ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Μοντέλα Transformer για την Ταξινόμηση Στιχουργών** **Transformer Models for Song Lyrics Writers Classification**
Ονοματεπώνυμο Φοιτητή	**Μιχαήλ Κοντοσώρος**
Πατρώνυμο	**Μιλτιάδης**
Αριθμός Μητρώου	**ΜΠΣΠ21022**
Επιβλέπων	**Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Μάιος 2023**

Τριμελής Εξεταστική Επιτροπή

Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής	Γεώργιος Τσιχριντζής Καθηγητής	Ευάγγελος Σακκόπουλος Αναπληρωτής Καθηγητής

Περιεχόμενα

Περίληψη

Η παρούσα πτυχιακή αναφέρεται στην αρχιτεκτονική των μοντέλων βαθιάς μάθησης (transformers). Γίνεται ανάλυση της αρχιτεκτονικής καθώς επίσης έχει πραγματοποιηθεί η υλοποίηση του συγκεκριμένου μοντέλου σε στοίχους τραγουδιών ώστε να ταξινομήσουν το κάθε στοίχο με τον αντίστοιχο καλλιτέχνη που το έχει δημιουργήσει.

Αρχικά αναλύονται συναφείς προσεγγίσεις προβλημάτων που σχετίζονται με δεδομένα κειμένων που χρησιμοποιούν ως μοντέλο την αρχιτεκτονική των transformers. Επίσης γίνεται περιγραφή των διαδικασιών που χρειάζονται τα δεδομένα πριν από την εκπαίδευση του μοντέλου καθώς και η ανάλυση των δεδομένων έτσι ώστε η διαδικασία της εκπαίδευσης να πραγματοποιηθεί με τον σωστό τρόπο.

Ακολούθως αναπτύσσεται αναλυτικά η αρχιτεκτονική των transformers έτσι ώστε να γίνει κατανοητή η λειτουργεία τους με την οποία αυτά τα μοντέλα μπορούν να επιτυγχάνουν αποτελέσματα που μοιάζουν με εκείνα του ανθρώπου. Σε αρκετά σημεία γίνεται αναλυτική επεξήγηση μέσω εικόνων έτσι ώστε να είναι πιο εύκολη η κατανόηση της αρχιτεκτονικής.

Στην συνέχεια γίνεται αναλυτική περιγραφή των δεδομένων με τα οποία πραγματοποιήθηκε η εκπαίδευση του μοντέλου καθώς επίσης αναφέρεται και όλη η προεργασία των δεδομένων που ήταν απαραίτητη να γίνει πριν την εκπαίδευση.

Στο τέλος παρουσιάζονται τα τελικά συμπεράσματα και οι περιοχές για περαιτέρω έρευνα.

Λέξεις – Κλειδιά: Μοντέλα Transformers , δεδομένα, εκπαίδευση, encoder , decoder

Abstract

This thesis focuses on the architecture of deep learning models (transformers). It analyzes the architecture and presents the implementation of this specific model on song lyrics to classify each line with its corresponding artist who created it.

Initially, related approaches to text data problems that utilize the transformer architecture as a model are examined. Furthermore, the necessary data preprocessing steps before model training are described, along with an analysis of the data to ensure proper training process.

Next, the transformer architecture is elaborated in detail to provide an understanding of how these models can achieve results similar to human-like performance. Visual illustrations are used at several points to facilitate comprehension of the architecture.

Then, a detailed description of the training data used for the model is provided, along with the preprocessing steps required prior to training.

Finally, the thesis concludes with the final findings and areas for further research.

ΚΕΦΑΛΑΙΟ 1

1.1 Εισαγωγή στη Τεχνητή Νοημοσύνη

Στις μέρες μας η τεχνίτη [1] νοημοσύνη έχει ευρεία χρήση στην καθημερινή μας ζωή και έχει ως σκοπό την μίμηση της ανθρώπινης συμπεριφοράς. Αυτή η προσομοίωση γίνεται μέσω της ανάπτυξης υπολογιστικών συστημάτων. Ο τρόπος με τον οποίο ένας υπολογιστής μπορεί να μάθει μέσα από την διαδικασία της εκπαίδευσης μοιάζει αρκετά με εκείνη του ανθρώπου. Μέσα από την επανάληψη και την συλλογή των δεδομένων οι αλγόριθμοι αρχίζουν να μαθαίνουν για το πρόβλημα στο οποίο εκπαιδεύονται να λύσουν. Η ΤΝ έχει 3 βασικά πλεονεκτήματα:

- Ταχύτητα μάθησης
- Εξοικονόμηση χρόνου
- Χρήση σε επαναλαμβανόμενα και επικίνδυνα επαγγέλματα

Η ταχύτητα είναι ένα από τα πιο βασικά πλεονεκτήματα. Ένας υπολογιστής μπορεί πλέον να μάθει πολύ πιο γρήγορα από έναν ανθρώπινο εγκέφαλο.

Επίσης, άλλο ένα πλεονέκτημα που σχετίζεται με την ταχύτητα είναι και η δυνατότητα γρηγορότερης λήψης αποφάσεων. Πολλές επιχειρήσεις χρησιμοποιούν την ΤΝ διότι μπορούν να έχουν πιο γρήγορα αποτελέσματα και πιο αξιόπιστα από ότι θα είχαν αν στη θέση της ΤΝ υπήρχε ο άνθρωπος. Για παράδειγμα, αν μια εταιρεία καθημερινά πρέπει να εξάγει πληροφορία από έναν μεγάλο αριθμό δεδομένων που μπορεί να δέχεται καθημερινά, από θέμα ταχύτητας θα ήταν πολύ χρονοβόρο στη θέση της ΤΝ να υπήρχε ο άνθρωπος αλλά και πιο δαπανηρό αφού η εταιρεία θα χρειαζόταν μεγάλο αριθμό υπαλλήλων.

Πολλές φορές σε επαγγέλματα όπου ο ρυθμός είναι μονότονος υπάρχει ο κίνδυνος της ανθρώπινης κόπωσης. Αυτό συμβαίνει γιατί ο ανθρώπινος εγκέφαλος δεν μπορεί να ενεργοποιήσει την δημιουργικότητα του και την περιέργεια του.

Με την ΤΝ πολλά τέτοια επαγγέλματα , όπως για παράδειγμα η επιλογή των ποιοτικών φρούτων στην παραγωγή ενός εργοστασίου, θα μπορούσε η ανθρώπινη παρέμβαση να μην υπάρχει. Η ΤΝ έχει ευρεία χρήση σχεδόν σε όλα τα επίπεδα, από προβλήματα οπτικής αντίληψης μέχρι εξαγωγή πληροφορίας από κείμενα ή ακόμα και ταξινόμηση κειμένων ανάλογα με το περιεχόμενο τους.

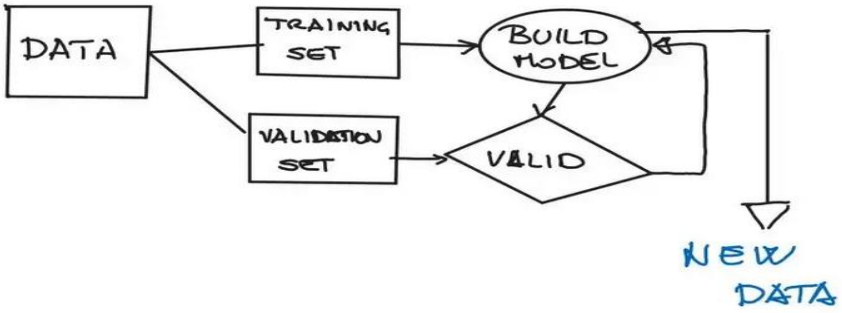
Μέχρι στιγμής αναλύσαμε τον όρο Τεχνίτη Νοημοσύνη και είδαμε τα οφέλη που μπορεί να προσφέρει. Με λίγα λόγια με τον όρο ΤΝ αναφερόμαστε στο πεδίο της επιστήμης των υπολογιστών που στόχος τους είναι η δημιουργία έξυπνων συστημάτων που εκτελούν εργασίες που συνήθως απαιτούν ανθρώπινη νοημοσύνη. Μία υποκατηγορία της ΤΝ είναι το Machine Learning ή ML [2] . Με τον όρο Machine Learning εννοούμαι όλες τις προσεγγίσεις που επιτρέπουν στις μηχανές να μαθαίνουν μέσα από δεδομένα. Οι αλγόριθμοι προσπαθούν να ελαχιστοποιήσουν τα σφάλματα και να μεγιστοποιήσουν την πιθανότητα ώστε η πρόβλεψη που θα κάνουν να είναι αληθείς.

Υπάρχουν τριών ειδών προσεγγίσεις μηχανικής μάθησης:

- Supervised learning
- Unsupervised learning
- Deep learning

Supervised learning

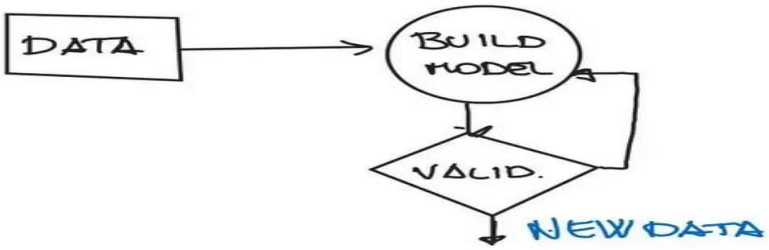
Η μέθοδος Supervised learning [10] είναι η πιο συνηθισμένη μέθοδος μηχανικής μάθησης. Η διαδικασία της εκπαίδευσης γίνεται με την χρήση μίας ομάδας δεδομένων που αναφέρεται και ως (training set [3]). Κάθε εισαγωγή δεδομένου μέσα στο σύστημα εμπεριέχει και μια ταμπέλα (label) το οποίο αντιπροσωπεύει το επιθυμητό αποτέλεσμα που θέλουμε να έχουμε (x1,y1),(x2,y2) …. (xn,yn).



Supervised learning schema [9]

Unsupervised learning

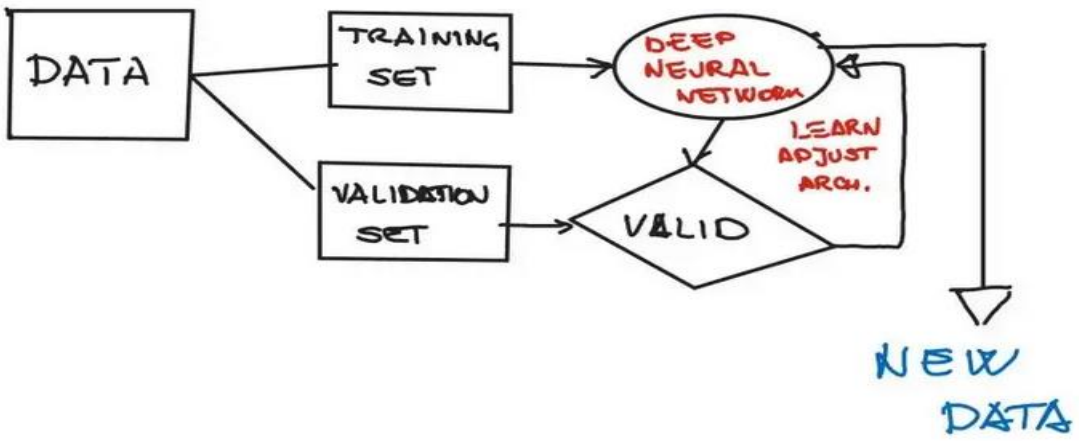
Αντίθετα με την μέθοδο Unsupervised learning [10] τα δεδομένα μας κατά την είσοδο στο μοντέλο δεν εμπεριέχουν labels, έτσι το μοντέλο οργανώνει τα δεδομένα αναζητώντας κοινά χαρακτηριστικά μεταξύ τους.



Unsupervised learning schema [9]

Deep learning

Τέλος , η μέθοδος Deep learning [10] αποτελεί ένα σημαντικό κομμάτι της μηχανικής μάθησης. Βασίζεται στον τρόπο με τον οποίο ο ανθρώπινος εγκέφαλος επεξεργάζεται τις πληροφορίες και μέσω αυτού μαθαίνει. Αποτελείται από πολλά επίπεδα όπου κάθε επίπεδο χρησιμοποιεί τις πληροφορίες από το προηγούμενο επίπεδο έτσι ώστε το μοντέλο να εκπαιδευτεί.



Deep learning schema [9]

1.2 Γενική περιγραφή του προβλήματος

Σκοπός της παρούσας πτυχιακής είναι να δούμε με τι ακρίβεια μπορεί ένας αλγόριθμος να αναγνωρίσει σε ποιόν καλλιτέχνη ανήκουν οι στοίχοι με τον οποίο θα τον τροφοδοτήσουμε. Το πρόβλημα αυτό ονομάζεται Text Classification [4]. Με τον όρο Text Classification ονομάζουμε την διαδικασία κατηγοριοποίησης ενός κειμένου σύμφωνα με το περιεχόμενο του. Αυτή η διαδικασία κατηγοριοποίησης γίνεται με την βοήθεια της NLP (Natural Language Processing) [5] διαδικασίας.

Το NLP είναι μία υποκατηγορία της ΤΝ και είναι αναγκαία η χρήση της έτσι ώστε ο υπολογιστής να καταλάβει την ανθρώπινη γλώσσα. Για να καταλάβουν οι υπολογιστές την φυσική γλώσσα και το νόημα – διασύνδεση που έχουν οι λέξεις μεταξύ τους μέσα σε ένα κείμενο πρέπει πρώτα να μετατραπούν σε ένα συγκεκριμένο format. Θα δούμε αυτή την διαδικασία πιο αναλυτικά στα επόμενα κεφάλαια.

Για την υλοποίηση της εργασίας χρησιμοποιήσαμε το Bert μοντέλο το οποίο ανήκει στη κατηγορία των Transformers. Το Bert [6] είναι ένα μοντέλο που δημιούργησε η Google το 2018 και πιο συγκεκριμένα είναι ένα pre-trained μοντέλο το οποίο έχει εκπαιδευτεί με έναν αρκετά μεγάλο αριθμό δεδομένων – κειμένων από την Wikipedia.

Για την υλοποίηση του προβλήματος η συλλογή των δεδομένων έγινε μέσα από την ιστοσελίδα Kaggle [7]. Τα δεδομένα περιείχαν 4 μουσικούς καλλιτέχνες και όλα τα τραγούδια που έχουν γράψει με τους αντίστοιχους στοίχους τους. Ο Classifier που θέλουμε να εκπαιδεύσουμε θα είναι σε θέση να ταξινομήσει μετά την εκπαίδευση τους στοίχους με τους αντίστοιχους δημιουργούς τους.

Να σημειωθεί ότι για καλύτερο αποτέλεσμα της εργασίας, εκτός από το backend που περιλαμβάνει τον κώδικα με το pre-processing των δεδομένων και την εκπαίδευση του μοντέλου δημιουργήθηκε και το frontend έτσι ώστε να υπάρχει μια ολοκληρωμένη μορφή του project. Το frontend δημιουργήθηκε με την βοήθεια του framework Django [8] και θα το δούμε πιο αναλυτικά σε επόμενο κεφάλαιο.

Στα επόμενα κεφάλαια θα δούμε πιο αναλυτικά την επεξεργασία των δεδομένων που πραγματοποιήθηκε πριν την εκπαίδευση καθώς επίσης και αρκετά σημεία από τον κώδικα σε μορφή εικόνων.

ΚΕΦΑΛΑΙΟ 2

Τα μοντέλα Transformers [11] αποτελούν την αρχή μίας τέταρτης βιομηχανικής επανάστασης η οποία ξεκίνησε το 2015. Από το 2015 η ΤΝ έχει αναπτυχθεί σε μεγάλο βαθμό σε προβλήματα που σχετίζονται με το NLP (Natural Language Processing) σε σχέση με το παρελθόν.

Σε λιγότερο από 5 χρόνια, η ΤΝ έχει ευρύ χρήση και μπορεί να χρησιμοποιηθεί ανά πάσα στιγμή με την χρήση API. Όλες αυτές οι τεχνολογικές επιτομές δεν δημιουργήθηκαν από πανεπιστήμια αλλά από μεγάλες εταιρείες όπως η Google που δημιούργησε το Bert αλλά και η Microsoft σε συνεργασία με την εταιρεία OpenAI που έφτιαξαν το GPT-3 [12] .

