ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	**Ταξινόμηση Πολιτικών Θέσεων σε Κειμενικά Δεδομένα με Χρήση Νευρωνικών Δικτύων BERT** **Classification of Political Positions in Textual Data using BERT Neural Networks**
Ονοματεπώνυμο Φοιτητή	**Νικόλαος Κρουστάλλης**
Πατρώνυμο	**Ευάγγελος**
Αριθμός Μητρώου	**Π20108**
Επιβλέπων	**Διονύσιος Σωτηρόπουλος**

Ημερομηνία Παράδοσης Σεπτέμβριος 2024

Copyright

Πίνακας Περιεχομένων

Επισκόπηση

Η επεξεργασία φυσικής γλώσσας (Natural Language Processing) είναι ένα πεδίο της επιστήμης των υπολογιστών που μελετά την αλληλεπίδραση του ανθρώπου και του υπολογιστή μέσω της κατανόησης της φυσικής γλώσσας με την οποία επικοινωνούν οι άνθρωποι. Λόγω των διάφορων εφαρμογών της επεξεργασίας φυσικής γλώσσας το πεδίο αυτό έχει παρουσιάσει μεγάλη ανάπτυξη τα τελευταία χρόνια. Η ανάπτυξη αυτή οφείλεται στις διάφορες νέες τεχνικές και προσεγγίσεις που έχουν εφαρμοστεί τα τελευταία χρόνια και έχουν συνεισφέρει στην καλύτερη κατανόηση του αντικειμένου και την εφαρμογή του σε όλο και περισσότερα σενάρια χρήσης.

Η αύξηση του όγκου δεδομένων που μπορούμε να αποθηκεύσουμε και να επεξεργαστούμε (Big Data) οδήγησε στην εισαγωγή μεθόδων μηχανικής μάθησης (Machine Learning). Ανά τα έτη έχουν προταθεί αρχιτεκτονικές Αναδρομικών Νευρωνικών Δικτύων (Recurrent Neural Networks, RNN) και Συνελικτικών Νευρωνικών Δικτύων (Convolutional Neural Networks, CNN). Η αρχιτεκτονική η οποία έχει επικρατήσει και χρησιμοποιείται σε διάφορες εφαρμογές[4] αυτό το πεδίο μελέτης παρουσιάζοντας τα καλύτερα αποτελέσματα είναι τα Μοντέλα Μετασχηματιστών (Transformer Models).

Τα Μοντέλα Μετασχηματιστών (Transformer Models) βασίζονται σε μία νέα βασίζονται σε μία βασική ιδέα τον μηχανισμό προσοχής (attention mechanism). Αυτού του είδους τα Νευρωνικά Δίκτυα έχουν επικρατήσει σε διάφορους τομείς εφαρμογών καθώς δεν χρησιμοποιούν τεχνικές αναδρομής ή συνέλιξης και επομένως δεν έχουν τόσο υψηλές υπολογιστικές απαιτήσεις. Επιπλέον παρουσιάζουν καλύτερα αποτελέσματα σε εφαρμογές που απαιτούν παράλληλους υπολογισμούς και επιπλέον μικρότερες χρονικές απαιτήσεις κατά την διαδικασία εκπαίδευσης τους.

Ο τομέας της επεξεργασίας φυσικής γλώσσας αποτελείται από διάφορες εφαρμογές με διαφορετικούς στόχους ή διαφορετική προσέγγιση κάθε φορά. Υπάρχουν εφαρμογές όπως η εξαγωγή χαρακτηριστικών η οποία εξάγει χαρακτηριστικά για το κείμενο που λαμβάνει σαν είσοδο, η σύνοψη του κειμένου η οποία παράγει μία περίληψη του δοσμένου κειμένου και η συμπλήρωση κενού η οποία προβλέπει τις λέξεις που λείπουν από ένα κείμενο.

Η εφαρμογή η οποία θα επικεντρωθούμε σε αυτή την εργασία είναι η ταξινόμηση κειμένου, στην οποία με ένα δοσμένο κείμενο εισόδου το σύστημα το ταξινομεί σε μία κατηγορία. Αυτό θα επιτευχθεί αναθέτοντας μία ετικέτα (label) που αντιστοιχεί σε μία κλάση (class) η οποία αναπαριστά μία γενική κατηγορία όμοιων αντικειμένων ή εννοιών. Αυτή η διαδικασία είναι πολύ χρήσιμη σε διάφορες εφαρμογές όπως ανάλυση συναισθήματος, την ανίχνευση ψευδών ειδήσεων και το φιλτράρισμα ανεπιθύμητων email (spam filtering).

Abstract

Natural Language Processing (NLP) is a field of computer science that studies the interaction between humans and computers through the understanding of natural language, which is the way humans communicate. Due to its various applications, NLP has experienced significant growth in recent years. This growth is attributed to the new techniques and approaches that have been applied in recent years, which have contributed to a better understanding of the field and its application in an increasing number of use cases.

The increase in the volume of data we can store and process (Big Data) has led to the introduction of Machine Learning methods. Over the years, architectures such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) have been proposed. However, the architecture that has prevailed and is used in various applications within this field, yielding the best results, is the Transformer Models.

Transformer Models are based on a fundamental idea: the attention mechanism. This type of neural network has gained prominence in various application domains because it does not use recurrence or convolution techniques, and therefore does not have such high computational demands. Furthermore, it produces better results in applications that require parallel computations and has shorter training times.

The field of Natural Language Processing comprises various applications with different objectives or approaches each time. For example, there are applications like feature extraction, which extracts key features from the input text, text summarization, which generates a summary of the given text, and fill-in-the-blank, which predicts the missing words in a text.

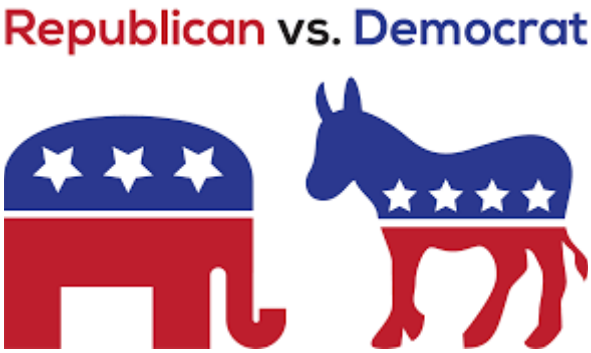
The application that we will focus on in this work is text classification, where the system categorizes a given input text into a specific class. This is achieved by assigning a label that corresponds to a class, which represents a general category of similar objects or concepts. This process is very useful in various applications such as sentiment analysis, fake news detection, and spam filtering.

Επισκόπιση

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός δυαδικού ταξινομητή (binary classifier) ο οποίος να μπορεί να κατηγοριοποιήσει διάφορα tweets των χρηστών του X (πρώην twitter).

Εικόνα 1: Το logo του X (πρώην twitter)

Στην διαδικασία αυτή θα χρησιμοποιήσουμε διάφορες τεχνικές μηχανικής μάθησης με στόχο την ανάπτυξη του ζητούμενου ταξινομητή[1]. Η γενική κατηγορία των εφαρμογών επεξεργασίας φυσικής γλώσσας παρουσιάζει ραγδαία ανάπτυξη τα τελευταία χρόνια και παρουσιάζει προοπτικές για διαφόρων ειδών εφαρμογές που καλύπτουν ανάγκες. Η εφαρμογή που θα επικεντρωθούμε σε αυτή την εργασία είναι η ταξινόμηση[5], δηλαδή η κατηγοριοποίηση (classification) ενός δείγματος σε μία από τις διαθέσιμες κατηγορίες. Πιο συγκεκριμένα στο παράδειγμα μας θα παρουσιάσουμε έναν δυαδικό ταξινομητή (binary classifier) ο οποίος θα διαχωρίζει τα tweets και θα κατηγοριοποιεί την πολιτική θέση του κάθε χρήστη είτε ως ψηφοφόρο των Δημοκρατικών (Democratic Party) ή των Ρεπουμπλικάνων (Republican Party)[2].



Εικόνα 2: Τα σύμβολα των δύο κομμάτων

Η ταξινόμηση αυτή θα γίνει με χρήση τεχνικών μηχανικής μάθησης και πιο συγκεκριμένα νευρωνικών δικτύων. Θα χρησιμοποιήσουμε ένα προ-εκπαιδευμένο (pre-trained) νευρωνικό δίκτυο αρχιτεκτονικής BERT (Bidirectional Encoder Representations from Transformers) το οποίο θα τροποποιήσουμε ώστε να υπάρχουν δύο έξοδοι που αντιστοιχούν στις δύο κλάσεις που αναφέραμε παραπάνω. Η τεχνική αυτή ονομάζεται μάθηση με μετάθεση (Transfer Learning)[8] και χρησιμοποιείται σε διάφορες εφαρμογές όπου η εκπαίδευση ενός ολόκληρου μοντέλου έχει μεγάλες απαιτήσεις σε χρόνο και πόρους.
Η εκπαίδευση του μοντέλου θα γίνει με χρήση του αντίστοιχου συνόλου δεδομένων (Dataset) από το Kaggle, η γλώσσα προγραμματισμού θα είναι η python και θα χρησιμοποιήσουμε το περιβάλλον Colab της Google[6] για την συγγραφή του κώδικα και την εκπαίδευση του μοντέλου.

