Seq2Seq με τις οποίες πειραματιστήκαμε. Όλες οι απαραίτητες λεπτομέρειες βρίσκονται στον πίνακα 4.3. Βλέπουμε ότι το DISTILBERT2GPT έχει τον μικρότερο χρόνο εκπαίδευσης λόγω της απόσταξης (distillation) που το καθιστά ταχύτερο και πιο αποτελεσματικό [32]. Επιπλέον, αξίζει να σημειωθεί ότι το ROBERTA2GPT επιτυγχάνει την υψηλότερη βαθμολογία σε όλες τις μετρικές παραγωγής κειμένου εκτός από τη βαθμολογία METEOR, όπου το ALBERT2GPT έχει την καλύτερη απόδοση.

Οι λεπτομέρειες σχετικά με τη χειροκίνητη αξιολόγηση των εξηγήσεων που παράγει κάθε μοντέλο συνοψίζονται στον πίνακα 4.4. Βλέπουμε ότι το ROBERTA2GPT καταφέρνει να παράγει τις πιο ακριβείς εξηγήσεις με βαθμολογία 77,17% και ακολουθεί το BERT2GPT με βαθμολογία 76,14%. Τέλος, έχουμε τις ALBERT2GPT και DISTILBERT2GPT με βαθμολογία 73.33% και 72.53% αντίστοιχα.

	BERT2GPT	**ALBERT2GPT**	**DISTILBERT2GPT**	**ROBERTA2GPT**
% correct explanations	76.14%	73.33%	72.53%	**77.17%**

Table 4.4. *Χειροκίνητη αξιολόγηση ενός τυχαίου υποσυνόλου των παραγόμενων εξηγήσεων. Το ποσοστό των ορθών εξηγήσεων αναφέρεται στα 100 δείγματα που συλλέξαμε.*

4.2.2.2 Expl2label

Για την εκπαίδευση του ταξινομητή εξηγήσεων, κατά τη φάση της εκπαίδευσης παρέχουμε τόσο την εξήγηση όσο και την πραγματική ετικέτα, ενώ κατά τη φάση του συμπερασμού το μοντέλο προβλέπει την ετικέτα εξόδου με βάση μόνο την εξήγηση εισόδου. Φυσικά, η *ακρίβεια* χρησιμοποιείται ως μετρική επικύρωσης για τον ταξινομητή. Όλες οι απαραίτητες λεπτομέρειες για την εκπαίδευση και αξιολόγηση του μοντέλου Expl2Label μπορούν να βρεθούν στον πίνακα 4.6. Βλέπουμε ότι ένας απλός ταξινομητής με βάση το BERT επιτυγχάνει ακρίβεια 97,47% στα δεδομένα δοκιμής e-SNLI. Αυτό είναι ένα τρομερά υψηλό αποτέλεσμα

και υποστηρίζει τον ισχυρισμό ότι, στο σύνολο δεδομένων e-SNLI, μπορεί κανείς εύκολα να συσχετίσει μια εξήγηση με μια ετικέτα (συνεπαγωγή, αντίφαση, συνεπαγωγή) χωρίς να έχει γνώση του premise και του hypothesis που αντιστοιχεί στην εξήγηση [4]. Αυτό υποστηρίζει περαιτέρω τον ισχυρισμό ότι η διάταξη *ExpainThenPredict* είναι πράγματι μια λογική αποσύνθεση του έργου ΣΦΓ. Ωστόσο, αυτό δεν ισχύει πάντα, επειδή υπάρχουν πολλοί τρόποι για να διατυπωθεί μια εξήγηση (βλ. πίνακα 4.5 για λεπτομέρειες).

premise	**hypothesis**	**label**	**explanation**
A woman is in the park	A person is in the park	entailment	A woman is a person
A woman is in the park	There is no person in the park	contradiction	A woman is a person

Table 4.5. *Ακριβώς η ίδια εξήγηση μπορεί να δικαιολογεί διαφορετική σχέση συνεπαγωγής, ανάλογα με τις προτάσεις premise και hypothesis [4].*

Fine-tuning details	
encoder checkpoint	bert-base-uncased
encoder max length	128
# epochs	5
batch size	32
# GPUs	1
GPU type	NVIDIA Volta V100
training time	5 hours & 42 mins
test acc	97.47%

Table 4.6. *Λεπτομέρειες επί της εκπαίδευσης και αξιολόγησης του ταξινομητή Expl2Label βασισμένου στο BERT.*

4.2.2.3 Διασύνδεση των μοντέλων για συμπερασμό

Έχοντας *ξεχωριστά* εκπαιδεύσει τα μοντέλα Seq2Seq και Expl2Label, ήρθε η ώρα να τα "συνδέσουμε" μεταξύ τους, τροφοδοτώντας την εξήγηση που παράγεται από το Seq2Seq στο Expl2Label και εκτελώντας *συμπεράσματα* σε ολόκληρο το μοντέλο, το οποίο μπορεί να θεωρηθεί ως μαύρο κουτί (αυτή την αρχιτεκτονική μπορεί να δει κανείς στο σχήμα 4.1). Αναφέρουμε τα αποτελέσματα μετά τον συμπερασμό των διαφόρων παραλλαγών του μοντέλου ExplainThenPredict στον πίνακα 4.7 παρακάτω.

Το μοντέλο μας δέχεται ένα ζεύγος (premise, hypothesis) στην είσοδο και προβλέπει την αντίστοιχη ετικέτα (συνεπαγωγή, αντίφαση, ουδέτερη), επομένως κατά τη διάρκεια της εξαγωγής συμπερασμάτων μας ενδιαφέρει η ακρίβεια του μοντέλου μας. Βλέπουμε ότι όλες οι παραλλαγές του μοντέλου έχουν χαμηλότερη ακρίβεια από την ακρίβεια του μοντέλου αναφοράς (90,13%), ωστόσο, όπως θα συζητήσουμε εκτενώς παρακάτω, αυτό δεν πρέπει να μας ενοχλεί καθόλου. Επιπλέον, βλέπουμε ότι η παραλλαγή ROBERTA2GPT Seq2Seq επιτυγχάνει την υψηλότερη ακρίβεια (87,97%), ακολουθούμενη από το BERT2GPT που έχει ακρίβεια 86,72% και τέλος έχουμε τα DISTILBERT2GPT και ALBERT2GPT. Η παρατήρηση αυτή είναι σύμφωνη με τα ευρήματα του πίνακα 4.4 και δίνει μια ένδειξη ότι οι ακριβείς εξηγήσεις οδηγούν γενικά σε πιο ακριβείς προβλέψεις. Στην ενότητα 4.3, θα προσπαθήσουμε

	BERT2GPT + BERT	**ALBERT2GPT + BERT**	**DISTILBERT2GPT + BERT**	**ROBERTA2GPT + BERT**
acc	86.72%	85.45%	85.15%	**87.97%**
acc (entailment)	89.13%	86.76%	87.29%	**90.17%**
acc (contradiction)	90.42%	88.35%	85.82%	**91.69%**
acc (neutral)	80.4%	81.14%	**82.20%**	82.01%

Table 4.7. *Αποτελέσματα συμπερασμού του μοντέλου ExplainThenPredict. Τα BERT2GPT, ALBERT2GPT, DISTILBERT2GPT and ROBERTA2GPT υποδεικνύουν την παραλλαγή που χρησιμοποιήθηκε για το Seq2Seq, ενώ το BERT υποδεικνύει τον ταξινομητή Expl2Label. Αναφέρουμε τόσο τη συνολική ακρίβεια όσο και την ακρίβεια ανά ετικέτα εξόδου.*

να συσχετίσουμε την ποιότητα των εξηγήσεων με την ανθεκτικότητα των αντιπάλων, η οποία είναι άλλωστε η ουσία της εργασίας μας. Τέλος, αξίζει να σημειωθεί ότι όλα τα εκπαιδευμένα μοντέλα ExplainThenPredict επιτυγχάνουν ακρίβεια συνεπαγωγής και αντίφασης κοντά στο 90% και ακρίβεια ουδετερότητας γύρω στο 80%, δηλαδή σημαντικά χαμηλότερη. Λαμβάνοντας υπόψη ότι το σύνολο δεδομένων e-SNLI είναι απόλυτα ισορροπημένο μεταξύ των 3 κατηγοριών του, οι χαμηλότερες τιμές της ουδέτερης ακρίβειας υποδηλώνουν ότι, γενικά, το μοντέλο μας μπορεί να ταξινομήσει ευκολότερα ζεύγη όπου η πρόταση hypothesis συνεπάγεται από την premise ή έρχεται σε αντίθεση με αυτήν.

4.3 Αποτελέσματα Ανταγωνιστικών Επιθέσεων

Χρησιμοποιούμε τις συνταγές επίθεσης BERT-attack [33] και TextFooler [3] προκειμένου να αξιολογήσουμε την ανθεκτικότητα των μοντέλων ExplainThenPredict απέναντι σε ανταγωνιστικές επιθέσεις. Οι BERT-attack και TextFooler είναι κορυφαίες συνταγές παραγωγής ανταγωνιστικών επιθέσεων όσον αφορά την επίθεση στην αρχιτεκτονική των μετασχηματιστών και ταιριάζουν απόλυτα στο έργο μας, καθώς και οι δύο παράγουν ανταγωνιστικά παραδείγματα που διατηρούν τη σημασιολογία και τη σύνταξη του αρχικού κειμένου εισόδου.[1]

Τα πειράματα που αφορούν τις επιθέσεις διαξάγονται σε δύο στάδια, όπως φαίνεται παρακάτω:

1. Στοχεύουμε στην πρόταση *premise* όταν επιτιθέμεθα στο μοντέλο αναφοράς και στα μοντέλα ExplainThenPredict διατηρώντας την υπόθεση ως έχει.

2. Στοχεύουμε στην πρόταση *hypothesis* όταν επιτιθέμεθα στο μοντέλο αναφοράς και στα μοντέλα ExplainThenPredict διατηρώντας το premise ως έχει.

Για να μετρήσουμε την ανθεκτικότητα των αντιπάλων, θα χρησιμοποιήσουμε μια μετρική που ονομάζεται *ποσοστό επιτυχίας επίθεσης (attack success rate)*. Το ποσοστό επιτυχίας επίθεσης είναι στην πραγματικότητα το ποσοστό των προσπαθειών επίθεσης που παράγουν επιτυχημένα ανταγωνιστικά παραδείγματα. Επομένως, υψηλότερες τιμές του ποσοστού επιτυχίας επίθεσης υποδηλώνουν υψηλό αριθμό επιτυχημένων προσπαθειών για τη

[1] Στο κεφάλαιο 2, τονίζουμε ότι μας ενδιαφέρει η αναγνώριση κειμενικής συνεπαγωγής μόνο μεταξύ *σημασιολογικά συνεκτικών* προτάσεων.

δημιουργία έγκυρου ανταγωνιστικού παραδείγματος, επομένως το μοντέλο-στόχος είναι ευάλωτο στην ανταγωνιστική επίθεση. Αντίθετα, χαμηλότερες τιμές υποδηλώνουν χαμηλό αριθμό επιτυχημένων προσπαθειών για τη δημιουργία έγκυρου ανταγωνιστικού παραδείγματος, επομένως το μοντέλο-στόχος είναι ανθεκτικό (robust) στην ανταγωνιστική επίθεση. Κατά συνέπεια, και λαμβάνοντας υπόψη ότι η ανθεκτικότητα σε αντίπαλα παραδείγματα είναι μια μέτρηση της ευαισθησίας ενός μοντέλου σε ανταγωνιστικά παραδείγματα, καθίσταται σαφές ότι πράγματι το ποσοστό επιτυχίας της επίθεσης είναι μια έγκυρη και λογική μετρική για το έργο μας.

4.3.1 TextFooler

Ο αλγόριθμος TextFooler διαθέτει 2 υπερ-παραμέτρους που μας επιτρέπουν να ελέγξουμε την επιθυμητή ποικιλομορφία και σημασιολογική ομοιότητα μεταξύ του αρχικού και του ανταγωνιστικού κειμένου. Πιο συγκεκριμένα, η υπερ-παράμετρος *max_candidates* (που ονομάζεται N στην εργασία TextFooler [3]) χρησιμοποιείται για να καθορίσει τον αριθμό των υποψήφιων συνωνύμων που μπορούν δυνητικά να αντικαταστήσουν μια "ευάλωτη" λέξη που υπάρχει στην αρχική ακολουθία και η υπερ- παράμετρος *min_cos_sim* (που ονομάζεται δ στην εργασία TextFooler [3]) χρησιμοποιείται για να καθορίσει το κατώφλι ομοιότητας συνημίτονου, ώστε η διαταραγμένη ακολουθία να θεωρείται έγκυρο ανταγωνιστικό παράδειγμα. Επομένως, καθίσταται σαφές ότι η μεγέθυνση της τιμής *max_candidates* ή η μείωση της τιμής *min_cos_sim* θα αναγκάσει τη δημιουργία πιο διαφορετικών υποψηφίων συνωνύμων εις βάρος της σημασιολογικής ομοιότητας. Αντίθετα, η μείωση του *max_candidates* ή η διεύρυνση του *min_cos_sim* θα αναγκάσει τη δημιουργία σημασιολογικά πιο όμοιων υποψηφίων συνωνύμων εις βάρος της ποικιλομορφίας.

Οι Jin et. al. [3] ισχυρίζονται ότι η ρύθμιση *max_candidates* = 50 και *min_cos_sim* = 0.7 επιτυγχάνει ισορροπία μεταξύ ποικιλότητας και ελέγχου σημασιολογικής ομοιότητας, οπότε συμπεριλαμβάνουμε αυτόν τον συνδυασμό τιμών στα πειράματά μας. Επιπλέον, προκειμένου να ευνοήσουμε τη σημασιολογική ομοιότητα έναντι της ποικιλομορφίας, αυξάνουμε ελαφρώς την τιμή του *min_cos_sim*, δηλαδή πειραματιζόμαστε επίσης με τη ρύθμιση *max_candidates* = 50 και *min_cos_sim* = 0.75. Τα αποτελέσματα των επιθέσεων για τις ρυθμίσεις που αναφέρθηκαν παραπάνω συνοψίζονται στους πίνακες 4.8 και 4.9.

Το Σχήμα 4.3 απεικονίζει τα επιτυχημένα ποσοστά επιτυχίας επίθεσης των μοντέλων μας ExplainThenPredict σε σχέση με το ποσοστό επιτυχίας επίθεσης του μοντέλου αναφοράς για τα σενάρια όπου η πρόταση-στόχος είναι το premise και η πρόταση-στόχος είναι το hypothesis, αντίστοιχα. Παρατηρούμε ότι ανεξάρτητα από την πρόταση-στόχο και την τιμή του *min_cos_sim*, όλες οι παραλλαγές του μοντέλου επιτυγχάνουν χαμηλότερο ποσοστό επιτυχίας επίθεσης σε σύγκριση με το μοντέλο αναφοράς. Αυτό σημαίνει ότι οι παραπάνω παραλλαγές είναι όντως πιο ανθεκτικές όσον αφορά τις ανταγωνιστικές επιθέσεις.

Η εικόνα 4.4 απεικονίζει το ποσοστό μείωσης του ποσοστού επιτυχίας των επιθέσεων που επιτεύχθηκε από τα μοντέλα μας. Φυσικά, η τιμή αναφοράς είναι το ποσοστό επιτυχίας επιθέσεων του μοντέλου αναφοράς για κάθε ρύθμιση *min_cos_sim* $in\{0.7, 0.75\}$. Έτσι, μπορούμε να συμπεράνουμε ότι η παραλλαγή ROBERTA2GPT είναι η πιο επιτυχημένη στην ενίσχυση της ανθεκτικότητας και ακολουθεί η παραλλαγή BERT2GPT. Αυτές οι δύο

TextFooler (target sentence: premise)					
max_candidates = 50 min_cos_sim = 0.7	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	24.93%	27.76%	24.2%	28.74%	24.08%
Attack success rate (↓)	72.16%	67.99%	70.9%	67.33%	71.1%
Average perturbed word %	11.32%	8.04%	7.75%	7.93%	8.12%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	43.74	44.1	41.75	44.57	42.16
max_candidates = 50 min_cos_sim = 0.75	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	33.22%	35.2%	30.92%	36.18%	31.1%
Attack success rate (↓)	62.9%	59.41%	61.5%	58.88%	61.2%
Average perturbed word %	11.52%	8.02%	7.79%	7.89%	8.23%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	37.02	36.68	35.11	37.14	35.49

Table 4.8. *Σύνοψη αποτελεσμάτων (αλγόριθμος παραγωγής επιθέσεων: TextFooler, πρόταση-στόχος: premise). Η ανάλυσή μας βασίζεται στο ποσοστό επιτυχημένων επιθέσεων, συνεπώς έμφαση πρέπει να δοθεί στις αντίστοιχες γραμμές.*

TextFooler (target sentence: hypothesis)					
max_candidates = 50 min_cos_sim = 0.7	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	10.33%	13.86%	12.68%	16.31%	11.83%
Attack success rate (↓)	88.46%	84.01%	85.16%	81.46%	86.11%
Average perturbed word %	8.3%	7.1%	7.22%	7.32%	7.02%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	24.3	24.6	25.05	25.92	24.16
max_candidates = 50 min_cos_sim = 0.75	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	15.89%	18.6%	18.19%	21.85%	16.73%
Attack success rate (↓)	82.26%	78.55%	78.71%	75.16%	80.35%
Average perturbed word %	8.61%	7.28%	7.35%	7.49%	7.2%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	20.64	20.64	21.08	21.67	20.27

Table 4.9. *Σύνοψη αποτελεσμάτων (αλγόριθμος παραγωγής επιθέσεων: TextFooler, πρόταση-στόχος: hypothesis). Η ανάλυσή μας βασίζεται στο ποσοστό επιτυχημένων επιθέσεων, συνεπώς έμφαση πρέπει να δοθεί στις αντίστοιχες γραμμές.*

παραλλαγές ξεχωρίζουν με συνέπεια, δηλαδή τόσο για τις premise/hypothesis ως προτάσεις-στόχους όσο και για κάθε ρύθμιση $min\_cos\_sim \in \{0.7, 0.75\}$. Οι ALBERT2GPT και DISTILBERT2GPT επιτυγχάνουν σημαντικά βελτιωμένη ανθεκτικότητα μόνο όταν η πρόταση-στόχος είναι υπόθεση, ενώ αγωνίζονται να επιτύχουν μειωμένο ποσοστό επιτυχίας επίθεσης όταν η πρόταση-στόχος είναι υπόθεση.

