ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	(Ελληνικά) ΜΟΝΤΕΛΑ TRANSFORMER ΓΙΑ ΑΝΑΛΥΣΗ ΣΥΝΑΙΣΘΗΜΑΤΟΣ **(Αγγλικά)** TRANSFORMER MODELS FOR SENTIMENT ANALYSIS
Ονοματεπώνυμο Φοιτητή	ΜΟΥΣΤΑΚΑΣ ΑΝΑΣΤΑΣΙΟΣ
Πατρώνυμο	ΓΕΩΡΓΙΟΣ
Αριθμός Μητρώου	Π/ 16194
Επιβλέπων	ΣΩΤΗΡΟΠΟΥΛΟΣ ΔΙΟΝΥΣΙΟΣ, Επίκουρος Καθηγητής

Ημερομηνία Παράδοσης Μήνας Έτος

Ιούνιος 2024

Copyright ©

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΑΝΑΣΤΑΣΙΟΣ ΜΟΥΣΤΑΚΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"ΜΟΝΤΕΛΑ TRANSFORMER ΓΙΑ ΑΝΑΛΥΣΗ ΣΥΝΑΙΣΘΗΜΑΤΟΣ - TRANSFORMER MODELS FOR SENTIMENT ANALYSIS"

ΔΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ:

ΔΙΟΝΥΣΙΟΣ ΣΩΤΗΡΌΠΟΥΛΟΣ

ΤΣΙΧΡΙΝΤΖΗΣ ΓΕΩΡΓΙΟΣ

Πίνακας Περιεχομένων

Εισαγωγή

Η μελέτη και η επεξεργασία της φυσικής γλώσσας (Natural Language Processing – NLP) είναι ένα πεδίο μελέτης που έχει σημειώσει μεγάλη εξέλιξη τα τελευταία χρόνια. Σε αυτή την περίοδο έχουν επινοηθεί, εφαρμοστεί και αξιολογηθεί διάφορες προσεγγίσεις και τεχνικές που έχουν προσφέρει στην πρόοδο του αντικειμένου συνολικά.

Η σύγχρονη προσέγγιση στο αντικείμενο της επεξεργασίας φυσικής γλώσσας είναι η χρήση μηχανικής μάθησης και συγκεκριμένα μοντέλων της αρχιτεκτονικής transformer. Αυτή η αρχιτεκτονική βαθιάς μάθησης εξελίχθηκε κληρονομώντας διάφορα χαρακτηριστικά από προηγούμενες προσεγγίσεις και πλέον έχει την δυνατότητα να καλύψει ένα ευρύ φάσμα εργασιών (tasks) επεξεργασίας φυσικής γλώσσας.

Μία από τις εργασίες για τις οποίες έχουν εκπαιδευτεί διάφορα τέτοια μοντέλα είναι η ανάλυση συναισθήματος του κειμένου[1] (sentiment analysis). Η ανάλυση συναισθήματος είναι ένα πεδίο το οποίο συνδέει την γλωσσολογία με την επιστήμη των υπολογιστών και έχει σαν τελικό στόχο τον καθορισμό του συναισθήματος στο κείμενο που αναλύθηκε. Τελικό αποτέλεσμα της ανάλυσης είναι η θετική ή αρνητική αξιολόγηση του συναισθήματος που εξάγεται από το κείμενο.

Η ανάλυση συναισθήματος είναι ένα χρήσιμο εργαλείο που πλέον συναντάται συχνά στην εργαλειοθήκη των αναλυτών κοινωνικών δικτύων καθώς μπορεί με αυτοματοποιημένο τρόπο να εξάγει το συναίσθημα του κειμένου, αξιοποιώντας το νόημα, λέξεις από το κείμενο και το νόημα τους, την δομή του κειμένου και πολλά άλλα.

Abstract

The study and processing of natural language (NLP) is a field of study that has shown significant progress during the last years. During this time period there have been various models that have been invented, applied and evaluated that have contributed to the overall progress of this field of study.
The contemporary approach to the subject of natural language processing is the usage of machine learning and specifically the usage of a neural network architecture called transformer model. This deep learning architecture has inherited various characteristics from previous approaches and is currently capable of covering a vast field of natural language processing tasks.
One of these tasks is called Sentiment Analysis. There have been many models trained for this specific task with the end goal of determining whether the sentiment of the text is overall positive or negative. Sentiment analysis is a field that bridges linguistics with computer science and has as an end goal the classification of the text to a specific sentiment either positive or negative.
Sentiment analysis is a useful and versatile tool that is being utilized by most social media analysts due to its ability to determine the overall sentiment of a text by taking into account the meaning, the words and the structure of the text between other factors.

Επισκόπηση

Η επεξεργασία φυσικής γλώσσας[2] είναι ένας τομέας μελέτης που συνδυάζει την γλωσσολογία και την μηχανική μάθηση και αποσκοπεί στην κατανόηση της γλώσσας όπως θα την κατανοούσε ένας άνθρωπος. Ο στόχος της επεξεργασίας φυσικής γλώσσας είναι η εξαγωγή του νοήματος του κειμένου σαν σύνολο και η εξαγωγή του ρόλου της κάθε λέξης μέσα στο σύνολο.
Το πεδίο μελέτης της επεξεργασίας φυσικής γλώσσας είναι ευρύ και καλύπτει διάφορες εφαρμογές και μεθοδολογίες και για αυτόν τον λόγο οι εφαρμογές έχουν χωριστεί σε διάφορες εργασίες (tasks). Οι κυριότερες εργασίες είναι οι εξής:

- **Ταξινόμηση προτάσεων:** Αυτή η εργασία έχει σαν στόχο την εξαγωγή κάποιου συμπεράσματος συνολικά για μία πρόταση. Το συμπέρασμα μπορεί να έχει να κάνει με το συναίσθημα της πρότασης (θετικό ή αρνητικό), την ορθότητα της διατύπωσης μίας πρότασης, την λογική σύνδεση δύο προτάσεων μεταξύ τους κλπ.
- **Ταξινόμηση λέξεων μέσα σε μία πρόταση:** Σε αυτή την εργασία στόχος είναι ο καθορισμός του ρόλου κάποιας λέξης μέσα σε μία πρόταση. Για παράδειγμα σε αυτή την κατηγορία εφαρμογής στόχος μπορεί να είναι ο εντοπισμός της οντότητας στην οποία αναφέρεται η πρόταση ή η γραμματική ανάλυση της πρότασης (υποκείμενο, ρήμα, αντικείμενο).
- **Δημιουργία κειμένου:** Η δημιουργία κειμένου με αυτόματο τρόπο εφαρμόζεται με διάφορους τρόπους. Για παράδειγμα η συμπλήρωση κενού σε μία πρόταση (fill mask) το οποίο μπορεί να αποτελείται από μία ή περισσότερες λέξεις ή η σύνθεση κειμένου με αυτοματοποιημένο τρόπο από ορίσματα (prompt) που δίνει ο χρήστης σαν είσοδο.

- **Εξαγωγή απάντησης από κείμενο:** Δοσμένου του κειμένου, μίας ερώτησης και του γενικού νοήματος το σύστημα θα πρέπει να είναι σε θέση να δώσει μία απάντηση που νοηματικά θα απαντά την ερώτηση με βάση το κείμενο που δόθηκε.
- **ΔημΙΟυργία νέου κειμένου από υπάρχον:** Σε αυτή την εργασία έξοδος είναι ένα κείμενο με βάση το κείμενο που δόθηκε σαν είσοδο. Τέτοιου είδους εργασίες είναι η αυτόματη δημιουργία περίληψης ενός κειμένου ή η μετάφραση ενός κειμένου σε άλλη γλώσσα.

Η επεξεργασία φυσικής γλώσσας δεν περιορίζεται μόνο σε αυτές τις εργασίες και επιπλέον επεκτείνεται πέραν του απλού κειμένου και σε πεδία που έχουν να κάνουν με την όραση υπολογιστών ή την αναγνώριση φωνής συνδυάζοντας άλλες εφαρμογές και συμπληρώνοντας την λειτουργικότητα τους όπως για παράδειγμα η περιγραφή μίας εικόνας σε κείμενο ή η απόδοση μίας ηχογράφησης ομιλίας σε γραπτό κείμενο.

Η πρόκληση σε όλες τις παραπάνω εργασίες που περιγράψαμε είναι ότι ο υπολογιστής δεν επεξεργάζεται τις πληροφορίες με τον ίδιο τρόπο που το επεξεργάζεται ένας άνθρωπος. Μία πρόταση της οποίας το νόημα μπορεί να γίνει εύκολα κατανοητό από έναν άνθρωπο είναι πολύ δύσκολο να κατανοηθεί από ένα μοντέλο μηχανικής μάθησης. Για την χρήση τέτοιων μοντέλων το κείμενο πρέπει να έχει επεξεργαστεί με τρόπο τέτοιο ώστε να είναι σε μορφή κατάλληλη για την εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Λόγω της πολυπλοκότητας της ανθρώπινης γλώσσας η διαδικασία της προετοιμασίας του κειμένου είναι μια περίπλοκη διαδικασία. Για αυτή την διαδικασία έχουν δοκιμαστεί και διατυπωθεί διάφοροι τρόποι αναπαράστασης του κειμένου

κάποιοι από τους οποίους θα εφαρμοστούν και στην εφαρμογή που παρουσιάζουμε.

Για πολλά χρόνια το πεδίο της επεξεργασίας φυσικής γλώσσας καλύπτονταν από κλασσικές μεθόδους ανάλυσης του κειμένου οι οποίες προσέφεραν στην μελέτη του αντικειμένου σαν σύνολο και βοήθησαν την ανάπτυξη των σύγχρονων μεθόδων, αλλά παρουσίασαν ελλείψεις σε σημαντικά ζητήματα όπως όταν έρχονταν αντιμέτωπα με μία καινούρια άγνωστη λέξη ή με ένα αρκετά αραιό νοηματικά κείμενο, την νοηματική σύνδεση αντικειμένων που απέχουν αρκετά μεταξύ τους στο κείμενο και άλλα. Παρακάτω θα παρουσιαστούν συνοπτικά κάποιες από τις κλασικές μεθόδους επεξεργασίας φυσικής γλώσσας.

- **n-gram[3]:** Μέθοδος η όποια βασίζεται στις ακολουθίες λέξεων μήκους n. Αυτή η μέθοδος αναθέτει μία πιθανότητα εμφάνισης σε κάθε λέξη για την λέξη της θέσης n δοσμένων των n-1 λέξεων.
- **TF-IDF[4]:** Μέθοδος της οποίας τα αρχικά σημαίνουν term frequency – inverse document frequency και είναι μία στατιστική μέθοδος η οποία αξιολογεί πόσο σχετική είναι μία λέξη σε ένα κείμενο ή ένα σύνολο κειμένων. Αυτή η μέθοδος λειτουργεί αυξάνοντας αναλογικά το πλήθος των εμφανίσεων μίας λέξης μέσα σε ένα κείμενο σταθμίζοντας την με βάση της εμφάνιση αυτής της λέξης σε όλα τα κείμενα που εξετάζονται. Για παράδειγμα τα άρθρα οι σύνδεσμοι και παρόμοιες λέξεις που εμφανίζονται σε κάθε κείμενο μπορεί να έχουν υψηλό ρυθμό εμφάνισης αλλά δεν έχουν υψηλή βαθμολογία καθώς σταθμίζονται από την εμφάνιση τους σε κάθε κείμενο και επομένως δεν προσφέρουν ιδιαίτερα στο νόημα του κειμένου.