Εικόνα 3: Το logo της γλώσσας προγραμματισμού python

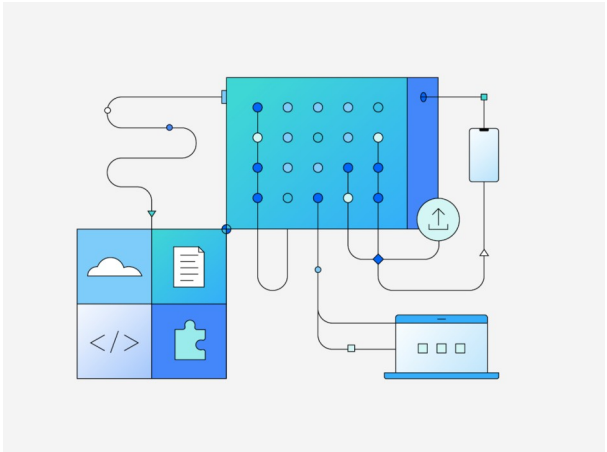
Εικόνα 4: Το logo του περιβάλλοντος Google Colab

Στην συνέχεια θα αναλύσουμε περαιτέρω τις τεχνικές που παρουσιάσαμε συνοπτικά παραπάνω και στην συνέχεια θα παρουσιάσουμε την τελική εφαρμογή.

Μηχανική Μάθηση και Νευρωνικά Δίκτυα

Η Μηχανική Μάθηση (Machine Learning, ML)[7] είναι ένας τομέας της τεχνητής νοημοσύνης και της επιστήμης των υπολογιστών που χρησιμοποιεί σύνολα δεδομένων και συγκεκριμένους αλγορίθμους και έχει σαν στόχο την δημιουργία ενός συστήματος τεχνητής νοημοσύνης που μπορεί να μιμηθεί τον τρόπο που μαθαίνουν οι άνθρωποι, βελτιώνοντας ταυτόχρονα την ακρίβεια του.

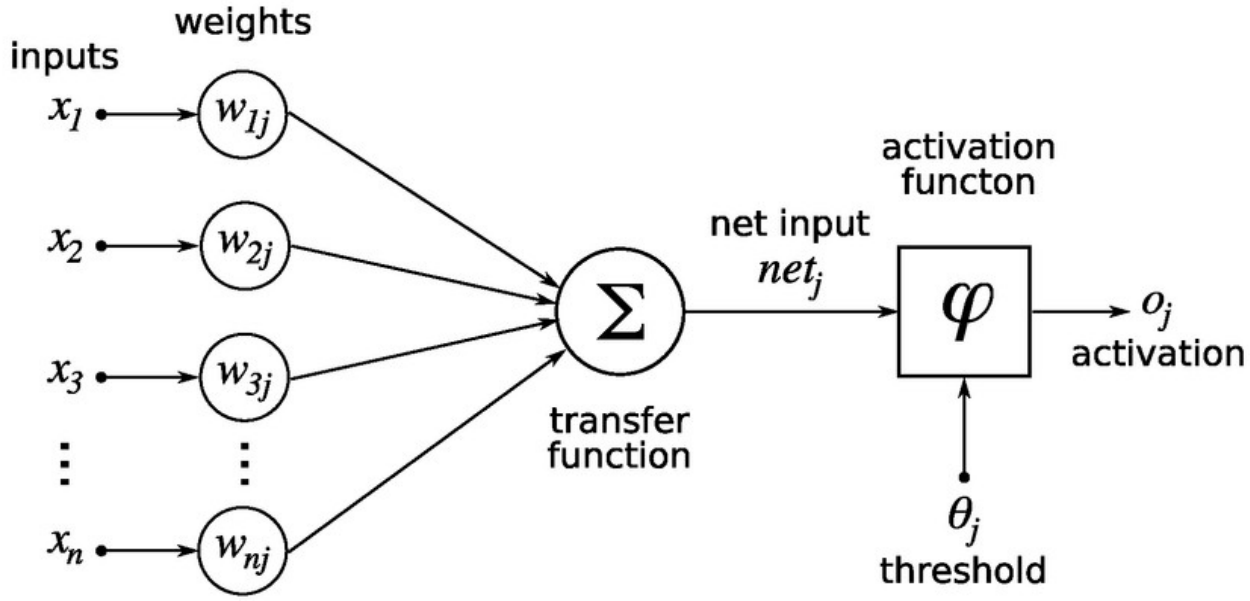
Ένας αλγόριθμος μηχανικής μάθησης αποτελείται από τρία επιμέρους τμήματα. Το πρώτο τμήμα είναι μία διαδικασία απόφασης (Decision Process). Σε αυτό το βήμα οι αλγόριθμοι μηχανικής μάθησης πραγματοποιούν μία πρόβλεψη ή μία ταξινόμηση. Με βάση κάποια δεδομένα εισόδου τα οποία μπορεί να είναι ονομασμένα η όχι, δηλαδή να γνωρίζουμε την κλάση στην οποία ανήκουν ή όχι, μπορούμε να παράγουμε μία εκτίμηση για το μοτίβο που ακολουθούν τα δεδομένα. Το επόμενο τμήμα είναι μία συνάρτηση σφάλματος (Error Function). Σκοπός της συνάρτησης σφάλματος είναι η αξιολόγηση της πρόβλεψης του μοντέλου. Εάν υπάρχουν γνωστά παραδείγματα, δηλαδή ονομασμένα δεδομένα, η συνάρτηση σφάλματος χρησιμοποιείται για να αξιολογήσει την ακρίβεια του μοντέλου. Το τελευταίο τμήμα του αλγορίθμου είναι η διαδικασία βελτιστοποίησης του μοντέλου (Model Optimization Process). Αυτό το βήμα βασίζεται στην υπόθεση πως εάν ένα μοντέλο μπορεί να πραγματοποιήσει με πιο ακριβή τρόπο την κατηγοριοποίηση/ταξινόμηση στο σύνολο των δεδομένων εκπαίδευσης, τότε τα βάρη αναπροσαρμόζονται με στόχο την μείωση της διαφοράς ανάμεσα στην εκτίμηση του μοντέλου και τις πραγματικές τιμές. Ο αλγόριθμος εκτελεί επαναληπτικά αυτά τα βήματα αναπροσαρμόζοντας τα βάρη με αυτόνομο τρόπο μέχρις ότου να επιτύχει το προκαθορισμένο όριο ακρίβειας του μοντέλου.



Εικόνα 5: Αφηρημένη αναπαράσταση της διαδικασίας της Μηχανικής Μάθησης

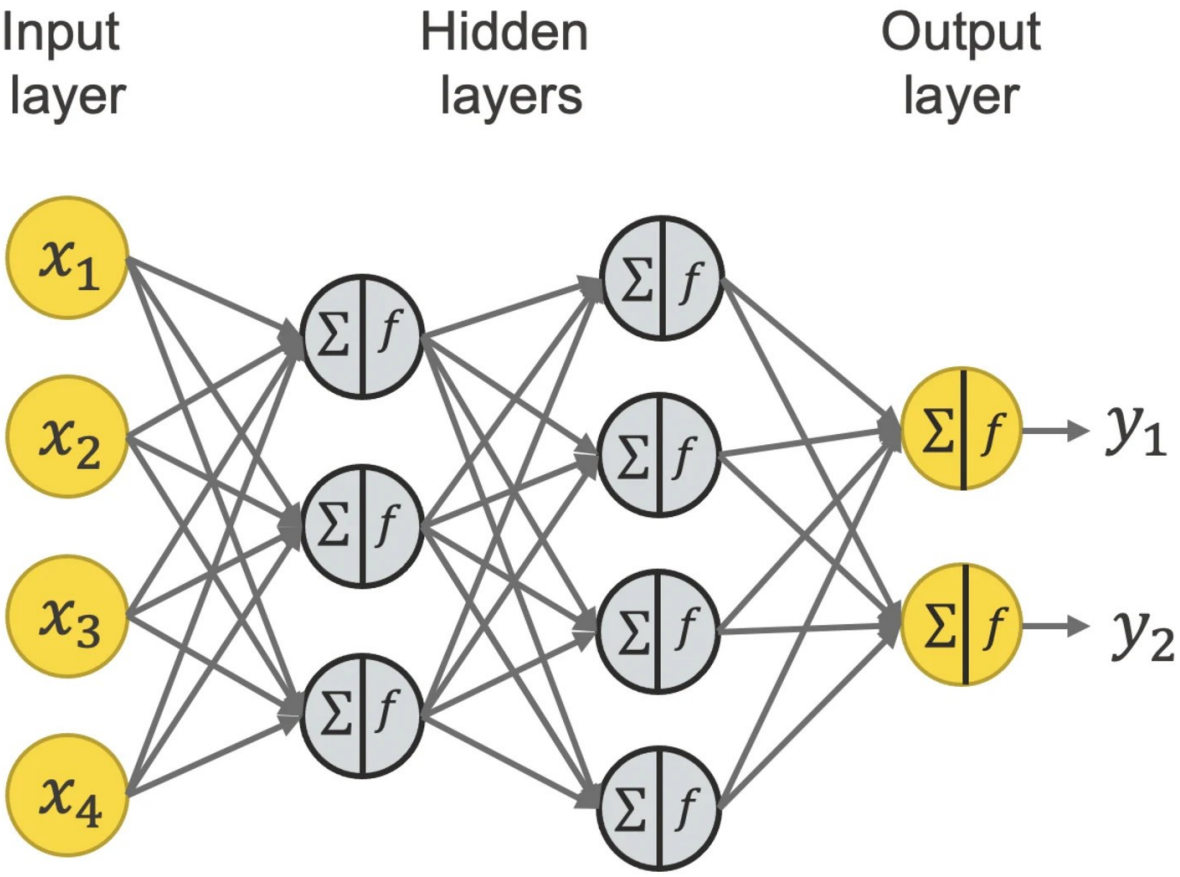
Πηγή: https://www.ibm.com/topics/machine-learning

Τα νευρωνικά δίκτυα (Neural Networks, NN) αποτελούνται από επίπεδα νευρώνων (node layers) και έχουν ένα επίπεδο εισόδου (input layer), ένα ή περισσότερα κρυφά επίπεδα (hidden layers) και ένα επίπεδο εξόδου. Ο κάθε νευρώνας ενώνεται με τους υπόλοιπους και έχει δικά του βάρη και δικό του κατώφλι. Εάν η τιμή αυτού του νευρώνα περνά το όριο που έχει οριστεί στο κατώφλι τότε ο νευρώνας ενεργοποιείται και στέλνει το αποτέλεσμα του στο επόμενο επίπεδο, ενώ σε διαφορετική περίπτωση τα δεδομένα δεν προχωρούν στο επόμενο επίπεδο. Έναν ένα νευρωνικό δίκτυο αποτελείται από πάνω από τρία επίπεδα νευρώνων, τότε ονομάζεται βαθύ (deep neural network), δηλαδή κάθε νευρωνικό δίκτυο με πάνω από ένα κρυφά στρώματα νευρώνων είναι ένα βαθύ νευρωνικό δίκτυο.