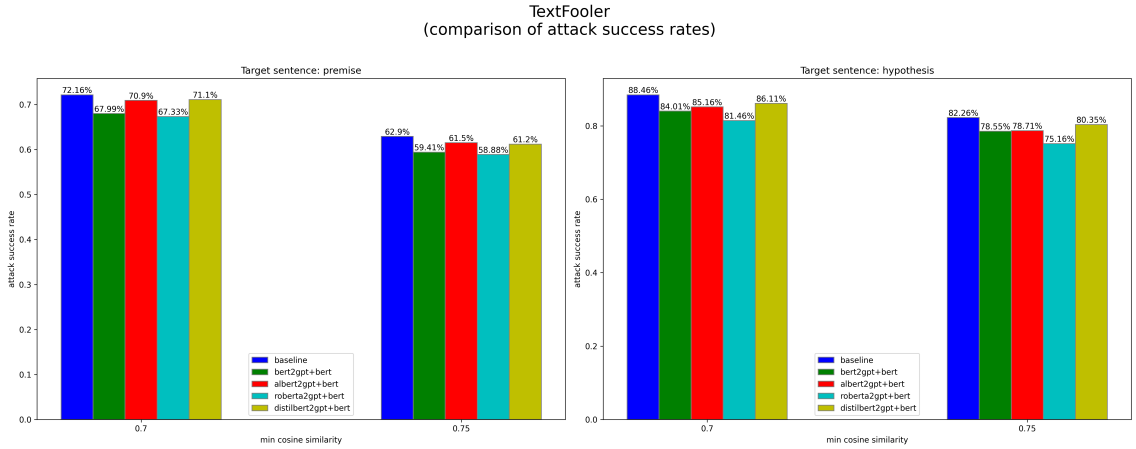

Figure 4.3. *Οπτικοποίηση των τιμών του ποσοστού επιτυχημένων ανταγωνιστικών επιθέσεων για τις παραλλαγές των ExplainThenPredict μοντέλων και του μοντέλου αναφοράς (αλγόριθμος παραγωγής ανταγωνιστικών επιθέσεων: TextFooler).*

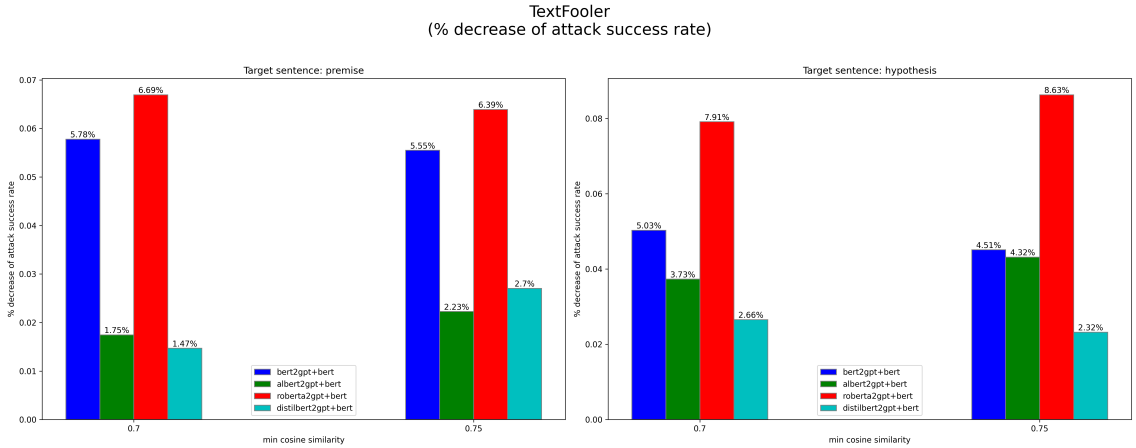

Figure 4.4. *Οπτικοποίηση της ποσοστιαίας ελάττωσης του ποσοστού επιτυχημένων ανταγωνιστικών επιθέσεων (ως προς το μοντέλο αναφοράς) που επιτυγχάνουν οι παραλλαγές των ExplainThenPredict μοντέλων (αλγόριθμος παραγωγής ανταγωνιστικών επιθέσεων: TextFooler).*

4.3.2 BERT-attack

Ο αλγόριθμος BERT-attack διαθέτει μία υπερ-παράμετρο που μας επιτρέπει να ελέγξουμε την επιθυμητή σημασιολογική ομοιότητα μεταξύ του αρχικού και του ανταγωνιστικού κειμένου. Πιο συγκεκριμένα, η υπερ-παράμετρος *max_candidates* (που ονομάζεται K στην εργασία BERT-attack [33]) χρησιμοποιείται για να καθορίσει τον αριθμό των υποψήφιων συνωνύμων που μπορούν δυνητικά να αντικαταστήσουν μια "ευάλωτη" λέξη που υπάρχει στην αρχική ακολουθία. Διαισθητικά, η μεγέθυνση του *max_candidates* θα αναγκάσει τη δημιουργία λιγότερο όμοιων σημασιολογικά υποψηφίων συνωνύμων και επομένως το ποσοστό επιτυχίας της επίθεσης αναμένεται να αυξηθεί [33]. Επιλέγουμε να πειραματιστούμε με τις τιμές *max_candidates* $in\{6, 8\}$, οι οποίες επιτυγχάνουν μια ισορροπία μεταξύ σημασιολογικής ομοιότητας και υπολογιστικών πόρων. Τα αποτελέσματα της επίθεσης για το προαναφερθέν περιβάλλον συνοψίζονται στους πίνακες 4.10 και 4.11 παρακάτω.

Η εικόνα 4.5 απεικονίζει τα ποσοστά επιτυχίας των επιθέσεων των μοντέλων μας Ex-

BERT-attack (target sentence: premise)					
max_candidates = 6	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	19.26%	25.18%	21.92%	25.4%	23.37%
Attack success rate (↓)	78.5%	70.96%	74.35%	71.13%	72.55%
Average perturbed word %	10.88%	7.97%	7.84%	7.84%	8.09%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	26.96	29.77	28.52	29.93	29.19
max_candidates = 8	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	14.79%	22.54%	19.02%	22.22%	20.02%
Attack success rate (↓)	83.48%	74.01%	77.74%	74.74%	76.49%
Average perturbed word %	10.78%	7.94%	7.78%	7.83%	8.15%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	31.27	35.9	33.84	35.68	34.88

Table 4.10. *Σύνοψη αποτελεσμάτων (αλγόριθμος παραγωγής επιθέσεων: BERT-attack, πρόταση-στόχος: premise). Η ανάλυσή μας βασίζεται στο ποσοστό επιτυχημένων επιθέσεων, συνεπώς έμφαση πρέπει να δοθεί στις αντίστοιχες γραμμές.*

BERT-attack (target sentence: hypothesis)					
max_candidates = 6	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	9.16%	15.23%	13.48%	16.11%	13.16%
Attack success rate (↓)	89.77%	82.44%	84.23%	81.68%	84.54%
Average perturbed word %	8.58%	7.22%	7.41%	7.24%	7.34%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	15.28	16.24	16.3	16.59	16.03
max_candidates = 8	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	4.87%	11.68%	10.19%	12.53%	9.61%
Attack success rate (↓)	94.57%	86.54%	88.08%	85.76%	88.71%
Average perturbed word %	8.16%	7.07%	7.24%	7.16%	7.21%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	17.37	18.93	19.03	19.46	18.69

Table 4.11. *Σύνοψη αποτελεσμάτων (αλγόριθμος παραγωγής επιθέσεων: BERT-attack, πρόταση-στόχος: hypothesis). Η ανάλυσή μας βασίζεται στο ποσοστό επιτυχημένων επιθέσεων, συνεπώς έμφαση πρέπει να δοθεί στις αντίστοιχες γραμμές.*

plainThenPredict σε σχέση με το ποσοστό επιτυχίας των επιθέσεων του μοντέλου αναφοράς για τα σενάρια όπου η πρόταση-στόχος είναι η premise και η πρόταση-στόχος είναι το hypothesis. Έχει ενδιαφέρον να παρατηρήσουμε ότι, και για τα δύο σενάρια, όλες οι παραλλαγές του μοντέλου ExplainThenPredict επιτυγχάνουν σταθερά χαμηλότερο ποσοστό επιτυχίας επίθεσης σε σύγκριση με το μοντέλο αναφοράς, επομένως ο στόχος της αύξησης

της ανθεκτικότητας επιτυγχάνεται.

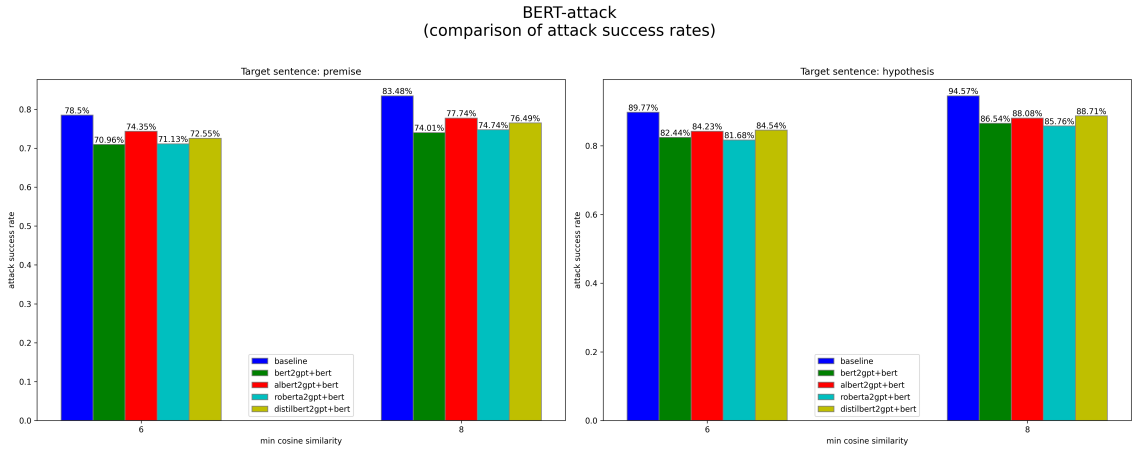

Figure 4.5. *Οπτικοποίηση της ποσοστιαίας ελάττωσης του ποσοστού επιτυχημένων ανταγωνιστικών επιθέσεων (ως προς το μοντέλο αναφοράς) που επιτυγχάνουν οι παραλλαγές των ExplainThenPredict μοντέλων (αλγόριθμος παραγωγής ανταγωνιστικών επιθέσεων: BERT-attack).*

Η εικόνα 4.6 απεικονίζει το ποσοστό μείωσης του ποσοστού επιτυχίας των επιθέσεων που επιτεύχθηκε από τα μοντέλα μας. Φυσικά, η τιμή αναφοράς είναι το ποσοστό επιτυχίας επίθεσης του μοντέλου αναφοράς για κάθε ρύθμιση *max_candidates in*$\{6, 8\}$. Μπορούμε να συμπεράνουμε ότι, για το σενάριο όπου η πρόταση-στόχος είναι η premise, η παραλλαγή BERT2GPT είναι η πιο επιτυχημένη στην ενίσχυση της ανθεκτικότητας των αντιπάλων και ακολουθεί η παραλλαγή ROBERTA2GPT. Ωστόσο, για το σενάριο όπου η πρόταση-στόχος είναι το hypothesis, συμβαίνει το αντίθετο, δηλαδή η ROBERTA2GPT επιτυγχάνει τη μεγαλύτερη μείωση του ποσοστού επιτυχίας επίθεσης και ακολουθεί η BERT2GPT. Σε κάθε περίπτωση, μπορούμε να συμπεράνουμε ότι, παρόμοια με τα παραπάνω πειράματα που αφορούν το TextFooler, οι BERT2GPT και ROBERTA2GPT εμφανίζονται ως οι πιο ανθεκτικές όσον αφορά το ποσοστό επιτυχίας επίθεσης και ακολουθούν οι DISTILBERT2GPT και ALBERT2GPT, οι οποίες, στην περίπτωση της επίθεσης BERT, καταφέρνουν να μειώσουν σημαντικά το ποσοστό επιτυχίας επίθεσης, σε σύγκριση με το TextFooler, όπου το πέτυχαν με δυσκολία.

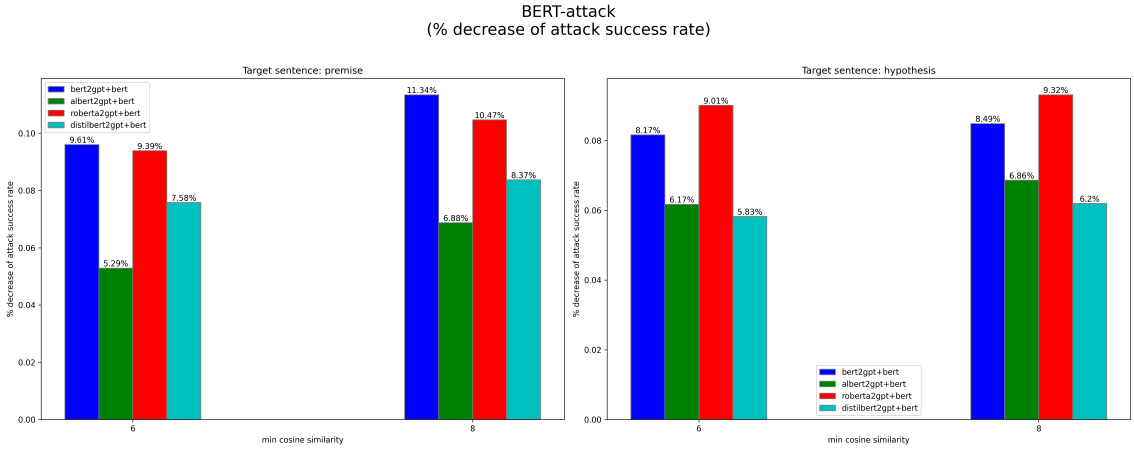

Figure 4.6. *Οπτικοποίηση της ποσοστιαίας ελάττωσης του ποσοστού επιτυχημένων ανταγωνιστικών επιθέσεων (ως προς το μοντέλο αναφοράς) που επιτυγχάνουν οι παραλλαγές των ExplainThenPredict μοντέλων (αλγόριθμος παραγωγής ανταγωνιστικών επιθέσεων: BERT-attack).*

Chapter 5

Συμπεράσματα

Στην παρούσα εργασία, τροποποιήσαμε την παραδοσιακή εργασία ΣΦΓ αξιοποιώντας εξηγήσεις φυσικής γλώσσας τόσο κατά την εκπαίδευση όσο και κατά την εξαγωγή συμπερασμάτων. Ακολουθώντας την εργασία των Camburu et. al. [4], χρησιμοποιήσαμε στα πειράματά μας τη ρύθμιση *ExplainThenPredict*, η οποία περιλαμβάνει πρώτα ένα παραγωγικό μοντέλο που, δεδομένης μιας πρότασης premise και μιας πρότασης hypothesis, προβλέπει μια εξήγηση που προσπαθεί να περιγράψει τη σημασιολογική σχέση μεταξύ της premise και της hypothesis, και δεύτερον αυτή η εξήγηση τροφοδοτείται σε έναν ταξινομητή που προβλέπει την ετικέτα εξόδου. Η εργασία μας βασίζεται στην παρατήρηση ότι, χρησιμοποιώντας την παραπάνω ρύθμιση, η προβλεπόμενη ετικέτα δεν εξαρτάται πλέον άμεσα από τις προτάσεις premise και hypothesis στην είσοδο. Αντ' αυτού, εξαρτάται από μια εξήγηση που προσπαθεί να περιγράψει τη σημασιολογική σχέση μεταξύ των δύο προτάσεων, επομένως αυτή η ενδιάμεση εξήγηση θα μπορούσε ενδεχομένως να φιλτράρει το θόρυβο που επικαλύπτεται στις προτάσεις εισόδου. Δόθηκε έμφαση στην παραγωγή εξηγήσεων και, συγκεκριμένα, τέσσερα μοντέλα κωδικοποιητή - αποκωδικοποιητή βασισμένα σε μετασχηματιστές (BERT2GPT, ALBERT2GPT, ROBERTA2GPT και DISTILBERT2GPT) εκπαιδεύτηκαν για τον σκοπό αυτό.

Επιτεθήκαμε στα μοντέλα μας χρησιμοποιώντας τους αλγορίθμους BERT-attack και TextFooler, οι οποίοι είναι οι πιο διαδεδομένοι στον τομέα του NLP και παράγουν ανταγωνιστικά παραδείγματα που διατηρούν τη σημασιολογική ομοιότητα ανάμεσα στην αυθεντική και ανταγωνιστική πρόταση και, επίσης, χρησιμοποιήσαμε το ποσοστό επιτυχίας των επιθέσεων για να μετρήσουμε την ανθεκτικότητα απέναντι στις ανταγωνιστικές επιθέσεις. Τα πειράματά μας έδειξαν ότι όλα τα εκπαιδευμένα μοντέλα είναι όντως πιο ανθεκτικά σε σύγκριση με το μοντέλο αναφοράς, το οποίο είναι ένας απλός ταξινομητής που δεν αξιοποιεί τις εξηγήσεις φυσικής γλώσσας. Ειδικότερα, η μείωση του ποσοστού επιτυχίας επίθεσης παρατηρήθηκε σταθερά, δηλαδή τόσο για το TextFooler όσο και για το BERT-attack, και για τις δύο προτάσεις-στόχους (premise και hypothesis) και για κάθε τιμή υπερπαραμέτρου του αλγορίθμου επίθεσης που χρησιμοποιήθηκε στα πειράματά μας. Επιπλέον, συσχετίσαμε πειραματικά την ποιότητα μιας εξήγησης με την ανθεκτικότητα (μείωση του ποσοστού επιτυχίας της επίθεσης). Συγκεκριμένα, δείξαμε, μέσω ανθρώπινης αξιολόγησης, ότι οι παραλλαγές BERT2GPT και ROBERTA2GPT Seq2Seq παράγουν τις πιο ακριβείς εξηγήσεις, ενώ ταυτόχρονα επιτυγχάνουν την υψηλότερη μείωση του ποσοστού επιτυχίας επιθέσεων.

Ελπίζουμε ότι, στο μέλλον, η εργασία μας θα χρησιμεύσει ως ένα ισχυρό βασικό κριτήριο για τη συσχέτιση των εξηγήσεων φυσικής γλώσσας με την ανθεκτικότητα απέναντι σε adversarial attacks. Ενθαρρύνεται κανείς να πειραματιστεί με άλλα σύνολα δεδομένων εκτός του e-SNLI ή να χρησιμοποιήσει σύνθετες αρχιτεκτονικές που αποτυπώνουν βαθύτερα τη σημασιολογική σχέση μεταξύ premise και hypothesis.

Part II

English Version

Chapter 6

Transformer-based Encoder-Decoder Models

6.1 Introduction

The transformer-based encoder-decoder model was introduced by Vaswani et al. [5] in the famous *Attention is all you need* paper and is today the de-facto standard encoder-decoder architecture in Natural Language Processing (NLP) [6, 7]. In this chapter, we attempt to clarify how the transformer-based encoder-decoder architecture can be used for modeling sequence-to-sequence problems. To this end, we break down the transformer-based encoder-decoder model into its encoder and decoder part and we illustrate how the model can be used for inference based on its mathematical definiton.

This chapter is divided into four sections:

- **Background:** We provide a short history of neural encoder-decoder models, mainly focusing on RNN-based models.
- **Encoder-Decoder:** We present the transformer-based encoder-decoder model and we explain how the model can be used for inference in sequence-to-sequence tasks.
- **Encoder:** We break down the encoder part of the model.
- **Decoder:** We break down the decoder part of the model.

6.2 Background

6.2.1 Natural Language Generation Task

6.2.1.1 Definition

Tasks in Natural Language Generation (NLG), a subfield of NLP, are best expressed as sequence-to-sequence problems. Such tasks can be defined as finding a model that maps a sequence of input words to a sequence of target words. Summarization and translation are the most well-known examples of NLG tasks.

In the following, we assume that each word is encoded into a vector representation. n input words can then be represented as a sequence of n input vectors as follows:

$$\mathbf{X}_{1:n} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$$

Consequently, sequence-to-sequence problems can be solved by finding a mapping f from an input sequence of n vectors $\mathbf{X}_{1:n}$ to a sequence of m target vectors $\mathbf{Y}_{1:m}$, where the number of target vectors m is unknown apriori and depends on the input sequence. This mapping can be represented as follows:

$$f : \mathbf{X}_{1:n} \rightarrow \mathbf{Y}_{1:m}$$

6.2.1.2 Limitations

Sutskever et al. [34] noted that Deep Neural Networks (DNNs), despite their flexibility and power, can only define a mapping whose inputs and targets can be sensibly encoded with vectors of *fixed* dimensionality. Using a DNN model[1] to solve sequence-to-sequence problems would therefore mean that the number of target vectors m has to be known apriori and would have to be independent of the input $\mathbf{X}_{1:n}$. This is suboptimal, because, for NLG tasks, the number of target words usually depends on the input $\mathbf{X}_{1:n}$ and not just on the input length n. For example, an article of 1000 words can be summarized to both 200 words and 100 words depending on its content.

6.2.2 RNN-based Approach

In 2014, Cho et al. [35] and Sutskever et al. [34] proposed to use an encoder-decoder model purely based on Recurrent Neural Networks (RNNs) for sequence-to-sequence tasks. In contrast to DNNs, RNNs are capable of modeling a mapping to a *variable* number of target vectors.