Αξίζει να σημειωθεί ότι τα μοντέλα Transformers διαφέρουν από τις άλλες αρχιτεκτονικές μοντέλων διότι μπορούν να χρησιμοποιηθούν σε ένα εύρη φάσμα εργασιών. Βέβαια για να δημιουργηθούν τέτοιου είδους τεχνολογίες δεν γίνεται μέσω ενός απλού υπολογιστή αλλά μέσω ειδικών υπολογιστών που ονομάζονται και ως supercomputers [13] .

Η εκπαίδευση των συγκεκριμένων μοντέλων γίνονται με δισεκατομμύρια δεδομένα, έτσι τα μοντέλα αυτά λόγο της δυναμικής τους ονομάζονται και μηχανές. Κατά την χρήση τους σε κάποιο πρόβλημα πολλές φορές δεν χρειάζεται κάποιο ιδιαίτερο fine tuning [14] ή κάποια αλλαγή των παραμέτρων τους.

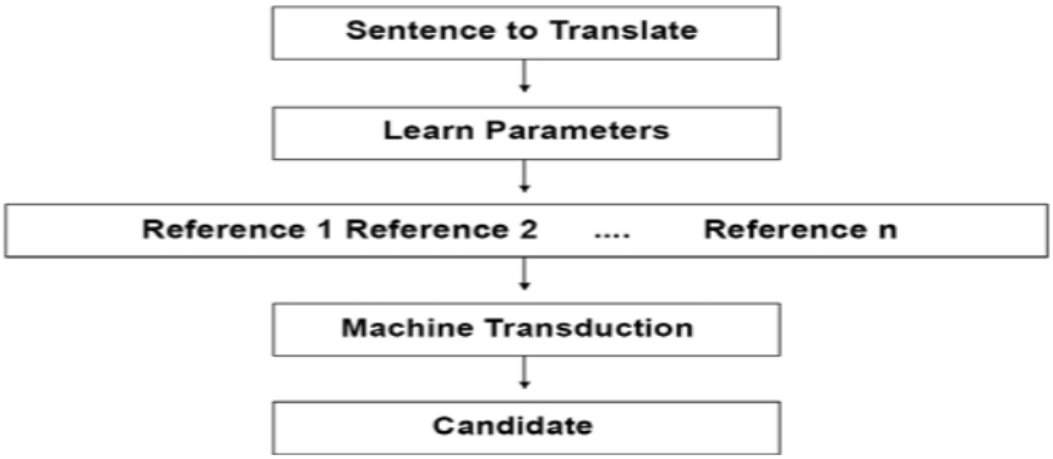
Στην συνέχεια θα δούμε προβλήματα NLP όπου μπορούμε να χρησιμοποιήσουμε την τεχνολογία Transformers.

2.1 Μετάφραση Κειμένων με την Χρήση Transformers

Μπορούμε πολύ εύκολα ως άνθρωποι όταν κάποιος μας περιγράψει κάτι να το φανταστούμε χρησιμοποιώντας την φαντασία μας. Για παράδειγμα αν κάποιος μας *πει τα λουλούδια στον κήπο μου είναι όμορφα* , έχουμε την ικανότητα να δημιουργήσουμε με την φαντασία μας μία εικόνα με λουλούδια μέσα στον κήπο. Παρόλο που δεν έχουμε δει τον συγκεκριμένο κήπο μπορούμε να φανταστούμε περίπου πως θα μοιάζει ο κήπος της περιγραφής πιο πάνω από τις εικόνες που έχουμε δει κατά την διάρκεια της ζωής μας.

Έτσι με παρόμοιο τρόπο μία μηχανή μαθαίνει από το μηδέν μέσω αναπαράστασης αριθμών οτιδήποτε θέλουμε να την διδάξουμε.

Για να γίνει η μετάφραση μίας γλώσσας χρειάζεται η γλώσσα Α να μετατραπεί στη γλώσσα Β. Η αρχιτεκτονική των Transformers βελτιώνει την νοημοσύνη αυτών των μοντέλων. Η αξιολόγηση των μοντέλων αυτών για το συγκεκριμένο πρόβλημα γίνεται με την χρήση του Blue Score [20] .



Διαδικασία μετάφρασης από το μοντέλο [18]

- Επιλογή της πρότασης που θέλουμε να μεταφράσουμε
- Εντοπισμός της σχέσης των λέξεων μεταξύ τους μέσα από την χρήση πολλαπλών παραμέτρων
- Εντοπισμός των διαφορετικών τρόπων με τους οποίους οι λέξεις συνδέονται μεταξύ τους
- Παραγωγή νέων ακολουθιών λέξεων
- Επιλογή μετάφρασης για μία λέξη ή πρόταση

Η διαδικασία της μετάφρασης [15] ξεκινάει πάντα από την επιλογή μίας πρότασης που θέλουμε να μεταφράσουμε από την γλώσσα Α. Η διαδικασία τελειώνει με το τελικό αποτέλεσμα που θα προκύψει από το μοντέλο. Το τελικό αποτέλεσμα αποτελεί την μετάφραση στην γλώσσα Β. Η ενδιάμεση διαδικασία μετατροπής της πρότασης από την γλώσσα Α στην γλώσσα Β ονομάζεται μεταγωγή της πρότασης.

Η διαδικασία της μεταγωγής επιτυγχάνεται στα μοντέλα Transformers με την χρήση encoder και του decoder.

2.2 Περίληψη Κειμένων με την Χρήση Transformers

Όπως είδαμε και πιο πάνω με την χρήση των Transformers μπορούμε να μεταφράσουμε μία γλώσσα σε μία άλλη. Μία άλλη χρήση που μπορούν να καλύψουν τα μοντέλα με αυτή την αρχιτεκτονική είναι η περίληψη κειμένων [16]. Τα μοντέλα T5 (Text-to-Text Transfer Transformer) [17] είναι μοντέλα επεξεργασίας φυσικής γλώσσας (NLP) που αναπτύχθηκαν από την ομάδα έρευνας της Google. Βασίζονται στην αρχιτεκτονική του Transformer και αποτελούν μεγάλα μοντέλα γλωσσικής επεξεργασίας που μπορούν να εκτελέσουν μια ευρεία γκάμα από εργασίες NLP, όπως απάντηση σε ερωτήσεις, σύνοψη κειμένου, μετάφραση και κατηγοριοποίηση κειμένου, μεταξύ άλλων.

Η μοναδική δυνατότητα των μοντέλων T5 είναι ότι σχεδιάστηκαν για να εκτελούν όλες αυτές τις εργασίες με τρόπο "κείμενο-προς-κείμενο" (text-to-text), δηλαδή μπορούν να λαμβάνουν ένα κομμάτι κειμένου ως είσοδο και να επιστρέφουν ένα διαφορετικό κομμάτι κειμένου που αντιστοιχεί σε μια συγκεκριμένη εργασία. Αυτή η προσέγγιση επιτρέπει την άνετη μεταφορά γνώσης μεταξύ των εργασιών, επιτρέποντας στα μοντέλα T5 να επιτυγχάνουν κορυφαίες επιδόσεις σε μια ευρεία γκάμα από NLP benchmarks.

Όταν ο άνθρωπος επικοινωνεί με κάποιον , πάντα ξεκινάμε με μία πρώτη ακολουθία λέξεων και στην συνέχεια με μία δεύτερη όπως βλέπουμε και από το παρακάτω σχήμα.

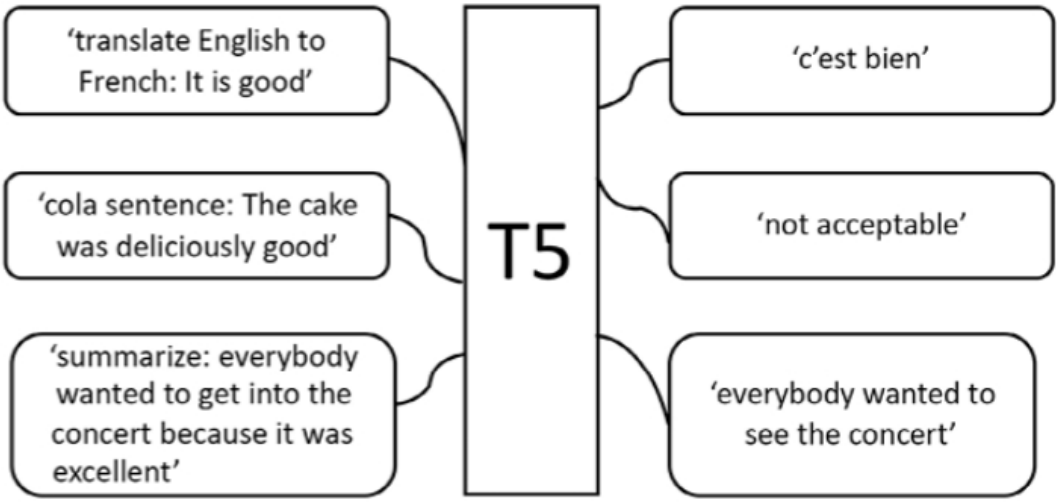
Διάγραμμα ακολουθίας λέξεων κατά την επικοινωνία του ανθρώπου [18]

Επίσης επικοινωνούμε μέσω της μουσικής και των ήχων. Καθώς επίσης η επικοινωνία μπορεί να γίνει μέσω της φυσικής κίνησης του σώματος και του χορού. Όταν η επικοινωνία γίνεται μέσω των λέξεων προσπαθούμε να κατανοήσουμε την σημασία όλων των λέξεων και δίνουμε μεγάλη προσοχή σε όλες τις λέξεις που βρίσκονται σε μία πρόταση. Όταν δεν μπορούμε να βγάλουμε εύκολο συμπέρασμα από μία πρόταση προσπαθούμε μέσω κάποιων λέξεων να καταλήξουμε σε ένα νόημα.

Τα μοντέλα T5 προκειμένου να αντιμετωπίσουν διαφορετικά προβλήματα που σχετίζονται με την επεξεργασία της φυσικής γλώσσας χρησιμοποιούν την λέξη “sequence”. Αυτή η μέθοδος είναι η

προσέγγιση “Text to Text” όπου το μοντέλο εκπαιδεύεται να μετατρέπει μια ακολουθία κειμένων σε μία άλλη ακολουθία κειμένων.

Στο παρακάτω διάγραμμα βλέπουμε διάφορες εισόδους – εξόδους που μπορεί να καλύψει ένα μοντέλο T5. Κάθε είσοδος – έξοδος αντιστοιχεί και σε διαφορετικό πρόβλημα.



Περιπτώσεις Text to Text με την χρήση T5 μοντέλου [18]

Η αρχιτεκτονική του T5 μοντέλου δεν διαφέρει πολύ από τα κλασικά Transformers μοντέλα. Τα μοντέλα αυτά έχουν έναν encoder και έναν decoder. Κάθε encoder αποτελείται από ένα multi – head attention layer και ένα feed forward layer. Ο ρόλος του multi-head attention layer είναι να επιλέγει την σημαντικότερη πληροφορία της εισόδου , ενώ το feed forward layer προωθεί τα αποτελέσματα του multi – head attention layer μέσα από ένα νευρωνικό δίκτυο. Επίσης κάθε decoder block αποτελείται από ένα multi-head attention layer, ένα encoder-decoder attention layer και ένα feed forward layer.

Το T5 μοντέλο χρησιμοποιεί έναν αριθμό encoder – decoder blocks έτσι ώστε να πετύχει τον σκοπό του. Κάθε block encoder – decoder δέχεται ως είσοδο τα αποτελέσματα του προηγούμενου block και δημιουργεί ένα νέο σύνολο το οποίο με την σειρά του θα περάσει στο επόμενο block.

Εφαρμογή των μοντέλων T5

Στην συνέχεια θα δούμε πως μπορούμε να χρησιμοποιήσουμε τα μοντέλα T5. Για την υλοποίηση του παραδείγματος θα πρέπει πρώτα να κάνουμε εγκατάσταση των Transformers. Η γλώσσα που θα χρησιμοποιήσουμε είναι η python, έτσι με την εντολή *pip install transformers* θα γίνει η εγκατάσταση των transformers. Στην συνέχεια θα εγκαταστήσουμε άλλη μια βιβλιοθήκη *pip install sentence piece.*

Αφού γίνει η εγκατάσταση θα εισάγουμε τις βιβλιοθήκες *torch* [19] και *json* στο πρόγραμμά μας:

```
import torch
import json
```

Μετά θα εισάγουμε τις παρακάτω βιβλιοθήκες:

```
from transformers import T5Tokenizer,
T5ForConditionalGeneration, T5Config
```

Στην συνέχεια θα εισάγουμε τον είδη εκπαιδευμένο tokenizer:

```
model =
T5ForConditionalGeneration.from_pretrained('
t5-large')
tokenizer = T5Tokenizer.from_pretrained('t5-
large')
```

Με την χρήση του tokenizer μπορούμε να μετατρέπουμε μία ακολουθία κειμένου σε μία ακολουθία από tokens. Τα tokens είναι μικρά κομμάτια του κειμένου όπως λέξεις ή σύμβολα. Υπάρχουν διάφοροι τρόποι ώστε να μετατρέψει κανείς το κείμενο σε tokens, αυτό εξαρτάται από το μοντέλο που θα χρησιμοποιήσει κάποιος. Για παράδειγμα, κάποια μοντέλα χρησιμοποιούν το κενό χαρακτήρα ως διαχωριστικό στοιχείο, ενώ άλλα χρησιμοποιούν τεχνικές όπως το byte-pair encoding (BPE) [21] για να διαχωρίσουν το κείμενο σε tokens.

Μετά θα επιλέξουμε την συσκευή με την οποία θα τρέξουμε το μοντέλο μας.

```
device = torch.device('cpu')
```

Στην συνέχεια θα ετοιμάσουμε μία συνάρτηση (function) η οποία θα δέχεται δύο παραμέτρους. Η πρώτη παράμετρος θα είναι το κείμενο και η δεύτερη θα είναι το μέγιστο μήκος του κειμένου που μπορεί να δεχτεί η συνάρτηση.

```
def summarize(text,ml):
```

Μέσα στην συνάρτηση θα προσθέσουμε λίγο κώδικα έτσι ώστε να γίνει ο καθαρισμός του κειμένου που πρόκειται να τροφοδοτήσουμε το μοντέλο . Αξίζει να σημειωθεί ότι για να χρησιμοποιήσουμε το μοντέλο T5 στο πρόβλημα της περίληψης πρέπει να προσθέσουμε το πρόθεμα *summarize* μπροστά από το κείμενό μας.

```
t5_prepared_Text = "summarize:
"+preprocess_text
```

Στην συνέχεια θα φορτώσουμε ένα κείμενο με σκοπό να παράξουμε την περίληψή του.

```
text ="""
The United States Declaration of Independence
was the first Etext
released by Project Gutenberg, early in 1971.
The title was stored
in an emailed instruction set which required
a tape or diskpack be
hand mounted for retrieval.  The diskpack was
the size of a large
cake in a cake carrier, cost $1500, and
contained 5 megabytes, of
which this file took 1-2%.  Two tape backups
were kept plus one on
paper tape.  The 10,000 files we hope to have
online by the end of
2001 should take about 1-2% of a comparably
priced drive in 2001.
"""
```

Αφού δηλώσαμε το κείμενο το οποίο θα τρέξουμε, πρέπει να καλέσουμε την συνάρτηση *summarize* έτσι ώστε να παράξουμε την περίληψή του.

```
print("Number of characters:",len(text))
summary=summarize(text,50)
print ("\n\nSummarized text: \n",summary)
```

```
Number of characters: 534
Preprocessed and prepared text:
 summarize: The United States Declaration of
Independence...
Summarized text:
 the united states declaration of independence
was the first etext published by project
gutenberg, early in 1971. the 10,000 files we
hope to have online by the end of2001 should
take about 1-2% of a comparably priced drive
in 2001. the united states declaration of
independence was the first Etext released by
project gutenberg, early in 1971
```

2.3 Ανάλυση συναισθημάτων με την Χρήση Transformers

Άλλη μία επιπλέον εφαρμογή των μοντέλων Transformers είναι η αναγνωρίσει των συναισθημάτων μέσα από το κείμενο [22] . Τα μοντέλα αυτά έχουν την ικανότητα να αναγνωρίζουν τις λέξεις που

εκφράζουν συναίσθημα και ουσιαστικά μπορούν να καταλήγουν σε ένα συμπέρασμα για το περιεχόμενο του κειμένου.

Ο στόχος είναι να μπορούν να εντοπίσουν αυτόματα και να εξάγουν την πληροφορία από το κείμενο, όπως το συναίσθημα που εκφράζει ο συγγραφέας μέσα από το κείμενο που έχει γράψει ή να εξάγουν την άποψη του συγγραφέα για το συγκεκριμένο κείμενο.