Εικόνα 6: Ένας απλός Νευρώνας (neuron). Σε αυτό το σχήμα οι είσοδοι συμβολίζονται με x τα βάρη με w και η πόλωση (bias) με b. Αρχικά η κάθε είοσοδος πολλαπλασιάζεται με το αντίστοιχο βάρος και στην συνέχεια αθροίζονται, δηλαδή με απλά λόγια υπολογίζεται τοι εσωτερικό γινόμενο των διανυσμάτων x και w το οποίο στην συνέχεια περνά από την συνάρτηση ενεργοποίησης σ η οποία ελέγχει εάν η τιμή είναι μεγαλύτερη από το προκαθορισμένο κάτω όριο. Εάν η τιμή περνά την τιμή κατωφλίωσης ο νευρώνας ενεργοποιείται και στέλνει την έξοδο του σαν είσοδο στο επόμενο επίπεδο.

Πηγή: https://www.researchgate.net/figure/Architecture-of-a-single-neuron-in-a-neural-network-5_fig2_326826129



Εικόνα 7: Αυτή η εικόνα παρουσιάζει ένα νευρωνικό δίκτυο με τέσσερις εισόδους x1,x2,x3,x4 στο επίπεδο εισόδου και δύο εξόδους y1,y2 στο επίπεδο εξόδου και δύο κρυφά επίπεδα. Ο κάθε νευρώνας αντιστοιχεί στο σχήμα που περιγράψαμε παραπάνω. Η διαδικασία εκπαίδευσης εφαρμόζεται σε νευρωνικά δίκτυα που στην γενική τους μορφή μοιάζουν με το παραπάνω.

Πηγή: https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks

Η μάθηση με μετάθεση (Transfer Learning) [8] είναι μία τεχνική της μηχανικής μάθησης στην οποία ένα ήδη εκπαιδευμένο μοντέλο σε ένα σύνολο δεδομένων χρησιμοποιείται για την βελτίωση της απόδοσης ενός άλλου μοντέλου σε ίδια η παρόμοια εφαρμογή με διαφορετικό σύνολο δεδομένων. Με πιο απλά λόγια η τεχνική της μάθησης με μετάθεση χρησιμοποιεί μία υπάρχουσα λύση ενός προβλήματος για την λύση ενός διαφορετικού προβλήματος.

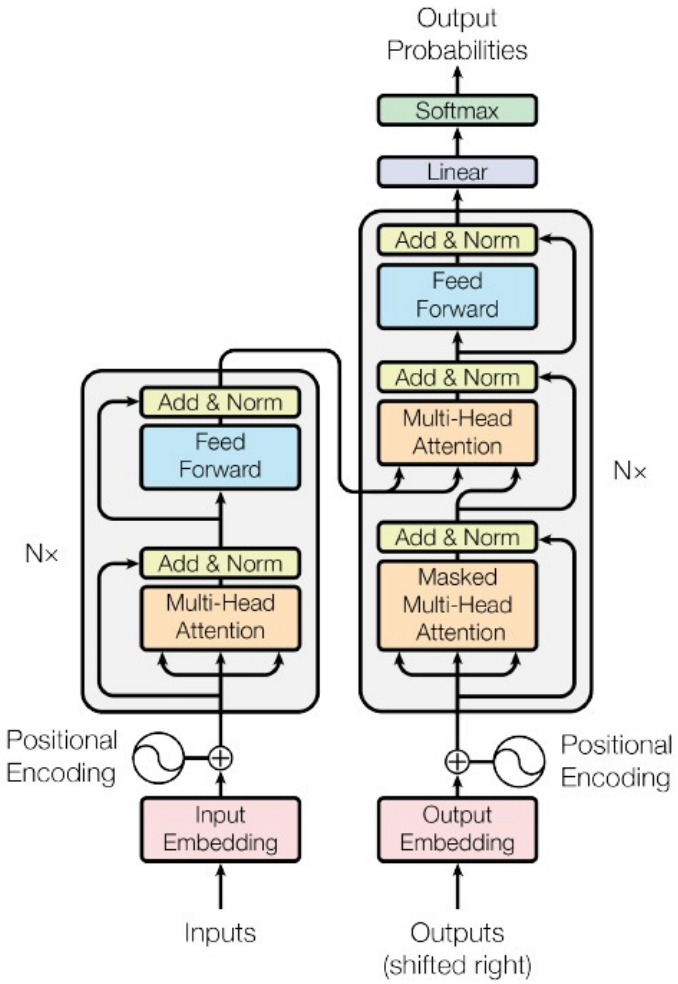
Σε τεχνικό επίπεδο η διαδικασία αυτή πραγματοποιείται σε ένα ήδη εκπαιδευμένο νευρωνικό δίκτυο, όπως αυτό στην εικόνα 7, στο οποίο αλλάζουμε τους νευρώνες του επιπέδου εξόδου ή και κάποιων αμέσως προηγούμενων επιπέδων και πρακτικά εκπαιδεύουμε μόνο αυτά τα επίπεδα που αλλάξαμε ώστε να ταιριάζουν στην εφαρμογή που θα αναπτύξουμε.

Αρχιτεκτονική Transformers

Η αρχιτεκτονική του μοντέλου μετασχηματιστή (Transformer Model)[3] βασίζεται στον μηχανισμό προσοχής (attention mechanism) ο οποίος χρησιμοποιείται για να δημιουργήσει καθολικές συσχετίσεις ανάμεσα στην είσοδο και την έξοδο του μοντέλου. Το κύριο πλεονέκτημα των Transformer είναι οι μεγάλες δυνατότητας παράλληλων υπολογισμών που μεταφράζεται σε σημαντική μείωση του χρόνου εκπαίδευσης των μοντέλων.

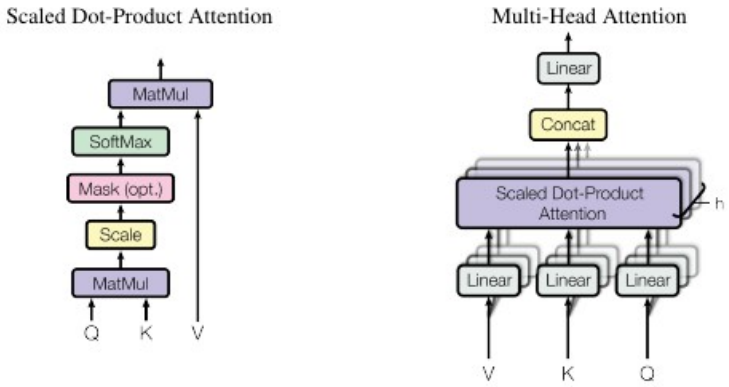
Η αρχιτεκτονική του μοντέλου βασίζεται στην δομή κωδικοποιητή – αποκωδικοποιητή (encoder – decoder). Ο κωδικοποιητής αντιστοιχεί την ακολουθία εισόδου που βρίσκεται στην μορφή συμβολικής αναπαράστασης x (x_1 , x_2 , … , x_n) σε μία μορφή συνεχούς αναπαράστασης z (z_1 , z_2 , … , z_n). Ο αποκωδικοποιητής έχοντας σαν είσοδο την ακολουθία z, παράγει μία ακολουθία εξόδου y (y_1 , y_2 , … , y_n) για ένα σύμβολο την φορά. Σε κάθε βήμα το μοντέλο είναι αυτό-αναδρομικό, δηλαδή χρησιμοποιεί τα σύμβολα που έχουν δημιουργηθεί ως επιπλέον είσοδο όταν δημιουργεί το κείμενο.

Το μοντέλο μετασχηματιστή ακολουθεί την αρχιτεκτονική των αυτό-αναφορικών μηχανισμών προσοχής (self-attention) και των ανά σημείο μηχανισμών (point-wise attention) προσοχής με πλήρως συνδεδεμένα επίπεδα νευρώνων (fully connected layers) τόσο για τον κωδικοποιητή όσο και για τον αποκωδικοποιητή.