Για την αντιμετώπιση των ελλείψεων που παρουσιάζουν κλασσικά μοντέλα όπως τα παραπάνω και την αξιοποίηση περισσότερων δυνατοτήτων του αντικειμένου της επεξεργασίας φυσικής γλώσσας έγινε η μετάβαση των μεθόδων σε μεθόδους μηχανικής και βαθιάς μάθησης.

Μηχανική Μάθηση και Νευρωνικά Δίκτυα

Η επεξεργασία φυσικής γλώσσας χρησιμοποιεί διάφορα μοντέλα μηχανικής μάθησης για την υλοποίηση των εργασιών (tasks) που περιγράψαμε παραπάνω. Τα μοντέλα αυτά αποτελούν την σύγχρονη προσέγγιση της επεξεργασίας φυσικής γλώσσας.
Με τον όρο μοντελοποίηση αναφερόμαστε στην περιγραφή μίας μαθηματικής ή πιθανοτικής σχέσης που υφίσταται μεταξύ διαφόρων μεταβλητών.
Η μηχανική μάθηση[5] είναι η διαδικασία δημιουργίας, εκπαίδευσης και χρήσης μοντέλων τα οποία εκπαιδεύουμε από δεδομένα. Συνήθως ο στόχος είναι η χρήση προϋπαρχόντων δεδομένων για την ανάπτυξη μοντέλων με τα οποία θα μπορούμε να προβλέψουμε διάφορα αποτελέσματα για νέα δεδομένα.
Η διαδικασία εκπαίδευσης μπορεί να πραγματοποιηθεί με επιβλεπόμενο τρόπο (supervised learning), δηλαδή για το σύνολο δεδομένων υπάρχει ένα σύνολο ετικετών με τις σωστές απαντήσεις από τις οποίες μαθαίνουμε ή μη-επιβλεπόμενη (unsupervised learning) στα δεδομένα της οποίας δεν υπάρχει κάποια τέτοια επισήμανση. Γενικά υπάρχουν διάφορα άλλα είδη μάθησης όπως ημιεπιβλεπόμενη (semisupervised) όπου κάποια από τα δεδομένα έχουν ετικέτες, διαδικτυακής (online) μάθησης στην οποία το μοντέλο προσαρμόζεται συνεχώς στα δεδομένα που λαμβάνει ως είσοδο και ενισχυτικής (reinforcement) στην οποία μετά από μία σειρά προβλέψεων το μοντέλο λαμβάνει μία ένδειξη που αξιολογεί το ποσοστό επιτυχίας των προβλέψεων του.

Για ακόμη και απλά προβλήματα υπάρχει ένας μεγάλος αριθμός μοντέλων τα οποία μπορούν να περιγράψουν την σχέση που θέλουμε να μοντελοποιήσουμε. Στις περισσότερες περιπτώσεις επιλέγουμε ένα ήδη παραμετροποιημένο μοντέλο και στην συνέχεια χρησιμοποιούμε τα δεδομένα μας για να μάθουμε τις παραμέτρους μας που μετά από αυτή την διαδικασία θα είναι κατά κάποιο τρόπο βέλτιστες. Αυτή η διαδικασία θα εφαρμοστεί στο παράδειγμα μας με την χρήση υπαρχόντων μοντέλων.

Τα μοντέλα μηχανικής μάθησης που χρησιμοποιούμε στο παράδειγμα μας και γενικά στην επεξεργασία φυσικής γλώσσας ανήκουν στην γενική κατηγορία των Νευρωνικών Δικτύων (Neural Networks). Ο όρος νευρωνικό δίκτυο προέρχεται από προσπάθειες αναπαράστασης επεξεργασίας πληροφοριών από βιολογικά συστήματα (McCulloch and Pitts,1943; Widrow and Hoff, 1960; Rosenblatt, 1962; Rumelhart et al., 1986). Από την οπτική των πρακτικών εφαρμογών της αναγνώρισης προτύπων και μηχανικής μάθησης ο ρεαλισμός όσον αφορά την βιολογική πλευρά του αντικειμένου αυτού θα δημιουργούσε περιττούς περιορισμούς.
Αρχικά θα ορίσουμε την συναρτησιακή μορφή του μοντέλου νευρωνικών δικτύων, ξεκινώντας από ένα απλό γραμμικό μοντέλο.

$$y(x, w) = f\left(\sum_{j=1}^{M} w_j \varphi_j(x)\right)$$

Όπου φ(.) η συνάρτηση βάσης και f(.) μία μη γραμμική συνάρτηση ενεργοποίησης[6] στην περίπτωση που το μοντέλο πραγματοποιεί ταξινόμηση και ταυτότητας σε περίπτωση παλινδρόμησης. Η κύρια διαφορά του γραμμικού μοντέλου με το νευρωνικό δίκτυο είναι ότι η συνάρτηση βάσης βασίζεται σε παραμέτρους οι οποίες μπορούν να τροποποιηθούν όπως και το

διάνυσμα βαρών w. Αυτή η υπόθεση οδηγεί στο βασικό μοντέλο νευρωνικών δικτύων το οποίο μπορεί να περιγραφεί σαν ένα σύνολο συναρτησιακών μετασχηματισμών.

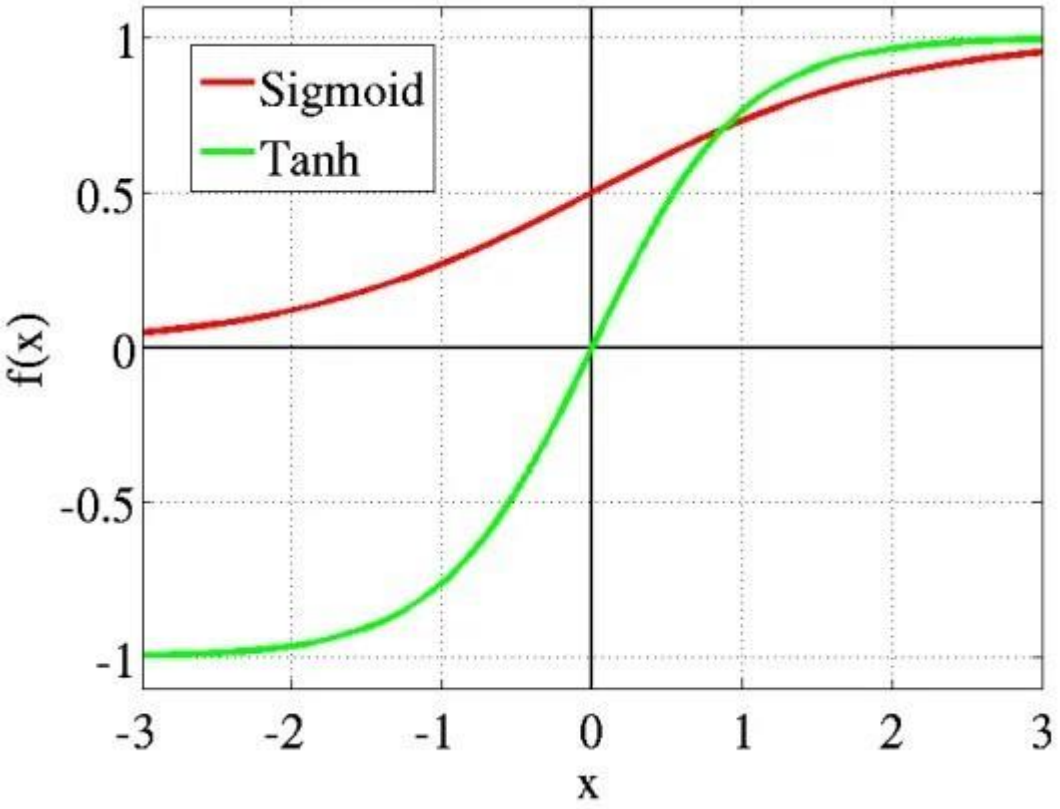
Αρχικά θα κατασκευάσουμε M γραμμικούς συνδυασμούς των παραμέτρων εισόδου $x_1, \ldots, x_D$ με την μορφή:

$$a_j = \sum_{i=1}^{D} [\, w_{ji}^{(1)} + w_{j0}^{(1)}]$$

Όπου j=1, … ,M και η σημείωση (1) συμβολίζουν τις παραμέτρους που αντιστοιχούν στο πρώτο επίπεδο νευρώνων. Οι παράμετροι της μορφής w_{ji} αναφέρονται ως βάρη(weights) και οι παράμετροι w_{j0} να αναφέρονται ως πόλωση (bias). Οι τιμές a_j είναι γνωστές ως τιμές ενεργοποίησης (activations). Κάθε μία από αυτές στην συνέχεια περνά από μία μη γραμμική και παραγωγίσιμη συνάρτηση ενεργοποίησης h(.), η οποία δίνει σαν έξοδο :

$$z_j = h(a_j)$$

Οι τιμές z αντιστοιχούν στις εξόδους των συναρτήσεων βάσης και στο γενικό θέμα των νευρωνικών δικτύων ονομάζονται κρυφές τιμές (hidden units). Οι συναρτήσεις ενεργοποίησης είναι συνήθως σιγμοειδούς μορφής όπως η sigmoid ή η tanh.

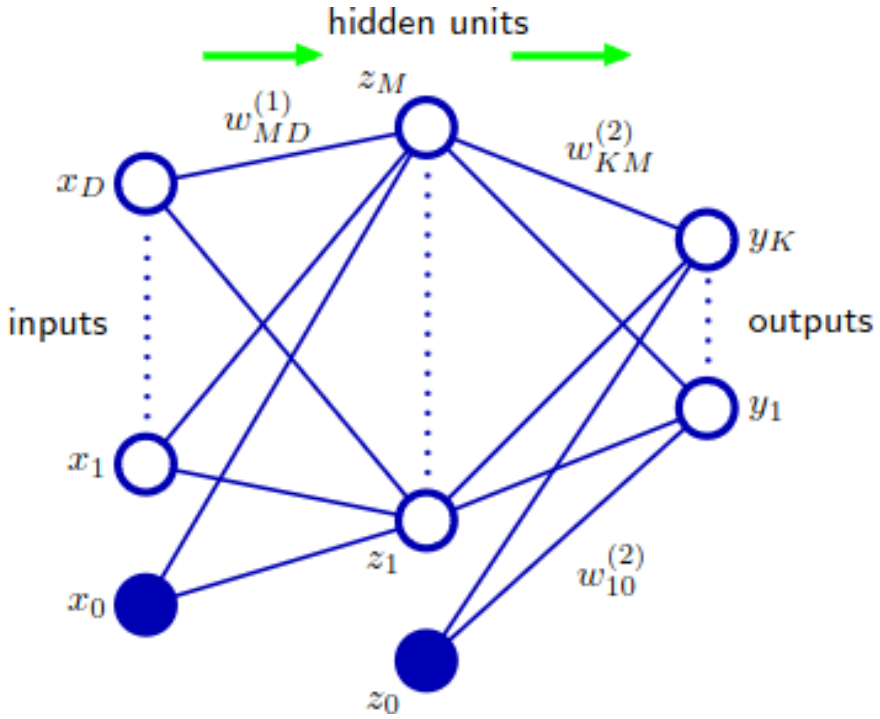


Συναρτήσεις ενεργοποίησης, Πηγή [5]

Στην συνέχεια οι τιμές z συνδυάζονται γραμμικά για να παράγουν την έξοδο του επόμενου επιπέδου.

$$a_k = \sum_{j=1}^{M} [\, w_{kj}^{(2)} z_j + w_{k0}^{(2)}]$$

Όπου k=1, … , K το πλήθος των τελικών εξόδων. Αυτός ο μετασχηματισμός αντιστοιχεί στο δεύτερο επίπεδο νευρώνων του δικτύου και ξανά το w_{k0} είναι η πόλωση του επιπέδου. Για ένα νευρωνικό δίκτυο με δύο επίπεδα το a_k μετά τον μετασχηματισμό του από μία κατάλληλη συνάρτηση h(.) όπως την περιγράψαμε παραπάνω δίνει το σύνολο των εξόδων του μοντέλου y_k.