Η χρήση της ανάλυσης των συναισθημάτων με την χρήση των transformers μπορεί να χρησιμοποιηθεί στους παρακάτω τομείς :

Επιχειρήσεις

Η χρήση της ανάλυσης των συναισθημάτων μπορεί να χρησιμοποιηθεί έτσι ώστε να γίνεται η αναγνώριση των συναισθημάτων του πελάτη μέσα από τα σχόλια που έχει γράψει κατά την αγορά ενός προϊόντος. Επίσης σημαντικό ρόλο παίζουν οι κριτικές από τους πελάτες αλλά και η αναρτήσεις μέσα από τα κοινωνικά δίκτυα. Η επιχείρηση όταν μπορεί να κατανοήσει τον πελάτη θα είναι πολύ πιο εύκολο να αναπτυχθεί και να διορθωθεί.

Πολιτική

Η ανάλυση της κοινής γνώμης και του συναισθήματος είναι πολύ σημαντική στον χώρο της πολιτικής. Η κατανόηση των απαιτήσεων του πολίτη είναι πολύ σημαντική για έναν πολιτικό προκυμμένου να κερδίσει την εμπιστοσύνη των πολιτών. Επίσης η κατανόηση των απαιτήσεων που έχει ο πολίτης μπορεί να διευκολύνει στην λήψη των αποφάσεων που μπορεί να πάρει ένας πολιτικός.

Υγειονομική περίθαλψη

Στις μέρες μας η υγειονομική περίθαλψη παίζει καθοριστικό ρόλο για τον άνθρωπο. Πολλές φορές οι ασθενείς δεν είναι ευχαριστημένοι από τις συνθήκες που επικρατούν σε ένα δημόσιο νοσοκομείο ή ακόμα και σε ένα ιδιωτικό. Έτσι και εδώ σημαντικό ρόλο παίζει η χρήση ενός αυτοματοποιημένου συστήματος έτσι ώστε να παίρνονται σωστές αποφάσεις από τα διοικητικά μέλη με στόχο την καλύτερη ανταπόκριση του συστήματος.

Στην συνέχεια θα δούμε μια εφαρμογή των Transformers σε ανάλυση συναισθημάτων μέσα από ένα κείμενο. Για την εφαρμογή του παραδείγματος θα χρειαστεί να κάνουμε εγκατάσταση κάποιες βιβλιοθήκες.

```
pip install transformers
pip install torch
```

Στην συνέχεια θα εισάγουμε τις απαραίτητες βιβλιοθήκες στο πρόγραμμά μας και θα φορτώσουμε το είδη εκπαιδευμένο μοντέλο DistilBert καθώς επίσης και τον εκπαιδευμένο tokenizer.

```
from transformers import DistilBertTokenizer, DistilBertForSequenceClassification
import torch

tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-uncased')
```

Να σημειωθεί ότι ο tokenizer μπορεί να ξανά εκπαιδευτεί αν θέλουμε να το προσαρμόσουμε σε ένα συγκεκριμένο λεξιλόγιο.

Στην συνέχεια θα ορίσουμε το κείμενο που θέλουμε να περάσουμε στο μοντέλο μας . Αφού ορίσουμε το κείμενο μετά μένει να ορίσουμε τον encoder ο οποίος σε συνδυασμό με τον tokenizer θα μας μετατρέψουν τις λέξεις σε αριθμούς και επίσης σημαντικό σημείο για την συνέχεια είναι η μετατροπή

του κειμένου σε tensor. Η παράμετρος *max_length* την χρησιμοποιούμαι έτσι ώστε το μέγιστο μήκος του κειμένου να αποτελείται από 128 λέξεις. Αν ένα κείμενο είναι μικρότερο τότε χρησιμοποιούμαι την παράμετρο *padding* έτσι ώστε οι θέσεις των λέξεων που δεν θα καλυφτούν να συμπληρωθούν από έναν ειδικό χαρακτήρα έτσι ώστε το μήκος του κειμένου να είναι σταθερό στο 128.

Παρακάτω ορίζουμε το αντικείμενο του μοντέλου έτσι ώστε να το χρησιμοποιήσουμε για την πρόβλεψη που θέλουμε. Η μεταβλητή predictions μετατρέπει τα *logits* [23] σε πιθανότητες μέσα από την χρήση του *softmax*.

Τέλος εξάγουμε το αποτέλεσμα του μοντέλου με την πιθανότητα του αποτελέσματος.

```
text = "I love this product! It's amazing!"
encoded_text = tokenizer.encode_plus(text, max_length=128, padding='max_length', truncation=True, return_tensors='pt')
```

```
sentiment_labels = ['Negative', 'Positive']
sentiment_label = sentiment_labels[predictions.index(max(predictions))]
sentiment_prob = max(predictions)

print(f"Sentiment: {sentiment_label}")
print(f"Probability: {sentiment_prob:.2f}")
```

```
Sentiment: Positive
Probability: 0.96
```

2.4 Διαδικασία εκπαίδευσης του Tokenizer

Ο tokenizer είναι ένα εργαλείο που το χρησιμοποιούμαι στην επεξεργασία της φυσικής γλώσσας (NLP). Με την χρήση του tokenizer μπορούμε να διαχωρίσουμε ένα κείμενο σε μικρότερα κομμάτια, αυτά τα κομμάτια ονομάζονται “tokens”.

Τα tokens μπορεί να είναι λέξεις, φράσεις , σημεία στίξης, αριθμοί κλπ.. Με την χρήση του tokenizer μπορούμε να πετύχουμε πολλούς σκοπούς όπως:

- Απομάκρυνση σημείων στίξης και άλλων χαρακτήρων που δεν είναι λέξεις.
- Κανονικοποίηση των λέξεων όπως η μετατροπή των πεζών γραμμάτων σε κεφαλαία
- Μετατροπή του κειμένου σε αριθμούς έτσι ώστε να μπορεί να επεξεργαστεί από αλγορίθμους μηχανικής μάθησης

Ο tokenizer μπορεί να χωρίσει μία λέξη σε μικρότερα μέρη όπως για παράδειγμα την λέξη smaller, μπορεί να την κάνει δύο ξεχωριστές λέξεις, η πρώτη είναι το *“small”* και η δεύτερη το *“er”*. Αν ο tokenizer δεν γνωρίζει μία λέξη και δεν μπορεί επίσης να την χωρίσει σε μικρότερα μέρη τότε τα χαρακτηρίζει ως άγνωστα με την λέξη “unk_token”.

Τον συγκεκριμένο tokenizer θα τον εκπαιδεύσουμε με τις παρακάτω παραμέτρους.

- **files** = paths (το path στο οποίο βρίσκεται το dataset)
- **vocab_size** = 52_000 (το μέγεθος του λεξιλογίου που θα μάθει ο tokenizer)

- **min_frequency** = [] (ο ελάχιστος αριθμός όπου εμφανίζεται μία λέξη μέσα στο dataset)
- **special_tokens** = [] (μία λίστα με special tokens)

Η λίστα με τα special tokens είναι η εξής:

- < s >: start token
- < pad >: padding token
- < /s >: end token
- < unk >: unknown token
- < mask >: mask token

Στην συνέχεια θα δούμε σε κώδικα την διαδικασία που χρειάζεται προκειμένου να εκπαιδεύσουμε έναν tokenizer με δικά μας δεδομένα.

```
from pathlib import Path
from tokenizers import ByteLevelBPETokenizer
paths = [str(x) for x in Path(".").glob("**/
*.txt")]
# Initialize a tokenizer
tokenizer = ByteLevelBPETokenizer()
# Customize training
tokenizer.train(files=paths,
vocab_size=52_000, min_frequency=2,
special_tokens=[
    "<s>",
    "<pad>",
    "</s>",
    "<unk>",
    "<mask>",
])
```

Ως μοντέλο χρησιμοποιούμε τον ByteLevelBPETokenizer. Ο Byte Pair Encoding tokenizer ή αλλιώς BPE [24] είναι ένας αλγόριθμος που χρησιμοποιείται στην επεξεργασία της φυσικής γλώσσας έτσι ώστε να διαιρεί τις λέξεις σε μικρότερα μέρη.

Ένα πλεονέκτημα του BPE είναι ότι μπορεί να χρησιμοποιηθεί σε οποιαδήποτε γλώσσα ή συμβολοσειρά καθώς δεν απαιτεί προκαθορισμένο λεξιλόγιο.

Το dataset που χρησιμοποιήσαμε για την εκπαίδευση του tokenizer είναι κείμενα αποθηκευμένα σε ένα txt.

Παρακάτω βλέπουμε τα βήματα που χρειάζονται έτσι ώστε να αποθηκεύσουμε το μοντέλο μετά την εκπαίδευσή του.

```
#@title Step 4: Saving the files to disk
import os
token_dir = '/content/KantaiBERT'
if not os.path.exists(token_dir):
  os.makedirs(token_dir)
tokenizer.save_model('KantaiBERT')
```

Αφού ολοκληρώσαμε την εκπαίδευση του tokenizer τώρα είναι η ώρα να τον χρησιμοποιήσουμε.

```
#@title Step 5 Loading the Trained Tokenizer
Files
from tokenizers.implementations import
ByteLevelBPETokenizer
from tokenizers.processors import
BertProcessing
tokenizer = ByteLevelBPETokenizer(
    "./KantaiBERT/vocab.json",
    "./KantaiBERT/merges.txt",
)
```

Στην συνέχεια αφού φορτώσαμε τον εκπαιδευμένο tokenizer θα τρέξουμε μία πρόταση έτσι ώστε να δούμε αν ο tokenizer θα αναγνωρίσει όλες τις λέξεις προκειμένου να τις κάνει tokens.

Για να γίνει αυτό πρέπει να καλέσουμε την μέθοδο encoder έτσι ώστε να ξεκινήσει η διαδικασία του tokenization.

```
tokenizer.encode("The Critique of Pure
Reason.").tokens
```

Το αποτέλεσμα που θα προκύψει είναι μία λίστα μέσα στην οποία είναι ξεχωριστά κάθε λέξη από την πρόταση. Να σημειωθεί ότι το γράμμα G μπροστά από κάθε λέξη εκφράζει το κενό διάστημα μεταξύ των λέξεων.

```
['The', 'ĠCritique', 'Ġof', 'ĠPure',
'ĠReason', '.']
```

Επίσης μπορούμε να ζητήσουμε από το μοντέλο να μας πει πόσα tokens βρήκε. Προφανώς στο παράδειγμα αυτό δεν μας χρησιμεύει σε κάτι αυτό διότι η πρόταση είναι πολύ μικρή , αλλά θα ήταν πολύ χρήσιμο σε ένα μεγάλο dataset όπου θα ήταν δύσκολο με το μάτι να δούμε πόσα tokens το μοντέλο εντόπισε. Για να γίνει αυτό αφαιρούμε από το τέλος το .tokens.

```
tokenizer.encode("The Critique of Pure Reason.")
```

Το αποτέλεσμα δείχνει ότι το μοντέλο βρήκε 6 tokens.

```
Encoding(num_tokens=6, attributes=[ids, type_ids, tokens,
offsets, attention_mask, special_tokens_mask, overflowing])
```

ΚΕΦΑΛΑΙΟ 3

3.1 Ανάλυση αρχιτεκτονικής Transformer μοντέλου

Στα προηγούμενα κεφάλαια έγινε αναλυτική περιγραφή σε διάφορα μοντέλα Transformers και είδαμε κάποια παραδείγματα όπου τα μοντέλα αυτά μπορούν να χρησιμοποιούνται με μεγάλη αποτελεσματικότητα.

Σε αυτό το κεφάλαιο θα αναφερθούμε αναλυτικά στην αρχιτεκτονική των μοντέλων Transformers έτσι ώστε να κατανοήσουμε την λειτουργία τους.

Η γλώσσα σε όλες τη χώρες είναι ένα μέσω επικοινωνίας μεταξύ των ανθρώπων. Η ανάπτυξη των πολιτισμών δεν θα είχε πραγματοποιηθεί χωρίς τις γλώσσες. Η καθημερινότητα μας απαιτεί την χρήση της γλώσσας όπως για παράδειγμα:

- Η χρήση των emails
- Οι αναρτήσεις σχολίων στα μέσα κοινωνικής δικτύωσης
- Η αποστολή μηνυμάτων με την χρήση των smartphones
- Αναζήτηση πληροφορίας μέσα από το internet

Τον Δεκέμβριο του 2017 η Google Brain και η Google Research

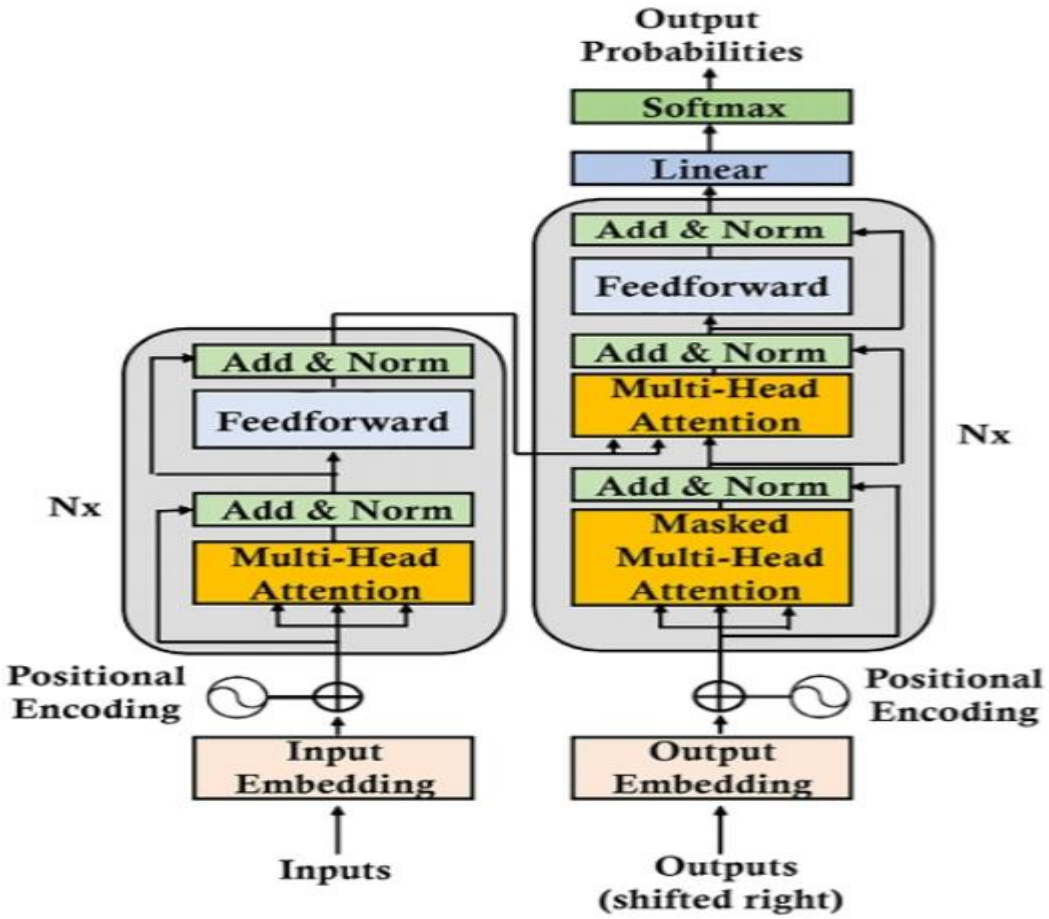
δημοσίευσαν το επιστημονικό άρθρο “Attention is All You Need”. Από εκείνη την στιγμή γεννήθηκε η ιδέα των Transformers. Η αρχιτεκτονική αυτή ξεπερνάει σε αποτελεσματικότητα κάθε μοντέλο που σχετίζεται με προβλήματα NLP. Τα μοντέλα Transformers εκπαιδεύονται πολύ πιο γρήγορα από τα υπόλοιπα μοντέλα και παρουσιάζουν μεγαλύτερη αποτελεσματικότητα.

Επίσης σε αυτό το κεφάλαιο θα αναφερθούμε στα βασικά χαρακτηριστικά που κάνουν αυτά τα μοντέλα ξεχωριστά.

Σε αυτό το κεφάλαιο θα καλύψουμε τα παρακάτω θέματα:

- Την αρχιτεκτονική των μοντέλων Transformers
- Το χαρακτηριστικό self-attention των μοντέλων αυτών
- Τον encoder – decoder
- Τα embeddings εισόδου – εξόδου
- Τα positional embeddings
- Multi – head attention
- Masked multi – attention
- Normalization
- Feedforward network
- Τις πιθανότητες εξόδου (Output probabilities)

Πάμε να δούμε κατευθείαν την αρχιτεκτονική των μοντέλων Transformers.