6.2.2.1 Basic Concepts

During inference, the encoder RNN encodes an input sequence $\mathbf{X}_{1:n}$ by successively updating its hidden state[2]. After having processed the last input vector $\mathbf{x}_n$, the encoder's hidden state defines the input encoding $\mathbf{c}$. Thus, the encoder defines the mapping:

$$f_{\theta_{enc}} : \mathbf{X}_{1:n} \rightarrow \mathbf{c}$$

Then, the decoder's hidden state is initialized with the input encoding $\mathbf{c}$ and, during inference, the decoder RNN is used to auto-regressively generate the target sequence. Mathematically, the decoder defines the probability distribution of a target sequence $\mathbf{Y}_{1:m}$ given the hidden state $\mathbf{c}$:

$$p_{\theta_{dec}}(\mathbf{Y}_{1:m}|\mathbf{c})$$

By Bayes' rule the distribution can be decomposed into conditional distributions of

[1] The same applies to Convolutional Neural Networks (CNNs). While an input sequence of variable length can be fed into a CNN, the dimensionality of the target will always be dependent on the input dimensionality or fixed to a specific value.

[2] At the first step, the hidden state is initialized as a zero vector and fed to the RNN together with the first input vector $\mathbf{x}_1$.

single target vectors as follows:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\mathbf{c}) = \prod_{i=1}^{m} p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \mathbf{c})$$

Thus, if the architecture can model the conditional distribution of the next target vector, given all previous target vectors $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \mathbf{c}), \forall i \in \{1, \dots, m\}$, then it can model the distribution of any target vector sequence given the hidden state $\mathbf{c}$ by simply multiplying all conditional probabilities. The RNN-based decoder architecture models $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \mathbf{c})$ by sequentially mapping the previous inner hidden state $\mathbf{c}_{i-1}$[3] and the previous target vector $\mathbf{y}_{i-1}$ to the current inner hidden state $\mathbf{c}_i$ and a logit vector $\mathbf{l}_i$. This mapping can be represented as follows:

$$f_{\partial_{\text{dec}}} : (\mathbf{y}_{i-1}, \mathbf{c}_{i-1}) \rightarrow (\mathbf{l}_i, \mathbf{c}_i)$$

Subsequently, the softmax operation is used to transform the logit vector $\mathbf{l}_i$ to a conditional probablity distribution of the next target vector:

$$p(\mathbf{y}_i|\mathbf{l}_i) = \textbf{Softmax}(\mathbf{l}_i), \text{ with } \mathbf{l}_i = f_{\partial_{\text{dec}}}(\mathbf{y}_{i-1}, \mathbf{c}_{i-1})$$

For more detail on the logit vector and the resulting probability distribution, please see footnote[4].

From the above equation, we can see that the distribution of the current target vector $\mathbf{y}_i$ is directly conditioned on the previous target vector $\mathbf{y}_{i-1}$ and the previous hidden state $\mathbf{c}_{i-1}$. Because the previous hidden state $\mathbf{c}_{i-1}$ depends on all previous target vectors $\mathbf{y}_0, \dots, \mathbf{y}_{i-2}$, it can be stated that the RNN-based decoder implicitly (e.g. indirectly) models the conditional distribution $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \mathbf{c})$.

The space of possible target vector sequences $\mathbf{Y}_{1:m}$ is prohibitively large so that at inference, one has to rely on decoding methods that efficiently sample high probability target vector sequences from $p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\mathbf{c})$.

Given such a decoding method, during inference, the next input vector $\mathbf{y}_i$ can then be sampled from $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \mathbf{c})$ and is consequently appended to the input sequence so that the decoder RNN then models $p_{\partial_{\text{dec}}}(\mathbf{y}_{i+1}|\mathbf{Y}_{0:i}, \mathbf{c})$ to sample the next input vector $\mathbf{y}_{i+1}$ and so on in auto-regressive fashion.

An important feature of RNN-based encoder-decoder models is the definition of special vectors, such as the EOS and BOS vector. The EOS vector often represents the final input vector $\mathbf{x}_n$ to "cue" the encoder that the input sequence has ended and also defines the end of the target sequence. As soon as the EOS is sampled from a logit vector, the generation is complete.

[3] $\mathbf{c}_0$ is thereby defined as $\mathbf{c}$ being the output hidden state of the RNN-based encoder.

[4] A neural network can define a probability distribution over all words, i.e. $p(\mathbf{y}|\mathbf{c}, \mathbf{Y}_{0:i-1})$ as follows. First, the network defines a mapping from the inputs $\mathbf{c}, \mathbf{Y}_{0:i-1}$ to an embedded vector representation $\mathbf{y}'$, which corresponds to the RNN target vector. The embedded vector representation $\mathbf{y}'$ is then passed to the *language model head* layer, which means that it is multiplied by the word embedding matrix, i.e. $\mathbf{Y}^{\text{vocab}}$, so that a score between $\mathbf{y}'$ and each encoded vector $\mathbf{y} \in \mathbf{Y}^{\text{vocab}}$ is computed. The resulting vector is called the logit vector $\mathbf{l} = \mathbf{Y}^{\text{vocab\_size}} \cdot \mathbf{y}'$ and can be mapped to a probability distribution over all words by applying a softmax operation: $p(\mathbf{y}|\mathbf{c}) = \text{Softmax}(\mathbf{Y}^{\text{vocab\_size}} \cdot \mathbf{y}') = \text{Softmax}(\mathbf{l})$.

The BOS vector represents the input vector $\mathbf{y}_0$ fed to the decoder RNN at the very first decoding step. To output the first logit $\mathbf{l}_1$, an input is required and since no input has been generated at the first step a special BOS input vector is fed to the decoder RNN.

6.2.2.2 Example walkthrough

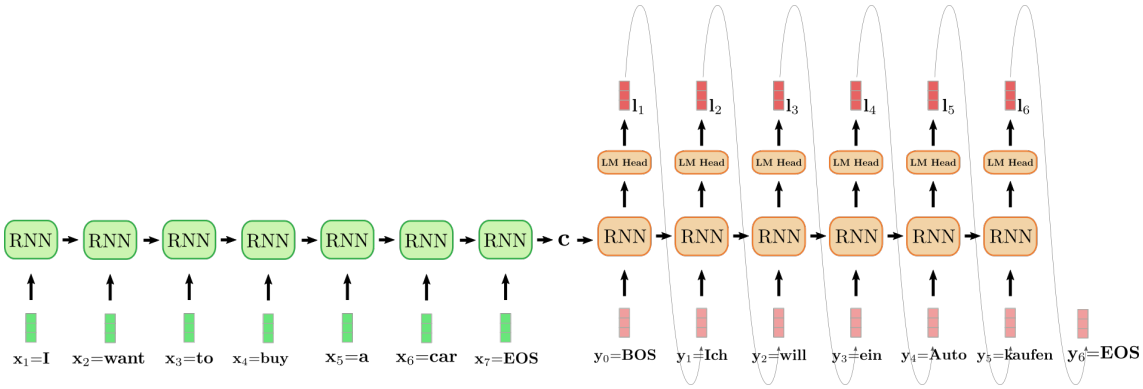

Figure 6.1. *Auto-regressive generation example of RNN-based encoder-decoder model.*

The unfolded RNN encoder is colored in green and the unfolded RNN decoder is colored in red.

The English sentence "I want to buy a car", represented by $\mathbf{x}_1$ = I, $\mathbf{x}_2$ = want, $\mathbf{x}_3$ = to, $\mathbf{x}_4$ = buy, $\mathbf{x}_5$ = a, $\mathbf{x}_6$ = car and $\mathbf{x}_7$ = EOS is translated into German: "Ich will ein Auto kaufen" defined as $\mathbf{y}_0$ = BOS, $\mathbf{y}_1$ = Ich, $\mathbf{y}_2$ = will, $\mathbf{y}_3$ = ein, $\mathbf{y}_4$ = Auto, $\mathbf{y}_5$ = kaufen and $\mathbf{y}_6$ = EOS.

To begin with, the input vector $\mathbf{x}_1$ = I is processed by the encoder RNN and updates its hidden state. Note that because we are only interested in the final encoder's hidden state $\mathbf{c}$, we can disregard the RNN encoder's target vector. The encoder RNN then processes the rest of the input sentence want, to, buy, a, car, EOS in the same fashion, updating its hidden state at each step until the vector $\mathbf{x}_7$ = *EOS* is reached[5]. In the illustration above the horizontal arrow connecting the unfolded encoder RNN represents the sequential updates of the hidden state.

The final hidden state of the encoder RNN, represented by $\mathbf{c}$ then completely defines the encoding of the input sequence and is used as the initial hidden stateof the decoder RNN. This can be seen as conditioning the decoder RNN on the encoded input.

To generate the first target vector, the decoder is fed the BOS vector, illustrated as $\mathbf{y}_0$ in the design above. The target vector of the RNN is then further mapped to the logit vector $\mathbf{l}_1$ by means of the LM Head feed-forward layer to define the conditional distribution of the first target vector as explained above:

$$p_{\theta_{dec}}(\mathbf{y}|\text{BOS}, \mathbf{c})$$

[5]Sutskever et al. [34] reverses the order of the input so that in the above example the input vectors would correspond to $\mathbf{x}_1$ = car, $\mathbf{x}_2$ = a, $\mathbf{x}_3$ = buy, $\mathbf{x}_4$ = to, $\mathbf{x}_5$ = want, $\mathbf{x}_6$ = I and $\mathbf{x}_7$ = EOS. The motivation is to allow for a shorter connection between corresponding word pairs such as $\mathbf{x}_6$ = I and $\mathbf{y}_1$ = Ich. The research group emphasizes that the reversal of the input sequence was a key reason for their model's improved performance on machine translation.

The word Ich is sampled (shown by the grey arrow, connecting $\mathbf{l}_1$ and $\mathbf{y}_1$) and consequently the second target vector can be sampled:

$$\text{will} \sim p_{\partial_{dec}}(\mathbf{y}|\text{BOS}, \text{Ich}, \mathbf{c})$$

And so on until at step $i = 6$, the EOS vector is sampled from $\mathbf{l}_6$ and the decoding is finished. The resulting target sequence amounts to $\mathbf{Y}_{1:6} = \{\mathbf{y}_1, \ldots, \mathbf{y}_6\}$, which is "Ich will ein Auto kaufen" in our example above.

To sum it up, an RNN-based encoder-decoder model, represented by $f_{\partial_{\text{enc}}}$ and $p_{\partial_{\text{dec}}}$ defines the distribution $p(\mathbf{Y}_{1:m}|\mathbf{X}_{1:n})$ by factorization:

$$p_{\partial_{\text{enc}},\partial_{\text{dec}}}(\mathbf{Y}_{1:m}|\mathbf{X}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{\text{enc}},\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \mathbf{X}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \mathbf{c}), \text{ with } \mathbf{c} = f_{\partial_{enc}}(X).$$

During inference, efficient decoding methods can auto-regressively generate the target sequence $\mathbf{Y}_{1:m}$. Nevertheless, RNN-based encoder-decoder models have two pitfalls:

- RNNs suffer from the vanishing gradient problem, making it very difficult to capture long-range dependencies [36].
- The inherent recurrent architecture of RNNs prevents efficient parallelization when encoding [5].

6.3 Encoder-Decoder

In 2017, Vaswani et al. [5] introduced the *Transformer* architecture and thereby gave birth to *transformer-based* encoder-decoder models.

Analogous to RNN-based encoder-decoder models, transformer-based encoder-decoder models consist of an encoder and a decoder which are both stacks of residual attention blocks. The key innovation of transformer-based encoder-decoder models is that such residual attention blocks can process an input sequence $\mathbf{X}_{1:n}$ of variable length n without exhibiting a recurrent structure. Not relying on a recurrent structure allows transformer-based encoder-decoders to be highly parallelizable, which makes the model orders of magnitude more computationally efficient than RNN-based encoder-decoder models on modern hardware.

As a reminder, to solve a sequence-to-sequence problem, we need to find a mapping of an input sequence $\mathbf{X}_{1:n}$ to an output sequence $\mathbf{Y}_{1:m}$ of variable length m. Let's see how transformer-based encoder-decoder models are used to find such a mapping.

Similar to RNN-based encoder-decoder models, the transformer-based encoder-decoder models define a conditional distribution of target vectors $\mathbf{Y}_{1:m}$ given an input sequence $\mathbf{X}_{1:n}$:

$$p_{\partial_{\text{enc}},\partial_{\text{dec}}}(\mathbf{Y}_{1:m}|\mathbf{X}_{1:n})$$

The transformer-based encoder part encodes the input sequence $\mathbf{X}_{1:n}$ to a sequence

of hidden states $\overline{\mathbf{X}}_{1:n}$, thus defining the mapping:

$$f_{\partial_{\text{enc}}} : \mathbf{X}_{1:n} \rightarrow \overline{\mathbf{X}}_{1:n}$$

The transformer-based decoder part then models the conditional probability distribution of the target vector sequence $\mathbf{Y}_{1:m}$ given the sequence of encoded hidden states $\overline{\mathbf{X}}_{1:n}$:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n}).$$

By Bayes' rule, this distribution can be factorized to a product of conditional probability distribution of the target vector $\mathbf{y}_i$ given the encoded hidden states $\overline{\mathbf{X}}_{1:n}$ and all previous target vectors $\mathbf{Y}_{0:i-1}$:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$$

The transformer-based decoder hereby maps the sequence of encoded hidden states $\overline{\mathbf{X}}_{1:n}$ and all previous target vectors $\mathbf{Y}_{0:i-1}$ to the logit vector $\mathbf{l}_i$. The logit vector $\mathbf{l}_i$ is then processed by the softmax operation to define the conditional probability distribution $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$, just as it is done for RNN-based decoders. However, in contrast to RNN-based decoders, the distribution of the target vector $\mathbf{y}_i$ is *explicitly* (or directly) conditioned on all previous target vectors $\mathbf{y}_0, \ldots, \mathbf{y}_{i-1}$ as we will see later in more detail. The 0th target vector $\mathbf{y}_0$ is hereby represented by a special "begin-of-sentence" BOS vector.

Having defined the conditional distribution $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$, we can now auto-regressively generate the output and thus define a mapping of an input sequence $\mathbf{X}_{1:n}$ to an output sequence $\mathbf{Y}_{1:m}$ at inference.

Let's visualize the complete process of auto-regressive generation of transformer-based encoder-decoder models.

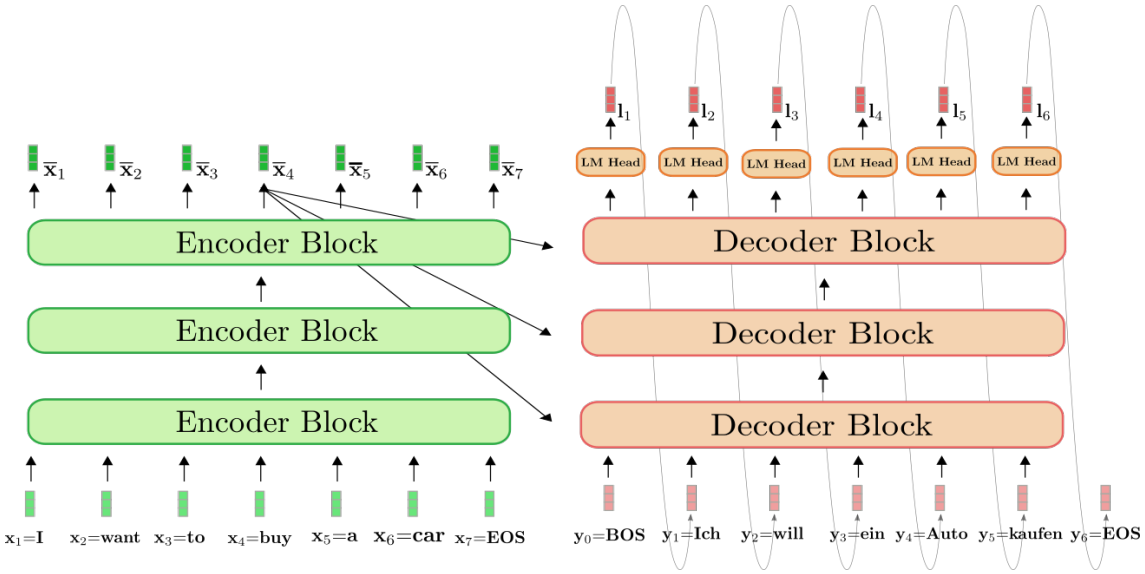

Figure 6.2. *Auto-regressive generation example of transformer-based encoder-decoder model.*

As can be seen in Figure 6.2, the transformer-based encoder is colored in green and the transformer-based decoder is colored in red. As in the previous section, we show how the English sentence "I want to buy a car", represented by $\mathbf{x}_1$ = I, $\mathbf{x}_2$ = want, $\mathbf{x}_3$ = to, $\mathbf{x}_4$ = buy, $\mathbf{x}_5$ = a, $\mathbf{x}_6$ = car, and $\mathbf{x}_7$ = EOS is translated into German: "Ich will ein Auto kaufen" defined as $\mathbf{y}_0$ = BOS, $\mathbf{y}_1$ = Ich, $\mathbf{y}_2$ = will, $\mathbf{y}_3$ = ein, $\mathbf{y}_4$ = Auto, $\mathbf{y}_5$ = kaufen, and $\mathbf{y}_6$ = EOS.

To begin with, the encoder processes the complete input sequence $\mathbf{X}_{1:7}$ = "I want to buy a car" (represented by the light green vectors) to a contextualized encoded sequence $\overline{\mathbf{X}}_{1:7}$. For example, $\overline{\mathbf{x}}_4$ defines an encoding that depends not only on the input $\mathbf{x}_4$ = "buy", but also on all other words "I", "want", "to", "a", "car" and "EOS", i.e. the context.

Next, the input encoding $\overline{\mathbf{X}}_{1:7}$ together with the BOS vector, i.e. $\mathbf{y}_0$, is fed to the decoder. The decoder processes the inputs $\overline{\mathbf{X}}_{1:7}$ and $\mathbf{y}_0$ to the first logit $\mathbf{l}_1$ (shown in darker red) to define the conditional distribution of the first target vector $\mathbf{y}_1$:

$$p_{\partial_{enc,dec}}(\mathbf{y}|\mathbf{y}_0, \mathbf{X}_{1:7}) = p_{\partial_{enc,dec}}(\mathbf{y}|\text{BOS, I want to buy a car EOS}) = p_{\partial_{dec}}(\mathbf{y}|\text{BOS}, \overline{\mathbf{X}}_{1:7})$$

Next, the first target vector $\mathbf{y}_1$ = Ich is sampled from the distribution (represented by the grey arrows) and can now be fed to the decoder again. The decoder now processes both $\mathbf{y}_0$ = "BOS" and $\mathbf{y}_1$ = "Ich" to define the conditional distribution of the second target vector $\mathbf{y}_2$:

$$p_{\partial_{dec}}(\mathbf{y}|\text{BOS Ich}, \overline{\mathbf{X}}_{1:7})$$

We can sample again and produce the target vector $\mathbf{y}_2$ = "will". We continue in auto-regressive fashion until at step 6 the EOS vector is sampled from the conditional distribution:

$$\text{EOS} \sim p_{\partial_{dec}}(\mathbf{y}|\text{BOS Ich will ein Auto kaufen}, \overline{\mathbf{X}}_{1:7})$$

And so on in auto-regressive fashion.

It is important to understand that the encoder is only used in the first forward pass to map $\mathbf{X}_{1:n}$ to $\overline{\mathbf{X}}_{1:n}$. As of the second forward pass, the decoder can directly make use of the previously calculated encoding $\overline{\mathbf{X}}_{1:n}$. For clarity, let's illustrate the first and the second forward pass for our example above.

As can be seen in Figure 6.3, only in step $i = 1$ do we have to encode "I want to buy a car EOS" to $\overline{\mathbf{X}}_{1:7}$. At step $i = 2$, the contextualized encodings of "I want to buy a car EOS" are simply reused by the decoder.