Διάγραμμα μοντέλου νευρωνικών δικτύων δύο επιπέδων με τα ονόματα μεταβλητών όπως παρουσιάστηκαν παραπάνω. Πηγή [5]

Επομένως μπορούμε να συνοψίσουμε το παραπάνω δίκτυο σε μία συναρτησιακή μορφή όπου:

$$y_k = \sigma(a_k)$$

Με σ να συμβολίζουμε την σιγμοειδή συνάρτηση με τύπο:

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

Με τον τύπο συνολικά να έχει την ακόλουθη μορφή:

$$y_k(x, w) = \sigma\left(\sum_{j-1}^{M}\left[w^{(2)}_{kj}\, h\left(\sum_{i=1}^{D}\left[w^{(1)}_{ji} + w^{(1)}_{j0}\right]\right) + w^{(2)}_{k0}\right]\right)$$

Ο παραπάνω τύπος μπορεί να επεκταθεί σε περισσότερα από δύο επίπεδα, με κατάλληλες τροποποιήσεις.

Σε τελική ανάλυση ένα νευρωνικό δίκτυο είναι μία μη γραμμική συνάρτηση που έχει σαν είσοδο ένα σύνολο παραμέτρων {x} και σαν έξοδο ένα σύνολο παραμέτρων {y} που ελέγχονται από ένα διάνυσμα βαρών {w} που αποτελείται από παραμέτρους που δέχονται τροποποίηση.

Η διαδικασία τροποποίησης των βαρών ενός νευρωνικού δικτύου ονομάζεται εκπαίδευση και αποτελείται από δύο διαδικασίες, την εμπρόσθια εκπαίδευση (forward training) του δικτύου και την οπισθοδρόμηση (back propagation).

Έως τώρα το νευρωνικό δίκτυο έχει παρουσιαστεί σαν ένα σύνολο μη γραμμικών παραμετρικών συναρτήσεων με είσοδο x και έξοδο y. Μία απλοϊκή προσέγγιση για τον καθορισμό των παραμέτρων του δικτύου είναι η ελαχιστοποίηση του αθροίσματος του τετραγωνικού σφάλματος.

Δοσμένου του συνόλου εισόδου $\{x_n\}$ όπου n=1, … ,N με το αντίστοιχο σύνολο $\{t_n\}$ που αποτελείται από τις επιθυμητές εξόδους, ελαχιστοποιούμε την ακόλουθη συνάρτηση σφάλματος:

$$E(w)=\frac{1}{2}\sum_{n=1}^{N}\|y(x_n,w)-t_n\|^2$$

Οι βέλτιστες παράμετροι στο σημείο όπου η παράγωγος μηδενίζεται, άρα οι βέλτιστες παράμετροι βρίσκονται όταν:

$$\nabla E(w)=0$$

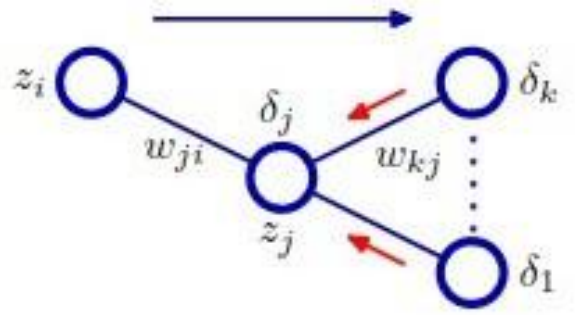
Επομένως κάθε φορά για να βελτιώσουμε τις παραμέτρους μπορούμε να κάνουμε ένα βήμα προς την κατεύθυνση που αντιστοιχεί σε $-\nabla E(w)=0$ και με αυτόν τον τρόπο να μειώσουμε το λάθος.
Ο τελικός στόχος της διαδικασίας είναι να βρούμε μία τιμή για το διάνυσμα w η οποία ελαχιστοποιεί το σφάλμα E(w). Το πρόβλημα που παρουσιάζει αυτή η διαδικασία είναι ότι υπάρχουν πολλά σημεία στα οποία το w είναι το τοπικό μέσο λόγω του μεγάλου πλήθους των παραμέτρων και των πολώσεων που χρησιμοποιεί το μοντέλο. Επειδή η εύρεση μίας αναλυτικής μεθόδου είναι σχεδόν αδύνατη για μοντέλα με μεγάλο αριθμό παραμέτρων οι περισσότερες τεχνικές χρησιμοποιούν μία αρχική, συνήθως τυχαία, τιμή για το $w^{(0)}$ και στην συνέχεια επαναληπτικά μετατοπίζονται μέσα στον χώρο των βαρών με επαναληπτικό τρόπο, όπως διατυπώνεται παρακάτω:

$$w^{\tau+1}=w^{\tau}+\Delta w^{\tau}$$

Ο πιο απλός τρόπος βελτιστοποίησης είναι η κατάβαση βαθμίδας (gradient descent) η οποία έχει σαν τεχνική την πραγματοποίηση ενός μικρού βήματος προς την αρνητική κατεύθυνση της παραγώγου, με την χρήση μίας παραμέτρου $\eta > 0$ που συνήθως λαμβάνει πολύ μικρή τιμή και ονομάζεται ρυθμός μάθησης (learning rate). Αυτή η μέθοδος ορίζεται ως εξής:

$$w^{\tau+1}=w^{\tau}-\eta \nabla E(w^{\tau})$$

Ο στόχος της παραπάνω διαδικασίας είναι η εύρεση μίας αποδοτικής μεθόδου για τον υπολογισμό της παραγώγου μίας συνάρτησης σφάλματος E(w). Αυτή η μέθοδος καλείται να διατρέχει το δίκτυο προς τα μπροστά (από την είσοδο στην έξοδο) και προς τα πίσω (από την έξοδο στην είσοδο) και ονομάζεται οπισθοδρόμηση σφάλματος (error backprogatation).



Με το κόκκινο χρώμα παρουσιάζεται η μέθοδος της οπισθοδρόμησης στο γράφημα. Πηγή [5]

Μοντέλα Transformers

Τα νευρωνικά δίκτυα που χρησιμοποιούνται στις σύγχρονες προσεγγίσεις της επεξεργασίας φυσικής γλώσσας ανήκουν στην κατηγορία των μοντέλων Transformer[8],[18]. Αυτά τα μοντέλα είναι το κύριο εργαλείο για τις εφαρμογές μηχανικής μάθησης στο πεδίο της επεξεργασίας φυσικής γλώσσας επειδή μπορούν να παρουσιάσουν αποτελέσματα μεγάλης ακρίβειας σε όλες τις εργασίες από την κατηγοριοποίηση κειμένου μέχρι την σύνθεση κειμένου. Ο μηχανισμός προσοχής (attention mechanism) είναι ένα πολύ σημαντικό μέρος αυτού του είδους μοντέλου και έχει πολύ σημαντικό ρόλο στο σύστημα. Ο μηχανισμός προσοχής είχε χρησιμοποιηθεί σε συμβατικά μοντέλα βαθιάς μάθησης πριν την χρήση του στα μοντέλα Transformer.

Ο μηχανισμός προσοχής (attention mechanism) προτάθηκε αρχικά για να δημιουργεί αντιστοιχία ανάμεσα στις εισόδους και τις εξόδους των κρυφών επιπέδων σε συμβατικά μοντέλα μηχανικής μάθησης. Ο μηχανισμός αυτός χρησιμοποιείται για να προσθέσει βάρη σε ενδιάμεσες κρυφές τιμές. Αυτά τα βάρη εξισορροπούν την προσοχή που πρέπει να δώσει το μοντέλο σε κάθε βήμα αποκωδικοποίησης. Το διάγραμμα ενός βασικού μηχανισμού μάθησης είναι το ακόλουθο.

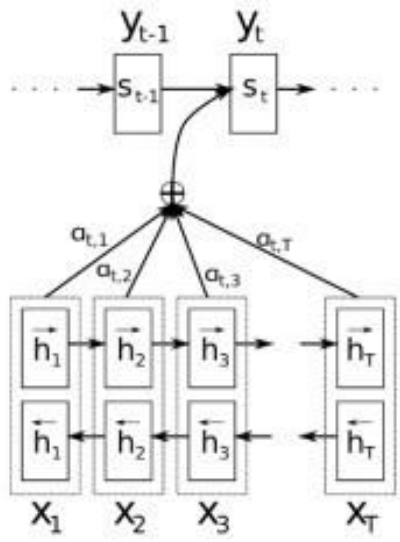


Figure 1.12 – Attention mechanism

Μηχανισμός Attention. Πηγή [18]

Ανά τα έτη έχουν προταθεί διάφορα είδη μηχανισμών προσοχής με διάφορες βελτιώσεις. Μερικοί από αυτούς τους μηχανισμούς προσοχής είναι οι: προσθετικοί (additive), πολλαπλασιαστικοί (multiplicative), γενικοί (general), πολλαπλασιαστικοί ανά στοιχείο (dot product). Ο τελευταίος μηχανισμός σε μία τροποποιημένη μορφή του με κλιμάκωση (scaling) παραμέτρων είναι και ο μηχανισμός προσοχής που συναντάμε στα μοντέλα Transformer. Ο μηχανισμός των μοντέλων αυτών έχει συγκεκριμένο όνομα και ονομάζεται multi-head attention mechanism. Παρακάτω παρουσιάζονται σε έναν πίνακα οι βασικοί μηχανισμοί προσοχής καθώς και το μοντέλο του multi-head attention mechanism.

Name	Attention score function	Citation
Content-based attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$	Loung2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$	Loung2015
Dot-product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Loung2015
Scaled dot-product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$	Vaswani2017

Table 2 – Types of attention mechanisms (Image inspired from https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html)

Πηγή [18]

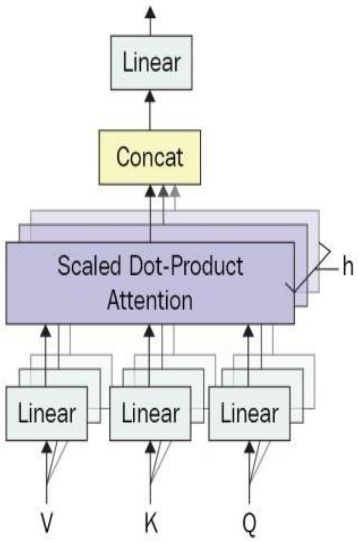


Figure 1.14 – Multi-head attention mechanism

Πηγή [18]

Πριν αναφερθούμε στους μηχανισμούς scaled dot product attention θα πραγματοποιήσουμε μία αναφορά στον μηχανισμό της αυτό-προσοχής (self attention). Αυτός ο μηχανισμός έχει σαν είσοδο τον πίνακα εισόδου X. Από αυτόν τον πίνακα

δημιουργούνται τρεις άλλοι πίνακες οι Q (query), K (key) και V (value) οι οποίοι είναι το γινόμενο του πίνακα X με τους πίνακες θ, Φ, g αντίστοιχα. Το γινόμενο του πίνακα Q με τον πίνακα K είναι η βαθμολογία του πίνακα προσοχής (attention score matrix). Το γινόμενο του πίνακα αυτού με τον πίνακα V δίνει το τελικό σκορ προσοχής του μοντέλου. Η διαδικασία που περιγράψαμε παραπάνω παρουσιάζεται στο επόμενο διάγραμμα.