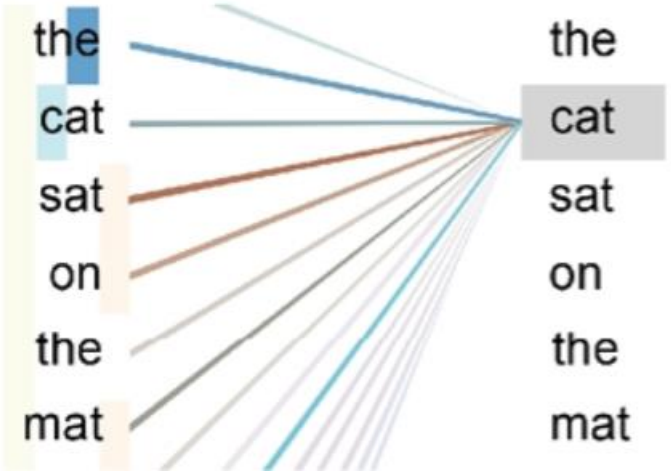
Διάγραμμα αρχιτεκτονικής των μοντέλων Transformers [18]

Στο δεξί τμήμα τα δεδομένα εισέρχονται μέσα στον encoder μέσω ενός *attention sublayer* και ενός *feedforward sublayer*.

Στο δεξί τμήμα βρίσκεται ο decoder μέσω του οποίου εισάγονται τα δεδομένα στόχου (target outputs). Η εισαγωγή αυτή γίνεται μέσω attention sublayers και ενός δικτύου feedforward sublayer. Ο attention μηχανισμός είναι ένας μηχανισμός “word to word” λειτουργίας. Ουσιαστικά ο attention μηχανισμός προσπαθεί να εντοπίσει την σημασία που έχει κάθε λέξη με όλες τις άλλες λέξεις μέσα σε ένα κείμενο ή πρόταση. Ας πάρουμε για παράδειγμα μία πρόταση:

```
The cat sat on the mat.
```

Ο attention μηχανισμός θα εντοπίσει την σχέση που έχει κάθε λέξη σε σχέση με τις υπόλοιπες λέξεις.



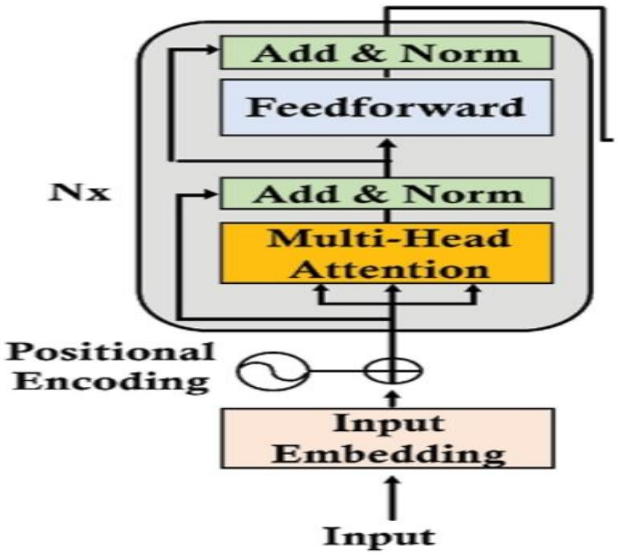
Λειτουργεία Attention μηχανισμού [18]

Ο attention μηχανισμός δημιουργεί μία πιο βαθιά σύγκριση μεταξύ της σχέσης που έχουν οι λέξεις μεταξύ τους.

Για κάθε attention sublayer , το μοντέλο Transformers δεν τρέχει μόνο μία φορά αλλά 8 φορές. Αυτή η λειτουργεία γίνεται σε παράλληλο χρόνο έτσι ώστε ο υπολογισμός που χρειάζεται να γίνει να μην καταναλώνει μεγάλο χρόνο.

3.2 Encoder

Τα layers του encoder και του decoder αποτελούνται από ένα σύνολο layers. Κάθε layer που βρίσκεται στον encoder έχει την παρακάτω μορφή:



Η αυθεντική δομή του encoder παραμένει ίδια και για τα 6 layers που βρίσκονται στο Transformer μοντέλο. Κάθε layer περιέχει δύο sublayers:

- Multi-headed attention mechanism

- Position – wise feedforward network

Η δομή των N = 6 layers του encoder είναι πανομοιότυπη, αλλά το περιεχόμενο σε κάθε layer δεν είναι πανομοιότυπο με το προηγούμενο layer.

Για παράδειγμα , το embedding layer βρίσκεται στο χαμηλό επίπεδο του encoder. Τα άλλα 5 layers του encoder δεν περιέχουν το embedding layer.

Επίσης, οι multi – head attention μηχανισμοί εκτελούν την ίδια λειτουργία από το 1 layer μέχρι το 6. Αξίζει να σημειωθεί ότι δεν εκτελούν το ίδιο έργο. Κάθε layer εκπαιδεύεται από το προηγούμενο layer και προσπαθούν να εξερευνήσουν διαφορετικούς τρόπους συσχέτισης μεταξύ των λέξεων μέσα σε μία πρόταση.

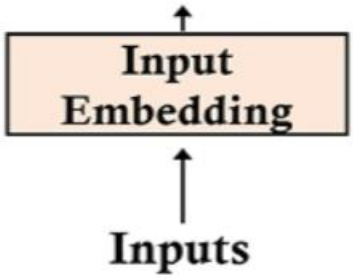
Η έξοδος από κάθε sublayer που βρίσκεται μέσα στο μοντέλο έχει μία σταθερή διάσταση, συμπεριλαμβανομένου και του embedding layer. Αυτή η διάσταση μπορεί να αλλάξει ανάλογα με το σκοπό που έχουμε. Στην αυθεντική αρχιτεκτονική των μοντέλων Transformers αυτή η διάσταση είναι 512.

Η χρήση της σταθερής διάστασης έχει πολλά θετικά που σχετίζονται με την εξοικονόμηση του χρόνου που χρειάζεται έτσι ώστε να εκπαιδευτεί το μοντέλο.

3.3 Embedding Layer

Σκοπός του Embedding layer είναι να μετατρέπει κάθε token που δέχεται κατά την είσοδο και να το μετατρέπει σε ένα vector με διαστάσεις 512.

Κατά την διαδικασία του tokenization όπως είπαμε και στα προηγούμενα κεφάλαια, μία πρόταση χωρίζεται σε λέξεις. Σκοπός είναι κάθε λέξη να μετατραπεί σε αριθμό, αυτό δεν δίνει όμως αρκετή πληροφορία για την σημασία της λέξης που έχει μέσα στην πρόταση. Έτσι, χρησιμοποιούμαι το Embedding layer έτσι ώστε να μας χαρτογράφησή και να μας δώσει τις συντεταγμένες που βρίσκεται κάθε λέξη μέσα σε έναν τρισδιάστατο χώρο.



Διάγραμμα Embedding Layer [18]

Ας δούμε παρακάτω ένα παράδειγμα:

```
The black cat sat on the couch and the brown
dog slept on the rug.
```

Θα δώσουμε προσοχή σε δύο λέξεις, την λέξη black και brown. Τα word embeddings των δύο λέξεων θα πρέπει να είναι όμοια. Παρακάτω θα παράξουμε τα word embeddings των δύο λέξεων.

```
black=[[-0.01206071  0.11632373  0.06206119
0.01403395  0.09541149  0.10695464 0.02560172
0.00185677 -0.04284821  0.06146432
0.09466285  0.04642421 0.08680347  0.05684567
-0.00717266 -0.03163519  0.03292002
-0.11397766 0.01304929  0.01964396
0.01902409  0.02831945  0.05870414  0.03390711
-0.06204525  0.06173197 -0.08613958
-0.04654748  0.02728105 -0.07830904
    …
0.04340003 -0.13192849 -0.00945092
-0.00835463 -0.06487109  0.05862355
-0.03407936 -0.00059001 -0.01640179
0.04123065
-0.04756588  0.08812257 0.00200338 -0.0931043
-0.03507337  0.02153351 -0.02621627
-0.02492662 -0.05771535 -0.01164199
-0.03879078 -0.05506947  0.01693138
-0.04124579 -0.03779858

-0.01950983 -0.05398201  0.07582296
0.00038318 -0.04639162
-0.06819214  0.01366171  0.01411388
0.00853774  0.02183574
-0.03016279 -0.03184025 -0.04273562]]
```

Word Embeddings για την λέξη "black" [18]

```
brown=[[ 1.35794589e-02 -2.18823571e-02
1.34526128e-02  6.74355254e-02
   1.04376070e-01  1.09921647e-02
-5.46298288e-02 -1.18385479e-02
   4.41223830e-02 -1.84863899e-02
-6.84073642e-02  3.21860164e-02
```

```
    4.09143828e-02  -2.74433400e-02
-2.47369967e-02   7.74542615e-02
    9.80964210e-03   2.94299088e-02
2.93895267e-02  -3.29437815e-02
...
   7.20389187e-02   1.57317147e-02
-3.10291946e-02  -5.51304631e-02
   -7.03861639e-02   7.40829483e-02
1.04319192e-02  -2.01565702e-03
    2.43322570e-02   1.92969330e-02
2.57341694e-02  -1.13280728e-01
    8.45847875e-02   4.90090018e-03
5.33546880e-02  -2.31553353e-02
    3.87288055e-05   3.31782512e-02
-4.00604047e-02  -1.02028981e-01
    3.49597558e-02  -1.71501152e-02
3.55573371e-02  -1.77437533e-02
   -5.94457164e-02   2.21221056e-02
9.73121971e-02  -4.90022525e-02]]
```

Word Embeddings για την λέξη "brown" [18]

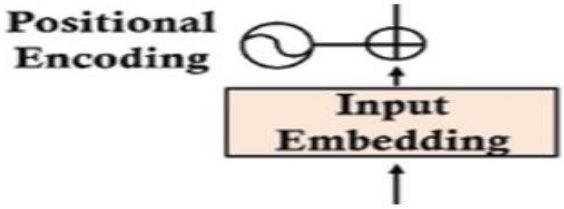
Για να ελέγξουμε αν τα δυο word embeddings που δημιουργήθηκαν αντιπροσωπεύουν την λέξη black και την λέξη brown μπορεί να γίνει με την χρήση της μεθόδους cosine_similarity από την βιβλιοθήκη του scikit learn.

```
cosine_similarity(black, brown)=
[[0.9998901]]
```

Χρήση της μεθόδου cosine_similarity [18]

3.4 Positional Encoding

Σε αυτό στο στάδιο εισέρχονται τα tokens χωρίς να γνωρίζουμε πραγματικά την θέση που έχουν μέσα σε μία ακολουθία (πρόταση).



Διάγραμμα του Positional Encoding [18]

Στο στάδιο του Positional Encoding δεν μπορούμε να παράξουμε επιπλέον vectors γιατί αυτό θα ήταν πολύ χρονοβόρο στην διαδικασία της εκπαίδευσης και τα attention sublayers του Transformer θα γινόντουσαν πολύ πιο σύνθετα. Έτσι, σαν εναλλακτική στο πρόβλημα αυτό είναι να γίνει προσθήκη

των τιμών του positional encoding μέσα στα embeddings vectors που έχουν προκύψει από το προηγούμενο στάδιο.

Το μοντέλο Transformer περιμένει μία σταθερή διάσταση των vectors που θα προκύψουν από το positional encoding. Στην περίπτωσή μας είναι 512. Αν ανατρέξουμε στην προηγούμενη πρόταση που είδαμε με τις λέξεις black και brown θα δούμε ότι αυτές οι λέξεις μπορεί να έχουν το ίδιο νόημα ως λέξεις αλλά ως προς την θέση στην οποία βρίσκονται, είναι πολύ μακριά η μία με την άλλη.

Η λέξη black είναι στην θέση 2 ενώ η λέξη brown βρίσκεται στην θέση 10.

Στόχος είναι να προστεθεί η τιμή του positional encoding στο word embedding της κάθε λέξεις που αντιστοιχεί έτσι ώστε το word embedding να αποκτήσει την πληροφορία της θέσης στην οποία βρίσκεται η λέξη.

Για κάθε word embedding vector χρειάζεται να βρεθεί ένας τρόπος έτσι ώστε να προστεθεί αυτή η πληροφορία σε κάθε τιμή i στο εύρος (0, 512) όπου είναι οι διαστάσεις του word embedding vector της λέξης brown και black.

Στο παρακάτω σχήμα απεικονίζεται το positional encoding vector που προκύπτει από τον Positional Encoder για την λέξη black.

```
PE(2)=
[[ 9.09297407e-01 -4.16146845e-01
9.58144367e-01 -2.86285430e-01
   9.87046242e-01 -1.60435960e-01
9.99164224e-01 -4.08766568e-02
   9.97479975e-01  7.09482506e-02
9.84703004e-01  1.74241230e-01
   9.63226616e-01  2.68690288e-01
9.35118318e-01  3.54335666e-01
   9.02130723e-01  4.31462824e-01
8.65725577e-01  5.00518918e-01
   8.27103794e-01  5.62049210e-01
7.87237823e-01  6.16649508e-01
   7.46903539e-01  6.64932430e-01
7.06710517e-01  7.07502782e-01
…
   5.47683925e-08  1.00000000e+00
5.09659337e-08  1.00000000e+00
   4.74274735e-08  1.00000000e+00
4.41346799e-08  1.00000000e+00
   4.10704999e-08  1.00000000e+00
3.82190599e-08  1.00000000e+00
   3.55655878e-08  1.00000000e+00
3.30963417e-08  1.00000000e+00
   3.07985317e-08  1.00000000e+00
2.86602511e-08  1.00000000e+00
   2.66704294e-08  1.00000000e+00
2.48187551e-08  1.00000000e+00
   2.30956392e-08  1.00000000e+00
2.14921574e-08  1.00000000e+00]]
```

Διάγραμμα του Positional Encoding vector για την λέξη black [18]

Αντίστοιχα πιο κάτω θα δούμε για την λέξη brown.

```
PE(10)=
[[-5.44021130e-01 -8.39071512e-01
1.18776485e-01 -9.92920995e-01
   6.92634165e-01 -7.21289039e-01
9.79174793e-01 -2.03019097e-01
   9.37632740e-01  3.47627431e-01
6.40478015e-01  7.67976522e-01
   2.09077001e-01  9.77899194e-01
-2.37917677e-01  9.71285343e-01
  -6.12936735e-01  7.90131986e-01
-8.67519796e-01  4.97402608e-01
  -9.87655997e-01  1.56638563e-01
-9.83699203e-01 -1.79821849e-01
...
  2.73841977e-07  1.00000000e+00
2.54829672e-07  1.00000000e+00
   2.37137371e-07  1.00000000e+00

 2.20673414e-07  1.00000000e+00
   2.05352507e-07  1.00000000e+00
 1.91095296e-07  1.00000000e+00
   1.77827943e-07  1.00000000e+00
 1.65481708e-07  1.00000000e+00
   1.53992659e-07  1.00000000e+00
 1.43301250e-07  1.00000000e+00
   1.33352145e-07  1.00000000e+00
 1.24093773e-07  1.00000000e+00
   1.15478201e-07  1.00000000e+00
 1.07460785e-07  1.00000000e+00]]
```

Διάγραμμα του Positional Encoding vector για την λέξη brown [18]

Τα συμπεράσματα για τις δύο λέξεις που προκύπτουν είναι ότι τα δύο vectors αφού περάσουν από τον Positional Encoding είναι ίδια σε μέγεθος δηλαδή 512.

Στην συνέχεια θα συγκρίνουμε την ομοιότητα των δύο λέξεων και το score που προκύπτει αν συγκρίνομε τις δύο λέξεις ως προς την θέση τους ή ως προς την σημασία σαν λέξεις μόνο χωρίς να λάβουμε υπόψιν την θέση στην οποία βρίσκονται μέσα στην ακολουθία (πρόταση).

```
cosine_similarity(pos(2), pos(10))=
[[0.8600013]]
```

Σύγκριση της ομοιότητας των δύο λέξεων μετά τον Positional Encoder

Παρατηρούμε ότι το score που προκύπτει από την σύγκριση των λέξεων μετά τον Positional Encoder είναι χαμηλότερο από το score που προκύπτει από την σύγκριση των λέξεων πριν από τον Positional Encoder. Ουσιαστικά, στην πρώτη σύγκριση βλέπουμε την σχέση που έχουν οι δύο λέξεις μέσα στην πρόταση αφού έχουμε συμπεριλάβει και την πληροφορία των θέσεων ενώ στην δεύτερη σύγκριση βλέπουμε ότι το score είναι πολύ υψηλό αφού συγκρίνουμε μόνο τα Word Embeddings δηλαδή τις λέξεις μόνο χωρίς την πληροφορία των θέσεων όπου βρίσκονται μέσα στο κείμενο.