To sum it up:

- The transformer-based encoder module defines a mapping from the input sequence $\mathbf{X}_{1:n}$ to a contextualized encoding sequence $\overline{\mathbf{X}}_{1:n}$.
- The transformer-based decoder module defines the conditional probability distribution $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$.
- Given an appropriate decoding mechanism, the output sequence $\mathbf{Y}_{1:m}$ can auto-regressively be sampled from $p_{\partial_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n}), \forall i \in \{1, \ldots, m\}$.

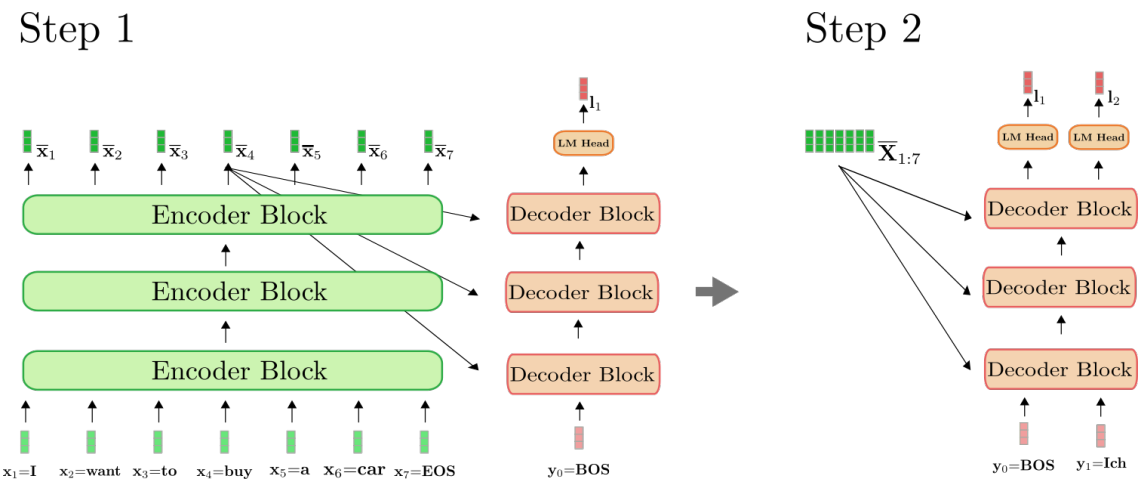

Figure 6.3. *Auto-regressive generation example of transformer-based encoder-decoder model step by step.*

Now that a general overview of how transformer-based encoder-decoder models work has been provided, we can dive deeper into both the encoder and decoder part of the model. More specifically, we will see exactly how the encoder makes use of the self-attention layer to yield a sequence of context-dependent vector encodings and how self-attention layers allow for efficient parallelization. Then, we will explain in detail how the self-attention layer works in the decoder model and how the decoder is conditioned on the encoder's output with *cross-attention* layers to define the conditional distribution $p_{\theta_{\text{dec}}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$. Along, the way it will become obvious how transformer-based encoder-decoder models solve the long-range dependencies problem of RNN-based encoder-decoder models.

6.4 Encoder

As mentioned in the previous section, the *transformer-based* encoder maps the input sequence to a contextualized encoding sequence:

$$f_{\theta_{\text{enc}}} : \mathbf{X}_{1:n} \rightarrow \overline{\mathbf{X}}_{1:n}$$

Taking a closer look at the architecture, the transformer-based encoder is a stack of residual encoder blocks. Each encoder block consists of a *bi-directional* self-attention layer, followed by two feed-forward layers. For simplicity, we disregard the normalization layers. Also, we will not further discuss the role of the two feed-forward layers, but simply see it as a final vector-to-vector mapping required in each encoder block[6]. The bi-directional self-attention layer puts each input vector $\mathbf{x}'_j, \forall j \in \{1, \ldots, n\}$ into relation with all input vectors $\mathbf{x}'_1, \ldots, \mathbf{x}'_n$ and by doing so transforms the input vector $\mathbf{x}'_j$ to a more "refined" contextual representation of itself, defined as $\mathbf{x}''_j$. Thereby, the first encoder block transforms each input vector of the input sequence $\mathbf{X}_{1:n}$ (shown in light

[6]It is argued in Yun et. al. [8] that feed-forward layers are crucial to map each contextual vector $\mathbf{x}'_i$ individually to the desired output space, which the *self-attention* layer does not manage to do on its own. It should be noted here, that each output token $\mathbf{x}'$ is processed by the same feed-forward layer.

green below) from a *context-independent* vector representation to a *context-dependent* vector representation, and the following encoder blocks further refine this contextual representation until the last encoder block outputs the final contextual encoding $\overline{\mathbf{X}}_{1:n}$ (shown in darker green below).

Let's visualize how the encoder processes the input sequence "I want to buy a car EOS" to a contextualized encoding sequence. Similar to RNN-based encoders, transformer-based encoders also add a special "end-of-sequence" input vector to the input sequence to hint to the model that the input vector sequence is finished[7].

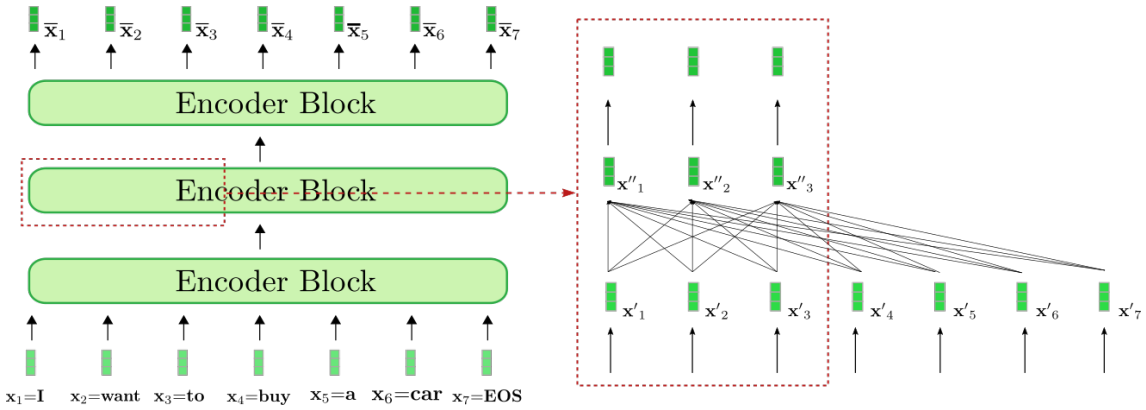

Figure 6.4. *Encoding process of the transformer-based encoder module.*

The exemplary transformer-based encoder is composed of three encoder blocks, whereas the second encoder block is shown in more detail in the red box on the right for the first three input vectors $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$. The bi-directional self-attention mechanism is illustrated by the fully-connected graph in the lower part of the red box and the two feed-forward layers are shown in the upper part of the red box. As stated before, we will focus only on the bi-directional self-attention mechanism.

As can be seen in Figure 6.4 each output vector of the self-attention layer $\mathbf{x''}_i, \forall i \in \{1, \ldots, 7\}$ depends *directly* on *all* input vectors $\mathbf{x'}_1, \ldots, \mathbf{x'}_7$. This means, for example, that the input vector representation of the word "want", i.e. $\mathbf{x'}_2$, is put into direct relation with the word "buy", i.e. $\mathbf{x'}_4$, but also with the word "I",i.e. $\mathbf{x'}_1$. The output vector representation of "want", i.e. $\mathbf{x''}_2$, thus represents a more refined contextual representation for the word "want".

Let's take a deeper look at how bi-directional self-attention works. Each input vector $\mathbf{x'}_i$ of an input sequence $\mathbf{X'}_{1:n}$ of an encoder block is projected to a key vector $\mathbf{k}_i$, value vector $\mathbf{v}_i$ and query vector $\mathbf{q}_i$ (shown in orange, blue, and purple respectively below) through three trainable weight matrices $\mathbf{W}_q, \mathbf{W}_v, \mathbf{W}_k$:

$$\mathbf{q}_i = \mathbf{W}_q \mathbf{x'}_i,$$

$$\mathbf{v}_i = \mathbf{W}_v \mathbf{x'}_i,$$

[7] However, the EOS input vector does not have to be appended to the input sequence, but has been shown to improve performance in many cases. On the contrary, the BOS target vector of the transformer-based decoder is required as a starting input vector to predict a first target vector.

$$\mathbf{k}_i = \mathbf{W}_k \mathbf{x'}_i,$$

$$\forall i \in \{1, \ldots n\}.$$

Note, that the **same** weight matrices are applied to each input vector $\mathbf{x}_i, \forall i \in \{i, \ldots, n\}$. After projecting each input vector $\mathbf{x}_i$ to a query, key, and value vector, each query vector $\mathbf{q}_j, \forall j \in \{1, \ldots, n\}$ is compared to all key vectors $\mathbf{k}_1, \ldots, \mathbf{k}_n$. The more similar one of the key vectors $\mathbf{k}_1, \ldots \mathbf{k}_n$ is to a query vector $\mathbf{q}_j$, the more important is the corresponding value vector $\mathbf{v}_j$ for the output vector $\mathbf{x''}_j$. More specifically, an output vector $\mathbf{x''}_j$ is defined as the weighted sum of all value vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ plus the input vector $\mathbf{x'}_j$. Thereby, the weights are proportional to the cosine similarity between $\mathbf{q}_j$ and the respective key vectors $\mathbf{k}_1, \ldots, \mathbf{k}_n$, which is mathematically expressed by $\textbf{Softmax}(\mathbf{K}_{1:n}^\intercal \cdot \mathbf{q}_j)$ as illustrated in the equation below.

Let's illustrate the bi-directional self-attention layer for one of the query vectors of our example above. For simplicity, it is assumed that our exemplary transformer-based encoder uses only a single attention head and that no normalization is applied.

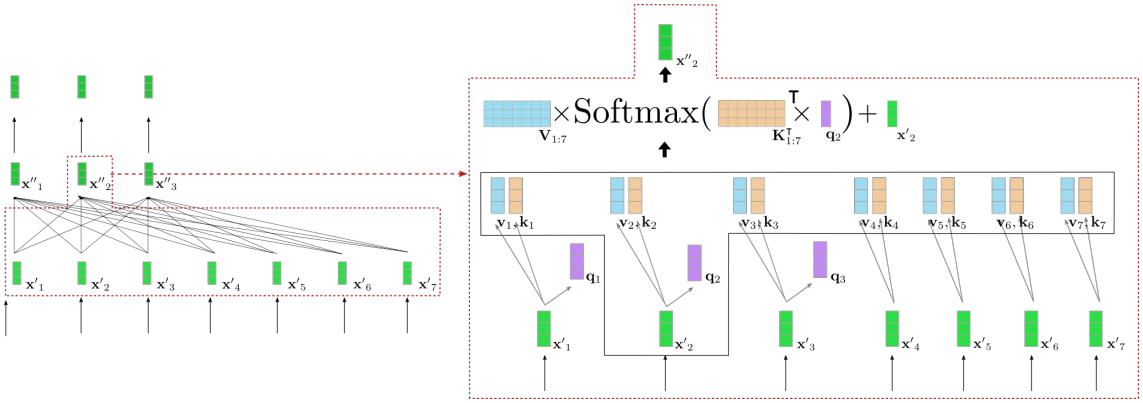

Figure 6.5. *Bi-directional self-attention layer of the transformer-based encoder module.*

On the left, the previously illustrated second encoder block is shown again and on the right, an in detail visualization of the bi-directional self-attention mechanism is given for the second input vector $\mathbf{x'}_2$ that corresponds to the input word "want". At first all input vectors $\mathbf{x'}_1, \ldots, \mathbf{x'}_7$ are projected to their respective query vectors $\mathbf{q}_1, \ldots, \mathbf{q}_7$ (only the first three query vectors are shown in purple above), value vectors $\mathbf{v}_1, \ldots, \mathbf{v}_7$ (shown in blue), and key vectors $\mathbf{k}_1, \ldots, \mathbf{k}_7$ (shown in orange). The query vector $\mathbf{q}_2$ is then multiplied by the transpose of all key vectors, i.e. $\mathbf{K}_{1:7}^\intercal$ followed by the softmax operation to yield the self-attention weights. The self-attention weights are finally multiplied by the respective value vectors and the input vector $\mathbf{x'}_2$ is added to output the "refined" representation of the word "want", i.e. $\mathbf{x''}_2$ (shown in dark green on the right). The whole equation is illustrated in the upper part of the box on the right. The multiplication of $\mathbf{K}_{1:7}^\intercal$ and $\mathbf{q}_2$ thereby makes it possible to compare the vector representation of "want" to all other input vector representations "I", "to", "buy", "a", "car", "EOS" so that the self-attention weights mirror the importance each of the other input vector representations $\mathbf{x'}_j$, with $j \neq 2$ for the refined representation $\mathbf{x''}_2$ of the word "want".

To further understand the implications of the bi-directional self-attention layer, let's

assume the following sentence is processed: "The house is beautiful and well located in the middle of the city where it is easily accessible by public transport". The word "it" refers to "house", which is 12 "positions away". In transformer-based encoders, the bi-directional self-attention layer performs a single mathematical operation to put the input vector of "house" into relation with the input vector of "it" (compare to the first illustration of this section). In contrast, in an RNN-based encoder, a word that is 12 "positions away", would require at least 12 mathematical operations meaning that in an RNN-based encoder a linear number of mathematical operations are required. This makes it much harder for an RNN-based encoder to model long-range contextual representations.

Also, it becomes clear that a transformer-based encoder is much less prone to lose important information than an RNN-based encoder-decoder model because the sequence length of the encoding is kept the same, i.e. $\textbf{len}(\mathbf{X}_{1:n}) = \textbf{len}(\overline{\mathbf{X}}_{1:n}) = n$, while an RNN compresses the length from $\textbf{len}((\mathbf{X}_{1:n}) = n$ to just $\textbf{len}(\mathbf{c}) = 1$, which makes it very difficult for RNNs to effectively encode long-range dependencies between input words.

In addition to making long-range dependencies more easily learnable, we can see that the Transformer architecture is able to process text in parallel.Mathematically, this can easily be shown by writing the self-attention formula as a product of query, key, and value matrices:

$$\mathbf{X''}_{1:n} = \mathbf{V}_{1:n}\text{Softmax}(\mathbf{Q}_{1:n}^{\intercal}\mathbf{K}_{1:n}) + \mathbf{X'}_{1:n}$$

The output $\mathbf{X''}_{1:n} = \{\mathbf{x''}_1, \ldots, \mathbf{x''}_n\}$ is computed via a series of matrix multiplications and a softmax operation, which can be parallelized effectively. Note, that in an RNN-based encoder model, the computation of the hidden state $\mathbf{c}$ has to be done sequentially: Compute hidden state of the first input vector $\mathbf{x}_1$, then compute the hidden state of the second input vector that depends on the hidden state of the first hidden vector, etc. The sequential nature of RNNs prevents effective parallelization and makes them much more inefficient compared to transformer-based encoder models on modern GPU hardware.

Now there should be a better understanding of how transformer-based encoder models effectively model long-range contextual representations and how they efficiently process long sequences of input vectors.

6.5 Decoder

As mentioned in the section 6.3 (Encoder-Decoder), the transformer-based decoder defines the conditional probability distribution of a target sequence given the contextualized encoding sequence:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n})$$

which by Bayes' rule can be decomposed into a product of conditional distributions of the next target vector given the contextualized encoding sequence and all previous target vectors:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m}|\overline{\mathbf{X}}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{dec}}(\mathbf{y}_i|\mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n})$$

Let's first understand how the transformer-based decoder defines a probability distri-

bution. The transformer-based decoder is a stack of decoder blocks followed by a dense layer, the "LM head". The stack of decoder blocks maps the contextualized encoding sequence $\overline{\mathbf{X}}_{1:n}$ and a target vector sequence prepended by the BOS vector and cut to the last target vector, i.e. $\mathbf{Y}_{0:i-1}$, to an encoded sequence of target vectors $\overline{\mathbf{Y}}_{0:i-1}$. Then, the "LM head" maps the encoded sequence of target vectors $\overline{\mathbf{Y}}_{0:i-1}$ to a sequence of logit vectors $\mathbf{L}_{1:n} = \mathbf{l}_1, \ldots, \mathbf{l}_n$, whereas the dimensionality of each logit vector $\mathbf{l}_i$ corresponds to the size of the vocabulary. This way, for each $i \in \{1, \ldots, n\}$ a probability distribution over the whole vocabulary can be obtained by applying a softmax operation on $\mathbf{l}_i$. These distributions define the conditional distribution:

$$p_{\partial_{dec}}(\mathbf{y}_i | \mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n}), \forall i \in \{1, \ldots, n\},$$

respectively. The "LM head" is often tied to the transpose of the word embedding matrix, i.e. $\mathbf{W}_{\text{emb}}^{\intercal} = \left[\mathbf{y}^1, \ldots, \mathbf{y}^{\text{vocab}}\right]^{\intercal}$[8]. Intuitively this means that for all $i \in \{0, \ldots, n-1\}$ the "LM Head" layer compares the encoded output vector $\overline{\mathbf{y}}_i$ to all word embeddings in the vocabulary $\mathbf{y}^1, \ldots, \mathbf{y}^{\text{vocab}}$ so that the logit vector $\mathbf{l}_{i+1}$ represents the similarity scores between the encoded output vector and each word embedding. The softmax operation simply transformers the similarity scores to a probability distribution. For each $i \in \{1, \ldots, n\}$, the following equations hold:

$$\begin{aligned} & p_{\partial_{dec}}(\mathbf{y} | \overline{\mathbf{X}}_{1:n}, \mathbf{Y}_{0:i-1}) \\ &= \text{Softmax}(f_{\partial_{\text{dec}}}(\overline{\mathbf{X}}_{1:n}, \mathbf{Y}_{0:i-1})) \\ &= \text{Softmax}(\mathbf{W}_{\text{emb}}^{\intercal} \overline{\mathbf{y}}_{i-1}) \\ &= \text{Softmax}(\mathbf{l}_i). \end{aligned}$$

Putting it all together, in order to model the conditional distribution of a target vector sequence $\mathbf{Y}_{1:m}$, the target vectors $\mathbf{Y}_{1:m-1}$ prepended by the special BOS vector, i.e. $\mathbf{y}_0$, are first mapped together with the contextualized encoding sequence $\overline{\mathbf{X}}_{1:n}$ to the logit vector sequence $\mathbf{L}_{1:m}$. Consequently, each logit target vector $\mathbf{l}_i$ is transformed into a conditional probability distribution of the target vector $\mathbf{y}_i$ using the softmax operation. Finally, the conditional probabilities of all target vectors $\mathbf{y}_1, \ldots, \mathbf{y}_m$ multiplied together to yield the conditional probability of the complete target vector sequence:

$$p_{\partial_{dec}}(\mathbf{Y}_{1:m} | \overline{\mathbf{X}}_{1:n}) = \prod_{i=1}^{m} p_{\partial_{dec}}(\mathbf{y}_i | \mathbf{Y}_{0:i-1}, \overline{\mathbf{X}}_{1:n}).$$

In contrast to transformer-based encoders, in transformer-based decoders, the encoded output vector $\overline{\mathbf{y}}_i$ should be a good representation of the next target vector $\mathbf{y}_{i+1}$ and not of the input vector itself. Additionally, the encoded output vector $\overline{\mathbf{y}}_i$ should be conditioned on all contextualized encoding sequence $\overline{\mathbf{X}}_{1:n}$. To meet these requirements each decoder block consists of a *uni-directional* self-attention layer, followed by a *cross-*

[8]The word embedding matrix $\mathbf{W}_{\text{emb}}$ gives each input word a unique *context-independent* vector representation. This matrix is often fixed as the "LM Head" layer. However, the "LM Head" layer can very well consist of a completely independent "encoded vector-to-logit" weight mapping.

attention layer and two feed-forward layers. The *uni-directional* self-attention layer puts each of its input vectors $\mathbf{y'}_j$ only into relation with all previous input vectors $\mathbf{y'}_i$, with $i \leq q$ for all $j \in \{1, \ldots, n\}$ to model the probability distribution of the next target vectors. The *cross-attention* layer puts each of its input vectors $\mathbf{y''}_j$ into relation with all contextualized encoding vectors $\overline{\mathbf{X}}_{1:n}$ to condition the probability distribution of the next target vectors on the input of the encoder as well.