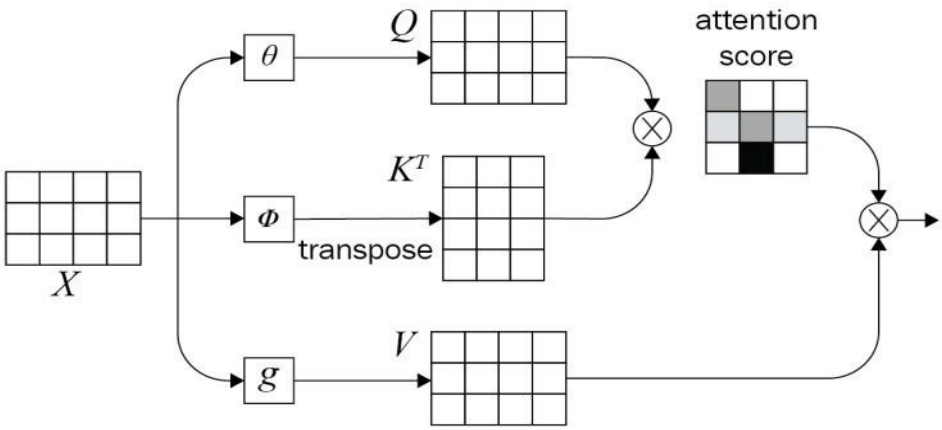


Figure 1.15 – Mathematical representation for the attention mechanism (Image inspired from https://blogs.oracle.com/datascience/multi-head-self-attention-in-nlp)

Πηγή [18]

Ο μηχανισμός κλιμακούμενου ανά σημείο πολλαπλασιασμό προσοχής (scaled dot product attention) είναι παρόμοιος με τον μηχανισμό αυτό-προσοχής (self attention) με την μόνη διαφορά ότι χρησιμοποιεί έναν παράγοντα κλιμάκωσης (scaling factor). Επιπλέον το μοντέλο θα πρέπει να είναι σε θέση να επεξεργάζεται διάφορες ιδιότητες της εισόδου από όλα τα επίπεδα. Τα μοντέλα τύπου Transformer ελέγχουν σχολιασμούς προηγούμενων επιπέδων των κωδικοποιητών και τιμές από προηγούμενα κρυφά επίπεδα.

Η αρχιτεκτονική Transformer δεν έχει μία βήμα προς βήμα επαναλαμβανόμενη ροή, αλλά χρησιμοποιεί κωδικοποίηση θέσης με στόχο την απόκτηση πληροφοριών για κάθε δείγμα της ακολουθίας εισόδου. Οι συνδεδεμένες έξοδοι του embedding και οι καθορισμένες τιμές της κωδικοποίησης διάταξης είναι οι είσοδοι που εισάγονται στο επίπεδο εισόδου του μοντέλου Transformer. Αυτή η λειτουργία της αρχιτεκτονικής Transformer παρουσιάζεται στο επόμενο διάγραμμα.

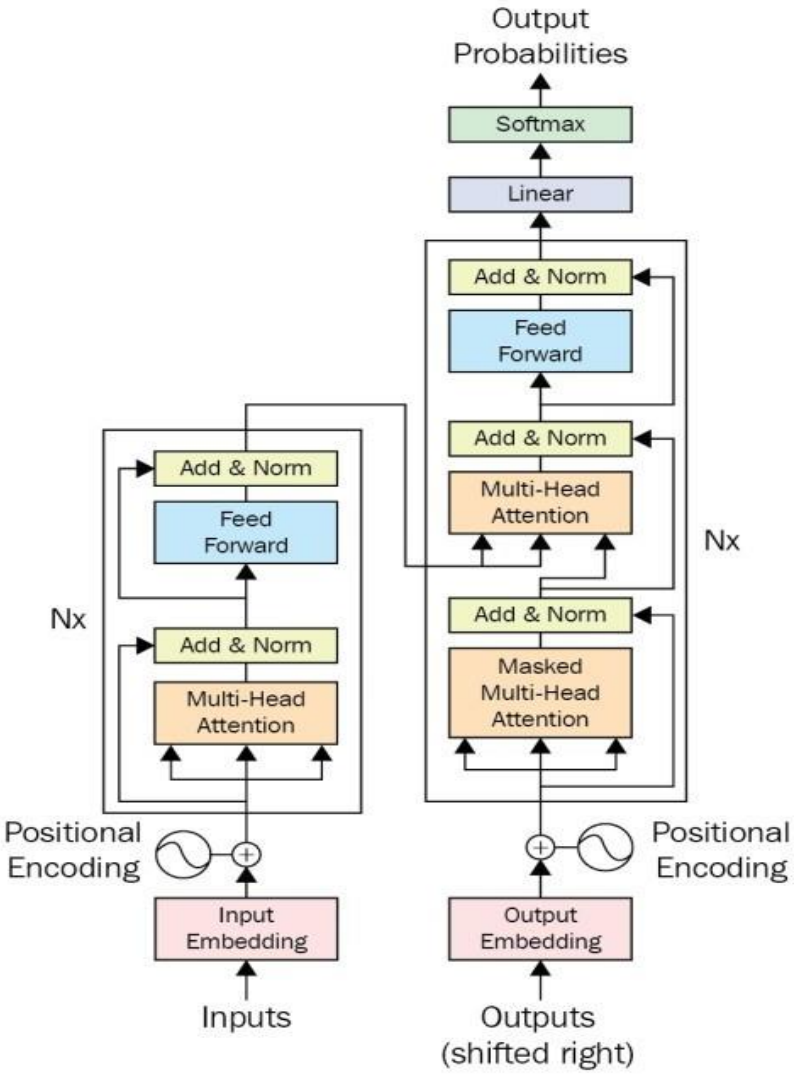


Figure 1.16 – A Transformer

Πηγή [18]

Ένα παράδειγμα που παρουσιάζει την απόδοση του μοντέλου Transformer και του scaled dot product παρουσιάζεται στην επόμενη εικόνα.

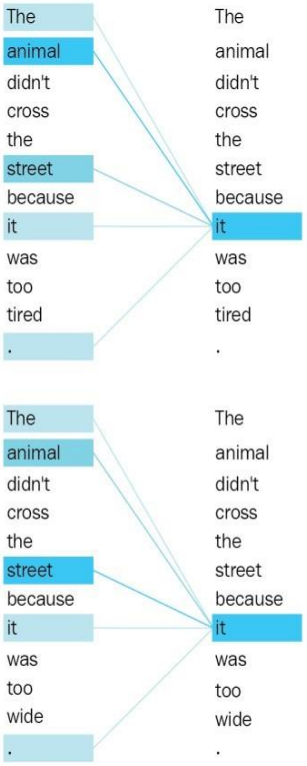
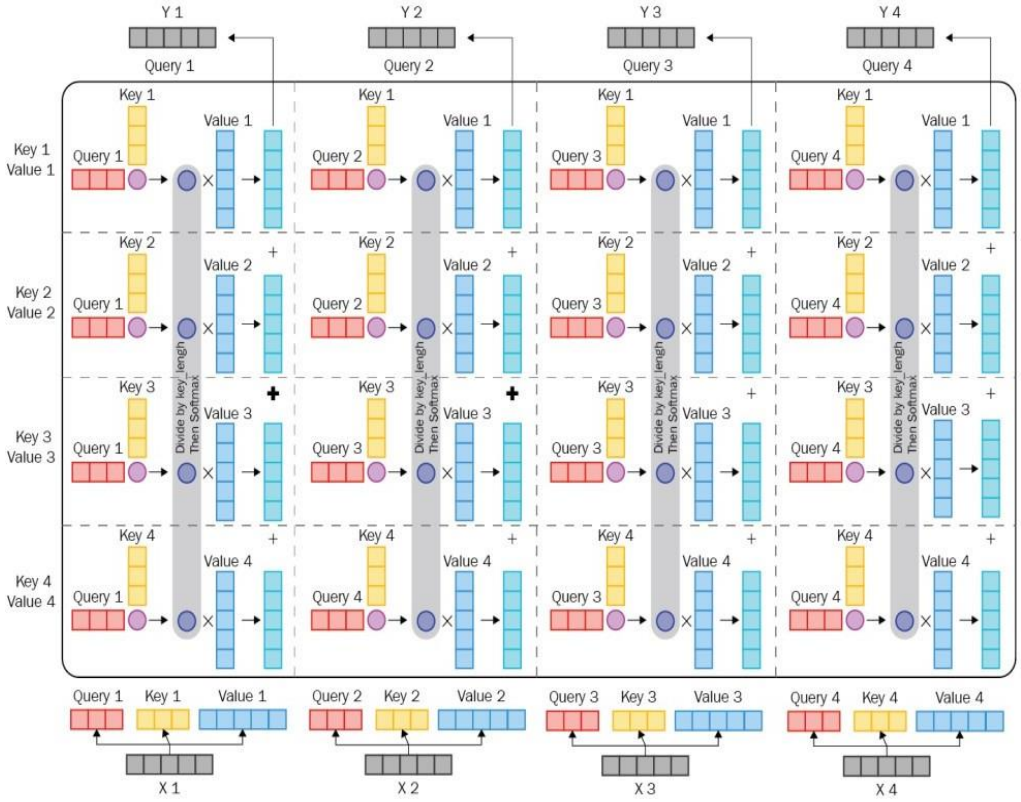


Figure 1.18 – Attention mapping for Transformers (Image inspired from https://ai.googleblog.com/2017/08/Transformer-novel-neural-network.html)

Πηγή [18]

Η λέξη it αναφέρεται σε διάφορα θέματα όπως μπορούμε να παρατηρήσουμε στην παραπάνω εικόνα. Μία άλλη βελτίωση που εμφανίστηκε λόγω των μοντέλων Transformer είναι ο παραλληλισμός. Τα κλασσικά ακολουθιακά μοντέλα δεν έχουν αυτή την δυνατότητα καθώς τα δείγματα επεξεργάζονται από το μοντέλο ένα προς ένα. Τα συστήματα εμπρόσθιας τροφοδότησης επιταχύνουν την διαδικασία καθώς οι πολλαπλασιασμοί πινάκων είναι ταχύτερη διαδικασία από μία ακολουθιακή μονάδα. Ένα σύνολο multi-head attention layers τοποθετημένα το ένα μετά το άλλο μπορούν να έχουν μία καλύτερη κατανόηση σε περίπλοκες προτάσεις. Μία οπτικοποίηση τέτοιου μοντέλου παρουσιάζεται στο επόμενο διάγραμμα.



Πηγή [18]

Στην πλευρά του μοντέλου που αντιστοιχεί στον αποκωδικοποιητή χρησιμοποιείται μία παρόμοια τεχνική με αυτή που χρησιμοποιείται στον κωδικοποιητή. Η κωδικοποίηση δίνεται σαν είσοδος σε κάθε στοίβα αποκωδικοποιητών στο δεύτερο επίπεδο multi-head attention. Αυτή η τροποποίηση δίνει την δυνατότητα στο μοντέλο να μπορεί να διατηρεί μία εικόνα της εξόδου του κωδικοποιητή ενώ βρίσκεται στο βήμα της αποκωδικοποίησης και την ίδια στιγμή να έχει μία καλύτερη ροή στην κατάβαση βαθμίδας που περιγράψαμε παραπάνω. Το τελευταίο επίπεδο με τον softmax χρησιμοποιείται για να παρέχει την έξοδο για διάφορες εφαρμογές για τις οποίες παρουσιάστηκε το μοντέλο Transformer εξ αρχής.