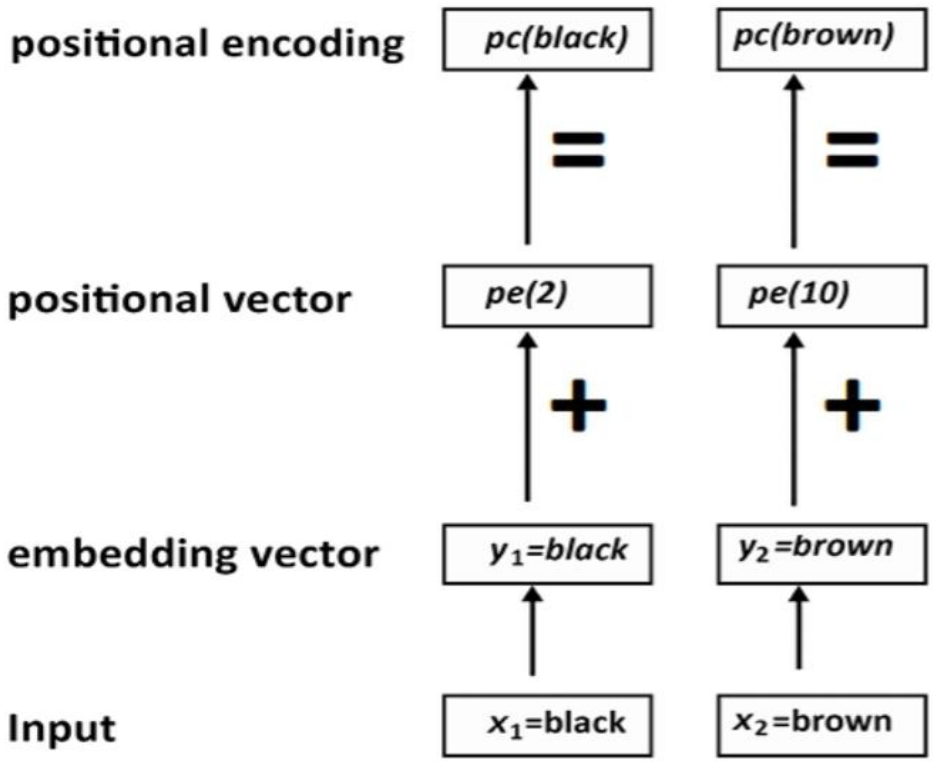
```
cosine_similarity(black, brown)=
[[0.9998901]]
```

Σύγκριση της ομοιότητας των δύο λέξεων πριν τον Positional Encoder [18]

Στην συνέχεια θα δούμε πως γίνεται η ενσωμάτωση των Positional vectors με τα Word Embedding vectors.

Προσθήκη του Positional Encoding στο Embedding Vector

Στην συνέχεια θα δούμε πως το Positional vector προστίθεται με το Embedding vector των λέξεων black και brown.



Ενσωμάτωση του Positional vector με το Embedding vector [18]

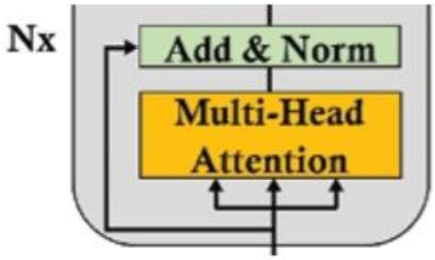
Από το παραπάνω διάγραμμα καταλαβαίνουμε ότι το Positional encoding προκύπτει από την πρόσθεση του Embedding vector σταθερού μήκους (512) με το Positional vector σταθερού μήκους (512).

Με αυτόν τον τρόπο το τελικό vector εμπεριέχει την πληροφορία του Word Embedding αλλά ταυτόχρονα έχει και την πληροφορία της θέσης της λέξης μέσα στην πρόταση.

Η έξοδος από το Positional Encoding αποτελεί την είσοδο για τον multi-head attention sublayer.

3.5 Multi-head attention sublayer

Ο multi-head attention sublayer αποτελείται από 8 κεφαλές και στην συνέχεια ακολουθεί το post-layer normalization σκοπός του οποίου είναι να ομαλοποιεί την έξοδο του.



Διάγραμμα του Multi-head attention sublayer [18]

Ο multi-head attention subιayer δεχεται ενα vector το οποιο περιέχει την πληροφορία από το embedding και το positional encoding κάθε λέξης.

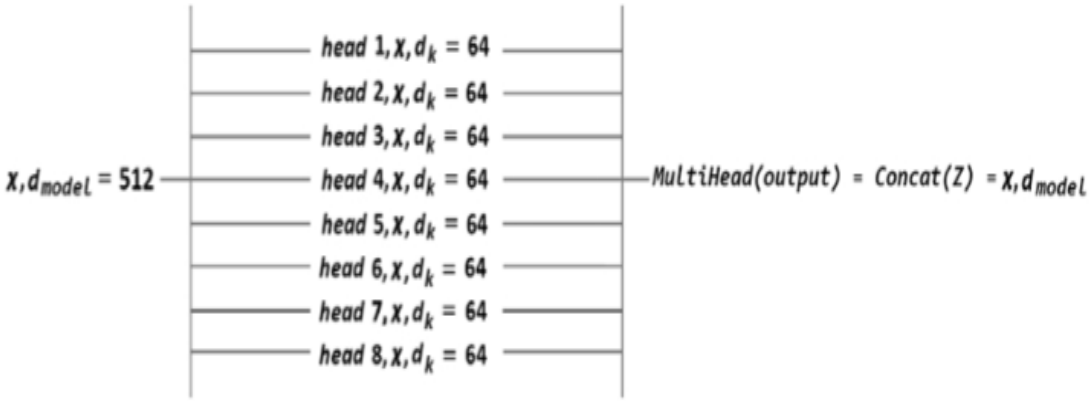
Η διάσταση κάθε vector όπως είπαμε και πιο πριν είναι διάστασης 512. Κάθε λέξη χαρτογραφείται σαν ένα heat map έτσι ώστε να καθοριστεί η λειτουργεία της μέσα στην πρόταση.

Στην παρακάτω πρόταση , θα δούμε πως η λέξη it μπορεί να συνδέεται με την λέξη cat και την λέξη rug.

```
Sequence =The cat sat on the rug and it was
dry-cleaned.
```

Το μοντέλο θα εκπαιδευτεί έτσι ώστε να καταλάβει αν η λέξη it συνδέεται με την λέξη cat ή την λέξη rug. Ένας καλύτερος τρόπος προσέγγισης του προβλήματος είναι να διαιρέσουμε τις διαστάσεις 512 κάθε vector, σε διαστάσεις 64 για κάθε vector. Έτσι θα έχουμε 8 vector όπου είναι οι συνολικές λέξεις που εμφανίζονται μέσα στην πρόταση χωρίς να λαμβάνουμε υπόψιν αυτές που εμφανίζονται παραπάνω από μία φορά , και κάθε λέξη από διάσταση 512 θα έχει διάσταση 64.

Παρακάτω θα δούμε και ένα αντίστοιχο παράδειγμα που θα μας βοηθήσει.



Διάγραμμα κεφαλών του multi-head [18]

Η μετατροπή των διαστάσεων γίνεται έτσι ώστε οι λέξεις σε μορφή vectors να τρέξουν σε παράλληλο χρόνο όλες μαζί κατά την εκπαίδευση. Αυτό έχει πολλά πλεονεκτήματα στην απόδοση κατά την διαδικασία του χρόνου εκπαίδευσης αλλά επίσης μειώνονται και οι κίνδυνοι για overfitting.

Από το παραπάνω διάγραμμα είναι φανερό ότι έχουμε 8 γραμμές που τρέχουν παράλληλα η μία με την άλλη. Κάποια κεφαλή από τις 8 θα αποφασίσει αν η λέξη it συνδέεται με την λέξη cat , κάποια άλλη θα αποφασίσει αν η λέξη it συνδέεται με την λέξη rug και κάποια άλλη αν η λέξη rug συνδέεται με την λέξη dry-cleaned. Η έξοδος κάθε κεφαλής είναι ένα matrix Z_i .

$$Z = (Z_0, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7)$$

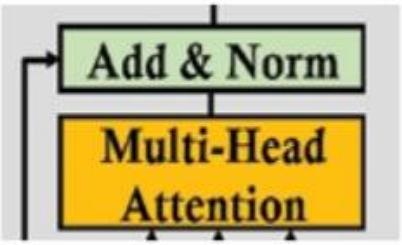
Πριν την έξοδο από τον multi-head attention sublayer τα στοιχεία Z_i ενώνονται σε ένα τελικό στοιχείο z το οποίο έχει διάσταση 512.

Στην συνέχεια θα δούμε μέσα σε κάθε κεφαλή τι μορφή αποκτά η κάθε λέξη σε επίπεδο matrix. Η κάθε λέξη που εισέρχεται απεικονίζεται με τρείς μορφές:

- Query matrix (Q) που έχει διαστάσεις d_{model} = 64. Η δομή είναι key – value κάθε λέξης.
- Key matrix (K) που έχει διαστάσεις d_{model} = 64.
- Value matrix (V) που έχει διαστάσεις d_{model} = 64.

3.6 Post-Layer Normalization

Μετά από κάθε attention sublayer και κάθε Feedforward sublayer στο μοντέλο Transformer υπάρχει ένα post-layer normalization.



Διάγραμμα Post-layer normalization [18]

Το Post-Ln αποτελείται από μία επιπλέον συνάρτηση (function) και ένα επιπλέον layer normalization. Αυτή η επιπλέον συνάρτηση συνδέεται με κάθε sublayer που τροφοδοτείται με δεδομένα εισόδου. Σκοπός της συνάρτησης είναι να ελέγξει ότι δεν θα χαθεί κάποια πολύτιμη πληροφορία.

$$LayerNormalization\,(x + Sublayer(x))$$

Το x αντιπροσωπεύει την πληροφορία η οποία εισέρχεται μέσα στο Sublayer. Η είσοδος στο LayerNormalization είναι ένα vector v το οποίο είναι διαστάσεων d_{model} = 512.

Υπάρχουν αρκετές μέθοδοι layer normalization, και οι διαφορές ποικίλουν ανάλογα με τα μοντέλα. Η βασική ιδέα για το LayerNormalization για τα Transformer μοντέλα δίνεται από τον παρακάτω τύπο.

$$LayerNormalization(\boldsymbol{v}) = \gamma \frac{\boldsymbol{v} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + \beta$$

Μαθηματικός τύπος του LayerNormalization

- μ είναι η μέση τιμή του v

$$\mu = \frac{1}{d}\sum_{k=1}^{d} v_k$$

Μαθηματικός τύπος μ

- σ είναι η τυπική απόκλιση του v

$$\sigma^2 = \frac{1}{d}\sum_{k=1}^{d} (v_{k-\mu})$$

Μαθηματικός τύπος σ

- γ είναι παράμετρος κλιμάκωσης
- β είναι ένα bias vector

Η παραπάνω μέθοδος LayerNormalization (ν) αποτελεί την γενική ιδέα τέτοιων μεθόδων. Στη συνέχεια θα αναφερθούμε στο επόμενο sublayer το οποίο είναι το Feedforward network.

3.7 Feedforward network

Η είσοδος που δέχεται το Feedforward network είναι η έξοδος που προέρχεται από το Post Layer Normalization με διαστάσεις $d_{model} = 512$.

Το FFN αποτελείται από τα παρακάτω στάδια:

- Ο encoder και ο decoder που βρίσκονται στον FFN επικοινωνούν μεταξύ τους.
- Το FFN έχει δύο layers και εφαρμόζει μία λειτουργεία που ονομάζεται ReLU.
- Η είσοδος και η έξοδος των FFN layers είναι διαστάσεων $d_{model} = 512$, αλλά το εσωτερικό layer είναι διαστάσεων $d_{ff} = 2048$.

3.8 Decoder

Τα layers από τα οποία αποτελείται ο decoder είναι ίδια με τα layers του encoder. Η δομή του decoder παραμένει ίδια όπως αυτή που βρίσκεται στον encoder για όλα τα N = 6 layers.

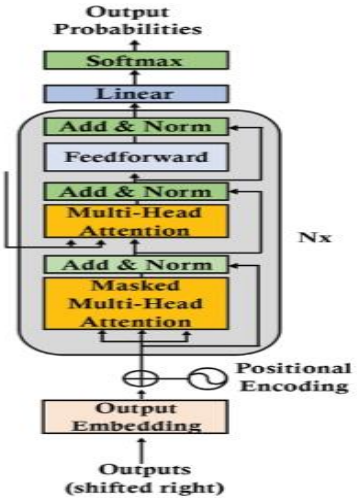
Κάθε layer περιέχει τρία sublayers:

- Multi – headed masked attention mechanism.
- Multi – headed attention mechanism.
- Feedforward network.

Ο decoder περιλαμβάνει ένα τρίτο sublayer το οποίο ονομάζεται Multi – headed masked attention mechanism όπου δεν το βρίσκουμε στον encoder.

Σε αυτό το sublayer οι λέξεις καλύπτονται έτσι ώστε το Transformer μοντέλο να μην μπορεί να δει τις διπλανές λέξεις, ουσιαστικά εμποδίζουμε το μοντέλο να βοηθηθεί από τις διπλανές λέξεις.

Παρακάτω απεικονίζεται η δομή του decoder:



Το embedding layer όπως και στον encoder έτσι και εδώ βρίσκεται στο κάτω μέρος. Επίσης όπως και στον encoder έτσι και εδώ οι διαστάσεις από κάθε sublayer είναι σταθερή δηλαδή d_{model} = 512.

Ας δούμε για παράδειγμα πως λειτουργεί ο decoder με την μετάφραση από την γλώσσα Α στην γλώσσα Β. Έστω ότι έχουμε την παρακάτω πρόταση στα Γαλλικά την οποία θέλουμε να την μεταφράσουμε στα Αγγλικά.

```
Output=Le chat noir était assis sur le canapé
et le chien marron dormait sur le tapis
```

Η παρακάτω πρόταση είναι η μετάφραση της προηγούμενης πρότασης.

```
Input=The black cat sat on the couch and the
brown dog slept on the rug.
```

Οι λέξεις εξόδου από την Α πρόταση που βγαίνουν από τον encoder εισέρχονται στο Embedding layer και στο Encoding layer του decoder όπως γίνεται και στον encoder.

Στα attention layers χρησιμοποιείται η προηγούμενη ακολουθία ως επιπλέον είσοδος στον decoder.

Ωστόσο, στο Masked multi-head attention sublayer οι μελλοντικές λέξεις που είναι να προβλέψει το μοντέλο Transformer κρύβονται έτσι ώστε να μάθει να προβλέπει σωστά.

Στην συνέχεια υπάρχει ένα Post-layer normalization το οποίο ακολουθείται μετά από τον masked-head attention sublayer 1 όπως και στον encoder.

Το multi-head attention sublayer 2 με τον ίδιο τρόπο καλύπτει τις λέξεις που είναι να προβλέψει το μοντέλο έτσι ώστε το Transformer μοντέλο να μην δει τις μετέπειτα λέξεις και βοηθηθεί.

Μετά από τον multi-head attention sublayer 2 ακολουθεί και εδώ ένα Post-layer normalization.

Στην συνέχεια υπάρχει το FFN sublayer το οποίο έχει την ίδια δομή με το FFN που βρίσκεται στον encoder.

Στο τέλος του decoder βρίσκονται το Linear layer και το SoftMax layer.

ΚΕΦΑΛΑΙΟ 4

4.1 Περιγραφή αρχιτεκτονικής Bert μοντέλου

Στα προηγούμενα κεφάλαια κάναμε εκτενή αναφορά για τα Transformer μοντέλα , αναλύσαμε την αρχιτεκτονική τους καθώς επίσης αναφερθήκαμε και σε διάφορες εφαρμογές που μπορούν να εφαρμοστούν και να βγάλουν εις πέρας καθημερινά προβλήματα ξεπερνώντας καμία φορά και την ανθρώπινη αποτελεσματικότητα.

Στην παρούσα εργασία χρησιμοποιήσαμε το Bert μοντέλο το οποίο είναι ένα Transformer μοντέλο. Η λέξη Bert σημαίνει (Bidirectional Encoder Representation from Transformers). Είναι ένα pre-trained deep learning μοντέλο το οποίο χρησιμοποιείται σε NLP (Natural Language Processing) προβλήματα συμπεριλαμβανομένου και του text classification.