Let's now visualize the transformer-based decoder for our English to German translation example.

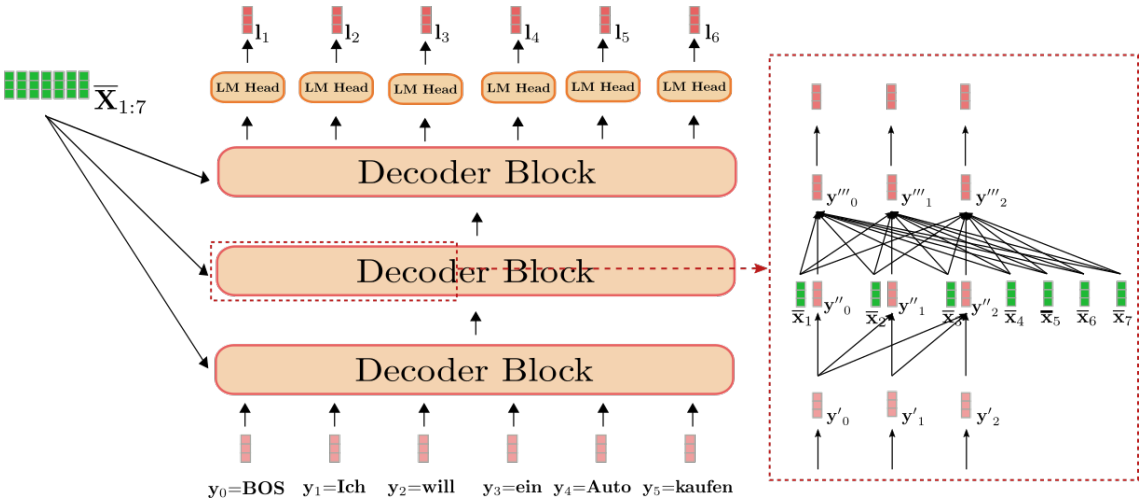

Figure 6.6. *Decoding process of the transformer-based decoder module.*

As can be seen in Figure 6.6, the decoder maps the input $\mathbf{Y}_{0:5}$ "BOS", "Ich", "will", "ein", "Auto", "kaufen" (shown in light red) together with the contextualized sequence of "I", "want", "to", "buy", "a", "car", "EOS", i.e. $\overline{\mathbf{X}}_{1:7}$ (shown in dark green) to the logit vectors $\mathbf{L}_{1:6}$ (shown in dark red).

Applying a softmax operation on each $\mathbf{l}_1, \mathbf{l}_2, \ldots, \mathbf{l}_5$ can thus define the conditional probability distributions:

$$p_{\theta_{dec}}(\mathbf{y}|\text{BOS}, \overline{\mathbf{X}}_{1:7}),$$

$$p_{\theta_{dec}}(\mathbf{y}|\text{BOS Ich}, \overline{\mathbf{X}}_{1:7}),$$

$$\ldots,$$

$$p_{\theta_{dec}}(\mathbf{y}|\text{BOS Ich will ein Auto kaufen}, \overline{\mathbf{X}}_{1:7}).$$

The overall conditional probability of:

$$p_{\theta_{dec}}(\text{Ich will ein Auto kaufen EOS}|\overline{\mathbf{X}}_{1:n})$$

can therefore be computed as the following product:

$$p_{\theta_{dec}}(\text{Ich}|\text{BOS}, \overline{\mathbf{X}}_{1:7}) \times \ldots \times p_{\theta_{dec}}(\text{EOS}|\text{BOS Ich will ein Auto kaufen}, \overline{\mathbf{X}}_{1:7}).$$

The red box on the right shows a decoder block for the first three target vectors

$\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2$. In the lower part, the *uni-directional* self-attention mechanism is illustrated and in the middle, the *cross-attention* mechanism is illustrated. Let's first focus on *uni-directional* self-attention.

As in bi-directional self-attention, in *uni-directional* self-attention, the query vectors $\mathbf{q}_0, \ldots, \mathbf{q}_{m-1}$ (shown in purple below), key vectors $\mathbf{k}_0, \ldots, \mathbf{k}_{m-1}$ (shown in orange below), and value vectors $\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}$ (shown in blue below) are projected from their respective input vectors $\mathbf{y'}_0, \ldots, \mathbf{y}_{m-1}$ (shown in light red below). However, in *uni-directional* self-attention, each query vector $\mathbf{q}_i$ is compared only to its respective key vector and all previous ones, namely $\mathbf{k}_0, \ldots, \mathbf{k}_i$ to yield the respective attention weights. This prevents an output vector $\mathbf{y''}_j$ (shown in dark red below) to include any information about the following input vector $\mathbf{y}_i$, with $i > 1$ for all $j \in \{0, \ldots, m-1\}$. As is the case in bi-directional self-attention, the attention weights are then multiplied by their respective value vectors and summed together.

We can summarize *uni-directional* self-attention as follows:

$$\mathbf{y''}_i = \mathbf{V}_{0:i}\mathbf{Softmax}(\mathbf{K}_{0:i}^\intercal \mathbf{q}_i) + \mathbf{y'}_i.$$

Note that the index range of the key and value vectors is $0 : i$ instead of $0 : m - 1$ which would be the range of the key vectors in bi-directional self-attention.

Let's illustrate *uni-directional* self-attention for the input vector $\mathbf{y'}_1$ for our example above.

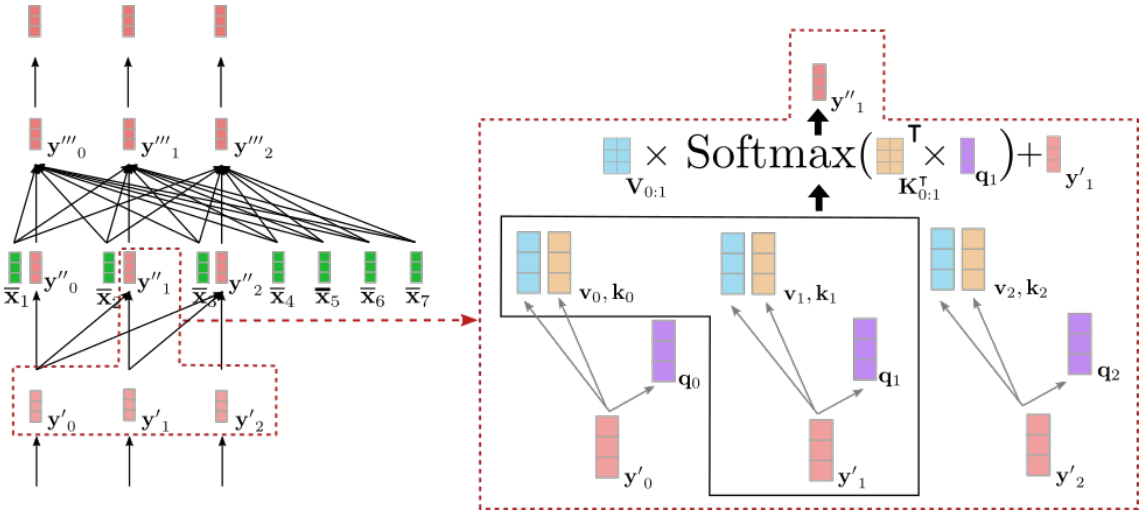

Figure 6.7. *Uni-directional self-attention layer of the transformer-based decoder module.*

As can be seen in Figure 6.7, $\mathbf{y''}_1$ only depends on $\mathbf{y'}_0$ and $\mathbf{y'}_1$. Therefore, we put the vector representation of the word "Ich", i.e. $\mathbf{y'}_1$ only into relation with itself and the "BOS" target vector, i.e. $\mathbf{y'}_0$, but not with the vector representation of the word "will", i.e. $\mathbf{y'}_2$.

So why is it important that we use *uni-directional* self-attention in the decoder instead of bi-directional self-attention? As stated above, a transformer-based decoder defines a mapping from a sequence of input vector $\mathbf{Y}_{0:m-1}$ to the logits corresponding to the next decoder input vectors, namely $\mathbf{L}_{1:m}$. In our example, this means, e.g. that the input vector $\mathbf{y}_1$ = "Ich" is mapped to the logit vector $\mathbf{l}_2$, which is then used to predict the input vector

$\mathbf{y}_2$. Thus, if $\mathbf{y'}_1$ would have access to the following input vectors $\mathbf{Y'}_{2:5}$, the decoder would simply copy the vector representation of "will", i.e. $\mathbf{y'}_2$, to be its output $\mathbf{y''}_1$. This would be forwarded to the last layer so that the encoded output vector $\overline{\mathbf{y}}_1$ would essentially just correspond to the vector representation $\mathbf{y}_2$.

This is obviously disadvantageous as the transformer-based decoder would never learn to predict the next word given all previous words, but just copy the target vector $\mathbf{y}_i$ through the network to $\overline{\mathbf{y}}_{i-1}$ for all $i \in \{1, \ldots, m\}$. In order to define a conditional distribution of the next target vector, the distribution cannot be conditioned on the next target vector itself. It does not make much sense to predict $\mathbf{y}_i$ from $p(\mathbf{y}|\mathbf{Y}_{0:i}, \overline{\mathbf{X}})$ because the distribution is conditioned on the target vector it is supposed to model. The *uni-directional* self-attention architecture, therefore, allows us to define a causal probability distribution, which is necessary to effectively model a conditional distribution of the next target vector.

Great! Now we can move to the layer that connects the encoder and decoder - the *cross-attention* mechanism!

The *cross-attention* layer takes two vector sequences as inputs: the outputs of the *uni-directional* self-attention layer, i.e. $\mathbf{Y''}_{0:m-1}$ and the contextualized encoding vectors $\overline{\mathbf{X}}_{1:n}$. As in the self-attention layer, the query vectors $\mathbf{q}_0, \ldots, \mathbf{q}_{m-1}$ are projections of the output vectors of the previous layer, i.e. $\mathbf{Y''}_{0:m-1}$. However, the key and value vectors $\mathbf{k}_0, \ldots, \mathbf{k}_{m-1}$ and $\mathbf{v}_0, \ldots, \mathbf{v}_{m-1}$ are projections of the contextualized encoding vectors $\overline{\mathbf{X}}_{1:n}$. Having defined key, value, and query vectors, a query vector $\mathbf{q}_i$ is then compared to all key vectors and the corresponding score is used to weight the respective value vectors, just as is the case for bi-directional self-attention to give the output vector $\mathbf{y'''}_i$ for all $i \in \{0, \ldots, m-1\}$. *Cross-attention* can be summarized as follows:

$$\mathbf{y'''}_i = \mathbf{V}_{1:n}\mathbf{Softmax}(\mathbf{K}_{1:n}^\intercal \mathbf{q}_i) + \mathbf{y''}_i$$

Note that the index range of the key and value vectors is $1 : n$ corresponding to the number of contextualized encoding vectors.

Let's visualize the *cross-attention* mechanism for the input vector $\mathbf{y''}_1$ for our example above.

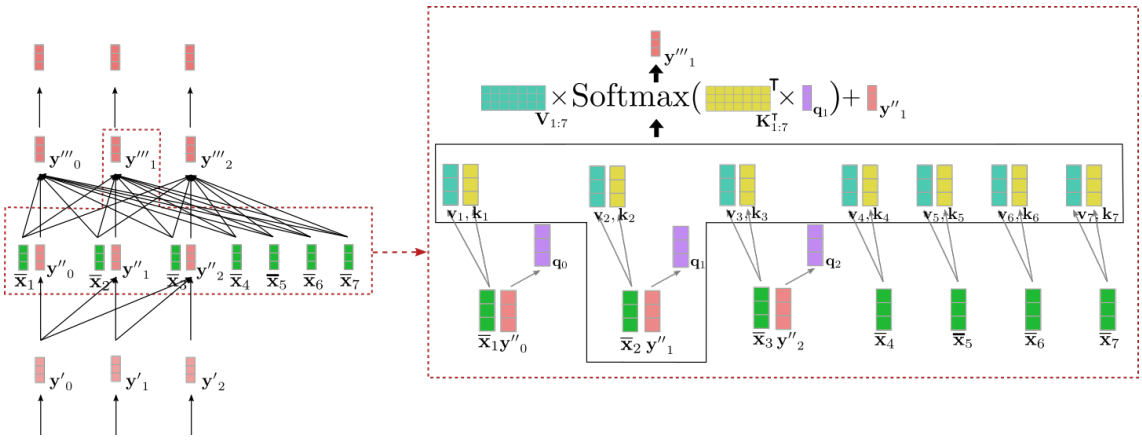

Figure 6.8. *Cross-attention layer of the transformer-based decoder module.*

As can be seen in Figure 6.8, the query vector $\mathbf{q}_1$ (shown in purple) is derived from $\mathbf{y''}_1$ and therefore relies on a vector representation of the word "Ich". The query vector $\mathbf{q}_1$ (shown in red) is then compared to the key vectors $\mathbf{k}_1, \ldots, \mathbf{k}_7$ (shown in yellow) corresponding to the contextual encoding representation of all encoder input vectors $\mathbf{X}_{1:n}$ = "I want to buy a car EOS". This puts the vector representation of "Ich" into direct relation with all encoder input vectors. Finally, the attention weights are multiplied by the value vectors $\mathbf{v}_1, \ldots, \mathbf{v}_7$ (shown in turquoise) to yield in addition to the input vector $\mathbf{y''}_1$ the output vector $\mathbf{y'''}_1$ (shown in dark red).

What happens here is that each output vector $\mathbf{y'}_i$ is a weighted sum of all value projections of the encoder inputs $\mathbf{v}_1, \ldots, \mathbf{v}_7$ plus the input vector itself $\mathbf{y}_i$ (c.f. illustrated formula above). The key mechanism to understand is the following: Depending on how similar a query projection of the input decoder vector $\mathbf{q}_i$ is to a key projection of the encoder input vector $\mathbf{k}_j$, the more important is the value projection of the encoder input vector $\mathbf{v}_j$. In loose terms this means, the more "related" a decoder input representation is to an encoder input representation, the more does the input representation influence the decoder output representation.

Now we can see how this architecture nicely conditions each output vector $\mathbf{y'''}_i$ on the interaction between the encoder input vectors $\overline{\mathbf{X}}_{1:n}$ and the input vector $\mathbf{y''}_i$. Another important observation at this point is that the architecture is completely independent of the number n of contextualized encoding vectors $\overline{\mathbf{X}}_{1:n}$ on which the output vector $\mathbf{y'''}_i$ is conditioned on. All projection matrices $\mathbf{W}_k^{\text{cross}}$ and $\mathbf{W}_v^{\text{cross}}$ to derive the key vectors $\mathbf{k}_1, \ldots, \mathbf{k}_n$ and the value vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ respectively are shared across all positions $1, \ldots, n$ and all value vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are summed together to a single weighted averaged vector. Now it becomes obvious as well, why the transformer-based decoder does not suffer from the long-range dependency problem, the RNN-based decoder suffers from. Because each decoder logit vector is directly dependent on every single encoded output vector, the number of mathematical operations to compare the first encoded output vector and the last decoder logit vector amounts essentially to just one.

To conclude, the *uni-directional* self-attention layer is responsible for conditioning each output vector on all previous decoder input vectors and the current input vector and the *cross-attention* layer is responsible to further condition each output vector on all encoded input vectors.

Chapter 7

Adversarial Attacks

7.1 Introduction

An *adversarial attack* on a DNN model refers to the systematic procedure of generating *adversarial examples*, that is carefully crafted inputs aiming to fool the target model. Specifically, adversarial examples have the key feature that they are crafted by adding **imperceptible perturbation (noise)** to the original input [9] and for this reason the terms adversarial examples and *adversarial perturbations* are used interchangeably in the present thesis. These slightly perturbed inputs may seem innocent, but often lead the model to output incorrect predictions with high confidence score [1]. It becomes clear that adversarial attacks are malicious attacks that can effectively decrease the performance and expose crucial vulnerabilities of state-of-the-art models used in several tasks [10, 11, 12, 13, 14, 15, 16, 17]. In other words, adversarial attacks pose a significant threat to crucial aspects of machine learning systems, such as security, transparency and credibility and therefore the need to establish counter-measures against adversarially crafted inputs is greater than ever [37].

7.2 Examples of Adversarial Attacks

Before we move on to further details about an adversarial attack, we present some examples of adversarial attacks on the two most well-known machine learning areas: Computer Vision (image domain) and NLP (text domain). These examples aim to provide a fundamental intuition of adversarial attacks and to illustrate the fact that the added perturbation is practically imperceptible to human, yet it can lead the model to generate erroneous predictions with high confidence score.

7.2.1 Computer Vision

Figure 7.1 below depicts an adversarial attack instance to an image classifier. Specifically, the target model is GoogLeNet which produces state-of-the-art results on the ImageNet classification challenge [18, 19]. We observe that superimposing a tiny (but deliberate) amount of noise causes the model to misclassify the panda as a gibbon with an extremely high confidence score (99.3%). We can also verify that the perturbed image is practically indistinguishable from the original.

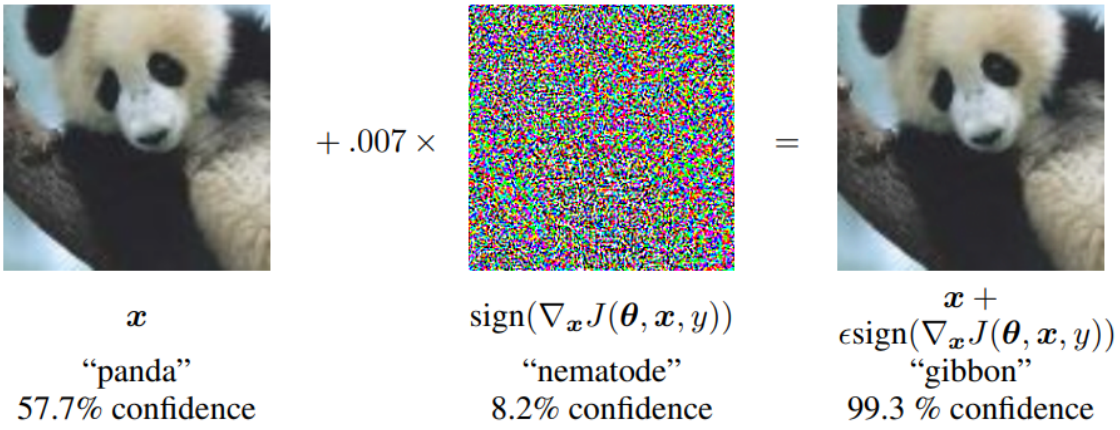

Figure 7.1. *Panda is misclassified as gibbon, after adding practically unnoticable perturbation to the original image (Image credit: Goodfellow et al. [1])*

7.2.2 Natural Language Processing

No natural analogy to the adversarial examples in Computer Vision exists for NLP. To be exact, the above example illustrates the fact that the original and perturbed image are literally indistinguishable to the human eye. However, how can two sequences of text be truly identical without being the same?

It has been already mentioned that the key feature of a valid adversarial example is its high *similarity* to the original input. In the image domain, we saw that it is possible to craft adversarial images that look exactly the same as the original, so there was no need to dive deeper into the notion of similarity, because the similarity was obviously visual. When it comes to the text domain, we must distinguish the notion of similarity into two categories:

- **Visual similarity** The perturbed text sequence *looks very similar* to the original input. Typically, adversarial attacks that retain visual similarity attempt to change as few characters as possible or insert human-like typos in the original sequence (see Table 7.1 below).

- **Semantic similarity** The perturbed text sequence is *semantically indistinguishable* from the original input. Typically, adversarial attacks that retain semantic similarity attempt to paraphrase the original input or perform synonyms substitution (see Table 7.1 below).