Ένα συνολικό παράδειγμα του μοντέλου Transformer δίνεται στο επόμενο διάγραμμα. Δείχνει ένα μοντέλο Transformer με δύο επίπεδα κωδικοποιητών και δύο επίπεδα αποκωδικοποιητών. Το επίπεδο με όνομα Add & Normalize του διαγράμματος προσθέτει και κανονικοποιεί την είσοδο που παίρνει από το επίπεδο εμπρόσθιας τροφοδότησης (Feed Forward).

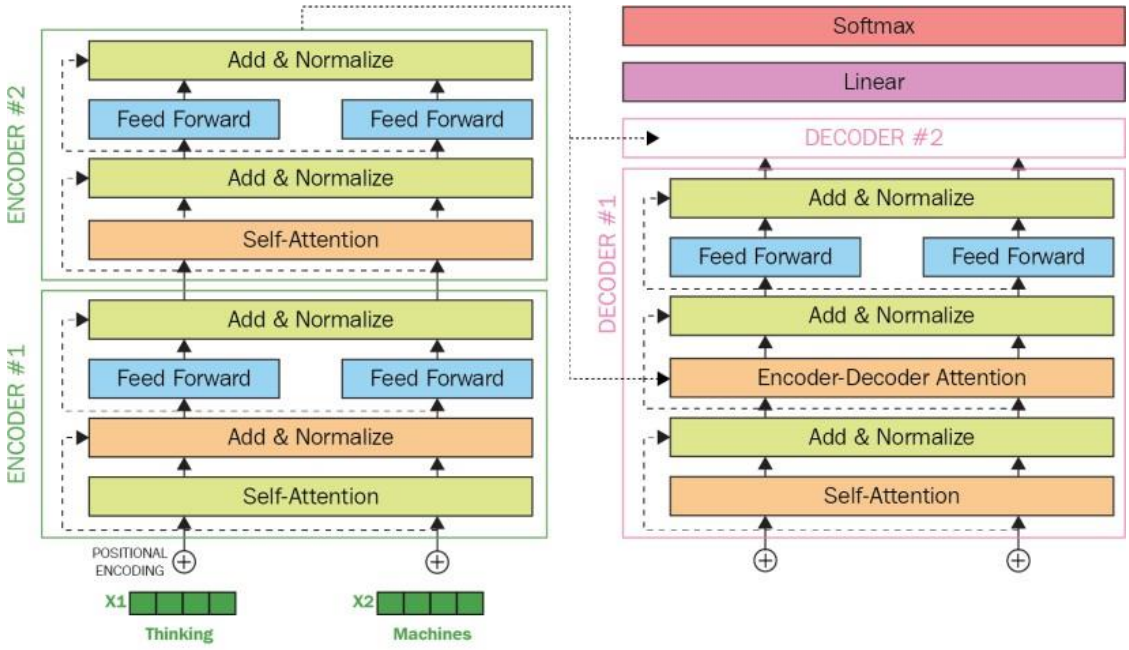


Figure 1.20 – Transformer model (Image inspired from http://jalammar.github.io/illustrated-Transformer/)

Πηγή [18]

Τα μοντέλα των οποίων οι αρχιτεκτονικές είναι βασισμένες στο μοντέλο Transformer έχουν αρκετά κοινά χαρακτηριστικά, για παράδειγμα όλα είναι βασισμένα στην ίδια αρχική αρχιτεκτονική με διαφορές στα βήματα που δεν χρησιμοποιούν. Σε κάποιες περιπτώσεις εφαρμόζονται μικρές αλλαγές όπως για παράδειγμα βελτιώσεις στον μηχανισμό multi-head που εφαρμόζουν κάθε φορά.

Περιγραφή Εφαρμογής

Μέχρι τώρα έχουμε καλύψει θεωρητικά τις έννοιες της επεξεργασίας φυσικής γλώσσας, των νευρωνικών δικτύων, της μηχανικής και της βαθιάς μάθησης και των μοντέλων Transformers. Σε αυτή την ενότητα θα γίνει αναφορά στις λειτουργίες που θα υλοποιήσουμε με την μορφή παραδειγμάτων για την επίδειξη της λειτουργικότητας των εφαρμογών που παρουσιάσαμε.
Σε αυτή την εφαρμογή θα χρησιμοποιηθούν πακέτα μηχανικής μάθησης και transformer μοντέλων όπως το PyTorch και το πακέτο transformers.

Το PyTorch[7] είναι ένα ανοικτού κώδικα (open source) πακέτο μηχανικής μάθησης που χρησιμοποιείται για την κατασκευή και εκπαίδευση μοντέλων μηχανικής και βαθιάς μάθησης. Το πακέτο αυτό έχει αναπτυχθεί κυρίως από την ομάδα έρευνας για το AI του Facebook. Το πλεονέκτημα του PyTorch αντίθεση με παρόμοια πακέτα μηχανικής μάθησης που χρησιμοποιούν στατικούς υπολογιστικούς γράφους, το PyTorch χρησιμοποιεί δυναμικούς υπολογισμούς που επιτρέπουν μεγαλύτερη ευελιξία στην κατασκευή πολύπλοκων μοντέλων.

Το logo του πακέτου PyTorch

Το πακέτο Transformers έχει αναπτυχθεί από τον οργανισμό Hugging Face και παρέχει σύγχρονα προ-εκπαιδευμένα μοντέλα μηχανικής μάθησης. Όλα αυτά τα μοντέλα μπορούν να χρησιμοποιηθούν προ-εκπαιδευμένα με την χρήση του πακέτου pipeline ή να εκπαιδευτούν εκ νέου σε συγκεκριμένο σύνολο δεδομένων για μία συγκεκριμένη εργασία με στόχο να παρουσιάζουν καλύτερη απόδοση σε αυτό.
Το μοντέλο στο οποίο θα βασιστούμε για αυτή την υλοποίηση είναι το BERT (Bidirectional Encoder Representations from Transformers) το οποίο είναι το πρώτο μοντέλο autoencoder το οποίο χρησιμοποίησε την αρχιτεκτονική κωδικοποιητή στοίβας Transformer με μικρές τροποποιήσεις για την μοντελοποίηση της φυσικής γλώσσας.
Η αρχιτεκτονική BERT[9] είναι ένας πολυεπίπεδος κωδικοποιητής βασισμένος στην αρχική υλοποίηση του μοντέλου Transformer. Το μοντέλο Transformer από την αρχή ήταν σχεδιασμένο για εργασίες αυτόματης μετάφρασης με το BERT να υλοποιεί ένα μέρος της αρχιτεκτονικής ώστε να μπορεί να προσφέρει καλύτερη μοντελοποίηση της φυσικής γλώσσας. Αυτό το μοντέλο αφού εκπαιδευτεί είναι σε θέση να παρέχει μία πλήρη κατανόηση της γλώσσας στην οποία εκπαιδεύτηκε.
Οι συναρτήσεις tokenizing[11] είναι ένα από τα σημαντικά μέρη μιας εφαρμογής επεξεργασίας φυσικής γλώσσας. Για το BERT χρησιμοποιείται το WordPiece. Γενικά οι tokenizers WordPiece, SentencePiece, BytePairEncoding είναι οι τρεις πιο γνωστοί και διαδεδομένοι tokenizers που χρησιμοποιούνται από τα μοντέλα Transformers. Ο κύριος λόγος που το BERT και άλλα μοντέλα Transformers χρησιμοποιούν tokenization υπολέξεων είναι η δυνατότητα των συναρτήσεων αυτών να αντιμετωπίζουν άγνωστα δείγματα.
Επιπλέον το BERT χρησιμοποιεί κωδικοποίηση θέσης για τα δείγματα (tokens) για να είναι σίγουρο ότι η θέση των δειγμάτων θα δωθεί στο μοντέλο με ορθό τρόπο.

Η παρακάτω εικόνα παρουσιάζει την αρχιτεκτονική του BERT για διάφορες εργασίες (tasks) μηχανικής μάθησης.

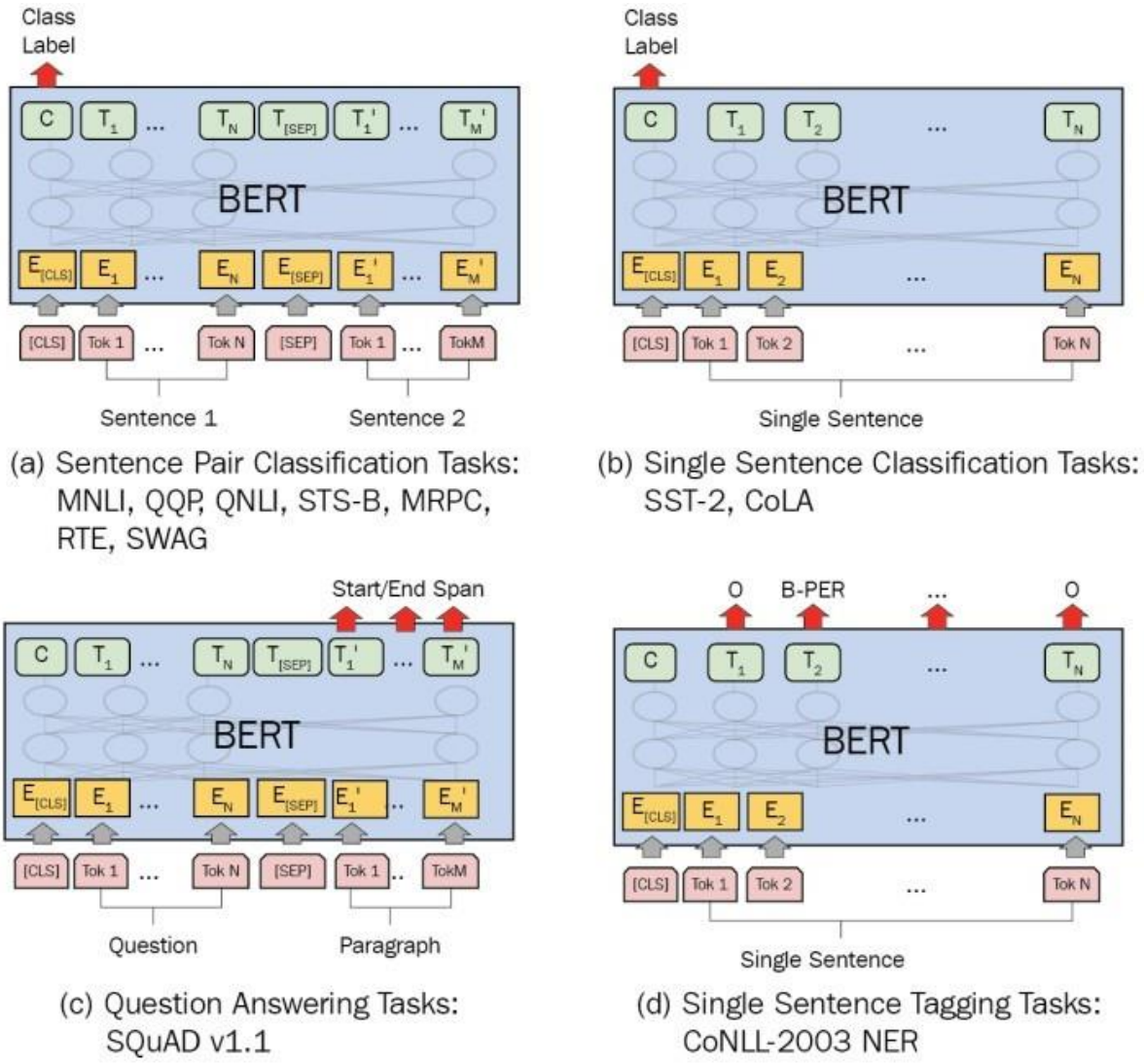


Figure 3.3 – BERT model for various NLP tasks

Πηγή [18]

Πειραματικό Βήμα και Περιγραφή Συνόλου Δεδομένων

Στο πειραματικό βήμα θα εκπαιδεύσουμε παραμετροποιημένα μοντέλα σε δεδομένα ώστε να βελτιώσουμε την ακρίβεια τους καθώς και θα παρουσιάσουμε την λειτουργικότητα έτοιμων μοντέλων σε διάφορες εργασίες (tasks) της επεξεργασίας φυσικής γλώσσας.