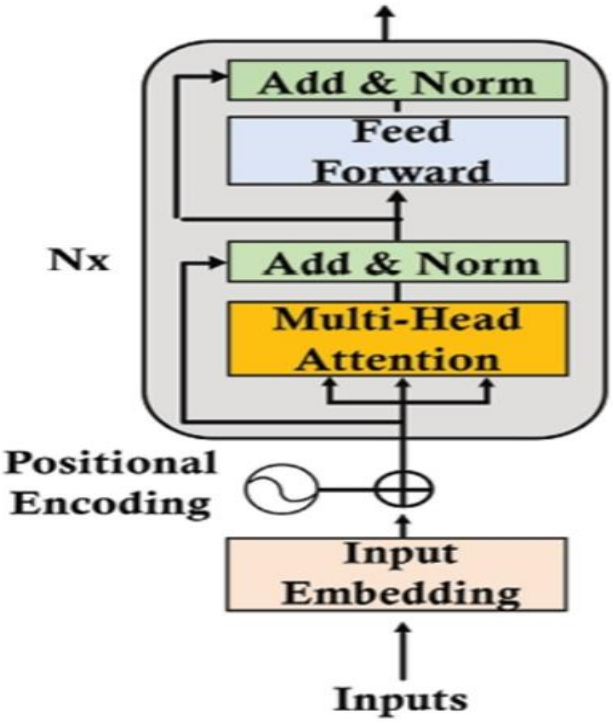
Στην περίπτωση του Bert μοντέλου χρησιμοποιείται μόνο ο encoder για προβλήματα text classification και όχι ο decoder όπως είδαμε στα προηγούμενα κεφάλαια.

Το Bert μοντέλο έχει εκπαιδευτεί σε έναν τεράστιο αριθμό κειμένων και η διαδικασία που ακολουθήθηκε για την εκπαίδευσή του ήταν unsupervised learning. Μέσα από την διαδικασία του unsupervised learning το μοντέλο εκπαιδεύεται και μαθαίνει μέσα από την μετάφραση του περιεχομένου των κειμένων, δηλαδή από συγκεκριμένα patterns ο αλγόριθμος μαθαίνει από μόνος του χωρίς να του υποδηλώνουμε μέσω labels αν αυτό που προέβλεψε ήταν σωστό ή λάθος.

Το Bert μοντέλο είναι ένα bidirectional multi-head attention sub-layer. Όταν εμείς οι άνθρωποι έχουμε δυσκολία στο να κατανοήσουμε μία ακολουθία από λέξεις κοιτάμε όλες τις λέξεις τις πρότασής και τις συγκρίνουμε μεταξύ τους προκειμένου να καταλήξουμε σε ένα συμπέρασμα. Έτσι και το Bert μοντέλο κοιτάει όλες τις λέξεις τις πρότασής την ίδια στιγμή.

Στην συνέχεια θα αναφερθούμε στην αρχιτεκτονική του Bert μοντέλου για προβλήματα που έχουν να κάνουν με το text classification.

Στο παρακάτω σχήμα απεικονίζεται ο Encoder για τον οποίο μιλήσαμε στο προηγούμενο κεφάλαιο και τον συναντάμε στο Bert μοντέλο.



Δομή του Encoder στο μοντέλο Bert [18]

Όπως είπαμε και προηγουμένως το μοντέλο Bert δεν έχει decoder αλλά μόνο encoder. Τα masked tokens γίνονται δηλαδή οι λέξεις που είναι να μαντέψει το μοντέλο Bert καλύπτονται στο Attention Layer του Encoder.

Τα κλασσικά Transformer μοντέλα αποτελούνται από N = 6 layers. Συνήθως οι διαστάσεις των κλασσικών μοντέλων Transformers είναι d_{model} = 512. Επίσης ο αριθμός των κεφαλών (heads) στα μοντέλα Transformers είναι A=8. Οι διαστάσεις τις κάθε κεφαλής που βρίσκεται μέσα στα μοντέλα δίνεται από τον παρακάτω υπολογισμό:

$$d_k = \frac{d_{model}}{A} = \frac{512}{8} = 64$$

Διαστάσεις κεφαλών που βρίσκονται στον Encoder

Αξίζει να σημειωθεί ότι τα layers του encoder που αποτελείται το Bert μοντέλο είναι μεγαλύτερα από τα κλασσικά μοντέλα Transformers.

Δύο μοντέλα Transformers μπορούν να κτιστούν με και να λειτουργήσουν με έναν Encoder.

- Το πρώτο είναι το Bert base το οποίο έχει = 12 encoder layers. Αυτό το μοντέλο έχει διαστάσεις d_{model} = 768 και επίσης το multi-head attention sub-layer αποτελείται από A = 12 κεφαλές. Η διαστάσεις κάθε κεφαλής παραμένει ίδια όπως στα κλασσικά μοντέλα Transformers δηλαδή 64. Η έξοδος από κάθε multi-head attention sub-layer πριν την ενσωμάτωσή τους είναι:

$$output\_multi\text{-}head\_attention = \{z_0, z_1, z_2, \ldots, z_{11}\}$$

Έξοδος από τον multi-head attention sub-layer του BERT BASE μοντέλου [18]

- Το δεύτερο μοντέλο ονομάζεται Bert large το οποίο αποτελείται από N = 24 encoder layers. Οι διαστάσεις του είναι d_{model} = 1024. Ο Multi-head attention sub-layer αποτελείται από A = 16 κεφαλές. Η διαστάσεις κάθε κεφαλής παραμένει ίδια δηλαδή 64.

$$d_k = \frac{d_{model}}{A} = \frac{1024}{16} = 64$$

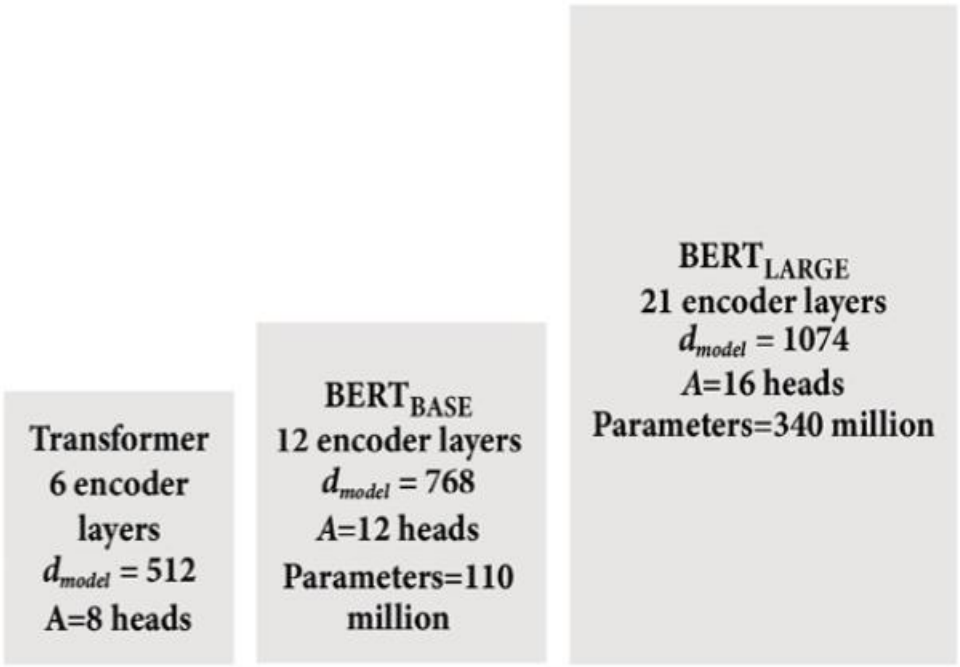
Διαστάσεις κεφαλών που βρίσκονται στο BERT LARGE μοντέλο [18]

Η έξοδος από κάθε multi-head attention sub-layer πριν την ενσωμάτωσή τους είναι:

$$output\_multi\text{-}head\_attention = \{z_0, z_1, z_2, \ldots, z_{15}\}$$

Έξοδος από τον multi-head attention sub-layer του BERT LARGE μοντέλου [18]

Στο παρακάτω σχήμα βλέπουμε μία περίληψη των Bert μοντέλων με τα χαρακτηριστικά από τα οποία αποτελούνται.



Διάγραμμα Bert μοντέλων [18]

4.2 Εκπαίδευση Multiclass Classifier Bert μοντέλου

Σκοπός της παρούσας πτυχιακής εργασίας είναι να δούμε πως ένα μοντέλο Transformer μπορεί να καταλάβει και να διακρίνει μέσα από τους στοίχους ενός τραγουδιού σε ποιόν καλλιτέχνη ανήκει.

Για την εκπόνηση της εργασίας χρησιμοποιήσαμε το Bert μοντέλο αφού αναφερόμαστε σε δεδομένα κειμένου. Με τον όρο Classifier εννοούμαι την κατηγοριοποίηση ενός προβλήματος σε μία κατηγορία το οποίο ανήκει. Αυτή η κατηγοριοποίηση μπορεί να χρησιμοποιηθεί σε κείμενο , δεδομένα ή ακόμα και σε εικόνα. Με τον όρο Multiclass Classification αναφερόμαστε σε μοντέλα τα οποία είναι ικανά να κατηγοριοποιήσουν προβλήματα τα οποία οι κατηγορίες είναι περισσότερες από δύο.

Όπως έχουμε αναφέρει και πιο πάνω τα μοντέλα Transformers είναι μοντέλα τα οποία έχουν είδη εκπαιδευτεί σε έναν μεγάλο αριθμό δεδομένων που βρίσκονται στην Wikipedia. Παρόλα αυτά για να εκπαιδεύσουμε το μοντέλο στο πρόβλημα το οποίο θέλουμε να λύσουμε χρειάστηκε να εντοπίσουμε επαρκεί δεδομένα. Για την άντληση δεδομένων πολύ χρήσιμη είναι η Kaggle ιστοσελίδα, εκεί μπορέσαμε να βρούμε τα δεδομένα που θέλαμε έτσι ώστε ο Multiclass Classifier να εκπαιδευτεί με τα κατάλληλα δεδομένα.

Η μορφή των δεδομένων μας είναι σε μορφή csv. Συνολικά είναι τέσσερα csv όπου κάθε ένα αντιστοιχεί σε έναν καλλιτέχνη. Οι πληροφορίες που έχουν τα csv είναι τα παρακάτω:

- Όνομα καλλιτέχνη
- Τίτλος τραγουδιού
- Ονομασία του Album
- Χρονολογία για κάθε τραγούδι
- Στοίχοι τραγουδιών

Λόγω του ότι θέλουμε το μοντέλο μας να αντιστοιχεί τους στοίχους με τον αντίστοιχο καλλιτέχνη τα δεδομένα που χρησιμοποιήσαμε κατά την εκπαίδευση είναι δύο, το όνομα του καλλιτέχνη και τους στοίχους των τραγουδιών.

```
,Artist,Title,Album,Year,Date,Lyrics
0,Coldplay,The Scientist,A Rush of Blood to the Head,2002,2002-08-26,come up to meet you tell you i'm sorry you dor
1,Coldplay,Viva la Vida,Viva La Vida or Death and All His Friends,2008,2008-05-25,chris martin i used to rule the w
2,Coldplay,Fix You,X&Y,2005,2005-06-06,chris martin when you try your best but you don't succeed when you get what
3,Coldplay,Yellow,Parachutes,2000,2000-06-26,chris martin look at the stars look how they shine for you and everyth
4,Coldplay,Hymn for the Weekend,A Head Full of Dreams,2016,2016-01-25,beyoncé and said drink from me drink from me
5,Coldplay,A Sky Full of Stars,Ghost Stories,2014,2014-05-02,'cause you're a sky 'cause you're a sky full of stars
6,Coldplay,Everglow,A Head Full of Dreams,2015,2015-11-26,oh they say people come say people go this particular dia
7,Coldplay,Adventure of a Lifetime,A Head Full of Dreams,2015,2015-11-06,indecipherable  turn your magic on umi sh
8,Coldplay,Orphans,Everyday Life,2019,2019-10-24,chris martin  choir moses martin i want to know when i can go back
9,Coldplay,Paradise,Mylo Xyloto,2011,2011-09-12,oohoohooh oohoohooh oohoohooh oohoohooh oohoohooh oohoohooh when sh
10,Coldplay,Magic,Ghost Stories,2014,2014-03-03,call it magic call it true i call it magic when i'm with you and i
11,Coldplay,Clocks,A Rush of Blood to the Head,2002,2002-08-26,the lights go out and i can't be saved tides that i
12,Coldplay,Daddy,Everyday Life,2019,2019-11-20,daddy are you out there daddy won't you come and play daddy do you
13,Coldplay,Everyday Life,Everyday Life,2019,2019-11-03,what in the world are we going to do look at what everybody
14,Coldplay,Sparks,Parachutes,2000,2000-07-10,did i drive you away i know what you'll say you say oh sing one you k
15,Coldplay,Up&Up,A Head Full of Dreams,2015,2015-12-04,chris martin fixing up a car driving it again searching for
```

Δομή δεδομένων σε μορφή csv

Αντίστοιχα ακολουθείται η ίδια δομή και για τους άλλους τρείς καλλιτέχνες. Στην συνέχεια για την πραγματοποίηση της εκπαίδευσης πρώτα έπρεπε να μετατρέψουμε τα δεδομένα μας από μορφή csv σε μία μορφή όπου θα μας ήταν εύκολη η επεξεργασία των δεδομένων αλλά και ο καθαρισμός των δεδομένων μας όπου χρειαζόταν.

```
def add_encode_cat(x):
    if x not in encode_dict.keys():
        encode_dict[x] = len(encode_dict)
    return encode_dict[x]


df_coldplay = pd.read_csv("./ColdPlay.csv")  # json file with examples of advices
df_edsheeran = pd.read_csv("./EdSheeran.csv")  # json file with examples of pe's
df_maroon5 = pd.read_csv("./Maroon5.csv")  # json file with examples of other (non-advice, non-pe)
df_taylorswift = pd.read_csv("./TaylorSwift.csv")
# Remove rows if Nan value is located in Lyrics column
df_coldplay = df_coldplay.dropna(subset=["Lyrics"])
df_edsheeran = df_edsheeran.dropna(subset=["Lyrics"])
df_maroon5 = df_maroon5.dropna(subset=["Lyrics"])
df_taylorswift = df_taylorswift.dropna(subset=["Lyrics"])
df_coldplay["Artist"] = "Coldplay"
df_edsheeran["Artist"] = "Ed Sheeran"
df_maroon5["Artist"] = "Maroon 5"
df_taylorswift["Artist"] = "Taylor Swift"
df_artists = pd.concat([df_coldplay, df_edsheeran, df_maroon5, df_taylorswift], ignore_index=True, sort=False).drop("Unnamed: 0", axis=1)
```

Συνάρτηση ανάγνωσης των δεδομένων και προσθήκη labels

Στην παραπάνω συνάρτηση διαβάζουμε κάθε αρχείο csv ξεχωριστά με την χρήση της βιβλιοθήκης pandas. Στην συνέχεια διαγράφουμε ολόκληρη την γραμμή από τα δεδομένα μας αν δεν υπάρχει πληροφορία στην στήλη Lyrics αφού θα μας ήταν άχρηστο να είχαμε κενή πληροφορία κατά την διαδικασία της εκπαίδευσης.

Η έξοδος της συγκεκριμένης συνάρτησης είναι ένας πίνακας ο οποίος περιέχει και τους τέσσερις καλλιτέχνες με τα αντίστοιχα δεδομένα τους. Παρακάτω θα δούμε την συνέχεια της επεξεργασίας των δεδομένων μας πριν την έναρξη της εκπαίδευσης.

```
56  df_artists = df_artists.drop_duplicates(subset=['Lyrics']).reset_index(drop=True)
57
58  # Allows only specific columns
59  df_artists = df_artists[["Artist", "Lyrics"]]
60
61  # Cleaning each Lyrics from extra spaces and punctuations
62  for index, row in df_artists.iterrows():
63      t = TextCleaning(row["Lyrics"])
64      df_artists["Lyrics"].at[index] = t.clean_up()
65  # add extra ENCODED_CAT column to store label encoding, i.e., 0 for Coldplay, 1 Ed Sheeran , 2 Maroon 5 and 3 Taylor Swift
66  df_artists["ENCODED_CAT"] = df_artists["Artist"].apply(lambda x: add_encode_cat(x))
67  # print(df_artists.loc[df_artists['Artist'] == "Ed Sheeran"])
```

Στην γραμμή 56 αφαιρούμαι από την στήλη Lyrics δεδομένα τα οποία είναι πανομοιότυπα. Κατά τη διάρκεια της εκπαίδευσης, το μοντέλο προσαρμόζεται στα δεδομένα εκπαίδευσης και μαθαίνει τη σχέση μεταξύ των χαρακτηριστικών των δεδομένων εισόδου και των αντίστοιχων εξόδων που πρέπει να παράγει.