In the present thesis, we focus on adversarial attacks in NLP and specifically on those that preserve the meaning of the original sequence and generate grammatically and syntactically valid text. This type of attacks is suitable for the NLI task, because we are interested in recognizing textual entailment only in meaningful and semantically coherent sentences.

Adversarial attack examples in NLP	
WordBug (visual similarity)	
Original input (label: POS)	This film has a special place in my heart
Adversarial input (label: NEG)	This film has a special plcae in my herat
TextFooler (semantic similarity)	
Original input (label: NEG)	The characters are totally estranged from reality
Adversarial input (label: POS)	The characters are fully estranged from reality

Table 7.1. *WordBug attack recipe (which aims to retain visual similarity) manages to flip the classifier's output label by inserting only 2 typos in the original sequence [2]. Similarly, TextFooler attack recipe (which aims to retain semantic similarity) manages to fool the classifier by replacing the word "totally" with its exact synonym word "fully" [3]. (POS: positive review, NEG: negative review)*

7.3 Basic Concepts and Definitions

This section gives the fundamental preliminary knowledge of adversarial attacks, including formula descriptions, basic taxonomy of adversarial attacks as well as metrics that measure a model's robustness against adversarial attacks.

7.3.1 Formula Descriptions

Section 7.2 provided a basic intuitive understanding of an adversarial attack, therefore we can now introduce the proper notation that formally describes the concept of adversarial attacks.

An adversarial attack can be fully defined by first defining its fundamental components. As mentioned above, an adversarial attack on a *DNN model* refers to the systematic procedure of generating *adversarial examples*, which means that the target *DNN model* and *adversarial examples* can be recognized as the integral components of an adversarial attack. In addition, the present thesis introduces counter-measures and defense mechanisms against adversarial attacks in NLI models, therefore we need to define the concept of *robustness*. The definitions of DNNs[1], adversarial examples and robustness are given below [9]:

DNN: A typical DNN can be represented as the function $F : X \rightarrow Y$, which maps an input set X to a label set Y. Y is a set of k classes, typically $Y = \{0, 1, ..., k-1\}$. A sample $x \in X$ is correctly classified by F to the ground-truth label y if and only if $F(x) = y$.

Adversarial Example: An attacker's goal is to add a small perturbation (noise) ϵ in sample x to create the perturbed input $x' = x + \epsilon$, such that $F(x') = y' \neq y$, where y is the ground truth label of x (that was also originally predicted by the DNN). Apart from

[1] In our work, we deal with adversarial attacks on NLI models. However, NLI is a sub-task of the classification task in machine learning, therefore the formal definition that is given for DNNs implicitly concerns DNN classifiers.

being misclassified by the DNN, the noise superimposed to x should be imperceptible to humans. To achieve this goal, the perturbation ϵ must not exceed a predefined threshold δ, i.e. $\|\epsilon\| < \delta$. Moreover, several imperceptibility metrics are adopted which are defined in section 7.3.3.

Robustness: A robust DNN model resists an adversarial attack by correctly classifying the adversarial examples x' generated by this attack. Hence, the predicted label that corresponds to x' should be y rather than y' in a robust DNN model, i.e. $F(x') = y$, where y is the label of x that was originally predicted by the DNN. The defense methods to enhance the robustness of models should tackle a wide range of perturbations ϵ effectively.

7.3.2 Classification of Adversarial Attacks

There are three prominent criterias based on which adversarial attacks can be categorized. These criterias are listed below:

- **Adversary's[2] goal** What is the goal or purpose of the attacker? Do they want to misguide the DNN's decision on one sample or influence the overall performance of the DNN model? [10]
- **Adversary's knowledge** What information is available to the attacker? Do they know the DNN's structure, its parameters or the training set used for the DNN's training? [10]
- **Perturbation unit** What is the basic unit within the input sample (word, character, sentence) that gets perturbed? (*applies only to text domain*) [9]

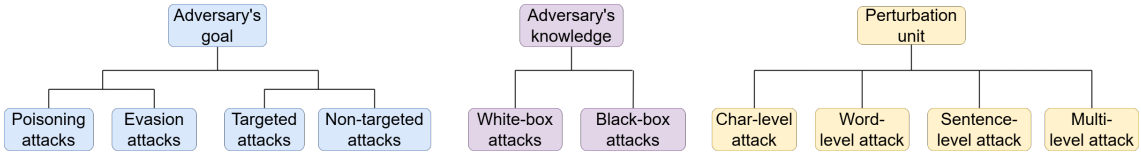

Figure 7.2. *Classification of adversarial attacks*

7.3.2.1 Adversary's goal

- **Poisoning attacks vs Evasion attacks**
  Poisoning attacks refer to the attacking algorithms that allow an attacker to insert fake samples into the training database of a DNN algorithm or modify legitimate training samples. These fake (*poisoned*) samples can have a significant impact on the performance of the trained classifier, e.g. they can result in the poor accuracy on test samples. This type of attacks frequently appears in the situation where the adversary has access to the training database. For example, web-based repositories often collect malware examples for training, which provides an opportunity for adversaries to poison the data [10].

[2]The term "adversary" refers to the attacker, i.e. the person that carries out the adversarial attack.

In **evasion attacks**, the classifiers are fixed and usually have good performance on benign testing samples. The adversaries do not have authority to change the classifier or its parameters, but they craft some fake samples that the classifier cannot recognize. In other words, the adversaries generate some fraudulent examples to evade detection by the classifier [10]. For example, in autonomous driving vehicles, sticking a few pieces of tapes on the stop signs can confuse the vehicle's road sign recognizer [20].

- **Targeted attacks vs Non-targeted attacks**
  In a **targeted attack**, when the victim sample (x, y) is given, where x is feature vector and $y \in Y$ is the ground truth label of x, the adversary aims to induce the classifier to give a specific label $t \in Y$ to the perturbed sample x'. For example, a fraudster is likely to attack a financial company's credit evaluation model to disguise himself as a highly credible client of this company [10].

  In a **non-targeted attack**, there is no specified target label t for the victim sample x and the adversary only wants the classifier to predict incorrectly [10].

7.3.2.2 Adversary's knowledge

- **White-box attacks**
  In a white-box attack setting, the adversary has access to all the information of the target DNN model, including its architecture, parameters, gradients etc. Therefore, the adversary can make full use of the network information to carefully craft adversarial examples. White-box attacks have been extensively studied because the disclosure of model architecture and parameters helps people understand the weaknesses of DNN models. According to Tramèr et al. [21], security against white-box attacks is the property that we desire ML models to have [10].

- **Black-box attacks**
  In a black-box attack setting, the inner configuration of DNN models is unavailable to adversaries. Adversaries can only feed the input data and query the outputs of the models. They usually attack the models by keeping feeding samples to the box and observing the output to exploit the model's input-output relationship. This setting can expose the weaknesses of a model, because it allows the adversary to identify potential shallow statistic properties between input and output. Compared to white-box attacks, black-box attacks are more practical in applications because model designers usually do not publicly release their model parameters for proprietary reasons [10].

7.3.2.3 Perturbation unit

In the text domain, adversarial attacks can be classified as char-level, word-level, sentence-level, and multi-level according to the perturbation units in generating adversarial examples.

- **Char-level attacks**
  Char-level attacks indicate that adversaries modify several characters in words to generate adversarial examples that can fool the detectors. Specifically, the modifications are mostly misspellings and the common operations include insertion, swap, deletion and flip [9].

- **Word-level attacks**
  Word-level attacks involve various word perturbations. Attackers generate adversarial examples by inserting, replacing, or deleting certain words in various manners [9].

- **Sentence-level attacks**
  Sentence-level attacks usually insert a sentence into a text or rewrite the sentence while maintaining its meanings [9].

- **Multi-level attacks**
  Multi-level attacks incorporate more than one of the three perturbation attacks mentioned above to achieve the imperceptible and high success rate attack [9].

7.3.3 Imperceptibility Metrics for Text Inputs

Adversarial examples are crafted by adding imperceptible perturbations into legitimate inputs to incur erroneous output labels [22]. In the image domain, various metrics are adopted to measure the imperceptibility of adversarial examples. L_p norm is the most commonly used method defined as

$$||\Delta x||_p = \sqrt[p]{\sum_{i=1}^{n} |x_i' - x_i|^p} \tag{7.1}$$

where:

- p represents the type of distance that is used to measure the resemblance of the original sample x and the perturbed sample x'. For example, $p = 2$ indicates the well-known euclidean distance
- $||\Delta x||_p$ represents the perturbation magnitude (added noise)
- n represents the dimensionality of x and x', i.e. both x and x' have n features
- x_i represents the i-th feature of the original sample x, $i = 1, 2, ..., n$
- x_i' represents the i-th feature of the perturbed sample x', $i = 1, 2, ..., n$

Nevertheless, it is not possible to use the L_p norm for accurately measuring imperceptibility in text inputs. In the image domain, one can introduce a pixel-level perturbation without actually changing how the image appears to the human eye (see figure 7.1). This is not applicable to the text domain, because every perturbation is visible to humans, due to the introduced grammatical and spelling errors. For the purposes of our thesis, an adversarial attack in the text domain is successful (in terms of imperceptibility) if:

- No obvious errors are observed by the human eye [9].
- The crafted adversarial text conveys the same semantic meaning as the original one [9].
- The model output on the adversarial text and the legitimate input is different, which means an erroneous output has occurred [9].

Thus, the majority of metrics adopted in images cannot be directly applied for evaluating the quality of adversarially crafted text sequences due to the inherently different perturbation mechanisms in these two types of data. Next, we present the metrics employed for measuring the imperceptibility of adversarial texts [9].

- **Euclidean distance** [23]. Euclidean distance refers to the similarity between two **vectors** by measuring their distance in the Euclidean space. Specifically, given two word vectors $\overrightarrow{u}, \overrightarrow{v}$ of dimensionality n, i.e. $\overrightarrow{u} = (u_1, u_2, ..., u_n)$ and $\overrightarrow{v} = (v_1, v_2, ..., v_n)$, the Euclidean distance ED of these two vectors is defined as

$$ED = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + ... + (u_n - v_n)^2} \tag{7.2}$$

  The lower the distance, the more similar the vectors.

- **Edit distance** [24]. Edit distance refers to the similarity between two **strings** by calculating the number of editing operations required to convert one string to another. *Levenshtein distance* [25] is a widely used edit distance. For two strings a and b, the Levenshtein distance lev is calculated by

$$lev(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} lev(i-1,j) + 1 \\ lev(i,j-1) + 1 \\ lev(i-1,j-1) + 1_{a_i \neq b_i} \end{cases} & \text{otherwise} \end{cases} \tag{7.3}$$

  where $lev(i,j)$ is the distance between the first i characters in a and the first j characters in b. The lower the Levenshtein distance, the more similar the two strings.

- **Cosine similarity** [3]. Cosine similarity refers to the similarity between two **vectors** by measuring the cosine of the angle between them. Specificlly, given two word vectors $\overrightarrow{u}, \overrightarrow{v}$ of dimensionality n, i.e. $\overrightarrow{u} = (u_1, u_2, ..., u_n)$ and $\overrightarrow{v} = (v_1, v_2, ..., v_n)$, the cosine similarity CD is calculated by

$$CD = \frac{\overrightarrow{u} \cdot \overrightarrow{v}}{\|\overrightarrow{u}\| \cdot \|\overrightarrow{v}\|} = \frac{\sum_{i=1}^{n} u_i \times v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}} \tag{7.4}$$

Compared to Euclidean distance, the cosine similarity pays more attention to the difference between the directions of the two vectors. The more consistent their directions, the more similar the vectors.

- **Jaccard similarity coefficient** [26]. The Jaccard similarity coefficient is used to compare the similarity between two objects, that are represented as sets. For two given sets A and B, the Jaccard similarity coefficient $J(A, B)$ is calculated by

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{7.5}$$

where $0 \leq J(A, B) \leq 1$. The closer the value of $J(A, B)$ is to 1, the more similar the two sets are. In the text domain, intersection $A \cap B$ is the set that contains words present in both text sequences, whereas union $A \cup B$ is the set that contains words present on either text sequence (without duplication).

Chapter 8

Proposal

8.1 Problem statement

Machine Learning models that are trained to achieve state-of-the-art accuracy on held-out datasets often learn to base their predictions on shallow input statistics that are deduced at training time [4]. Therefore, a small perturbation to the input can significantly impact the predictions of such models, thus making them susceptible to adversarial attacks. For example, Jin et. al. [3] showed that attacking the state-of-the-art NLI classifier drops the accuracy from 89.4% to only 4%.

8.2 Motivation

In 2018, Camburu et. al. [4] introduced the e-SNLI dataset, which extends the SNLI dataset [27] by incoprorating human-annotated free-form natural language explanations that justify why a specific pair of sentences (premise and hypothesis) are associated with the semantic relation of entailment, contradiction or neutrality.

Camburu et. al. [4] claim that the e-SNLI dataset can be exploited in the direction of training interpretable models that provide robust explanations for their predictions. Nevertheless, no systematic research or experiments have been conducted in order to verify or reject this claim. In this work, we investigate whether the above claim holds true in terms of adversarial robustness. Specifically, we conduct a series of experiments in order to check whether NLI models that are trained with natural language explanations are more resistant to adversarial attacks.

8.3 Proposal

Our experiments strongly rely on the *ExplainThenPredict* framework introduced by Camburu et. al. [4]. According to this setup, the predicted label is not directly conditioned on the input premise and hypothesis. Instead, it is conditioned on an explanation that attempts to describe the semantic relationship between the input premise and hypothesis. The intermediate explanation is the novelty of our work and we investigate whether it can filter noise superimposed in the input sentences, therefore improving adversarial robustness.

Chapter 9

Experiments and Results

9.1 Experimental setup

9.1.1 Explanations-based setup

This setup exploits natural language explanations and actually corresponds to the ExplainThenPredict framework (see figure 9.1) introduced by Camburu et. al. [4]. According to ExplainThenPredict, the NLI task is modified as follows:

1. Given a pair of premise and hypothesis, a generative model (called *Seq2Seq* as a codename) produces a free-form natural language explanation that justifies the semantic relationship between the premise and hypothesis (entailment, contradiction or neutral).

2. This explanation is fed into a classifier (called *Expl2Label* as a codename) that predicts the output label (entailment, contradiction or neutral).

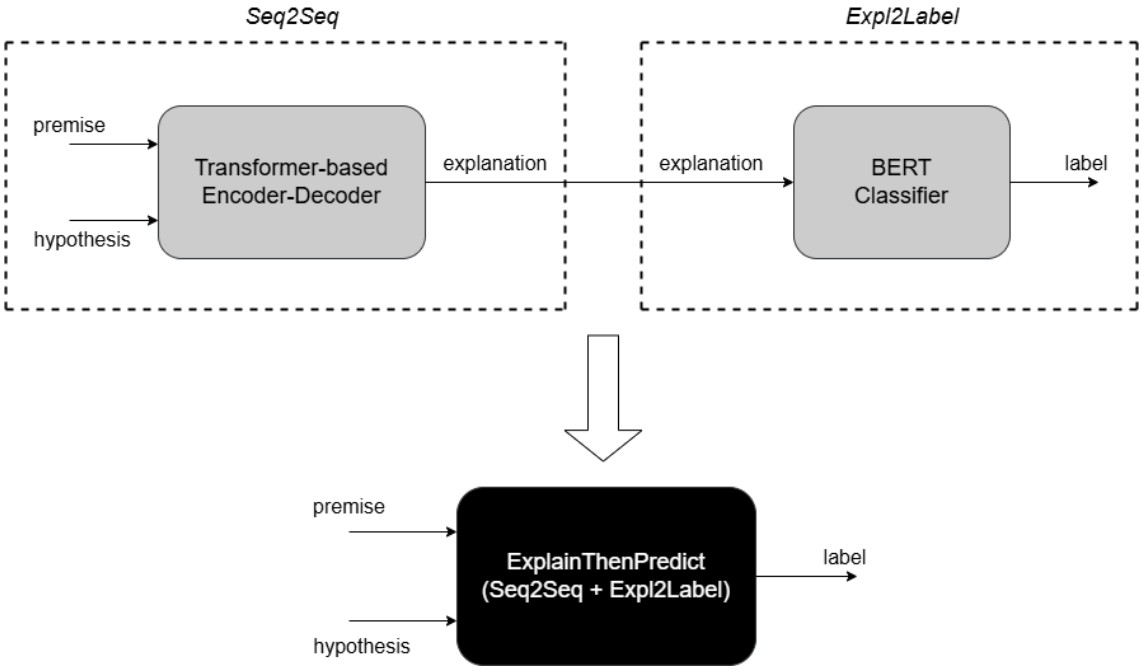

Figure 9.1. *ExplainThenPredict setup*

Figure 9.2 illustrates an exemplary inference process that follows the above setup. For demonstration purposes, we select a pair of brief and straightforward sentences from the e-SNLI dataset, i.e. *A Land Rover is being driven across a river* serves as premise and *A vehicle is crossing a river* serves as hypothesis. During inference, the generative model produces the explanation *Land Rover is a vehicle*, which correctly captures the semantic relation of the two sentences, and finally, this explanation is fed to the classifier, which, in turn, predicts the label *entailment*. This label matches the ground-truth and is conceptually aligned with the generated explanation.

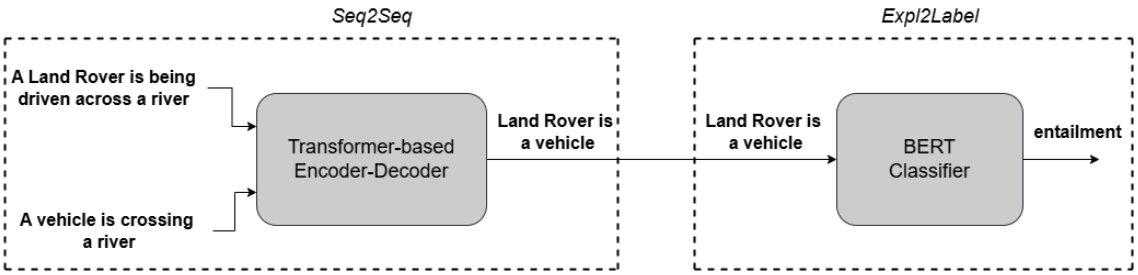

Figure 9.2. *Exemplary inference process, according to ExplainThenPredict setup*

It becomes clear that the predicted label is now conditioned on the intermediate explanation produced by the generative model, instead of directly depending on the input premise and hypothesis. Moreover, by providing the intermediate explanation as input to the explanation classifier, we can consider the combination of the two models as a black box that is fed with a pair of premise and hypothesis and predicts the corresponding label. Therefore, the modified NLI task comes down to the traditonal NLI task.

Camburu et. al. [4] claim that this is a sensible decomposition of the NLI task, as in the vast majority of cases, the explanation has a specific template depending on the semantic relation it justifies. In other words, the explanation is strongly indicative of the label it justifies.

In all the experiments that will be mentioned below, we consider the following architectures for our models:

- **Generative model:** Camburu et. al. [4] used RNN-based generative models in their experiments. We choose to differentiate by considering the transformers architecture, which is more efficient and widely used nowadays. So, our generative model is essentially a *transformers-based encoder-decoder* model (see chapter 6 for details). Specifically, for the decoder part we make use of GPT-2 and for the encoder part we make use of the most prevalent BERT variations, i.e. BERT, DistilBERT, ALBERT and RoBERTa. For example, if the encoder follows BERT architecture, we call our model *BERT2GPT*, if the encoder follows RoBERTa architecture, we call the model *ROBERTA2GPT* and so on.

- **Explanation classifier:** We use a BERT-based classifier, i.e. a BERT encoder with a classification head on top of the embeddings. This is a simple, yet effective architecture which, as will be shown later, achieves a surprisingly high accuracy.

It is worth highlighting that the novelty of our work revolves around the explanation generation which is attributed to the generative model. Consequently, our experimental analysis focuses on different architecture variations of the generative model, while keeping the simple BERT architecture as basis for the explanation classifier.