Εικόνα 8: Η αρχιτεκτονική του μοντέλου Transformer

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



Εικόνα 9: Οι μηχανισμοί προσοχής (attention mechanisms)

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

Τα μοντέλα μετασχηματιστών είναι ένα μεγάλο μέρος της μελέτης του τομέα της επεξεργασίας φυσικής γλώσσας καθώς παρουσιάζουν πολύ καλά αποτελέσματα σε ένα ευρύ φάσμα των προβλημάτων της επεξεργασίας φυσικής γλώσσας, από την ταξινόμηση κειμένου στην σύνθεση κειμένου. Ο μηχανισμός προσοχής που περιγράψαμε παραπάνω παίζει πολύ σημαντικό ρόλο στην λειτουργία αυτών των μοντέλων.

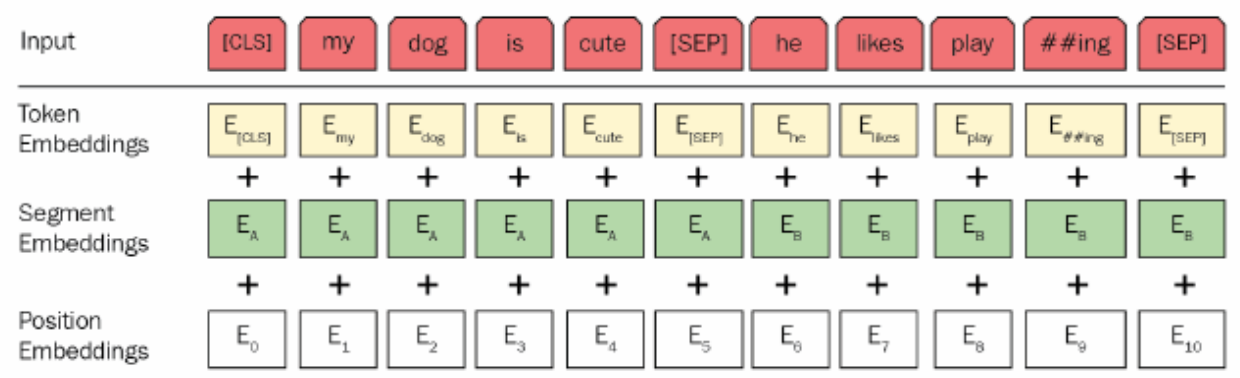
Η διαδικασία της μάθησης με μετάθεση που παρουσιάσαμε σε προηγούμενη ενότητα εφαρμόζεται με μεγάλη επιτυχία σε τέτοιου τύπου μοντέλα. Αυτό συμβαίνει καθώς υπάρχουν πολλά μοντέλα μετασχηματιστή τα οποία χρησιμοποιούνται για την κατανόηση της ίδιας της γλώσσας. Αυτού του είδους τα μοντέλα ονομάζονται μοντέλα γλώσσας (language models) και παρέχουν ένα μοντέλο για την γλώσσα την οποία εκπαιδευτήκαν.

Αυτή η διαδικασία υλοποιείται χρησιμοποιώντας έναν ήδη εκπαιδευμένο ταξινομητή μαζί με ένα μοντέλο που μετατρέπει το κείμενο σε μορφή κατάλληλη για είσοδο στο μοντέλο. Αυτή η διαδικασία ονομάζεται συμβολοποίηση (Tokenization)[9]. Τα μοντέλα της κατηγορίας συμβολοποίησης (tokenizer) είναι μοντέλα τα οποία αντιστοιχούν την αναπαράσταση κειμένου σε μία αριθμητική αναπαράσταση η οποία στην συνέχεια μπορεί να επεξεργαστεί από το μοντέλο.

Μοντέλο BERT

Το μοντέλο μετασχηματιστή που θα χρησιμοποιήσουμε στην παρούσα εργασία ονομάζεται BERT. Το BERT είναι ακρωνύμιο και σημαίνει Bidirectional Encoder Representations from Transformers και είναι ένας πολυστρωματικός κωδικοποιητής (multilayer encoder) βασισμένος στην αρχική αρχιτεκτονική των μοντέλων Transformer. Το αρχικό μοντέλο μετασχηματιστή είχε αναπτυχθεί για επεξεργασία φυσικής γλώσσας, αλλά το BERT παρουσιάζει βελτιώσεις στην αρχιτεκτονική αυτή με στόχο την καλύτερη μοντελοποίηση της γλώσσας (language modelling). Αυτό το προ-εκπαιδευμένο μοντέλο παρουσιάζει δυνατότητες καθολικής κατανόησης (global understanding) της γλώσσας στην οποία εκπαιδεύτηκε.

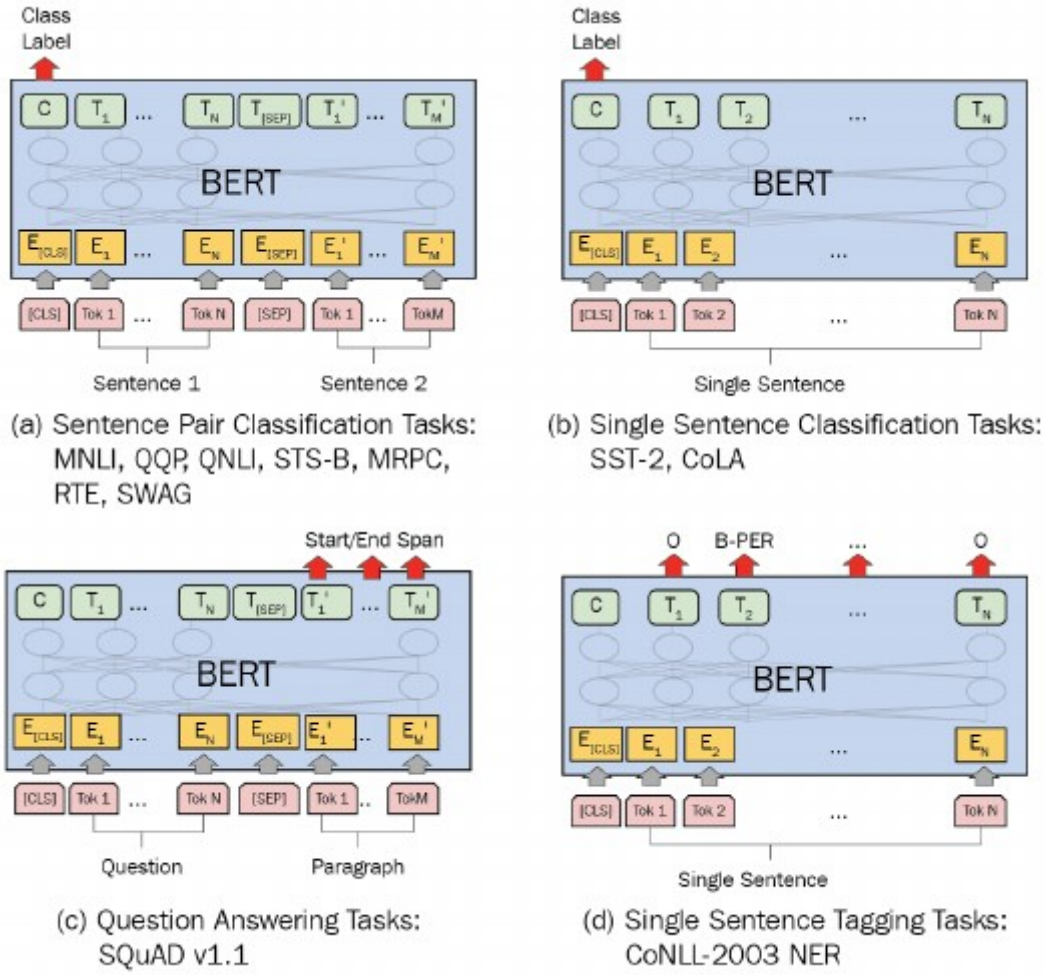
Το μοντέλο BERT παίρνει σαν είσοδο αναπαραστάσεις που έχουν παραχθεί από τον συμβολοποιητή που ονομάζεται WordPiece. Ο κύριος λόγος που το BERT και τα λοιπά μοντέλα Transformer χρησιμοποιούν συμβολοποιητές υπολέξεων (subword) καθώς μπορούν να διαχειριστούν άγνωστα σύμβολα. Επιπλέον το BERT χρησιμοποιεί κωδικοποιητές θέσης (positional encodings) ώστε να είναι γνωστή η σειρά με την οποία τα tokens δίνονται στο μοντέλο σαν είσοδο.