Η αφαίρεση ίδιων δεδομένων από τα δεδομένα εκπαίδευσης είναι απαραίτητη για να διασφαλιστεί ότι το μοντέλο δεν μαθαίνει απλά να "αποθηκεύει" τα δεδομένα εκπαίδευσης και να τα αντιγράφει στις προβλέψεις του. Αν αφήσουμε το μοντέλο να μάθει τα ίδια δεδομένα που χρησιμοποιούνται κατά τη διάρκεια της εκπαίδευσης, τότε το μοντέλο μπορεί να γίνει υπερβολικά εξειδικευμένο και να μην μπορεί να γενικεύσει την εκμάθηση σε νέα δεδομένα. Αυτό ονομάζεται "overfitting" και αποτελεί συχνό πρόβλημα στην εκπαίδευση μοντέλων AI.

Στην γραμμή 66 κάνουμε προσθήκη μίας επιπλέον στήλης την οποία δημιουργήσαμε εμείς. Αυτή η στήλη αντιπροσωπεύει κάθε καλλιτέχνη αλλά αυτή την φορά όχι με το όνομα του αλλά με μία αρίθμηση από το 0 έως το 3.

Για παράδειγμα ο καλλιτέχνης Ed Sheeran έχει τον αριθμό 1, αυτό γίνεται διότι το μοντέλο μπορεί να εκπαιδευτεί μόνο με αριθμούς. Έτσι μετατρέπουμε κάθε πληροφορία η οποία μπορεί να είναι με την μορφή λέξεων σε αριθμούς.

Μετά τον καθαρισμό τον δεδομένων έγινε μέτρηση των δεδομένων για κάθε καλλιτέχνη. Σημαντικό είναι να υπάρχει ομοιομορφία μεταξύ των δεδομένων, δηλαδή όλες οι κατηγορίες να έχουν τον ίδιο αριθμό περίπου δεδομένων. Στην περίπτωσή μας κάθε καλλιτέχνης έχει 300 δεδομένα δηλαδή 300 στίχους τραγουδιών.

Ο καθαρισμός του κειμένου των στοίχων γίνεται στην γραμμή 64 με την χρήση της συνάρτησης clean_up που γίνετε μέσα από το αντικείμενο t της κλάσης TextCleaning.

```python

REMOVE_MULTIPLE_SPACES = re.compile(r" +")

Kontosoros, 2 months ago | 1 author (Kontosoros)
@dataclass
class TextCleaning:
    text: str

    def tokenize_tok(self, txt_body):
        txt_body = txt_body.split()
        text_to_yield = []
        for tok in txt_body:
            toks = re.sub(r"([^\w\s]|_)", r" \1 ", tok.replace("\\", " ").strip()).split()
            if len(toks) > 1:
                for t in toks:
                    if t:
                        text_to_yield.append(t)
            else:
                if toks:
                    text_to_yield.append(toks[0])
        txt_body = " ".join(text_to_yield)
        return txt_body

    def clean_up(self):
        # Remove punctuations
        txt_body = re.sub(r"[^\w\s]|[_]", " ", self.text)
        # Remove digits
        txt_body = re.sub("\d+", "", txt_body)
        # Remove multiple spaces
        txt_body = self.tokenize_tok(REMOVE_MULTIPLE_SPACES.sub(" ", txt_body))
        return txt_body

```

Παραπάνω βλέπεται το εσωτερικό της κλάσης TextCleaning. Αφαιρούμαι διάφορους αριθμούς που μπορεί να υπάρχουν μέσα στους στοίχους , διάφορα σημεία στίξης καθώς επίσης και επιπλέον κενά που μπορούν να υπάρχουν μέσα στο κείμενο. Ο καθαρισμός γίνεται πολύ εύκολα με την χρήση της βιβλιοθήκης Regex.

```python
train_dataset = df_artists.sample(frac=TRAIN_SIZE, random_state=200)  # select randomly a
test_dataset = df_artists.drop(train_dataset.index).reset_index(
    drop=True
)  # delete the previously selected examples from total dataset, the remaining will be us
train_dataset = train_dataset.reset_index(drop=True)

```

Στην γραμμή 68 δημιουργούμαι το set της εκπαίδευσης ή αλλιώς training dataset. Με την παράμετρο random_state επιλέγουμε τυχαία τα δεδομένα που θα ανήκουν στο training set έτσι ώστε να αποφύγουμε τυχόν επαναλήψεις στα δεδομένα μας. Συνήθως το training dataset περιέχει το 80% των δεδομένων έτσι ώστε το μοντέλο να εκπαιδευτεί με έναν αρκετά μεγάλο βαθμό δεδομένων.

Στην γραμμή 69 αντίστοιχα δημιουργούμαι το set για το test που θα υποβάλουμε το μοντέλο μας μετά την εκπαίδευση έτσι ώστε να δούμε πόσο αποδοτικό είναι το μοντέλο σε δεδομένα τα οποία δεν έχει δει κατά την εκπαίδευση. Το test dataset αποτελείται από το 20% των δεδομένων.

Αφού φτιάξαμε το training dataset και το test dataset στην συνέχεια φορτώνουμε τον tokenizer ο οποίος είναι pretrained και είναι εκείνος ο οποίος θα χωρίσει κάθε πρόταση σε λέξεις. Στην γραμμή 84 δημιουργούμαι ένα αντικείμενο training set το οποίο βγαίνοντας από την συνάρτηση Tokenized_set θα έχει υποστεί padding δηλαδή κάθε στοίχος θα έχει ένα σταθερό μήκος 512 θέσεων.

Το ίδιο θα κάνουμε και στην γραμμή 86 αντίστοιχα για το testing set.

```
81  tokenizer = DistilBertTokenizer.from_pretrained(
82      "distilbert-base-uncased"
83  )  # load tokenizer for Distilbert model, "uncased" means that only lower case wordpie
84  training_set = Tokenized_Set(train_dataset, tokenizer, MAX_LEN)
85  # print(training_set.__getitem__(1))
86  testing_set = Tokenized_Set(test_dataset, tokenizer, MAX_LEN)
87  train_params = {"batch_size": BATCH_SIZE, "shuffle": True, "num_workers": 0}
88  test_params = {"batch_size": BATCH_SIZE, "shuffle": True, "num_workers": 0}
89  # data loaders take tokenised sets and produce small sets of batch size at every step
90  training_loader = DataLoader(training_set, **train_params)
91  testing_loader = DataLoader(testing_set, **test_params)
```

Στην γραμμή 87 και 88 θα ορίσουμε κάποιες παραμέτρους όπως το batch size. Τέλος, στην γραμμή 90 και 91 η συνάρτηση Dataloader θα παράγει ένα set δεδομένων ανάλογα με το Batch size που έχουμε ορίσει, έτσι ώστε το μοντέλο να τροφοδοτείται κατά την διαδικασία της εκπαίδευσης και του testing.

Στην συνέχεια ορίζουμε το μοντέλο το οποίο θα εκπαιδεύσουμε.

```
94  # Create the classification model
95  model = DistilBertForSequenceClassification.from_pretrained(
96      "distilbert-base-uncased",
97      num_labels=NUM_CLASSES,
98      output_attentions=False,
99      output_hidden_states=False,
100 )
101 model.to(device)
102 loss_function = torch.nn.CrossEntropyLoss()  # For binary classification, BCEWithLogitsLoss() is recommended
103 optimizer = AdamW(params=model.parameters(), lr=LEARNING_RATE, eps=EPS)  # AdamW is better than Adam
104 total_steps = len(training_loader) * EPOCHS
105 # Create the learning rate scheduler
106 scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps=0, num_training_steps=total_steps)
```

Στην γραμμή 101 επιλέγουμε να εκπαιδεύσουμε το μοντέλο με την χρήση της gpu ή της cpu αντίστοιχα. Στην γραμμή 102 καλούμαι την συνάρτηση Cross Entropy που θα μας βοηθήσει στον υπολογισμό του loss, μετά καλούμαι την συνάρτηση του optimizer με τις παραμέτρους που θέλουμε να χρησιμοποιήσει (Learning rate = 2e-05). Τέλος στην γραμμή 106 ορίζουμε τον scheduler.

Στην συνέχεια είμαστε έτοιμη να ορίσουμε την συνάρτηση της εκπαίδευσης. Στην παρακάτω συνάρτηση θα μπαίνουμε μέσα για κάθε epoch εκπαίδευσης, θα υπολογίζεται σε κάθε βήμα το loss και στη συνέχεια θα προκύπτει ένα συνολικό loss για το κάθε epoch.

Σε κάθε βήμα εκπαίδευσης τα βάρη του μοντέλου αλλάζουν προκειμένου το μοντέλο να βελτιωθεί και να κάνει σωστή εκτίμηση κατά την έξοδο του. Το loss υπολογίζεται μέσα από την σύγκριση των προβλέψεων που έχει κάνει το μοντέλο σε σύγκριση με τα πραγματικά αποτελέσματα που θα έπρεπε να έχει προβλέψει το μοντέλο.

```python
loss_list = []
def train(epoch):
    tr_loss = 0
    nb_tr_steps = 0
    y_true = []
    y_pred = []
    model.train()
    # ids, mask, targets, outputs are all tensors
    for step, batch in enumerate(tqdm(training_loader, 0)):
        ids = batch["ids"].to(device, dtype=torch.long)
        # ids example (token ids for each token of each message of the batch):
        # tensor([  [  101,  1049,  1013,  ..., 22932, 24434,   102],
        #           [  101, 19842,  2019,  ...,     0,     0,     0],
        #           [  101, 22857,  2692,  ...,     0,     0,     0]    ])
        mask = batch["mask"].to(device, dtype=torch.long)
        # mask example (mark useful part and padded part of each message of the batch):
        # tensor([  [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        #            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        #            1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        #            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], .... ])
        targets = batch["targets"].to(device, dtype=torch.long)
        # targets example (0 or 1 for each message of the batch), batch=8 :
        # tensor([1, 1, 0, 0, 1, 1, 1, 1])
        # clear previously calculated gradients
        model.zero_grad()
        # get model predictions for the current batch
        outputs = model(
            ids,
            # token_type_ids = None, # single sentence task, no segments
            attention_mask=mask,
        )

        # output example (output classification score (logit) for each message of the batch), 2 classes, batch=8:
        # tensor([[ 0.1027,  0.1023],[ 0.0319,  0.1203],[ 0.0019,  0.0196],[ 0.0774,  0.0595],
        # [ 0.0585, -0.0451],[ 0.0545,  0.0008],[ 0.1142,  0.0034],[ 0.1503,  0.0091]])
        # note: score (logits) is not a probability yet
        loss = loss_function(outputs.logits, targets.view(-1))
        # loss example: tensor(0.6856)
        loss_list.append(loss.item())
        tr_loss += loss.item()
        nb_tr_steps += 1
        if step % 500 == 0 and not step == 0:
            loss_step = tr_loss / nb_tr_steps
            print()
            print(f"Training Loss after batch: {loss_step}")

        optimizer.zero_grad()
        # Perform backward pass to claculate the gradients
        loss.backward()
        # clip the the gradients to 1.0. It helps in preventing the exploding gradient problem
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        # Update parameters based on computed gradients
        optimizer.step()
        # Update the learning rate
        scheduler.step()
    epoch_loss = tr_loss / nb_tr_steps
    print()
    print(f"Training Avg Loss Epoch: {epoch_loss}")
```

Τέλος, στην πιο κάτω εικόνα βλέπουμε την συνάρτηση του test. Σε αυτή την συνάρτηση μπορούμε να ελέγξουμε την αποδοτικότητα του μοντέλου μετά την διαδικασία της εκπαίδευσης.

```python
176   # The validation function
177   def valid(model, testing_loader):
178       y_true = []
179       y_pred = []
180       model.eval()
181       with torch.no_grad():
182           for _, batch in enumerate(tqdm(testing_loader, 0)):
183               ids = batch["ids"].to(device, dtype=torch.long)
184               mask = batch["mask"].to(device, dtype=torch.long)
185               targets = batch["targets"].to(device, dtype=torch.long)
186               with torch.no_grad():
187                   # we dont provide targets as arguments in model(), it returns logits not loss
188                   outputs = model(
189                       ids,
190                       # token_type_ids = None,
191                       attention_mask=mask,
192                   )
193               big_idx = outputs.logits.argmax(dim=1)  # index of the item with the highest logit/score for each message in the batch
194               y_true.extend(targets.tolist())
195               y_pred.extend(big_idx.tolist())
196       print(classification_report(y_true, y_pred, target_names=["Coldplay", "Ed Sheeran", "Maroon 5", "Taylor Swift"], digits=4))
197       return
198
199
200   # Train the model
201   for epoch in range(EPOCHS):
202       print("EPOCH:", epoch + 1)
203       train(epoch)
204
```

Και εδώ όπως και στην διαδικασία της εκπαίδευσης στο μοντέλο εισέρχονται δεδομένα και το μοντέλο κάνει προβλέψεις. Σε αυτό το σημείο δεν υπολογίζεται κάποιο loss error.

Στη γραμμή 196 με την συνάρτηση classification_report δημιουργούμαι έναν συγκεντρωτικό πίνακα με τα αποτελέσματα του μοντέλου και το score από την διαδικασία του test.

Τέλος, αφού ολοκληρώσουμε την διαδικασία της εκπαίδευσης και την διαδικασία του test σώζουμε τα βάρη του μοντέλο μας στο directory που εμείς θα επιλέξουμε.

```
205    # Save trained model and tokenizer
206    output_dir = "./save_model/"
207    model_to_save = model.module if hasattr(model, "module") else model
208    model_to_save.save_pretrained(output_dir)
209    tokenizer.save_pretrained(output_dir)
210    print("TRAINING COMPLETED")
```

Στην γραμμή 208 σώζουμε το μοντέλο μας ενώ στην γραμμή 209 σώζουμε τον tokenizer ο οποίος εκπαιδεύτηκε και αυτός κατά την διαδικασία της εκπαίδευσης.

4.3 Χρήση εκπαιδευμένου Multiclass Classifier Bert μοντέλου

Αφού ολοκληρώσαμε την διαδικασία της εκπαίδευσης μένει να φορτώσουμε το μοντέλο έτσι ώστε να το χρησιμοποιήσουμε σε καινούργια δεδομένα.

```
1   import pandas as pd
2   import torch
3   import transformers
4   from transformers import DistilBertForSequenceClassification, DistilBertTokenizer
5   import numpy as np
6   import torch.nn as nn
7   # Setting up the device for GPU usage
8   from torch import cuda
9   device = "cuda" if cuda.is_available() else "cpu"
10  MAX_LEN = 512
11  NUM_CLASSES = 4
12
13  # Load model and vocabulary from disk
14  model_file = "./save_model/"
15  model = DistilBertForSequenceClassification.from_pretrained(
16      model_file,
17      num_labels=NUM_CLASSES,
18      output_attentions=False,
19      output_hidden_states=False,
20  )
21  model.to(device)      Kontosoros, 2 months ago • 1
```

Από την γραμμή 1 μέχρι την γραμμή 8 εισάγουμε τις απαραίτητες βιβλιοθήκες που χρειαζόμαστε. Στην γραμμή 9 επιλέγουμε να τρέξουμε το μοντέλο μας σε gpu ή σε cpu. Στην συνέχεια επιλέγουμε το μέγιστο μήκος του κειμένου μας να είναι 512 λέξεις καθώς και τον αριθμό των κλάσεων που θέλουμε να προβλέψει το μοντέλο μας.