9.1.2 Explanations-free setup

This setup corresponds to the traditonal NLI task, where a classifier is fed with a pair of premise and hypothesis and predicts their semantic relation (entailment, contradiction, neutral). In this case, the output label is directly conditioned on the input premise and hypothesis. This setup does not include explanations and thus will serve as a baseline. The classifier is essentially a BERT-based classifier, i.e. a BERT encoder with a classification head on top of the embeddings.

9.2 Training and Evaluation

9.2.1 Baseline

The fine-tuning process of the baseline model is pretty straightforward. We remind that the baseline is essentially a BERT-based classifier, i.e. a BERT encoder with a classification head on top of the embeddings. At train time, both the pair of (premise, hypothesis) and the ground-truth label is provided, while at inference time the classifier predicts the label based on the input pair of premise and hypothesis. Naturally, *accuracy* is used as validation metric for the classifier. All the necessary fine-tuning details of the baseline can be found in table 9.1 below.

Fine-tuning details	
encoder checkpoint	bert-base-uncased
encoder max length	128
# epochs	5
batch size	32
# GPUs	1
GPU type	NVIDIA Volta V100
training time	~6 hours
test acc	90.13%

Table 9.1. *Fine-tuning details of the BERT-based classifier that serves as baseline.*

9.2.2 ExplainThenPredict model

In figure 9.2, the generative model and the explanation classifier appear to be interconnected provided that the output of the former serves as input for the latter. However, this is not entirely true, because the two models are trained (fine-tuned) *separately* and they become joint only at inference time. Indeed, this is a sensible approach, as the two

models serve different tasks, i.e. generative model is under the task of text generation and explanation classifier is under the task of text classification.

Having fine-tuned the generative model and the explanation classifier for their corresponding tasks, the two models can be interconnected as can be seen in figure 9.2. Therefore, we can consider this model combination as a black-box and perform inference in this black-box, where the input is now a pair of (premise, hypothesis) and the output is the label that describes their semantic relationship (entailment, contradiction, neutral).

9.2.2.1 Seq2Seq

For fine-tuning the transformer-based generative model, at training time we provide both the pair of (premise, hypothesis) and the ground-truth explanation, while at inference time the model predicts an explanation based only on the input pair of premise and hypothesis.

Now, it is of paramount importance to determine the validation metric that we will use for selecting the optimal Seq2Seq model at training time. To be able to do so, we must consider the task that our Seq2Seq models try to solve. The specific task is explanation generation, which falls under the category of *text generation*. Some popular text generation metrics are BLEU [28], METEOR [29], ROUGE [30] and BERT-score [31].

All of these metrics evaluate the predicted text against a set of ground-truth texts that are called *references*. This is in line with our task, as both validation and test e-SNLI data include *3 human-annotated explanations for each sample*. This is a key feature of the e-SNLI dataset and is actually necessary, because an explanation can be formulated in many different ways, while preserving its meaning. Therefore, the 3 explanations per sample introduce a set of versatile alternative explanations and render the values of the aforementioned text generation metrics more representative of the generated explanation quality.

However, we must also consider that METEOR and BLEU were invented to address the task of machine translation [29, 28], while ROUGE was invented to address the task of summarization [30]. In other words, METEOR, BLEU and ROUGE are task-specific text generation metrics and only BERT-score remains task-agnostic. The main problem lies in the fact that, so far, there is no metric invented exclusively for the explanation generation task, so we have to either use BERT-score, which is task-agnostic, or borrow a metric (BLEU, METEOR, ROUGE) that addresses a different task. Of course, none of these options is optimal, because the absence of a task-specific text generation metric does not allow us to fully rely on the automatic evaluation of the generated explanations quality.

For this reason, we perform a post-hoc human evaluation of the generated explanations, which is the standard methodology in the field of explanation generation and is leveraged in similar works [4, 38, 39]. Specifically, we randomly collect 100 samples from the e-SNLI dataset and we concatenate each sample with the corresponding explanation generated by the Seq2Seq model in use (BERT2GPT, DISTILBERT2GPT, ROBERTA2GPT, ALBERT2GPT). This way, we create a custom, minimal dataset that has the pair of

columns (premise, hypothesis, ground-truth label, predicted explanation) and we manually determine whether the generated explanation is *correct*, i.e. whether it *fully* justifies the ground-truth label. Examples of manually annotated explanations can be found in table 9.2.

premise	hypothesis	gold label	predicted explanation	Does the explanation fully justify the label?
A blond headed child in yellow boots and yellow jacket vest playing in the gravel with his pail, shovel and trucks	A blonde child is playing	entailment	A child playing is the same as a child playing	NO
A blond headed child in yellow boots and yellow jacket vest playing in the gravel with his pail, shovel and trucks	A blonde child is playing	entailment	A blond headed child is a type of blond child and playing in the gravel is a type of playing	**YES**
An elderly woman wearing a skirt is picking out vegetables at a local market	A young girl is blowing bubbles	contradiction	An elderly woman is not a young girl	NO
An elderly woman wearing a skirt is picking out vegetables at a local market	A young girl is blowing bubbles	contradiction	An elderly woman is not a young girl. Picking out vegetables is not the same as blowing bubbles	**YES**

Table 9.2. *Manual annotation examples from our sampled collection. In case of entailment, we consider an explanation accurate, if it includes all the reasons why hypothesis is entailed by premise. In case of contradiction, an explanation is accurate, if it includes all the reasons why hypothesis contradicts the premise.*

Acknowledging that the selection of a metric that fits our task is crucial for the quality of the generated explanations, we conducted thorough experiments in order to determine which of the text generation metrics METEOR, BLEU, ROUGE and BERT-score is the right for our task. Specifically, we fine-tuned our Seq2Seq models experimenting with each of the above metrics as a validation metric and we manually checked the quality of generated explanations using the methodology that was described above. The validation metric selection was based on the human evaluation of the generated explanations and the results showed that METEOR is the most suitable metric for our task and is followed by BERT-score. This is another evidence that METEOR and BERT-score have a high correlation with human judgement [40, 39] and indeed the whole concept of NLI and natural language explanation generation involves a significant amount of human judgement. As future work, one can experiment with a combination of the above text generation metrics to use as validation metric, e.g. an idea is to use the harmonic mean of METEOR and BERT-score in order to potentially make the best of both metrics.

Having established METEOR score as the validation metric for our task, we can now proceed to the fine-tuning process of the different Seq2Seq model that we experimented with. All the necessary fine-tuning details can be found in the table 9.3. We can see that DISTILBERT2GPT has the smallest training time due to its distillation which makes it faster and more efficient [32]. Furthermore, it is worth noting that ROBERTA2GPT achieves the highest score in all text generation metrics except for METEOR score, where ALBERT2GPT has the best performance.

The details concerning manual evaluation of the explanations generated by each model are summarized in the table 9.4. We can see that ROBERTA2GPT manages to generate the most accurate explanations with a score of 77.17% and is followed by BERT2GPT with a score of 76.14%. Last, we have ALBERT2GPT and DISTILBERT2GPT with a score of 73.33% and 72.53% respectively.

	BERT2GPT	**ALBERT2GPT**	**DISTILBERT2GPT**	**ROBERTA2GPT**
encoder checkpoint	bert-base-uncased	albert-base-v2	distilbert-base-uncased	roberta-base
encoder max length	128	128	128	128
decoder checkpoint	gpt2	gpt2	gpt2	gpt2
decoder max length	64	64	64	64
text generation strategy	greedy search	greedy search	greedy search	greedy search
# epochs	5	5	5	5
batch size	32	32	32	32
# GPUs	1	1	1	1
GPU type	NVIDIA Volta V100	NVIDIA Volta V100	NVIDIA Volta V100	NVIDIA Volta V100
training time (↓)	12 hrs & 20 mins	12 hrs & 16 mins	**9 hrs & 35 mins**	12 hrs & 29 mins
meteor (↑)	0.5332	**0.5591**	0.5393	0.5509
bert-score (↑)	0.8707	0.8742	0.8701	**0.8744**
rouge (↑)	0.5885	0.6005	0.5859	**0.6011**
bleu (↑)	0.3859	0.3911	0.3719	**0.3992**

Table 9.3. *Fine-tuning details of the transformer-based encoder-decoder models (Seq2Seq) that were used for the explanation generation task. All reported scores refer to inference, not validation. The optimal values among all 4 Seq2Seq models are denoted with bold font.*

	BERT2GPT	**ALBERT2GPT**	**DISTILBERT2GPT**	**ROBERTA2GPT**
% correct explanations	76.14%	73.33%	72.53%	**77.17%**

Table 9.4. *Manual evaluation of a random subset of the generated explanations. The correct explanations percentage is reported for all the 100 collected samples. The optimal values among all 4 Seq2Seq models are denoted in bold font.*

9.2.2.2 Expl2label

For fine-tuning the explanation classifier, at training time we provide both the explanation and the ground-truth label, while at inference time the model predicts the output label based only on the input explanation. Naturally, *accuracy* is used as validation metric for the classifier. All the necessary fine-tuning details of the Expl2Label model can be found in table 9.6. We can see that a simple BERT-based classifier achieves 97.47% accuracy on the e-SNLI test data. This is a tremendously high score and supports the claim that, in e-SNLI dataset, one can easily associate an explanation with a label (entailment, contradiction, entailment) without having knowledge of the premise and hypothesis that correspond the explanation [4]. This further supports the claim that the *ExpainThenPredict* setup is indeed a sensible decomposition of our task. However, this is not always the case, because there are multiple ways to formulate an explanation (see table 9.5 for details).

premise	**hypothesis**	**label**	**explanation**
A woman is in the park	A person is in the park	entailment	A woman is a person
A woman is in the park	There is no person in the park	contradiction	A woman is a person

Table 9.5. *The same explanation can justify a different label depending on the input premise and hypothesis. For the contradiction pair, one could also explain that "There can be no person in the park if a woman is in the park" which is more indicative of contradiction [4].*

Fine-tuning details	
encoder checkpoint	bert-base-uncased
encoder max length	128
# epochs	5
batch size	32
# GPUs	1
GPU type	NVIDIA Volta V100
training time	5 hours & 42 mins
test acc	97.47%

Table 9.6. *Fine-tuning details of the BERT-based Expl2Label classifier.*

9.2.2.3 Putting it all together

Having *separately* fine-tuned the models Seq2Seq and Expl2Label, it is time to interconnect them, by feeding the explanation generated by Seq2Seq to the Expl2Label, and perform *inference* to the entire model, which can be considered a black-box (this architecture can be seen in figure 9.1). We report the inference results of the different variations of our ExplainThenPredict model in table 9.7 below.

	BERT2GPT + BERT	**ALBERT2GPT + BERT**	**DISTILBERT2GPT + BERT**	**ROBERTA2GPT + BERT**
acc	86.72%	85.45%	85.15%	**87.97%**
acc (entailment)	89.13%	86.76%	87.29%	**90.17%**
acc (contradiction)	90.42%	88.35%	85.82%	**91.69%**
acc (neutral)	80.4%	81.14%	**82.20%**	82.01%

Table 9.7. *ExplainThenPredict inference results. BERT2GPT, ALBERT2GPT, DISTILBERT2GPT and ROBERTA2GPT indicate the Seq2Seq model variation, while BERT indicates the Expl2Label classifier. We report the accuracy among all test samples as well as the accuracy per label (entailment, contradiction, neutral). The optimal values among all 4 ExplainThenPredict models are denoted in bold font.*

Our model takes a pair of (premise, hypothesis) as input and predicts the corresponding label (entailment, contradiction, neutral), therefore during inference we are interested in the resulting accuracy of our model. We can see that all model variations have a lower accuracy than the baseline's accuracy (90.13%), however, as we will extensively discuss below, this should not bother us at all. Moreover. we can see that ROBERTA2GPT Seq2Seq variation achieves the highest acccuracy (87.97%) followed by BERT2GPT which has 86.72% accuracy and last we have DISTILBERT2GPT and ALBERT2GPT. This observation is in line with the findings of table 9.4 and gives a hint that accurate explanations generally lead to more accurate predictions. In section 9.3, we will attempt to associate the quality of explanations with adversarial robustness, which is, after all, the essence of our work. Finally, it is worth noting that all the fine-tuned ExplainThenPredict models achieve entailment and contradiction accuracy close to 90% and neutral accuracy around

80%, which is significantly lower. Taking into account that the e-SNLI dataset is perfectly balanced among its 3 classes, the lower values of neutral accuracy indicate that, in general, our model can more easily classify pairs where the hypothesis sentence is entailed by the premise or contradicts it.

9.3 Attack results

We use the BERT-attack [33] and TextFooler [3] attack recipes in order to evaluate the the adversarial robustness of our ExplainThenPredict models. BERT-attack and TextFooler are state-of-the-art editors in terms of attacking the transformers architecture and they perfectly fit our task as they both generate fluent adversarial examples that preserve the semantics and the syntax of the original text input.[1]

Our experiments regarding the attacks are carried out in 2 stages as follows:

1. We target the *premise* sentence when attacking the baseline model and the ExplainThenPredict models while keeping the hypothesis as is.

2. We target the *hypothesis* sentence when attacking the baseline model and the ExplainThenPredict models while keeping the premise as is.

Indeed, the strategy of separately attacking premise and hypothesis provides a better alignment among our experiments which renders our conclusions reliable and concrete. Moreover, this strategy can help us gain a better insight regarding the sensitivity of each input component (premise and hypothesis) against adversarial attacks.

In order to measure adversarial robustness, we will use a metric called *attack success rate*. Attack success rate is actually the percentage of attack attempts that produce successful adversarial examples. Therefore, higher values of attack success rate indicate a high number of successful attempts to craft a valid adversarial example, thus the target model is susceptible to the adversarial attack. On the contrary, lower values indicate a low number of successful attempts to craft a valid adversarial example, thus the target model is resistant (robust) to the adversarial attack. Consequently, and taking into account that adversarial robustness is a measurement of a model's susceptibility to adversarial examples, it becomes clear that indeed attack success rate is a valid and sensible metric for our task.

For the sake of completeness, we report all the statistics that were automatically collected during the attacks. However, throughout our analysis, emphasis is placed primarily on attack success rate and secondarily on *after-attack accuracy*, i.e. the percentage of inputs that are both correctly classified and unsuccessfully attacked.

9.3.1 TextFooler

TextFooler algorithm has 2 hyper-parameters that allow us to control the desired diversity and semantic similarity between the original and adversary text. More specifi-

[1] In chapter 7, we highlight that we are interested in recognizing textual entailment only between *meaningful* and *semantically coherent* sentences.

cally, hyper-parameter *max_candidates* (called N in the TextFooler paper [3]) is used to specify the number of candidate synonyms that can potentially replace a "vulnerable" word present in the original sequence and hyper-parameter *min_cos_sim* (called δ in the TextFooler paper [3]) is used to specify the cosine similarity threshold so that the perturbed sequence is considered a valid adversarial example. Therefore, it becomes clear that enlarging *max_candidates* or lowering *min_cos_sim* will force the generation of more diverse synonym candidates in the expense of semantic similarity. On the contrary, lowering *max_candidates* or enlarging *min_cos_sim* will force the generation of semantically more similar synonym candidates in the expense of diversity.

Jin et. al. [3] claim that setting *max_candidates* = 50 and *min_cos_sim* = 0.7 strikes a balance between diversity and semantic similarity control, so we include this combination of values in our experiments. Furthermore, in order to favor semantic similarity over diversity, we slightly increase the value of *min_cos_sim*, i.e. we also experiment with the setting *max_candidates* = 50 and *min_cos_sim* = 0.75. The attack results for the settings mentioned above are summarized in tables 9.8 and 9.9.

TextFooler (target sentence: premise)					
max_candidates = 50 min_cos_sim = 0.7	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	24.93%	27.76%	24.2%	28.74%	24.08%
Attack success rate (↓)	72.16%	67.99%	70.9%	67.33%	71.1%
Average perturbed word %	11.32%	8.04%	7.75%	7.93%	8.12%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	43.74	44.1	41.75	44.57	42.16
max_candidates = 50 min_cos_sim = 0.75	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	33.22%	35.2%	30.92%	36.18%	31.1%
Attack success rate (↓)	62.9%	59.41%	61.5%	58.88%	61.2%
Average perturbed word %	11.52%	8.02%	7.79%	7.89%	8.23%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	37.02	36.68	35.11	37.14	35.49

Table 9.8. *Attack results synopsis (attack recipe: TextFooler, target sentence: premise). Our analysis is based on attack success rate, so emphasis must be placed only on the corresponding rows.*

Figure 9.3 depicts the achieved attack success rates of our ExplainThenPredict models' vs the attack success rate of the baseline for the scenarios where the target sentence is premise and the target sentence is hypothesis, respectively. We observe that regardless of the target sentence and the value of *min_cos_sim*, all model variations achieve a lower attack success rate compared to the baseline. This means that the above variations are indeed more robust in terms of adversarial attacks.

Figure 9.4 depicts the percentage of attack success rate decrease that was achieved by our models. Of course, the reference value is the attack success rate of the baseline

TextFooler (target sentence: hypothesis)					
max_candidates = 50 min_cos_sim = 0.7	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	10.33%	13.86%	12.68%	16.31%	11.83%
Attack success rate (↓)	88.46%	84.01%	85.16%	81.46%	86.11%
Average perturbed word %	8.3%	7.1%	7.22%	7.32%	7.02%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	24.3	24.6	25.05	25.92	24.16
max_candidates = 50 min_cos_sim = 0.75	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	15.89%	18.6%	18.19%	21.85%	16.73%
Attack success rate (↓)	82.26%	78.55%	78.71%	75.16%	80.35%
Average perturbed word %	8.61%	7.28%	7.35%	7.49%	7.2%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	20.64	20.64	21.08	21.67	20.27

Table 9.9. *Attack results synopsis (attack recipe: TextFooler, target sentence: hypothesis). Our analysis is based on attack success rate, so emphasis must be placed only on the corresponding rows.*

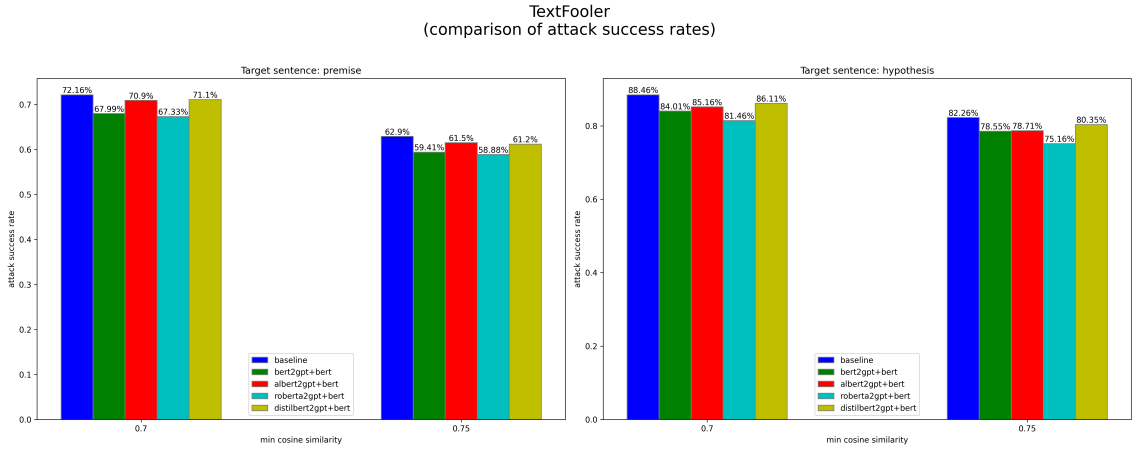

Figure 9.3. *Visualization of attack success rate of ExplainThenPredict model variations vs. baseline model (attack recipe: TextFooler).*

for each setting *min_cos_sim* $\in \{0.7, 0.75\}$. Figure 9.4 can be practically used to:

- Compare each ExplainThenPredict model variation with the baseline in terms of adversarial robustness. Specifically, when the bar is above 0, this means that the attack success rate of our model is lower that the baseline's and therefore our model achieves an increased adversarial robustness. However, when the bar is below 0, this means that the attack success rate of our model is higher that the baseline's and therefore our model fails at increasing adversarial robustness.