Το πρώτο πείραμα που θα παρουσιάσουμε είναι η εκπαίδευση ενός μοντέλου BERT (Bidirectional Encoder Representations from Transformers). Το BERT είναι σχεδιασμένο με τρόπο τέτοιο ώστε να είναι σε θέση να εκπαιδεύει αρχιτεκτονικές βαθιάς μάθησης με αμφίδρομες αναπαραστάσεις από κείμενο που δεν έχει ετικέτες με την τακτική της από κοινού προετοιμασίας στο νόημα στο αριστερό άκρο και το δεξιό άκρο σε όλα τα επίπεδα νευρώνων. Σαν αποτέλεσμα της ιδιότητας που παρουσιάσαμε το προ-εκπαιδευμένο μοντέλο BERT μπορεί να βελτιώσει την ακρίβεια του με την προσθήκη ενός τελευταίου επιπέδου νευρώνων στην έξοδο και με αυτόν τον τρόπο να δημιουργηθούν μοντέλα για διάφορες εφαρμογές επεξεργασίας φυσικής γλώσσας χωρίς κάποια τροποποίηση ειδική για κάθε αρχιτεκτονική.

Το σύνολο δεδομένων που θα χρησιμοποιήσουμε για την εκπαίδευση είναι το CNN/DailyMail[10] Dataset το οποίο χρησιμοποιείται για την εργασία περίληψης κειμένου (text summarization task). Τα δεδομένα προήλθαν από άρθρα του CNN και της Daily Mail τα οποία λογίζονται ως ερωτήσεις στο σύνολο δεδομένων και ως σύνολο απαντήσεων χρησιμοποιούνται περιλήψεις οι οποίες έχουν γραφτεί από ανθρώπους.
Το σύνολο δεδομένων αποτελείται από 286.817 σύνολα εκπαίδευσης, 13.368 σύνολα επικύρωσης και 11.487 σύνολα δοκιμής. Τα άρθρα έχουν μήκος 766 λέξεις και 29,74 προτάσεις κατά μέσο όρο, ενώ οι περιλήψεις έχουν μήκος 53 λέξεις και 3.72 προτάσεις.

Το μοντέλο δεν είναι σε θέση να επεξεργαστεί το κείμενο με την αρχική μορφή του η οποία είναι κατανοητή από τον άνθρωπο, επομένως θα πρέπει να πραγματοποιηθεί κάποια διαδικασία η οποία θα μετατρέψει τα δεδομένα σε μορφή τέτοια ώστε να είναι κατάλληλα για είσοδο στο μοντέλο.
Την διαδικασία αυτή πραγματοποιεί ένας ενδιάμεσος μηχανισμός που ονομάζεται tokenizer. Η διαδικασία Tokenization (διακριτοποίηση) είναι μία απαραίτητη διαδικασία στον κόσμο της επεξεργασίας φυσικής γλώσσας και χρησιμοποιείται για να διαχωρίσει το κείμενο σε μικρότερες μονάδες που ονομάζονται tokens (δείγματα). Τα tokens μπορούν να είναι λέξεις, φράσεις, τμήματα λέξεων η ακόμη και χαρακτήρες ανάλογα τις απαιτήσεις της εφαρμογής. Σε εφαρμογές που χρησιμοποιούν μοντέλα Transformer η διαδικασία διακριτοποίησης (tokenization) είναι ένα σημαντικό βήμα στην προ-επεξεργασία δεδομένων καθώς τα δεδομένα μετασχηματίζονται σε μορφή έτοιμη για χρήση από τα μοντέλα.
Η διαδικασία της διακριτοποίησης (tokenization) βοηθά το μοντέλο να αναγνωρίσει την δομή του κειμένου και να το επεξεργαστεί με πιο αποδοτικό τρόπο. Μέσω αυτής της διαδικασίας δίνεται η δυνατότητα στο μοντέλο να διαχειριστεί διάφορες εναλλαγές της φυσικής γλώσσας οι οποίες παρουσιάζουν πολλαπλά νοήματα ή μορφές. Η επιλογή της μεθόδου διακριτοποίησης εξαρτάται από την εργασία (task) που πραγματοποιείται και την γλώσσα που επεξεργάζεται το μοντέλο κάθε φορά.
Η διαδικασία διακριτοποίησης (tokenization) είναι ένα σημαντικό βήμα στην διαδικασία προ-επεξεργασίας των μοντέλων Transformers και παίζει πολύ σημαντικό ρόλο στην τελική απόδοση του μοντέλου που εκπαιδεύεται ή χρησιμοποιείται.

Στο μοντέλο που εκπαιδεύσαμε η προετοιμασία της οποίας μεγάλο μέρος ήταν η διαδικασία του tokenizer είναι η ακόλουθη.

```
batch_size=4
encoder_max_length=512
decoder_max_length=128

def process_data_to_model_inputs(batch):
    #tokenizing inputs and labels
    inputs = tokenizer(batch["article"], padding="max_length", truncation=True, max_length=encoder_max_length)
    outputs = tokenizer(batch["highlights"], padding="max_length", truncation=True, max_length=decoder_max_length)

    batch["input_ids"] = inputs.input_ids
    batch["attention_mask"] = inputs.attention_mask
    batch["decoder_input_ids"] = outputs.input_ids
    batch["decoder_attention_mask"] = outputs.attention_mask
    batch["labels"] = outputs.input_ids.copy()

  # because BERT automatically shifts the labels, the labels correspond exactly to `decoder_input_ids`.
  # We have to make sure that the PAD token is ignored
    batch["labels"] = [[-100 if token == tokenizer.pad_token_id else token for token in labels] for labels in batch["labels"]]

    return batch
```

Όπως μπορούμε να δούμε η διαδικασία διακριτοποίησης ορίζει τις εισόδους να έχουν μήκος 512 χαρακτήρες και τις εξόδους 128 χαρακτήρες και το μέγεθος του συνόλου εκτέλεσης κάθε φορά είναι ίσο με 4.

Στην συνέχεια προετοιμάζουμε τα σύνολα δεδομένων εκπαίδευσης και επικύρωσης με βάση τις παραμέτρους που ορίσαμε παραπάνω.

```
train_data = train_data.select(range(32))

#preparing training data
train_data=train_data.map(
    process_data_to_model_inputs,
    batched=True,
    batch_size=batch_size,
    remove_columns=['article','highlights','id']
)

train_data.set_format(
    type='torch',columns=["input_ids", "attention_mask", "decoder_input_ids", "decoder_attention_mask", "labels"],
)

#preparing validation data
val_data=val_data.map(
    process_data_to_model_inputs,
    batched=True,
    batch_size=batch_size,
    remove_columns=['article','highlights','id']
)

val_data.set_format(
    type='torch',columns=["input_ids", "attention_mask", "decoder_input_ids", "decoder_attention_mask", "labels"],
)
```

Η διαδικασία εκπαίδευσης που ακολουθούμε είναι η διαδικασία sequence to sequence[12] (seq2seq) η οποία χρησιμοποιείται για την μετατροπή ακολουθιών, στην περίπτωση μας κειμένου, από ένα νοηματικό πλαίσιο σε ένα άλλο. Να σημειωθεί σε αυτό το σημείο ότι ένα μοντέλο Encoder – Decoder σαν αυτό που εκπαιδεύουμε σε αυτό το παράδειγμα μπορεί να αναφέρεται και από μόνο του σαν sequence to sequence.
Οι παράμετροι που δόθηκαν για την εκπαίδευση είναι οι ακόλουθες.

```
@dataclass
class Seq2SeqTrainingArguments(TrainingArguments):
    label_smoothing: Optional[float] = field(
        default=0.0, metadata={"help": "The label smoothing epsilon to apply (if not zero)."}
    )
    sortish_sampler: bool = field(default=False, metadata={"help": "Whether to SortishSamler or not."})
    predict_with_generate: bool = field(
        default=False, metadata={"help": "Whether to use generate to calculate generative metrics (ROUGE, BLEU)."}
    )
    adafactor: bool = field(default=False, metadata={"help": "whether to use adafactor"})
    encoder_layerdrop: Optional[float] = field(
        default=None, metadata={"help": "Encoder layer dropout probability. Goes into model.config."}
    )
    decoder_layerdrop: Optional[float] = field(
        default=None, metadata={"help": "Decoder layer dropout probability. Goes into model.config."}
    )
    dropout: Optional[float] = field(default=None, metadata={"help": "Dropout probability. Goes into model.config."})
    attention_dropout: Optional[float] = field(
        default=None, metadata={"help": "Attention dropout probability. Goes into model.config."}
    )
    lr_scheduler: Optional[str] = field(
        default="linear", metadata={"help": f"Which lr scheduler to use."}
    )
```

Με τις παραπάνω παραμέτρους ορισμένες και μετά την λήψη των συνόλων που θα χρησιμοποιηθούν ως μετρικές (rouge) μπορούμε να προχωρήσουμε στον ορισμό των παραμέτρων εκπαίδευσης, την εκπαίδευση και την αξιολόγηση του μοντέλου αρχικά με το εσωτερικό σύνολο επικύρωσης και στην συνέχεια με τις μετρικές που ορίσαμε. Παρακάτω θα παρουσιαστούν οι παράμετροι εκπαίδευσης και στην συνέχεια η αξιολόγηση του μοντέλου. Να σημειωθεί ότι οι παράμετροι εκπαίδευσης διαφέρουν ανάλογα τις δυνατότητες του κάθε μηχανήματος.

```
training_args = Seq2SeqTrainingArguments(
    output_dir="./",
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    predict_with_generate=True,
    evaluate_during_training=True,
    do_train=True,
    do_eval=True,
    logging_steps=2,
    save_steps=16,
    eval_steps=4,
    warmup_steps=1,
    max_steps=16,
    overwrite_output_dir=True,
    save_total_limit=3,
    fp16=True,
)

trainer = Seq2SeqTrainer(
    model=bert2bert,
    args=training_args,
    compute_metrics=compute_metrics,
    train_dataset=train_data,
    eval_dataset=val_data,
)
trainer.train()
```

Μετά την εκπαίδευση του μοντέλου παρουσιάζονται οι μετρικές αξιολόγησης της απόδοσης του, οι οποίες παρουσιάζονται στον επόμενο πίνακα.