Εικόνα 10: Το μοντέλο BERT

Πηγή: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

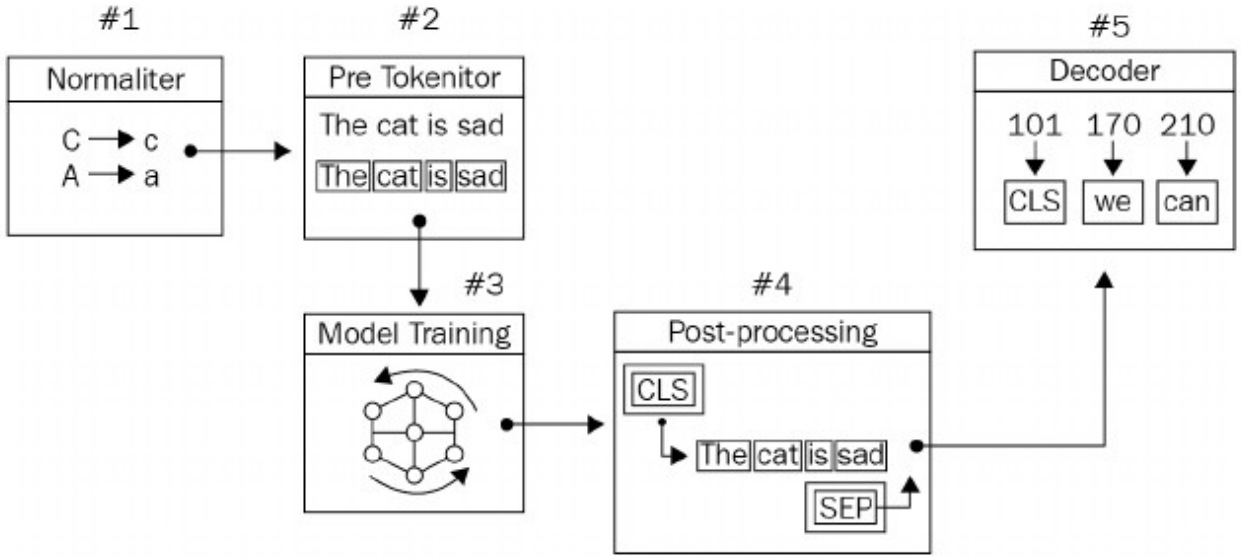
Η εκπαίδευση του BERT στην γενική περίπτωση γίνεται σαν αυτόματος κωδικοποιητής (autoencoder), δηλαδή με την ίδια είσοδο και την ίδια έξοδο, αλλά ανά την εφαρμογή για την οποία εκπαιδεύεται το μοντέλο χρησιμοποιούνται διαφορετικά μέρη της εξόδου με στόχο να "απαντά" σε διαφορετικά ερωτήματα ανάλογα με την εφαρμογή.



Εικόνα 11: Διαφορετικές εφαρμογές του μοντέλου BERT

Πηγή: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Για τον αλγόριθμο BERT έχει κατασκευαστεί ένας ειδικός αλγόριθμος για την διαδικασία της συμβολοποίησης, το BertWordPieceTokenizer. Πιο συγκεκριμένα αυτή η διαδικασία έχει σκοπό με είσοδο ένα τμήμα κειμένου να παράγει σαν έξοδο μία ακολουθία κατάλληλη για είσοδο σε ένα μοντέλο μετασχηματιστή.
Πιο συγκεκριμένα ο συμβολοποιητής χωρίζει την είσοδο που βρίσκεται σε μορφή κειμένου σε σύμβολα και να αναθέτει σε κάθε σύμβολο ένα αναγνωριστικό (identifier) και στην συνέχεια αυτά τα σύμβολα χρησιμοποιούνται σαν είσοδο στο νευρωνικό δίκτυο. Τα περισσότερα μοντέλα μετασχηματιστή βασίζονται στην διαδικασία συμβολοποίησης σε υπολέξεις το οποίο δίνει την δυνατότητα το μοντέλο να μπορεί να κωδικοποιήσει άγνωστες ή σπάνιες λέξεις στις οποίες δεν εκπαιδεύτηκε το μοντέλο.



Εικόνα 12: Διαδικασία Tokenization

Επισκόπηση Εφαρμογής

Η εφαρμογή που υλοποιήσαμε βασίζεται στον συνδυασμό των παραπάνω εννοιών για την παραγωγή ενός μοντέλου που ταξινομεί τα δείγματα κειμένου που λαμβάνει σαν είσοδο. Σε αυτή την εφαρμογή θα χρησιμοποιήσουμε ένα προ-εκπαιδευμένο μοντέλο BERT και θα το εκπαιδεύσουμε ξανά για το δικό μας σύνολο δεδομένων βασιζόμενοι στην τεχνική του Transfer Learning που περιγράψαμε παραπάνω.

Η εφαρμογή αυτή θα πραγματοποιεί ταξινόμηση ανάμεσα σε δύο κατηγορίες, δηλαδή θα είναι δυαδικός ταξινομητής (binary classifier). Οι κατηγορίες που επιλέξαμε είναι η πολιτική θέση του χρήστη του twitter η οποία μπορεί να είναι είτε στους δημοκρατικούς είτε στους ρεπουμπλικάνους. Η επιλογή του συνόλου δεδομένων έγινε καθώς υπάρχουν αρκετά δεδομένα που εκφράζουν αυτές τις δύο θέσεις και επιπλέον βασιζόμενοι στην παραδοχή πως το μοντέλο αναλύει νοηματικά το κείμενο βοηθά πολύ στην επιτυχία της ταξινόμησης ότι όσοι αναρτούν πολιτικές δημοσιεύσεις στο twitter έχουν αρκετά πολωμένο λόγο και επομένως φαίνεται έντονα η διαφορά ανάμεσα στις πολιτικές θέσεις.

Η διαδικασία βασίζεται σε τρία διαφορετικά βήματα. Την προ-επεξεργασία των δεδομένων (preprocessing), την εκπαίδευση του μοντέλου με Transfer Learning και την αξιολόγηση του εκπαιδευμένου μοντέλου στο σύνολο επικύρωσης (validation set).

Σκοπός της εφαρμογής είναι η υλοποίηση ενός συστήματος που μπορεί να πραγματοποιήσει την ταξινόμηση ανάμεσα στις δύο κατηγορίες με αρκετά υψηλή ακρίβεια. Αυτό θα επιτευχθεί εφαρμόζοντας κατάλληλες πρακτικές οι οποίες σε τεχνικό επίπεδο θα παρουσιαστούν στην επόμενη ενότητα.

Πειραματικά Αποτελέσματα

Σε αυτό το βήμα θα εφαρμόσουμε τις τεχνικές και τους αλγορίθμους που παρουσιάσαμε παραπάνω για να υλοποιήσουμε μία εφαρμογή που πραγματοποιεί την λειτουργία που περιγράψαμε προηγουμένως. Η εφαρμογή αυτή με την χρήση ενός προ-εκπαιδευμένου μοντέλου και ενός συνόλου δεδομένων θα παρουσιάσει ένα μοντέλο που έχει εκπαιδευτεί ξανά για την χρήση του σε συγκεκριμένη εφαρμογή (task specific model). Η εφαρμογή θα υλοποιηθεί σε περιβάλλον google colab το οποίο χρησιμοποιεί απομακρυσμένη υπολογιστική ισχύ σε κάποιον απομακρυσμένο διακομιστή της google. Η επιλογή αυτού του περιβάλλοντος έγινε καθώς η εκπαίδευση ενός τέτοιου μοντέλου απαιτεί υψηλή υπολογιστική ισχύ και μεγάλο χρονικό διάστημα και με την χρήση του google colab αποδεσμεύουμε πόρους του τοπικού μηχανήματος οι οποίοι θα ήταν απασχολημένοι με την εκπαίδευση του μοντέλου. Επιπλέον σε περίπτωση που θελήσουμε να εκτελέσουμε τοπικά τον κώδικα το colab μας δίνει την δυνατότητα να κάνουμε export σε μορφή είτε απλού αρχείου python είτε jupyter notebook το οποίο έχει την ίδια μορφή με το περιβάλλον colab.

Σαν πρώτο βήμα θα “φορτώσουμε” τα δεδομένα στο σύστημα και θα τα επεξεργαστούμε ώστε να έρθουν σε μορφή τέτοια ώστε να μπορούν να χρησιμοποιηθούν ως είσοδος για το μοντέλο. Για να μετασχηματιστούν τα δεδομένα σε αυτή την μορφή θα πρέπει να αφαιρεθούν οι ειδικοί χαρακτήρες (όπως @, #, & κ.λπ.), να αντικατασταθούν όλα τα κεφαλαία γράμματα από μικρά και να ανατεθεί το κάθε κείμενο σε μία από τις δύο κλάσεις.

```
[ ]  import pandas as pd

     raw_dataset=pd.read_csv('ExtractedTweets.csv')
     print('Dataset Shape: ',raw_dataset.shape)
     print('Dataset Columns: ',raw_dataset.columns)
```

```
Dataset Shape:  (86460, 3)
Dataset Columns:  Index(['Party', 'Handle', 'Tweet'], dtype='object')
```

As we can see from the above output we have 86460 samples to train the model.
From the Columns we need Party and Tweet.

Εικόνα 13: Εισαγωγή των δεδομένων στο σύστημα

```
[ ]  raw_dataset=raw_dataset[['Tweet','Party']]
     raw_dataset.head()
```

	Tweet	Party
0	Today, Senate Dems vote to #SaveTheInternet. P...	Democrat
1	RT @WinterHavenSun: Winter Haven resident / Al...	Democrat
2	RT @NBCLatino: .@RepDarrenSoto noted that Hurr...	Democrat
3	RT @NALCABPolicy: Meeting with @RepDarrenSoto ...	Democrat
4	RT @Vegalteno: Hurricane season starts on June...	Democrat

We will now clear the text of mentions, hashtags and special characters.