Στην γραμμή 14 φορτώνουμε τα βάρη του μοντέλου που έχουμε αποθήκευση και στην γραμμή 15 δημιουργούμαι ένα αντικείμενο του μοντέλου έτσι ώστε να ξεκινήσουμε την διαδικασία εφαρμογής του.

```
23  model.eval()
24  tokenizer = DistilBertTokenizer.from_pretrained("./save_model/")
25  inputs = tokenizer.encode_plus(text="", add_special_tokens=True, max_length=MAX_LEN, padding="max_length", return_token_type_ids=True, truncation=True)
26
27  with torch.no_grad():
28      ids = torch.tensor([inputs["input_ids"]], dtype=torch.long).to(device, dtype=torch.long)
29      mask = torch.tensor([inputs["attention_mask"]], dtype=torch.long).to(device, dtype=torch.long)
30      outputs = model(ids, mask)
31      probability = torch.softmax(outputs[0],dim=1)
32
33  result =probability.detach() .to("cpu").numpy()[0]
34  max_value = max(result)
35  probability_list = result.tolist()
36  max_index = probability_list.index(max_value)
37
```

Στην γραμμή 23 ενεργοποιούμαι το μοντέλο μας να τρέξει σε evaluation κατάσταση, στην γραμμή 24 φορτώνουμε τον tokenizer που προέκυψε από την εκπαίδευσή που κάναμε. Στην συνέχεια μετατρέπουμε κάθε λέξη του κειμένου σε αριθμό με την εφαρμογή του tokenizer και επίσης δημιουργούμαι ένα σταθερό μήκος δεδομένων με την εφαρμογή του padding.

Στην συνέχεια τροφοδοτούμαι το μοντέλο μας στην γραμμή 30 και τέλος μεταφράζουμε τις πιθανότητες που εξέρχονται από το μοντέλο ως αποτέλεσμα.

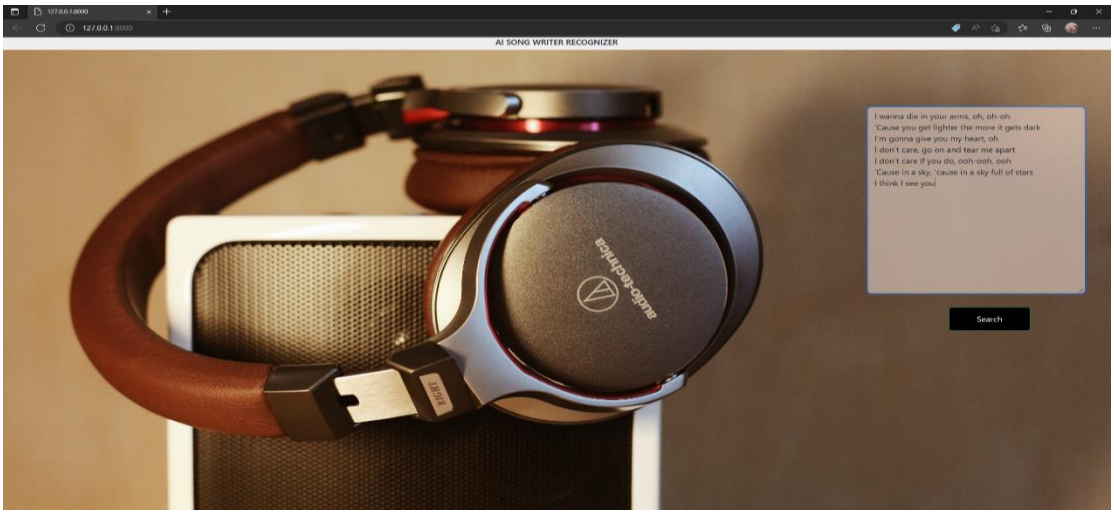
4.4 Κατασκευή Front – end περιβάλλοντος

Για την ολοκλήρωση της πτυχιακής εργασίας δημιουργήθηκε μία ολοκληρωμένη εφαρμογή με την χρήση του framework Django.

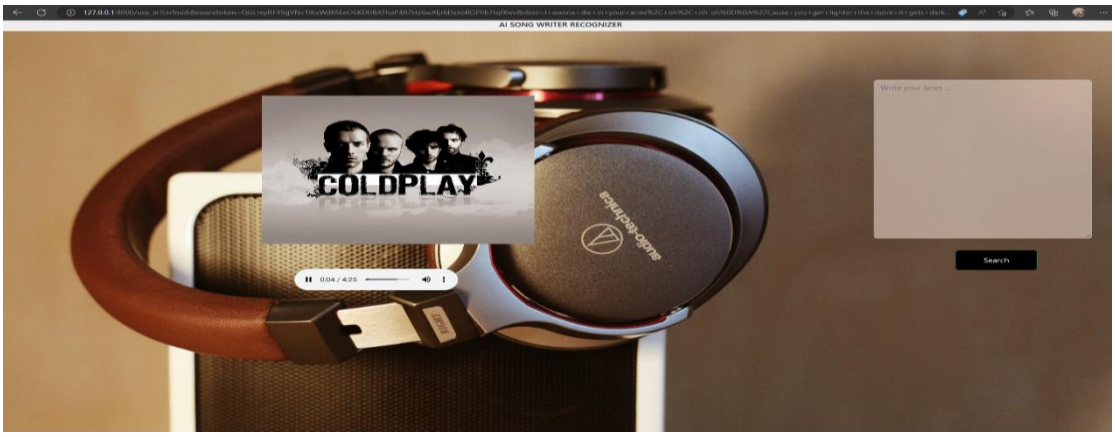
Το περιβάλλον έχει ένα πεδίο μέσα στο οποίο κάποιος μπορεί να γράψει τους στοίχους ενός τραγουδιού (να σημειωθεί ότι οι στοίχοι πρέπει να έχουν γραφτεί από τους καλλιτέχνες στους οποίους εκπαιδεύτηκε το μοντέλο).

Οι στοίχοι στην συνέχεια με το πάτημα ενός κουμπιού μεταφέρονται στο back – end μέρος της εφαρμογής όπου από εκεί περνάνε από το μοντέλο.

Πριν περάσουν από το μοντέλο ακολουθείτε όλη η διαδικασία καθαρισμού του κειμένου καθώς επίσης εφαρμόζεται ο tokenizer.



Στην πιο πάνω εικόνα έχουμε πληκτρολογήσει ένα μέρος από τους στοίχους ενός τραγουδιού που ονομάζεται “A sky full of stars” των Golden Play. Οι στοίχοι του τραγουδιού που βάλαμε βρίσκονται περίπου στην μέση. Αυτό το κάναμε έτσι ώστε το μοντέλο να μην δοκιμαστεί σε δεδομένα που έμοιαζαν και έχει δει κατά την διαδικασία της εκπαίδευσης.



Αφού πατήσουμε το κουμπί και στείλουμε τους στοίχους στο μοντέλο στην συνέχεια μεταφράζουμε την απάντηση του μοντέλου και δείχνουμε στην συνέχεια τον καλλιτέχνη ή το συγκρότημα που δημιούργησε τους συγκεκριμένους στοίχους.

4.5 Αποτελέσματα της Εκπαίδευσης

Τα αποτελέσματα που προέκυψαν από την εκπαίδευση του μοντέλου απεικονίζονται παρακάτω:

	precision	recall	f1-score	support
Coldplay	0.8689	0.8281	0.8480	64
Ed Sheeran	0.7222	0.6964	0.7091	56
Maroon 5	0.6750	0.7500	0.7105	36
Taylor Swift	0.8000	0.8085	0.8042	94
accuracy			0.7800	250
macro avg	0.7665	0.7708	0.7680	250
weighted avg	0.7822	0.7800	0.7806	250

Αποτελέσματα εκπαίδευσης του μοντέλου

Το μοντέλο μετά την εκπαίδευσή του δοκιμάστηκε μέσα από την διαδικασία του Test. Κατά την διαδικασία αυτή τροφοδοτούμαι το μοντέλο με δεδομένα που δεν είδε κατά την εκπαίδευση έτσι ώστε να δούμε πόσο καλά μπορεί να προβλέψει.

Στην τεχνητή νοημοσύνη, το precision (ακρίβεια) αναφέρεται στο ποσοστό των προβλέψεων που είναι σωστές θετικές προβλέψεις σε σχέση με όλες τις θετικές προβλέψεις.

Για παράδειγμα, ας υποθέσουμε ότι έχουμε ένα σύστημα ταξινόμησης email σε spam και μη spam. Εάν το σύστημα προβλέπει ότι 100 email είναι spam, και από αυτά τα 100, τα 90 είναι πραγματικά spam και τα υπόλοιπα 10 είναι λάθος ταξινομημένα, τότε το precision είναι 90%.

Συνοπτικά, το precision μετρά την ακρίβεια του μοντέλου στο να προβλέπει θετικά αποτελέσματα, δηλαδή την ικανότητά του να αναγνωρίζει σωστά τα στοιχεία που αναζητούμε.

Αντίστοιχα η μέτρηση recall (ανάκληση) αναφέρεται στο ποσοστό των σωστών θετικών προβλέψεων σε σχέση με όλες τις πραγματικές θετικές εμφανίσεις στα δεδομένα. Εάν υπάρχουν συνολικά 150 spam email και το σύστημα καταφέρνει να τα αναγνωρίσει σωστά τα 90, τότε το recall είναι 60%.

Συνοπτικά, το recall μετρά την ικανότητα του μοντέλου να εντοπίζει όλα τα θετικά αποτελέσματα, δηλαδή την ικανότητά του να αναγνωρίζει σωστά όλα τα στοιχεία που ενδιαφέρουν το μοντέλο.

Η μέτρηση F1 score είναι ένας μετρικός δείκτης που συνδυάζει τα precision και recall με σκοπό να αξιολογήσει την συνολική απόδοση ενός μοντέλου στην ταξινόμηση ή στην αναγνώριση αντικειμένων σε εικόνες.

Ο τύπος του F1 score είναι ο εξής:

F1 score = 2 * (precision * recall) / (precision + recall)

Το F1 score είναι ένας αριθμός μεταξύ 0 και 1 και το καλύτερο δυνατό F1 score είναι 1, που σημαίνει ότι το μοντέλο έχει τέλεια ακρίβεια και ανάκληση. Αντίθετα, αν το F1 score είναι 0, αυτό σημαίνει ότι το μοντέλο δεν έχει καθόλου ακρίβεια ή ανάκληση.

Συνολικά, το F1 score παρέχει μια εξισορροπημένη αντίληψη για την απόδοση ενός μοντέλου στον εντοπισμό και την αναγνώριση αντικειμένων στα δεδομένα, καθώς λαμβάνει υπόψη τόσο την ακρίβεια όσο και την ανάκληση του μοντέλου.

Τέλος, η μέτρηση accuracy ακρίβεια) αναφέρεται στο ποσοστό των σωστών προβλέψεων που κάνει ένα μοντέλο στα δεδομένα ελέγχου.

Η ακρίβεια είναι μια απλή μετρική που μας δίνει μια γενική ιδέα για την απόδοση ενός μοντέλου, αλλά μπορεί να είναι ανεπαρκής σε περιπτώσεις όπου η κλάση που πρέπει να προβλεφθεί είναι ανισορροπημένη , δηλαδή υπάρχουν περισσότερα δείγματα μιας κλάσης σε σχέση με μια άλλη. Σε αυτές τις περιπτώσεις, άλλες μετρικές όπως το precision, το recall και το F1 score μπορούν να παρέχουν πιο ενδεικτικά αποτελέσματα για την απόδοση του μοντέλου.

ΚΕΦΑΛΑΙΟ 5

5.1 Γενικά Συμπεράσματα

Στην παρούσα πτυχιακή εργασία έγινε εκτενής αναφορά της αρχιτεκτονική των μοντέλων Transformers. Πραγματοποιήθηκε αναλυτική περιγραφή των διαφόρων layers από τα οποία αποτελείται ένα μοντέλο Transformer καθώς επίσης έγινε ανάλυση των δεδομένων και τα στάδια που περνάνε τα δεδομένα μέσα από τα μοντέλα Transformer.

Στην συνέχεια δόθηκαν παραδείγματα και αναλύθηκαν διάφορες εφαρμογές που σχετίζονται με NLP προβλήματα και πως η χρήση των Transformers μπορούν να βοηθήσουν σε αυτά.

Επίσης έγινε αναφορά για το μοντέλο Tokenizer και πώς μπορεί κάποιος να εκπαιδεύσει από την αρχή ένα τέτοιο μοντέλο με δεδομένα που συνδέονται εξ ολοκλήρου με το πρόβλημα που θέλει να λύσει.

Στην συνέχεια έγινε ανάλυση της αρχιτεκτονικής του μοντέλου Bert το οποίο στην συνέχεια εκπαιδεύτηκε πάνω σε δεδομένα τα οποία ήταν στοίχοι τραγουδιών. Πριν την διαδικασία της εκπαίδευσης έγινε αναφορά στα δεδομένα τα οποία χρησιμοποιήθηκαν για την εκπαίδευσή του καθώς επίσης αναλύθηκε ο κώδικας της εκπαίδευσης του μοντέλου.

Ύστερα έγινε ανάλυση του κώδικα που αφορά την φόρτωση του μοντέλου και την χρήση του σε δεδομένα τα οποία δεν είχε δει κατά την διαδικασία της εκπαίδευσης.

Τέλος, έγινε σύντομη αναφορά για το front – end μέρος της εφαρμογής μέσω εικόνων. Σαν τελευταίο στάδιο αναλύθηκαν τα αποτελέσματα που προέκυψαν μετά την διαδικασία της εκπαίδευσης καθώς επίσης έγινε αναλυτική περιγραφή συγκεκριμένων μετρήσεων.

5.2 Περιοχές για Περεταίρω Ανάπτυξη

Οι περιοχές για περεταίρω ανάπτυξη θα μπορούσαν να είναι:

- Η εκπαίδευση ενός επιπλέον μοντέλου που θα μπορούσε να αναγνωρίζει και τον ήχο του τραγουδιού έτσι ώστε να βοηθάει το πρώτο μοντέλο στην τελική απόφαση που θα πάρει έτσι ώστε να επιλέξει τον καλλιτέχνη στον οποίο ανήκουν οι στοίχοι. Ουσιαστικά θα έχουμε εκπαιδεύσει δυο μοντέλα, το πρώτο θα είναι εκπαιδευμένο με στοίχους και το δεύτερο θα είναι εκπαιδευμένο με την μελωδία του συγκεκριμένου τραγουδιού.
- Να εκπαιδευτεί ένας Tokenizer από την αρχή με τα δεδομένα των στοίχων και να μην είναι ένας pretrained tokenizer. Σαν διαδικασία δεν θα οδηγήσει σε βελτίωση της απόδοσης του μοντέλου απλά θα είναι πιο σωστή η εκπαίδευση του μοντέλου και σίγουρα σε δεδομένα όπου το μέγεθος ξεπερνάει τα 512 tokens θα βελτιώσει πολύ αυτό το πρόβλημα.
- Βελτίωση του μοντέλου με περαιτέρω εκπαίδευση. Το transformer μοντέλο απαιτεί μεγάλη ποσότητα δεδομένων για να εκπαιδευτεί και να επιτύχει καλή απόδοση. Η περαιτέρω εκπαίδευση του μοντέλου σε περισσότερα δεδομένα ή η χρήση τεχνικών όπως η αύξηση δεδομένων (data augmentation) μπορεί να βελτιώσει την απόδοσή του.
- Πειραματισμός με διαφορετικά hyperparameters. Η ρύθμιση των hyperparameters του μοντέλου μπορεί να οδηγήσει σε πολύ καλύτερα αποτελέσματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ – ΙΣΤΟΣΕΛΙΔΕΣ – ΕΠΙΣΤΗΜΟΝΙΚΑ ΑΡΘΡΑ

[1] https://www.ibm.com/topics/artificial-intelligence

[2] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6381354/

[3] https://www.techopedia.com/definition/33181/training-data

[4] https://www.exxactcorp.com/blog/Deep-Learning/What-is-Text-Classification

[5] https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP

[6] https://arxiv.org/pdf/1810.04805v2.pdf

[7] https://www.kaggle.com/

[8] https://www.djangoproject.com/

[9] https://towardsdatascience.com/supervised-unsupervised-and-deep-learning-aa61a0e5471c

[10] https://www.researchgate.net/profile/Vladimir-Nasteski/publication/328146111_An_overview_of_the_supervised_machine_learning_methods/links/5c1025194585157ac1bba147/An-overview-of-the-supervised-machine-learning-methods.pdf

[11] https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[12] https://arxiv.org/pdf/2005.14165.pdf

[13] https://en.wikipedia.org/wiki/Supercomputer

[14] https://arxiv.org/pdf/2206.02659.pdf

[15] https://www.hindawi.com/journals/misy/2022/6603576/

[16] file:///C:/Users/mkont/Downloads/Automated_News_Summarization_Using_Transformers.pdf

[17] https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

[18] Denis Rothman : “Transformers for Natural Language Processing, Second Edition”

[19] http://torch.ch/

[20] https://aclanthology.org/P02-1040.pdf

[21] https://aclanthology.org/P16-1162.pdf

[22] https://aclanthology.org/2022.wassa-1.25.pdf

[23] https://deepai.org/machine-learning-glossary-and-terms/logit

[24] https://aclanthology.org/2020.acl-main.170.pdf