Step	Training Loss	Validation Loss	Rouge2 Precision	Rouge2 Recall	Rouge2 Fmeasure
4	10.039099	10.122349	0.000000	0.000000	0.000000
8	7.995964	7.960626	0.001800	0.002300	0.002000
12	7.675583	7.756617	0.005500	0.015100	0.008000
16	7.481323	7.709305	0.004800	0.013600	0.007000

Η δεύτερη εφαρμογή που θα παρουσιάσουμε είναι και ο τελικός στόχος της εργασίας η ανάλυση συναισθήματος[13] (sentiment analysis). Η διαδικασία αυτή είναι η αυτοματοποιημένη αναγνώριση του συναισθήματος του κειμένου και εάν αυτό είναι θετικό, αρνητικό ή ουδέτερο. Με άλλα λόγια είναι μία εργασία (task) της επεξεργασίας φυσικής γλώσσας η οποία αναλύει την πολικότητα του κειμένου όσον αφορά το συνολικό συναίσθημα που παρουσιάζει.

Το σύνολο δεδομένων που θα χρησιμοποιήσουμε σε αυτό το παράδειγμα ονομάζεται IMDB sentimental analysis[14]. Η IMDb είναι μία διαδικτυακή βάση δεδομένων που έχει να κάνει με σειρές, ταινίες, βιντεοπαιχνίδια και άλλα περιεχόμενα διασκέδασης περιέχοντας τους συντελεστές, τις εταιρίες παραγωγής, περιλήψεις της υπόθεσης, αξιολογήσεις και κρητικές από χρήστες.
Το σύνολο δεδομένων που θα χρησιμοποιήσουμε αποτελείται από 50.000 κριτικές ταινιών και είναι ειδικά σχεδιασμένο για επεξεργασία φυσικής γλώσσας και ανάλυση κειμένου. Το συγκεκριμένο σύνολο δεδομένων είναι ειδικά σχεδιασμένο για δυαδική ανάλυση συναισθήματος, δηλαδή είτε θετικό είτε αρνητικό και αποτελείται από 25.000 δείγματα για εκπαίδευση και 25.000 δείγματα για επικύρωση. Επομένως ο τελικός στόχος όσον αφορά την χρήση αυτού του συνόλου δεδομένων είναι ο καθορισμός του συναισθήματος όσον αφορά κάθε κριτική ταινίας.

Όπως αναφέραμε και στην πρώτη εφαρμογή το κείμενο πρέπει να μετατραπεί σε μορφή κατανοητή από το μοντέλο, επομένως θα χρησιμοποιήσουμε ένα tokenizer βασισμένο στο μοντέλο BERT το distilbert[15], το οποίο έχει εκπαιδευτεί χρησιμοποιώντας 40% λιγότερες παραμέτρους από το αρχικό BERT.

```
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
```

Ομοίως θα χρησιμοποιήσουμε και το ίδιο μοντέλο ώστε να το εκπαιδεύσουμε και να βελτιώσουμε την ακρίβεια του στην εργασία (task) της ανάλυσης συναισθήματος σε κείμενο. Το παραμετροποιημένο μοντέλο που θα εκπαιδεύσουμε είναι το ακόλουθο:

```
model=AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=2)
```

Στην συνέχεια με την χρήση του tokenizer θα οργανώσουμε και θα προ-επεξεργαστούμε τα σύνολα εκπαίδευσης και δοκιμής που θα χρησιμοποιηθούν στο μοντέλο. Το βήμα της προ-επεξεργασίας είναι το ακόλουθο:

```
def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True)

tokenized_train = small_train_dataset.map(preprocess_function, batched=True)
tokenized_test = small_test_dataset.map(preprocess_function, batched=True)
```

Στην συνέχεια θα ορίσουμε τις παραμέτρους εκπαίδευσης, θα αρχικοποιήσουμε το μοντέλο και θα το εκπαιδεύσουμε με το σύνολο δεδομένων εκπαίδευσης. Η διαδικασία που περιγράψαμε είναι η ακόλουθη:

```
training_args = TrainingArguments(
    output_dir=repo_name,
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=2,
    weight_decay=0.01,
    save_strategy="epoch",
    push_to_hub=True,
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_train,
    eval_dataset=tokenized_test,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)
```

Μετά την διαδικασία εκπαίδευσης η εσωτερική αξιολόγηση του μοντέλου (το σύνολο δοκιμής) δίνει τα ακόλουθα αποτελέσματα.

```
TrainOutput(global_step=376, training_loss=0.3022774229658411, metrics={'train_runtime': 17824.5947,
'train_samples_per_second': 0.337, 'train_steps_per_second': 0.021, 'total_flos': 782725021021056.0,
'train_loss': 0.3022774229658411, 'epoch': 2.0})
```

Έχοντας ορίσει την συνάρτηση υπολογισμού των μετρικών του μοντέλου η οποία έχει την ακόλουθη μορφή μπορούμε να αξιολογήσουμε την απόδοση του μοντέλου που μόλις εκπαιδεύσαμε.

```
def compute_metrics(eval_pred):
    load_accuracy = load_metric("accuracy")
    load_f1 = load_metric("f1")

    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    accuracy = load_accuracy.compute(predictions=predictions, references=labels)["accuracy"]
    f1 = load_f1.compute(predictions=predictions, references=labels)["f1"]
    return {"accuracy": accuracy, "f1": f1}
```

Τα αποτελέσματα της αξιολόγησης είναι τα ακόλουθα.

```
{'eval_loss': 0.3215665817260742,
 'eval_accuracy': 0.8833333333333333,
 'eval_f1': 0.8844884488448845,
 'eval_runtime': 243.785,
 'eval_samples_per_second': 1.231,
 'eval_steps_per_second': 0.078,
 'epoch': 2.0}
```

Οι δύο βασικότερες τιμές που παρουσιάζονται είναι το eval_loss και το eval_accuracy, οι οποίες αναπαριστούν την απόσταση της πρόβλεψης με την πραγματική τιμή και την ακρίβεια στις προβλέψεις του μοντέλου αντίστοιχα. Οι αριθμοί που παρουσιάζονται έχουν ελάχιστη τιμή το 0 και μέγιστη το 1 και αντιστοιχούν σε ποσοστιαίες μονάδες, με το eval_loss να

αντιστοιχεί σε τιμή σχεδόν ίση με 32% και το eval_accuracy σχεδόν ίσο με 88%.
Η μετρική loss έχει σαν ιδανική τιμή το 0%, το οποίο σημαίνει ότι όλες οι προβλέψεις γίνονται με 100% επιτυχία, ενώ η μετρική accuracy έχει ως ιδανική τιμή το 100%. Οι τιμές που επέστρεψε η δική μας εκπαίδευση μπορούν να ερμηνευτούν ως απόκλιση από την πραγματική τιμή ίση με 32% και ακρίβεια πρόβλεψης ίση με 88%.

Τέλος με την χρήση του πακέτου pipeline[16] θα παρουσιάσουμε έτοιμα παραδείγματα ανάλυσης συναισθημάτων σε κείμενο.
Το pipeline είναι ένα πακέτο που χρησιμοποιεί προγραμματιστική διεπαφή (API, application programming interface) και χρησιμοποιείται για να καλέσει προ-εκπαιδευμένα μοντέλα μηχανικής μάθησης. Αποτελείται από 170 προ-εκπαιδευμένα μοντέλα και τα πακέτα που χρειάζονται για να λειτουργήσουν.
Σε αυτό το παράδειγμα θα χρησιμοποιήσουμε pipelines τα οποία χρησιμοποιούν μοντέλα της αρχιτεκτονικής Transformers. Με αυτό το πακέτο μπορούμε να χρησιμοποιήσουμε προ-εκπαιδευμένα μοντέλα για διάφορες εργασίες (tasks) επεξεργασίας φυσικής γλώσσας.

Θα παρουσιάσουμε την εισαγωγή (import) του πακέτου στο σύστημα και στην συνέχεια την επίδειξη της λειτουργικότητας του σε κάποιες εργασίες.
Η εισαγωγή του πακέτου γίνεται με τον εξής τρόπο:

```
from transformers import pipeline
```

Η πρώτη εργασία που θα παρουσιάσουμε είναι η ανάλυση συναισθήματος στην οποία έχουμε ήδη αναφερθεί. Σε αυτό το παράδειγμα θα χρησιμοποιήσουμε το σύνολο δεδομένων από το twitter με κάποια συγκεκριμένα παραδείγματα για να παρουσιάσουμε την λειτουργικότητα[17]. Σε αυτό το παράδειγμα θα χρησιμοποιήσουμε το ίδιο μοντέλο που εκπαιδεύσαμε παραπάνω, με την μόνη διαφορά ότι είναι προ-εκπαιδευμένο. Η επιλογή του μοντέλου γίνεται με τον ακόλουθο τρόπο.

```
sentiment_analysis = pipeline(model="distilbert-base-uncased-finetuned-sst-2-english")
```

Με την χρήση αυτού του μοντέλου θα προχωρήσουμε σε δοκιμές με την έξοδο να παρουσιάζει το συνολικό συναίσθημα του κειμένου.

```
text="almost got a kitty yesterday...but it didn`t work out"
sentiment = sentiment_analysis(text)
print(sentiment)
```

```
[{'label': 'NEGATIVE', 'score': 0.9992730021476746}]
```

```
text="Happy Mothers Day!!!"
sentiment = sentiment_analysis(text)
print(sentiment)
```

```
[{'label': 'POSITIVE', 'score': 0.999862790107727}]
```

```
text="I am having serious cig craving... head for kitchen, let the non smoker weight gain begin"
sentiment = sentiment_analysis(text)
print(sentiment)
```

```
[{'label': 'NEGATIVE', 'score': 0.9974679946899414}]
```

```
text="I am freezing"
sentiment = sentiment_analysis(text)
print(sentiment)
```

```
[{'label': 'NEGATIVE', 'score': 0.9966983199119568}]
```

Όπως μπορούμε να παρατηρήσουμε για κάθε κείμενο μας δίνεται το συνολικό συναίσθημα καθώς και το ποσοστό σωστής πρόβλεψης του συναισθήματος όπου όπως μπορούμε να παρατηρήσουμε είναι αρκετά υψηλό, σχεδόν 100%.

Μία άλλη εργασία (task) την οποία αναφέραμε στην αρχή είναι η εργασία της αναγνώρισης της αναφερόμενης οντότητας. Σε αυτή την εργασία στόχος είναι η αναγνώριση της οντότητας στην οποία αναφέρεται η πρόταση, καθώς και η γραμματική υπόσταση της οντότητας.
Το μοντέλο που θα χρησιμοποιήσουμε βασίζεται πάλι στο BERT, με την διαφορά ότι έχει εκπαιδευτεί για ακριβώς αυτή την εργασία (Named Entity Recognition, NER) και το καλούμε με τον εξής τρόπο:

```
ner_classifier = pipeline(model="dslim/bert-base-NER-uncased", aggregation_strategy="simple")
```

Παρακάτω θα παρουσιάσουμε κάποια αποτελέσματα από την χρήση αυτού του προ-εκπαιδευμένου μοντέλου.

```
text="The bad weather is also blamed for a plane crash that killed the pilot and two passengers in Belize ."
entity = ner_classifier(text)
print(entity)
```

```
[{'entity_group': 'LOC', 'score': 0.99856573, 'word': 'belize', 'start': 93, 'end': 99}]
```

```
text="Guerrilla activity in Afghanistan has increased after a winter lull , but activity is down compared to past years ."
entity = ner_classifier(text)
print(entity)
```

```
[{'entity_group': 'LOC', 'score': 0.99853134, 'word': 'afghanistan', 'start': 22, 'end': 33}]
```

Στο αποτέλεσμα παρουσιάζεται η λέξη η οποία είναι η οντότητα, η θέση της μέσα στην πρόταση σε αρίθμηση χαρακτήρων, ο τύπος της οντότητας και η πιθανότητα σωστής επιλογής της οντότητας, η οποία όπως και στο προηγούμενο παράδειγμα είναι αρκετά υψηλή.