- Compare the performance, in terms of adversarial robustness, among the ExplainThenPredict model variations. Specifically, the variation with the largest bar

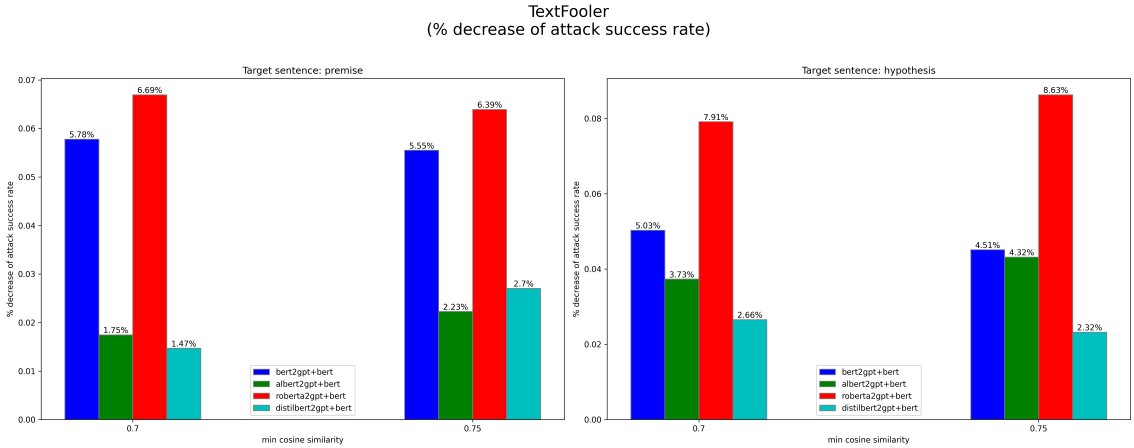

Figure 9.4. *Visualization of the % attack success rate decrease achieved by the 4 ExplainThenPredict model variations.*

height in the graph has the best performance, as it achieves the biggest percentage attack success rate decrease compared to the baseline.

Having said that, we can conclude that ROBERTA2GPT variation is the most successful in enhancing adversarial robustness and is followed by BERT2GPT variation. These two variations stand out consistently, i.e. both for premise/hypothesis as target sentences and for each setting *min_cos_sim* $\in \{0.7, 0.75\}$. ALBERT2GPT and DISTILBERT2GPT achieve significantly enchanced robustness only when the target sentence is hypothesis, whereas they struggle to achieve a decreased attack success rate when when the target sentence is premise.

9.3.2 BERT-attack

BERT-attack algorithm has 1 hyper-parameter that allows us to control the desired semantic similarity between the original and adversary text. More specifically, hyper-parameter *max_candidates* (called K in the BERT-attack paper [33]) is used to specify the number of candidate synonyms that can potentially replace a "vulnerable" word present in the original sequence. Intuitively, enlarging *max_candidates* will force the generation of semantically less similar synonym candidates and therefore the attack success rate is expected to increase [33]. We opt to experiment with the values *max_candidates* $\in \{6, 8\}$, which strike a balance between semantic similarity and computational resources. The attack results for the setting mentioned above are summarized in tables 9.10 and 9.11 below.

Figure 9.5 depicts the achieved attack success rates of our ExplainThenPredict models' vs the attack success rate of the baseline for the scenarios where the target sentence is premise and the target sentence is hypothesis. Interestingly, we observe that, for both scenarios, all ExplainThenPredict model variations consistently achieve a lower attack success rate compared to the baseline, therefore the goal of increasing adversarial robustness is accomplished.

Figure 9.6 depicts the percentage of attack success rate decrease that was achieved

BERT-attack (target sentence: premise)					
max_candidates = 6	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	19.26%	25.18%	21.92%	25.4%	23.37%
Attack success rate (↓)	78.5%	70.96%	74.35%	71.13%	72.55%
Average perturbed word %	10.88%	7.97%	7.84%	7.84%	8.09%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	26.96	29.77	28.52	29.93	29.19
max_candidates = 8	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	14.79%	22.54%	19.02%	22.22%	20.02%
Attack success rate (↓)	83.48%	74.01%	77.74%	74.74%	76.49%
Average perturbed word %	10.78%	7.94%	7.78%	7.83%	8.15%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	31.27	35.9	33.84	35.68	34.88

Table 9.10. *Attack results synopsis (attack recipe: BERT-attack, target sentence: premise). Our analysis is based on attack success rate, so emphasis must be placed only on the corresponding rows.*

BERT-attack (target sentence: hypothesis)					
max_candidates = 6	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	9.16%	15.23%	13.48%	16.11%	13.16%
Attack success rate (↓)	89.77%	82.44%	84.23%	81.68%	84.54%
Average perturbed word %	8.58%	7.22%	7.41%	7.24%	7.34%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	15.28	16.24	16.3	16.59	16.03
max_candidates = 8	Baseline	BERT2GPT + BERT	ALBERT2GPT + BERT	ROBERTA2GPT + BERT	DISTILBERT2GPT + BERT
Original accuracy	90.13%	86.72%	85.45%	87.97%	85.15%
Accuracy under attack	4.87%	11.68%	10.19%	12.53%	9.61%
Attack success rate (↓)	94.57%	86.54%	88.08%	85.76%	88.71%
Average perturbed word %	8.16%	7.07%	7.24%	7.16%	7.21%
Average num. words per input	21.39	21.39	21.39	21.39	21.39
Avg num queries	17.37	18.93	19.03	19.46	18.69

Table 9.11. *Attack results synopsis (attack recipe: BERT-attack, target sentence: hypothesis). Our analysis is based on attack success rate, so emphasis must be placed only on the corresponding rows.*

by our models. Of course, the reference value is the attack success rate of the baseline for each setting *max_candidates* $\in \{6, 8\}$. We can conclude that, for the attack premise scenario, BERT2GPT variation is the most successful in enhancing adversarial robustness and is followed by ROBERTA2GPT variation. However, for the attack hypothesis scenario, the opposite happens, i.e. ROBERTA2GPT achieves the biggest attack success

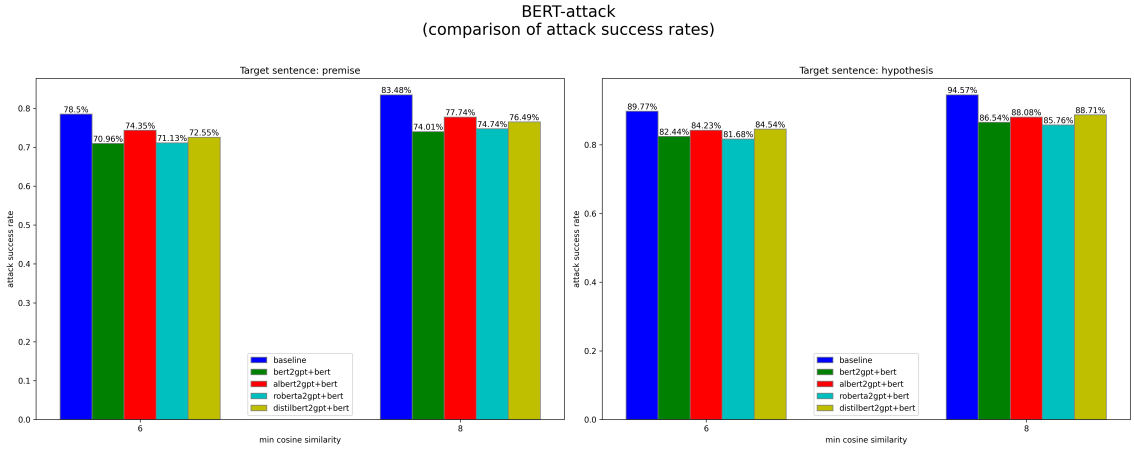

Figure 9.5. *Visualization of attack success rates of ExplainThenPredict model variations vs. baseline model (attack recipe: BERT-attack).*

rate decrease and is followed by BERT2GPT. In any case, we can deduce that, similarly to TextFooler experiments above, BERT2GPT and ROBERTA2GPT appear to be the most robust in terms of attack success rate and are followed by DISTILBERT2GPT and ALBERT2GPT, which, in the case of BERT-attack, manage to significantly decrease the attack success rate, compared to TextFooler, where they struggled to do so.

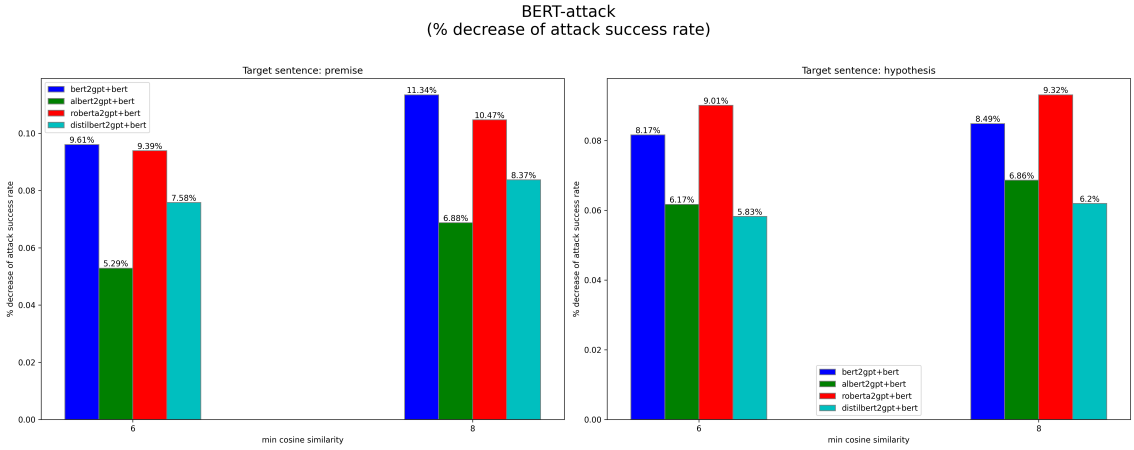

Figure 9.6. *Visualization of the % attack success rate decrease achieved by the 4 ExplainThenPredict model variations.*

Chapter 10

Conclusion

In this work, we modified the traditional NLI task by leveraging natural language explanations during both training and inference. Following the work of Camburu et. al. [4], we utilized the *ExplainThenPredict* setup in our experiments, which first involves a generative model that, given a premise and hypothesis, predicts an explanation that attempts to describe the semantic relationship between premise and hypothesis, and secondly this explanation is fed to a classifier that predicts the output label. Our work is based on the observation that, using the above setup, the predicted label is no longer directly conditioned on the input premise and hypothesis. Instead, it is conditioned on an explanation that attempts to describe the semantic relationship between the input premise and hypothesis, therefore this intermediate explanation could potentially filter noise superimposed in the input sentences. Emphasis was placed on the explanation generation and, specifically, four transformer-based encoder-decoder models (BERT2GPT, ALBERT2GPT, ROBERTA2GPT and DISTILBERT2GPT) were fine-tuned for this goal.

We attacked our models using BERT-attack and TextFooler algorithms, which are state-of-the-art in NLP and generate adversarial examples that retain semantic similarity and fluency and we used attack success rate in order to measure adversarial robustness. Our experiments showed that all the fine-tuned models are indeed more adversarially robust compared to the baseline, which is a simple classifier that does not leverage natural language explanations. Notably, the decrease in attack success rate was observed consistently, i.e. for both TextFooler and BERT-attack, for both target sentences (premise and hypothesis) and for each hyper-parameter value of the attack algorithm that was used in our experiments. Furthermore, we experimentally associated the quality of an explanation with the adversarial robustness (decrease in attack success rate). Specifically, we showed, through human evaluation, that BERT2GPT and ROBERTA2GPT Seq2Seq variations generate the most accurate explanations, while simultaneously achieving the highest decrease in attack success rate.

We hope that, in the future, our work will serve as a strong baseline when it comes to associating natural language explanations with adversarial robustness. One is encouraged to experiment with other datasets than e-SNLI or employ complex architectures that more deeply capture the semantic relationship between premise and hypothesis.

Bibliography

[1] Ian J. Goodfellow, Jonathon Shlens και Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. *International Conference on Learning Representations*, 2014.

[2] Ji Gao, Jack Lanchantin, Mary Lou Soffa και Yanjun Qi. *Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers*. *2018 IEEE Security and Privacy Workshops (SPW)*, σελίδες 50–56, 2018.

[3] Di Jin, Zhijing Jin, Joey Tianyi Zhou και Peter Szolovits. *Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment*. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025, 2020.

[4] Oana Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz και Phil Blunsom. *e-SNLI: Natural Language Inference with Natural Language Explanations*. *Advances in Neural Information Processing Systems*, τόμος 31, 2018.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser και Illia Polosukhin. *Attention is All you Need*. *Advances in Neural Information Processing Systems*, τόμος 30, 2017.

[6] Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos και Giorgos Stamou. *Puzzle Solving using Reasoning of Large Language Models: A Survey*. *arXiv preprint arXiv:2402.11291*, 2024.

[7] Ioannis Panagiotopoulos, Giorgos Filandrianos, Maria Lymperaiou και Giorgos Stamou. *AILS-NTUA at SemEval-2024 Task 9: Cracking Brain Teasers: Transformer Models for Lateral Thinking Puzzles*. *arXiv preprint arXiv:2404.01084*, 2024.

[8] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi και Sanjiv Kumar. *Are Transformers universal approximators of sequence-to-sequence functions?* *International Conference on Learning Representations*, 2020.

[9] Wenqi Wang, Run Wang, Lina Wang, Zhibo Wang και Aoshuang Ye. *Towards a Robust Deep Neural Network Against Adversarial Texts: A Survey*. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):3159–3179, 2023.

[10] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang και Anil K. Jain. *Adversarial Attacks and Defenses in Images, Graphs and Text: A Review*. *International Journal of Automation and Computing*, 17:151–178, 2019.

[11] Giorgos Filandrianos, Konstantinos Thomas, Edmund Dervakos και Giorgos Stamou. *Conceptual Edits as Counterfactual Explanations.* *AAAI Spring Symposium: MAKE*, 2022.

[12] Maria Lymperaiou, George Manoliadis, Orfeas Menis Mastromichalakis, Edmund G. Dervakos και Giorgos Stamou. *Towards Explainable Evaluation of Language Models on the Semantic Similarity of Visual Concepts*. *Proceedings of the 29th International Conference on Computational Linguistics*, σελίδες 3639–3658, 2022.

[13] Edmund Dervakos, Konstantinos Thomas, Giorgos Filandrianos και Giorgos Stamou. *Choose your Data Wisely: A Framework for Semantic Counterfactuals*. *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, σελίδες 382–390, 2023.

[14] Theodoti Stoikou, Maria Lymperaiou και Giorgos Stamou. *Knowledge-Based Counterfactual Queries for Visual Question Answering*, 2023.

[15] Angeliki Dimitriou, Maria Lymperaiou, Giorgos Filandrianos, Konstantinos Thomas και Giorgos Stamou. *Structure Your Data: Towards Semantic Graph Counterfactuals*. *arXiv preprint arXiv:2403.06514*, 2024.

[16] Angeliki Dimitriou, Nikolaos Chaidos, Maria Lymperaiou και Giorgos Stamou. *Graph Edits for Counterfactual Explanations: A comparative study*, 2024.

[17] Maria Lymperaiou, Giorgos Filandrianos, Konstantinos Thomas και Giorgos Stamou. *Counterfactual Edits for Generative Evaluation*. *arXiv preprint arXiv:2303.01555*, 2023.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke και Andrew Rabinovich. *Going deeper with convolutions*. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg και Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. *International Journal of Computer Vision*, σελίδα 211–252, 2015.

[20] I. Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati και Dawn Xiaodong Song. *Robust Physical-World Attacks on Machine Learning Models*. *ArXiv*, abs/1707.08945, 2017.

[21] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh και Patrick McDaniel. *Ensemble Adversarial Training: Attacks and Defenses*. *International Conference on Learning Representations*, 2018.

[22] Ali Shafahi, W. Ronny Huang, Christoph Studer, Soheil Feizi και Tom Goldstein. *Are adversarial examples inevitable?* *International Conference on Learning Representations*, 2019.

[23] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo Jhang Ho, Mani Srivastava και Kai Wei Chang. *Generating Natural Language Adversarial Examples*. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, σελίδες 2890–2896, 2018.

[24] Ji Gao, Jack Lanchantin, Mary Lou Soffa και Yanjun Qi. *Black-Box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers*. *2018 IEEE Security and Privacy Workshops (SPW)*, σελίδες 50–56, 2018.

[25] Vladimir I. Levenshtein. *Binary codes capable of correcting deletions, insertions, and reversals*. *Soviet physics. Doklady*, 10:707–710, 1965.

[26] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li και Ting Wang. *TextBugger: Generating Adversarial Text Against Real-world Applications*. *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

[27] Samuel R. Bowman, Gabor Angeli, Christopher Potts και Christopher D. Manning. *A large annotated corpus for learning natural language inference*. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, σελίδες 632–642, 2015.

[28] Kishore Papineni, Salim Roukos, Ward Todd και Wei Jing Zhu. *BLEU: a Method for Automatic Evaluation of Machine Translation*. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, σελίδες 311–318, 2002.

[29] Satanjeev Banerjee, Alon Lavie, Ward Todd και Wei Jing Zhu. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translatio and/or Summarization*, σελίδες 65–72, 2005.

[30] Chin Yew Lin. *ROUGE: A Package for Automatic Evaluation of Summaries*. *Text Summarization Branches Out*, σελίδες 74–81, Barcelona, Spain, 2004.

[31] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger και Yoav Artzi. *BERTScore: Evaluating Text Generation with BERT*. *International Conference on Learning Representations (ICLR)*, 2020.

[32] Victor Sanh, Lysandre Debut, Julien Chaumond και Thomas Wolf. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. *ArXiv*, abs/1910.01108, 2019.

[33] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue και Xipeng Qiu. *BERT-ATTACK: Adversarial Attack Against BERT Using BERT*. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 6193–6202, Online, 2020.

[34] Ilya Sutskever, Oriol Vinyals και Quoc V. Le. *Sequence to sequence learning with neural networks*. *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, σελίδα 3104–3112, Cambridge, MA, USA, 2014.

[35] Kyunghyun Cho, Bartvan Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk και Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, σελίδες 1724–1734, 2014.

[36] John F. Kolen και Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*, σελίδες 237–243. Wiley-IEEE Press, 2001.

[37] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras και Adrian Vladu. *Towards Deep Learning Models Resistant to Adversarial Attacks*. *International Conference on Learning Representations*, 2017.

[38] Sawan Kumar και Partha Pratim Talukdar. *NILE : Natural Language Inference with Faithful Natural Language Explanations*. *Annual Meeting of the Association for Computational Linguistics*, 2020.

[39] Sia, Suzanna and Belyy, Anton and Almahairi, Amjad and Khabsa, Madian and Zettlemoyer, Luke and Mathias, Lambert. *Logical Satisfiability of Counterfactuals for Faithful Explanations in NLI*. *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligenc*, 2023.

[40] Maxime Kayser, Oana Maria Camburu, Leonard Salewski, Cornelius Emde, Virginie Do, Zeynep Akata και Thomas Lukasiewicz. *e-ViL: A Dataset and Benchmark for Natural Language Explanations in Vision-Language Tasks*. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, σελίδες 1224–1234, 2021.

List of Abbreviations

CNN	Convolutional Neural Network
DNN	Deep Neural Network
NLG	Natural Language Generation
NLI	Natural Language Inference
NLP	Natural Language Processing
RNN	Recurrent Neural Network
ΑΝΔ	Αναδρομικά Νευρωνικά Δίκτυα
ΕΦΓ	Επεξεργασία Φυσικής Γλώσσας
ΣΦΓ	Συμπερασμίς Φυσικής Γλώσσας