Εικόνα 14: Παρουσίαση ενός δείγματος των αρχικών δεδομένων

```
import re

raw_dataset['clean_tweet'] = raw_dataset.apply(lambda row: row['Tweet'].lower(),axis=1)
#removing mentions
raw_dataset['clean_tweet']= raw_dataset.apply(lambda row: re.sub('@[A-Za-z0-9]+',' ',row['clean_tweet']),axis=1 )
#removing hashtags
raw_dataset['clean_tweet']= raw_dataset.apply(lambda row: re.sub('#[A-Za-z0-9]+',' ',row['clean_tweet']),axis=1 )
#removing hyperlinks
raw_dataset['clean_tweet']= raw_dataset.apply(lambda row: re.sub(r'http\S+',' ',row['clean_tweet']),axis=1 )
raw_dataset['clean_tweet']= raw_dataset.apply(lambda row: re.sub(r'www.\S+',' ',row['clean_tweet']),axis=1 )
#removing non alphabetic characters
raw_dataset['clean_tweet'] = raw_dataset.apply(lambda row: re.sub('[^a-z0-9,.!?]',' ', row['clean_tweet']),axis=1)
#removing the retweet buzzword
raw_dataset['clean_tweet'] = raw_dataset.apply(lambda row: re.sub(r'rt ','', row['clean_tweet']),axis=1)
raw_dataset[['Tweet','clean_tweet']].head(10)
```

	Tweet	clean_tweet
0	Today, Senate Dems vote to #SaveTheInternet. P...	today, senate dems vote to . proud to supposi...
1	RT @WinterHavenSun: Winter Haven resident / Al...	winter haven resident alta vista teacher ...
2	RT @NBCLatino: .@RepDarrenSoto noted that Hurr...	. noted that hurricane maria has left appr...
3	RT @NALCABPolicy: Meeting with @RepDarrenSoto ...	meeting with . thanks for taking the time...
4	RT @Vegalteno: Hurricane season starts on June...	hurricane season starts on june 1st puerto...
5	RT @EmgageActionFL: Thank you to all who came ...	thank you to all who came out to our orland...
6	Hurricane Maria left approx $90 billion in dam...	hurricane maria left approx 90 billion in dam...
7	RT @Tharryry: I am delighted that @RepDarrenSo...	i am delighted that will be voting for th...
8	RT @HispanicCaucus: Trump's anti-immigrant pol...	trump s anti immigrant policies are hurting...
9	RT @RepStephMurphy: Great joining @WeAreUnidos...	great joining and for a roundtable in ...

As we can see on the output above we have cleared the Dataset and it is ready to be used on the model as the input.

Εικόνα 15: Τα δεδομένα στην αρχική τους μορφή και την επεξεργασμένη μορφή

Στην συνέχεια για να μπορέσουμε να εισάγουμε τα δεδομένα στο μοντέλο BERT θα πρέπει να εφαρμόσουμε την διαδικασία συμβολοποίησης που παρουσιάσαμε παραπάνω. Στην διαδικασία αυτή το κείμενο που παρουσιάζεται παραπάνω θα μετατραπεί σε σύμβολα, δηλαδή σε αριθμητικά δεδομένα συγκεκριμένης μορφής τα οποία αναγνωρίζονται ως είσοδος από το μοντέλο.

```python
import torch
from datasets import Dataset
from transformers import AutoTokenizer
tokenizer=AutoTokenizer.from_pretrained('bert-base-cased')

def tokenize_function(examples):
    return tokenizer(examples['tweet'],padding='max_length',truncation=True)

dataset=Dataset.from_pandas(dataset)

tokenized_datasets=dataset.map(tokenize_function,batched=True)
```

Εικόνα 16: Εφαρμογή Tokenizer στα δεδομένα

```python
tokenized_datasets = tokenized_datasets.remove_columns(["tweet"])
tokenized_datasets = tokenized_datasets.rename_column("party", "labels")
tokenized_datasets.set_format("torch")

small_train_dataset = tokenized_datasets.shuffle().select(range(100))
small_eval_dataset = tokenized_datasets.shuffle().select(range(10))
```

```
We will train on 5000 samples and evaluate on 1000 samples
```

Εικόνα 17: Δημιουργία συνόλου δεδομένων εκπαίδευσης και αξιολόγησης

Έχοντας πραγματοποιήσει την παραπάνω διαδικασία τα δεδομένα είναι έτοιμα να εισαχθούν στο μοντέλο. Λόγω του μεγάλου όγκου των δειγμάτων που είναι διαθέσιμα στο σύνολο δεδομένων θα εκπαιδεύσουμε το μοντέλο σε 5000 δείγματα και θα το αξιολογήσουμε σε 1000 δείγματα. Για την εισαγωγή των δεδομένων στο μοντέλο θα χρησιμοποιήσουμε ένα ενσωματωμένο πακέτο του PyTorch που ονομάζεται dataloader και είναι υπεύθυνο για την εισαγωγή των δεδομένων στο μοντέλο καθώς και τον διαχωρισμό τους σε δέσμες (batches) που συγκεκριμένα στο παράδειγμα μας έχουν πλήθος ίσο με οκτώ δείγματα.

```
from torch.utils.data import DataLoader

train_dataloader = DataLoader(small_train_dataset, shuffle=True, batch_size=8)
eval_dataloader = DataLoader(small_eval_dataset, batch_size=8)
```

Εικόνα 18: Εισαγωγή των δεδομένων στο Dataloader

Έχοντας ορίσει τους dataloaders, στην συνέχεια θα ορίσουμε το μοντέλο και θα το παραμετροποιήσουμε. Η βασική παραμετροποίηση που καλούμαστε να πραγματοποιήσουμε είναι η επιλογή του μοντέλου και το πλήθος των εξόδων που θα παρουσιάζονται. Στην περίπτωση μας, όπως αναφέραμε παραπάνω, το πλήθος των εξόδων είναι ίσο με δύο και το μοντέλο είναι το BERT για μικρά γράμματα (bert uncased).

```
from transformers import AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased", num_labels=2)
```

```
model.safetensors: 100% 436M/436M [00:07<00:00, 59.9MB/s]
```

Εικόνα 19: Επιλογή και χρήση του μοντέλου

Στην συνέχεια θα επιλέξουμε τον βελτιστοποιητή (optimizer) του μοντέλου που θα χρησιμοποιήσουμε. Ο βελτιστοποιητής που επιλέξαμε είναι το AdamW, το οποίο είναι ένας αλγόριθμος βασισμένος στην στοχαστική κατάβαση βαθμίδας (stochastic gradient descent), ο οποίος χρησιμοποιεί τον αλγόριθμο Adam[10] (adaptive learning rate algorithm).

Ο αλγόριθμος Adam βασίζεται σε τρεις αρχές λειτουργίας.
Η πρώτη αρχή λειτουργίας είναι η ορμή (momentum). Ο αλγόριθμος διατηρεί μία μέση τιμή η οποία μειώνεται εκθετικά (exponentially decaying) των προηγούμενων τιμών της κλίσης της κατάβασης που πραγματοποιούμε (gradient descent).
Ο αλγόριθμος επιπλέον βασίζεται στον μεταβλητό βαθμό εκπαίδευσης (adaptive learning rate) δηλαδή για κάθε παράμετρο μάθησης είναι σε θέση να μεταβάλλει τον ρυθμό μάθησης το οποίο αποσκοπεί σε μεγαλύτερη διόρθωση της κλίσης για χαρακτηριστικά που εμφανίζονται σπανιότερα και μικρότερη διόρθωση σε πιο συχνά χαρακτηριστικά.
Η τελευταία αρχή λειτουργίας είναι η διόρθωση της πόλωσης (bias correction). Αυτή η λειτουργία είναι απαραίτητη επειδή στα αρχικά βήματα ο μέσος όρος που υπολογίζεται τείνει προς το μηδέν και η διόρθωση της πόλωσης διορθώνει αυτό το πρόβλημα.

Ο αλγόριθμος αυτός παρουσιάζει το πλεονέκτημα ότι είναι ικανός να προσπεράσει τα τοπικά ελάχιστα, δηλαδή να ξεπεράσει σημεία που ένας πιο βασικός αλγόριθμος (πχ. gradient descent) θα εγκλωβιζόταν. Για αυτό τον λόγο επιλέχθηκε ο αλγόριθμος Adam.

```
from transformers import AdamW

optimizer = AdamW(model.parameters(), lr=5e-5)
```

Εικόνα 20: Αρχικοποίηση του βελτιστοποιητή του μοντέλου.

Στην συνέχεια θα αρχικοποιήσουμε τις υπόλοιπες παραμέτρους του μοντέλου. Αυτές οι παράμετροι αναφέρονται στο πλήθος των εποχών εκπαίδευσης του μοντέλου, το πλήθος των βημάτων εκπαίδευσης καθώς και τις τιμές ορόσημα, οι οποίες είναι οι τιμές στις οποίες διακόπτεται νωρίτερα η διαδικασία της εκπαίδευσης.

```
from ignite.contrib.handlers import PiecewiseLinear

num_epochs = 10
num_training_steps = num_epochs * len(train_dataloader)

milestones_values = [
        (0, 5e-5),
        (num_training_steps, 0.0),
    ]
lr_scheduler = PiecewiseLinear(
        optimizer, param_name="lr", milestones_values=milestones_values
    )
```

Εικόνα 21: Αρχικοποίηση λοιπών παραμέτρων του μοντέλου

Στην συνέχεια θα ορίσουμε την συνάρτηση η οποία εκπαιδεύει το μοντέλο που θέλουμε να αρχικοποιήσουμε.

```
def train_step(engine, batch):
    model.train()

    batch = {k: v.to(device) for k, v in batch.items()}
    outputs = model(**batch)
    loss = outputs.loss
    loss.backward()

    optimizer.step()
    optimizer.zero_grad()

    return loss
```

The above function will be used to train the model

Εικόνα 22: Συνάρτηση εκπαίδευσης του μοντέλου

Παρακάτω θα παρουσιαστεί το μοντέλο, όπως αυτό παρουσιάζεται από το ίδιο σύστημα του PyTorch.

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(28996, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=2, bias=True)
)
```

Εικόνα 23: Η πλήρης εσωτερική αρχιτεκτονική του μοντέλου

Επιπλέον θα ορίσουμε την συνάρτηση που θα χρησιμοποιήσουμε για την αξιολόγηση του μοντέλου, η οποία θα χρησιμοποιηθεί μετά την εκπαίδευση με σκοπό τον καθορισμό της ακρίβειας σε προβλέψεις εκτός του συνόλου δεδομένων εκπαίδευσης.