Η τελευταία εργασία(task) που θα παρουσιάσουμε είναι η γραμματική ανάλυση του κειμένου η οποία αναλύει τον ρόλο της κάθε λέξης σε μία πρόταση. Το μοντέλο που χρησιμοποιούμε είναι βασισμένο στην αρχιτεκτονική BERT και είναι το ακόλουθο.

```
] pos_tagger=pipeline(model="vblagoje/bert-english-uncased-finetuned-pos",aggregation_strategy="simple")
```

Μερικά αποτελέσματα από την χρήση αυτού του μοντέλου είναι τα ακόλουθα. Λόγω του μεγάλου μήκους του αποτελέσματος θα παρουσιαστούν σε μορφή κειμένου και όχι σε μορφή στιγμιότυπου οθόνης.

```
"Bulgaria 's center-left government has survived a no-confidence vote prompted
by a suspension of European Union subsidies ."
```

```
[{'entity_group': 'PROPN', 'score': 0.9988028, 'word': 'bulgaria', 'start': 0,
'end': 8}, {'entity_group': 'PART', 'score': 0.9976402, 'word': "' s", 'start':
9, 'end': 11}, {'entity_group': 'ADJ', 'score': 0.5753033, 'word': 'center',
'start': 12, 'end': 18}, {'entity_group': 'PUNCT', 'score': 0.99964154, 'word':
'-', 'start': 18, 'end': 19}, {'entity_group': 'ADJ', 'score': 0.961699, 'word':
'left', 'start': 19, 'end': 23}, {'entity_group': 'NOUN', 'score': 0.99877983,
'word': 'government', 'start': 24, 'end': 34}, {'entity_group': 'AUX', 'score':
0.9981902, 'word': 'has', 'start': 35, 'end': 38}, {'entity_group': 'VERB',
'score': 0.99935097, 'word': 'survived', 'start': 39, 'end': 47},
{'entity_group': 'DET', 'score': 0.9993857, 'word': 'a', 'start': 48, 'end':
49}, {'entity_group': 'ADJ', 'score': 0.4705848, 'word': 'no', 'start': 50,
'end': 52}, {'entity_group': 'PUNCT', 'score': 0.99964607, 'word': '-', 'start':
52, 'end': 53}, {'entity_group': 'NOUN', 'score': 0.99880654, 'word':
'confidence vote', 'start': 53, 'end': 68}, {'entity_group': 'VERB', 'score':
0.99918884, 'word': 'prompted', 'start': 69, 'end': 77}, {'entity_group': 'ADP',
'score': 0.9994517, 'word': 'by', 'start': 78, 'end': 80}, {'entity_group':
'DET', 'score': 0.9994814, 'word': 'a', 'start': 81, 'end': 82},
{'entity_group': 'NOUN', 'score': 0.9989151, 'word': 'suspension', 'start': 83,
'end': 93}, {'entity_group': 'ADP', 'score': 0.9994892, 'word': 'of', 'start':
94, 'end': 96}, {'entity_group': 'PROPN', 'score': 0.9936139, 'word': 'european
union', 'start': 97, 'end': 111}, {'entity_group': 'NOUN', 'score': 0.9989875,
'word': 'subsidies', 'start': 112, 'end': 121}, {'entity_group': 'PUNCT',
'score': 0.9996456, 'word': '.', 'start': 122, 'end': 123}]
```

```
"Thousands of birds were slaughtered after scientists detected H5N1 in two other parts of southern Niger back in February ."
```

```
[{'entity_group': 'NOUN', 'score': 0.9986935, 'word': 'thousands', 'start': 0, 'end': 9}, {'entity_group': 'ADP', 'score': 0.9991615, 'word': 'of', 'start': 10, 'end': 12}, {'entity_group': 'NOUN', 'score': 0.99918705, 'word': 'birds', 'start': 13, 'end': 18}, {'entity_group': 'AUX', 'score': 0.99757236, 'word': 'were', 'start': 19, 'end': 23}, {'entity_group': 'VERB', 'score': 0.99934465, 'word': 'slaughtered', 'start': 24, 'end': 35}, {'entity_group': 'SCONJ', 'score': 0.9986363, 'word': 'after', 'start': 36, 'end': 41}, {'entity_group': 'NOUN', 'score': 0.9982816, 'word': 'scientists', 'start': 42, 'end': 52}, {'entity_group': 'VERB', 'score': 0.9996012, 'word': 'detected', 'start': 53, 'end': 61}, {'entity_group': 'NOUN', 'score': 0.7593071, 'word': 'h5n', 'start': 62, 'end': 65}, {'entity_group': 'NUM', 'score': 0.98131245, 'word': '##1', 'start': 65, 'end': 66}, {'entity_group': 'ADP', 'score': 0.9995858, 'word': 'in', 'start': 67, 'end': 69}, {'entity_group': 'NUM', 'score': 0.9982451, 'word': 'two', 'start': 70, 'end': 73}, {'entity_group': 'ADJ', 'score': 0.9988933, 'word': 'other', 'start': 74, 'end': 79}, {'entity_group': 'NOUN', 'score': 0.9987759, 'word': 'parts', 'start': 80, 'end': 85}, {'entity_group': 'ADP', 'score': 0.9996026, 'word': 'of', 'start': 86, 'end': 88}, {'entity_group': 'ADJ', 'score': 0.99571437, 'word': 'southern', 'start': 89, 'end': 97}, {'entity_group': 'PROPN', 'score': 0.99870014, 'word': 'niger', 'start': 98, 'end': 103}, {'entity_group': 'ADV', 'score': 0.99727684, 'word': 'back', 'start': 104, 'end': 108}, {'entity_group': 'ADP', 'score': 0.9995678, 'word': 'in', 'start': 109, 'end': 111}, {'entity_group': 'PROPN', 'score': 0.99831486, 'word': 'february', 'start': 112, 'end': 120}, {'entity_group': 'PUNCT', 'score': 0.99963975, 'word': '.', 'start': 121, 'end': 122}]
```

```
"Many Sunnis have held street protests demanding a re-vote ."
```

```
[{'entity_group': 'ADJ', 'score': 0.9982584, 'word': 'many', 'start': 0, 'end': 4}, {'entity_group': 'PROPN', 'score': 0.99825925, 'word': 'sunni', 'start': 5, 'end': 10}, {'entity_group': 'NOUN', 'score': 0.7037432, 'word': '##s', 'start': 10, 'end': 11}, {'entity_group': 'AUX', 'score': 0.9984767, 'word': 'have', 'start': 12, 'end': 16}, {'entity_group': 'VERB', 'score': 0.99950516, 'word': 'held', 'start': 17, 'end': 21}, {'entity_group': 'NOUN', 'score': 0.99714625, 'word': 'street protests', 'start': 22, 'end': 37}, {'entity_group': 'VERB', 'score': 0.99946195, 'word': 'demanding', 'start': 38, 'end': 47}, {'entity_group': 'DET', 'score': 0.999463, 'word': 'a', 'start': 48, 'end': 49}, {'entity_group': 'NOUN', 'score': 0.9961145, 'word': 're', 'start': 50, 'end': 52}, {'entity_group': 'PUNCT', 'score': 0.9995486, 'word': '-', 'start': 52, 'end': 53}, {'entity_group': 'NOUN', 'score': 0.9983461, 'word': 'vote', 'start': 53, 'end': 57}, {'entity_group': 'PUNCT', 'score': 0.9996395, 'word': '.', 'start': 58, 'end': 59}]
```

Όπως μπορούμε να παρατηρήσουμε παραπάνω για κάθε λέξη της πρότασης το σύστημα επιστρέφει τον ρόλο της κάθε λέξης στην πρόταση, την πιθανότητα της σωστής κατηγοριοποίησης, την ίδια την λέξη και την αρχή και το τέλος της κάθε λέξης μετρώντας χαρακτήρες.

Συμπεράσματα

Η επεξεργασία φυσικής γλώσσας έχει παρουσιάσει πολύ μεγάλη εξέλιξη τα τελευταία χρόνια. Αυτή η εξέλιξη οφείλεται στην εισαγωγή μεθόδων μηχανικής μάθησης στο αντικείμενο και συγκεκριμένα στα μοντέλα Transformers, τα οποία από μόνα τους παρουσιάστηκαν ως μεγάλο βήμα προόδου στον τομέα της μηχανικής και της βαθιάς μάθησης.
Το μοντέλο BERT που παρουσιάσαμε σε αυτή την εργασία είναι ένα μοντέλο με ευρύ πεδίο εφαρμογών στον τομέα της επεξεργασίας φυσικής γλώσσας και παρουσιάζει μεγάλες δυνατότητες σε ένα ευρύ φάσμα εφαρμογών μηχανικής μάθησης. Με την χρήση αυτού του μοντέλου παρουσιάσαμε την εφαρμογή της ανάλυσης συναισθημάτων (sentiment analysis) η οποία εμφανίζει μεγάλη χρησιμότητα σε εφαρμογές του πραγματικού κόσμου καθώς αυτή η λειτουργία μπορεί να χρησιμοποιηθεί για αυτοματοποιημένη εξυπηρέτηση πελατών ή ταξινόμηση μηνυμάτων ηλεκτρονικού ταχυδρομείου ή όπως και το σύνολο δεδομένων στο οποίο εκπαιδεύσαμε το μοντέλο μας, την αυτοματοποιημένη κατηγοριοποίηση των κριτικών τις οποίες μπορούν να αναρτούν οι χρήστες σε πεδίο αξιολόγησης.

Πηγές

[1] Sentiment Analysis: https://www.annualreviews.org/content/journals/10.1146/annurev-linguistics-011415-040518

[2] Hugging Face Natural Language Processing: https://huggingface.co/learn/nlp-course/chapter1/2

[3] n-gram grammars: https://web.stanford.edu/~jurafsky/slp3/3.pdf

[4] tf-idf: https://monkeylearn.com/blog/what-is-tf-idf/

[5] Pattern Recognition and Machine Learning: https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf

[6] Activation Functions: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[7] What is PyTorch?: https://towardsdatascience.com/introduction-to-py-torch-13189fb30cb3

[8] Hugging Face Transformers: https://huggingface.co/docs/transformers/index

[9] Hugging Face BERT: https://huggingface.co/docs/transformers/model_doc/bert

[10] CNN/DailyMail Dataset: https://paperswithcode.com/dataset/cnn-daily-mail-1

[11] Tokenization: https://medium.com/@lokaregns/preparing-text-data-for-transformers-tokenization-mapping-and-padding-9fbfbce28028

[12] Seq2Seq: https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

[13] Sentiment Analysis Hugging Face: https://huggingface.co/blog/sentiment-analysis-python

[14] IMDb Dataset: https://www.kaggle.com/code/karan842/imdb-sentimental-analysis-ml-lstm-bert-88

[15] distilbert: https://huggingface.co/docs/transformers/model_doc/distilbert

[16] Pipelines KDNuggets: https://www.kdnuggets.com/2023/02/simple-nlp-pipelines-huggingface-transformers.html

[17] Dataset for Sentiment Analysis: https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset

[18] Mastering Transformers:https://www.packtpub.com/product/mastering-transformers/9781801077651