```
def evaluate_step(engine, batch):
    model.eval()

    batch = {k: v.to(device) for k, v in batch.items()}
    with torch.no_grad():
        outputs = model(**batch)

    logits = outputs.logits
    predictions = torch.argmax(logits, dim=-1)

    return {'y_pred': predictions, 'y': batch["labels"]}
```

Εικόνα 24: Συνάρτηση αξιολόγησης του μοντέλου

```
train_evaluator = Engine(evaluate_step)
validation_evaluator = Engine(evaluate_step)
```

Now we have active evaluators for both train and validation sets

Εικόνα 25: Δημιουργία μεθόδων αξιολόγησης για τα δύο σύνολα δεδομένων

```
from ignite.metrics import Accuracy

Accuracy().attach(train_evaluator, 'accuracy')
Accuracy().attach(validation_evaluator, 'accuracy')
```

Now we have attached accuracy as a metric of the model.

Εικόνα 26: Εφαρμογή των αξιολογητών

Στην συνέχεια θα ορίσουμε την μέθοδο η οποία θα μπορεί να διακόπτει την εκπαίδευση πριν την ολοκλήρωση των εποχών εκπαίδευσης εάν επιτευχθούν οι στόχοι του μοντέλου.

```
from ignite.handlers import EarlyStopping

def score_function(engine):
    val_accuracy = engine.state.metrics['accuracy']
    return val_accuracy

handler = EarlyStopping(patience=2, score_function=score_function, trainer=trainer)
validation_evaluator.add_event_handler(Events.COMPLETED, handler)
```

Εικόνα 27: Συνάρτηση διακοπής εκπαίδευσης

Επιπλέον θα ορίσουμε συναρτήσεις οι οποίες θα εμφανίζουν τα αποτελέσματα της διαδικασίας εκπαίδευσης με συγκεντρωμένο τρόπο ανά εποχή εκπαίδευσης.

```
@trainer.on(Events.EPOCH_COMPLETED)
def log_training_results(engine):
    train_evaluator.run(train_dataloader)
    metrics = train_evaluator.state.metrics
    avg_accuracy = metrics['accuracy']
    print(f"Training Results - Epoch: {engine.state.epoch}  Avg accuracy: {avg_accuracy:.3f}")

def log_validation_results(engine):
    validation_evaluator.run(eval_dataloader)
    metrics = validation_evaluator.state.metrics
    avg_accuracy = metrics['accuracy']
    print(f"Validation Results - Epoch: {engine.state.epoch}  Avg accuracy: {avg_accuracy:.3f}")

trainer.add_event_handler(Events.EPOCH_COMPLETED, log_validation_results)
```

Εικόνα 28: Συναρτήσεις παρουσίασης αποτελεσμάτων

```
from ignite.handlers import ModelCheckpoint

checkpointer = ModelCheckpoint(dirname='models', filename_prefix='bert-base-cased', n_saved=2, create_dir=True)
trainer.add_event_handler(Events.EPOCH_COMPLETED, checkpointer, {'model': model})
```

Εικόνα 29: Προσθήκη των παραπάνω παραμέτρων στο μοντέλο.

Έχοντας ετοιμάσει όλα τα στοιχεία που συνθέτουν το μοντέλο, θα προχωρήσουμε στην εκπαίδευση του.

```
trainer.run(train_dataloader, max_epochs=num_epochs)
```

Εικόνα 30: Εκπαίδευση του μοντέλου

Αξιολόγηση Μοντέλου και Συμπεράσματα

Το παραπάνω μοντέλο εκπαιδεύτηκε μία φορά σε ένα μεγάλο σύνολο δεδομένων, το οποίο παρουσιάζει υψηλά επίπεδα ακρίβειας. Στην συνέχεια θα παρουσιαστούν τα αποτελέσματα αυτού του μοντέλου καθώς και αποτελέσματα σε ένα μικρότερο σύνολο εκπαίδευσης το οποίο χρησιμοποιήθηκε για την διαδικασία της δεκαπλής διεπικύρωσης (ten-fold cross-validation). Σε αυτή την διαδικασία πραγματοποιούμε δέκα φορές εκπαίδευση και αξιολόγηση σε σύνολα δεδομένων τα οποία διαφέρουν κάθε φορά με σκοπό την αξιολόγηση του μοντέλου ανεξάρτητα από τα δεδομένα.
Μετά την ολοκλήρωση της διαδικασίας εκπαίδευσης που παρουσιάσαμε παραπάνω συνεχίζουμε στην αξιολόγηση του μοντέλου. Το μοντέλο που δημιουργήσαμε βασιζόμενοι σε προϋπάρχον μοντέλο BERT, χρειάστηκε τρεις από τις δέκα εποχές εκπαίδευσης για να επιτύχει ακρίβεια ίση με 93% στο σύνολο εκπαίδευσης και ίση με 60% στο σύνολο αξιολόγησης που αποτελείται από άγνωστα στο μοντέλο δεδομένα.

```
Training Results - Epoch: 1  Avg accuracy: 0.480
Validation Results - Epoch: 1  Avg accuracy: 0.700


Training Results - Epoch: 2  Avg accuracy: 0.910
Validation Results - Epoch: 2  Avg accuracy: 0.200


Training Results - Epoch: 3  Avg accuracy: 0.930
2024-07-10 08:53:14,175 ignite.handlers.early_stopping.EarlyStopping INFO: EarlyStopping: Stop training
Validation Results - Epoch: 3  Avg accuracy: 0.600
State:
        iteration: 39
        epoch: 3
        epoch_length: 13
        max_epochs: 10
        output: <class 'torch.Tensor'>
        batch: <class 'dict'>
        metrics: <class 'dict'>
        dataloader: <class 'torch.utils.data.dataloader.DataLoader'>
        seed: <class 'NoneType'>
        times: <class 'dict'>
```

Εικόνα 31: Αποτελέσματα διαδικασίας εκπαίδευσης

Όπως μπορούμε να δούμε στην εικόνα 33, το μοντέλο σε κάθε εποχή εκπαίδευσης βελτίωνε την ακρίβεια του, μέχρι που με την ολοκλήρωση της τρίτης εποχής εκπαίδευσης το μοντέλο έφτασε τα ορόσημα (milestones) που είχαν τεθεί από την αρχή και ολοκλήρωσε την διαδικασία εκπαίδευσης.

Στην συνέχεια προχωρήσαμε σε εκτέλεση της διαδικασίας με την εφαρμογή της τεχνικής της δεκαπλής διεπικύρωσης που περιγράψαμε παραπάνω. Η διαδικασία αυτή χρησιμοποιείται για την απόδειξη της αποδοτικότητας του μοντέλου και βασίζεται στην υπόθεση ότι η εκπαίδευση δεν θα πρέπει να παρουσιάζει πολύ μεγάλες αποκλίσεις ανεξάρτητα του συνόλου δεδομένων που έχουν χρησιμοποιηθεί. Στο παράδειγμα μας παρουσιάζουμε ένα πείραμα που εκτελεί την διαδικασία εκπαίδευσης δέκα φορές με δέκα διαφορετικά σύνολα δεδομένων. Λόγω του υπολογιστικού φόρτου που απαιτεί η εκπαίδευση του μοντέλου, επιλέξαμε να μειώσουμε στο 1/10 του αρχικού τα σύνολα εκπαίδευσης και επικύρωσης.

```
---Current Fold: 1 ---
Training Results - Epoch: 1  Avg accuracy: 0.600
Validation Results - Epoch: 1  Avg accuracy: 0.000
Training Results - Epoch: 2  Avg accuracy: 0.600
Validation Results - Epoch: 2  Avg accuracy: 0.000
Training Results - Epoch: 3  Avg accuracy: 0.650
2024-10-13 18:41:57,832 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.000
---End of Fold: 1 ---
---Current Fold: 2 ---
Training Results - Epoch: 1  Avg accuracy: 0.700
Validation Results - Epoch: 1  Avg accuracy: 0.000
Training Results - Epoch: 2  Avg accuracy: 0.850
Validation Results - Epoch: 2  Avg accuracy: 0.000
Training Results - Epoch: 3  Avg accuracy: 0.950
2024-10-13 18:45:21,604 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.000
---End of Fold: 2 ---
---Current Fold: 3 ---
Training Results - Epoch: 1  Avg accuracy: 0.800
Validation Results - Epoch: 1  Avg accuracy: 0.000
Training Results - Epoch: 2  Avg accuracy: 1.000
Validation Results - Epoch: 2  Avg accuracy: 0.000
Training Results - Epoch: 3  Avg accuracy: 1.000
2024-10-13 18:48:48,496 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.000
---End of Fold: 3 ---
---Current Fold: 4 ---
Training Results - Epoch: 1  Avg accuracy: 0.550
Validation Results - Epoch: 1  Avg accuracy: 0.000
Training Results - Epoch: 2  Avg accuracy: 0.850
Validation Results - Epoch: 2  Avg accuracy: 0.500
Training Results - Epoch: 3  Avg accuracy: 0.900
Validation Results - Epoch: 3  Avg accuracy: 0.500
Training Results - Epoch: 4  Avg accuracy: 1.000
2024-10-13 18:53:17,634 ignite.handlers.early_stopping
Validation Results - Epoch: 4  Avg accuracy: 0.500
---End of Fold: 4 ---
---Current Fold: 5 ---
Training Results - Epoch: 1  Avg accuracy: 0.750
Validation Results - Epoch: 1  Avg accuracy: 0.500
Training Results - Epoch: 2  Avg accuracy: 0.700
Validation Results - Epoch: 2  Avg accuracy: 0.500
Training Results - Epoch: 3  Avg accuracy: 0.750
2024-10-13 18:56:38,653 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.500
---End of Fold: 5 ---
```

Εικόνα 32: Διαδικασίες εκπαίδευσης 1 έως 5

```
---Current Fold: 6 ---
Training Results - Epoch: 1  Avg accuracy: 0.800
Validation Results - Epoch: 1  Avg accuracy: 0.500
Training Results - Epoch: 2  Avg accuracy: 0.950
Validation Results - Epoch: 2  Avg accuracy: 0.500
Training Results - Epoch: 3  Avg accuracy: 1.000
2024-10-13 19:00:02,328 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.500
---End of Fold: 6 ---
---Current Fold: 7 ---
Training Results - Epoch: 1  Avg accuracy: 0.900
Validation Results - Epoch: 1  Avg accuracy: 0.500
Training Results - Epoch: 2  Avg accuracy: 0.900
Validation Results - Epoch: 2  Avg accuracy: 0.500
Training Results - Epoch: 3  Avg accuracy: 0.950
2024-10-13 19:03:25,766 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.500
---End of Fold: 7 ---
---Current Fold: 8 ---
Training Results - Epoch: 1  Avg accuracy: 0.700
Validation Results - Epoch: 1  Avg accuracy: 1.000
Training Results - Epoch: 2  Avg accuracy: 0.850
Validation Results - Epoch: 2  Avg accuracy: 0.500
Training Results - Epoch: 3  Avg accuracy: 1.000
2024-10-13 19:06:49,890 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.500
---End of Fold: 8 ---
---Current Fold: 9 ---
Training Results - Epoch: 1  Avg accuracy: 0.750
Validation Results - Epoch: 1  Avg accuracy: 0.000
Training Results - Epoch: 2  Avg accuracy: 0.850
Validation Results - Epoch: 2  Avg accuracy: 0.000
Training Results - Epoch: 3  Avg accuracy: 0.850
2024-10-13 19:10:14,773 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.000
---End of Fold: 9 ---
---Current Fold: 10 ---
Training Results - Epoch: 1  Avg accuracy: 0.850
Validation Results - Epoch: 1  Avg accuracy: 0.500
Training Results - Epoch: 2  Avg accuracy: 1.000
Validation Results - Epoch: 2  Avg accuracy: 0.500
Training Results - Epoch: 3  Avg accuracy: 1.000
2024-10-13 19:13:39,309 ignite.handlers.early_stopping
Validation Results - Epoch: 3  Avg accuracy: 0.500
---End of Fold: 10 ---
```

Εικόνα 33: Διαδικασίες εκπαίδευσης 6 έως 10

Παρακάτω παρουσιάζονται τα αποτελέσματα των πειραμάτων τόσο για την εκπαίδευση στο μεγάλο σύνολο δεδομένων όσο και στην διαδικασία της δεκαπλής διεπικύρωσης.

	Σύνολο Εκπαίδευσης	**Σύνολο Επικύρωσης**
Μοναδική Εκτέλεση στο πλήρες σύνολο δεδομένων	**93%**	**60%**

	Σύνολο Εκπαίδευσης	**Σύνολο Επικύρωσης**
Εποχή Εκπαίδευσης 1	**65%**	**0**
Εποχή Εκπαίδευσης 2	**95%**	**0**
Εποχή Εκπαίδευσης 3	**100%**	**0**
Εποχή Εκπαίδευσης 4	**100%**	**50%**
Εποχή Εκπαίδευσης 5	**75%**	**50%**
Εποχή Εκπαίδευσης 6	**100%**	**50%**
Εποχή Εκπαίδευσης 7	**95%**	**50%**
Εποχή Εκπαίδευσης 8	**100%**	**50%**
Εποχή Εκπαίδευσης 9	**85%**	**0**
Εποχή Εκπαίδευσης 10	**100%**	**50%**

Όπως μπορούμε να παρατηρήσουμε τα δεδομένα της μοναδικής εκτέλεσης εμφανίζουν αρκετά υψηλή ακρίβεια το οποίο μεταφράζεται σε αρκετά υψηλά ποσοστά εύστοχης πρόβλεψης της κλάσης που αντιστοιχεί στο δείγμα που δίνεται προς εξέταση.

Στην διαδικασία της διεπικύρωσης λόγω του μικρού συνόλου δεδομένων υπάρχουν διαδικασίες εκπαίδευσης οι οποίες δεν παρουσιάζουν κάποιο ιδιαίτερα καλό αποτέλεσμα, ενώ κάποιοι άλλοι κύκλοι εκπαίδευσης παρουσιάζουν παρόμοια αποτελέσματα χαμηλότερης ακρίβειας από το μοντέλο που έχει εκπαιδευτεί στο μεγάλο σύνολο δεδομένων. Το συμπέρασμα που μπορούμε να εξάγουμε από τις δύο διαδικασίες είναι ότι το μοντέλο είναι αρκετά αποδοτικό στην διαδικασία για την οποία το χρησιμοποιούμε ανεξαρτήτως των δεδομένων που εισάγουμε, δηλαδή γενικεύει σε μεγάλο βαθμό την εφαρμογή που καλείται να πραγματοποιήσει.

Η αξιολόγηση των αποτελεσμάτων που παρουσιάσαμε αφορά τόσο την αξιολόγηση της διαδικασίας εκπαίδευσης όσο και της εφαρμογής του μοντέλου που δημιουργήσαμε.

Η διαδικασία εκπαίδευσης πραγματοποιήθηκε με τον συνδυασμό διαφόρων μεθόδων μηχανικής μάθησης, προσαρμοσμένων στο πρόβλημα που ορίσαμε στην αρχή. Πέραν της βασικής διαδικασίας εκπαίδευσης ενός μοντέλου, η οποία αποτελείται από το στάδιο της προ-παρασκευής των δεδομένων, της εκπαίδευσης του μοντέλου και της αξιολόγησης των αποτελεσμάτων εφαρμόσαμε και την διαδικασία της μάθησης με μετάθεση η οποία εισήγαγε ένα προ-εκπαιδευμένο μοντέλο στο σύστημα το οποίο εμείς εκπαιδεύσαμε ξανά με συγκεκριμένο σκοπό. Εν κατακλείδι όσον αφορά την διαδικασία εκπαίδευσης παρουσιάζουμε την εκπαίδευση ενός μοντέλου βαθιάς μάθησης και μεγάλης έκτασης παραμέτρων με παραπάνω στοιχεία και πρακτικές απ’ ότι ένα απλό μοντέλο μηχανικής μάθησης.

Το μοντέλο παρουσιάζει μεγάλες δυνατότητες εφαρμογής σε διάφορα προβλήματα ταξινόμησης. Παρόλο που χρησιμοποιήσαμε ένα μικρό σύνολο δεδομένων λόγω των περιορισμών σε υπολογιστική ισχύ το μοντέλο παρουσίασε αρκετά υψηλή ακρίβεια, επομένως μία ασφαλής υπόθεση θα ήταν πως με μεγαλύτερες δυνατότητες σε σύνολα δεδομένων και ισχυρότερα συστήματα θα μπορούσαμε να επιτύχουμε μεγαλύτερη ακρίβεια στα αποτελέσματα του μοντέλου. Επιπλέον μία εφικτή επέκταση στο μοντέλο που παρουσιάσαμε είναι η δημιουργία ενός ταξινομητή με περισσότερες από δύο εξόδους ο οποίος για παράδειγμα θα μπορούσε να εντοπίσει την πολιτική θέση ενός Ευρωπαίου ψηφοφόρου που στις περισσότερες περιπτώσεις υποστηρίζει κάποιο από πολλά περισσότερα από δύο κόμματα. Μία άλλη κατεύθυνση η οποία θα μπορούσε να δοθεί στο μοντέλο είναι η ταξινόμηση με βάση κάποιο ενδιαφέρον ή ασχολία. Γενικώς ένα τέτοιο μοντέλο παρουσιάζει πολλές δυνατότητες και θα μπορούσε να χρησιμοποιηθεί για την αυτοματοποιημένη ταξινόμηση κειμένου που έχει γράψει κάποιος άνθρωπος και αναφέρεται στον γενικότερο τομέα εφαρμογής για τον οποίο έχει εκπαιδευτεί το μοντέλο.

Πηγές

[1] Transformers for Text Classification with IMDb Reviews:
https://pytorch-ignite.ai/tutorials/beginner/02-transformers-text-classification/

[2] Democrat Vs. Republican Tweets:
https://www.kaggle.com/datasets/kapastor/democratvsrepublicantweets

[3] Attention is All You Need:
https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[4] Tasks HuggingFace:
https://huggingface.co/tasks

[5] Text Classification HuggingFace:
https://huggingface.co/tasks/text-classification

[6] Google Colab:
https://colab.research.google.com/

[7] What is ML?:
https://www.ibm.com/topics/machine-learnin

[8] What is Transfer Learning ?:
https://www.ibm.com/topics/transfer-learning

[9] Tokenizer:
https://huggingface.co/docs/transformers/main_classes/tokenizer

[10] Demystifying the Adam Optimizer in Machine Learning:
https://medium.com/@weidagang/demystifying-the-adam-optimizer-in-machine-learning-4401d162cb9e

[11] BERT Explained: State of the art language model for NLP:
https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270