Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Ψηφιακός Πολιτισμός, Έξυπνες Πόλεις, IoT και Προηγμένες Ψηφιακές Τεχνολογίες»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Νευρωνικά Δίκτυα Transformer για την Ταξινόμηση Συναισθήματος** **Transformer Neural Networks for Sentiment Classification**
Ονοματεπώνυμο Φοιτητή	**Αργυρώ Παναγιώτα Νικολούζου**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΨΠΟΛ2229**
Επιβλέπων Καθηγητής	**Διονύσιος Σωτηρόπουλος**

Ημερομηνία παράδοσης **Δεκέμβριος 2024**

Τριμελής εξεταστική επιτροπή:

(υπογραφή)	(υπογραφή)	(υπογραφή)
Δ. Σωτηρόπουλος Επικ. Καθηγητής	Δ. Βέργαδος Καθηγητής	Ν. Μυριδάκης Επικ. Καθηγητής

ΣΥΝΟΨΗ

Σε αυτήν την εργασία φαίνεται πώς μπορούν να χρησιμοποιηθούν τα νευρωνικά δίκτυα Transformer για την ταξινόμηση συναισθήματος. Πιο συγκεκριμένα χρησιμοποιείται το προεκπαιδευμένο μοντέλο BERT το οποίο είναι βασισμένο στην αρχιτεκτονική των μετασχηματιστών (Transformer) για γλωσσική μοντελοποίηση ώστε να πραγματοποιηθεί ανάλυση συναισθήματος στο σύνολο δεδομένων GoEmotions. Αρχικά γίνεται αναφορά στους λόγους για τους οποίους είναι σημαντική η ταξινόμηση συναισθήματος καθώς και σε έρευνες που έχουν γίνει πάνω σε αυτή. Στην συνέχεια φαίνονται οι αρχιτεκτονικές βασικών τύπων νευρωνικών δικτύων, του μοντέλου transformer και του μοντέλου BERT και γίνεται περιγραφή του συνόλου δεδομένων GoEmotions. Ύστερα, παρουσιάζονται τα αποτελέσματα της ταξινόμησης και πραγματοποιείται αξιολόγηση αυτής μέσω των κατάλληλων μέτρων. Τέλος, παρατίθενται τα συμπεράσματα που βγήκαν από την ταξινόμηση και δίνονται προτάσεις για μελλοντική έρευνα.

ABSTRACT

This thesis shows how Transformer neural networks can be used for sentiment classification. Specifically, the pre-trained BERT model which is based on the Transformer architecture for linguistic modelling is used to perform sentiment analysis on the GoEmotions dataset. Firstly, the reasons why sentiment classification is important and the research that has been done on it are discussed. Then, the architectures of basic types of neural networks, the transformer model and the BERT model are shown and a description of the GoEmotions dataset is given. Afterwards, the results of the classification are presented and its evaluation through the appropriate metrics is done. Finally, the conclusions drawn from the classification are listed and suggestions for future research are given.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

Η Ανάλυση Συναισθήματος είναι η μελέτη που ασχολείται με την επεξεργασία φυσικής γλώσσας και προσανατολίζεται στις απόψεις και τα συναισθήματα των ανθρώπων [2,3]. Τα συναισθήματα παίζουν σημαντικό ρόλο στις επιτυχημένες και αποτελεσματικές ανθρώπινες σχέσεις. Στην πραγματικότητα, σε πολλές περιπτώσεις, η ανθρώπινη «συναισθηματική νοημοσύνη» είναι ο κύριος παράγοντας για επιτυχημένη αλληλεπίδραση. Υπάρχουν επίσης σημαντικές ενδείξεις ότι η ορθολογική μάθηση στους ανθρώπους εξαρτάται από τα συναισθήματα. Ως εκ τούτου, η συναισθηματική υπολογιστική (affective computing) και η ανάλυση συναισθήματος είναι το κλειδί για την πρόοδο της τεχνητής νοημοσύνης και όλων των ερευνητικών πεδίων που απορρέουν από αυτήν. Επιπλέον, έχουν εφαρμογή σε πολλά διαφορετικά σενάρια και υπάρχει ένας ικανοποιητικός αριθμός εταιρειών, μεγάλων και μικρών, που περιλαμβάνουν την ανάλυση συναισθημάτων ως μέρος του σκοπού τους [8].

Αξίζει να σημειωθεί ότι το πεδίο της ταξινόμησης συναισθημάτων αποτελεί μια ενδιαφέρουσα ερευνητική κατεύθυνση λόγω της άμεσης σύνδεσής της με τον πραγματικό κόσμο. Η διερεύνηση της γνώμης των ανθρώπων είναι σημαντική για καλύτερη λήψη αποφάσεων. Όλες οι αποφάσεις έχουν αντίκτυπο στην καθημερινή ζωή. Στην προ-διαδικτυακή εποχή, ήταν σύνηθες για ένα άτομο να ρωτάει τους φίλους και τους συγγενείς του για την γνώμη τους πριν πάρει μια απόφαση. Οι οργανισμοί διεξήγαγαν δημοσκοπήσεις, έρευνες για να κατανοήσουν τα συναισθήματα και τη γνώμη του ευρύτερου κοινού για τα προϊόντα ή τις υπηρεσίες τους [3].

Οι οργανισμοί έχουν εξελιχθεί και τώρα εξετάζουν τους ιστότοπους αξιολογήσεων για να γνωρίζουν πώς έχει λάβει το κοινό το προϊόν τους αντί να διεξάγουν έρευνες. Αυτές οι πληροφορίες που είναι διαθέσιμες στον Ιστό είναι μια πολύτιμη πηγή τόσο για το μάρκετινγκ, όσο και για όποιους ενδιαφέρονται να ερευνήσουν απόψεις [3]. Η ανάλυση συναισθήματος έχει παρατηρηθεί ως μια νέα τάση στα μέσα κοινωνικής δικτύωσης καθώς έχει πολλές εφαρμογές και μπορεί να ενισχύσει την επιχειρηματική ευφυΐα, να διευρύνει τις δυνατότητες των πλάνων διαχείρισης πελατειακών σχέσεων και συστάσεων επιτρέποντας, για παράδειγμα, να ανακαλύψουν από τι έμειναν ιδιαίτερα ευχαριστημένοι οι καταναλωτές ή από που έχουν λάβει πολύ αρνητικά σχόλια. Αυτό μπορεί να οδηγήσει στην βελτίωση προϊόντων και υπηρεσιών. Επίσης, μπορεί να αξιοποιηθεί για φιλτράρισμα των «τρολ» σχολίων και για την ανίχνευση ανεπιθύμητων μηνυμάτων στις διαδικτυακές επικοινωνίες [8].

Πολλές είναι οι εταιρείες που επενδύουν ένα αυξανόμενο χρηματικό ποσό σε στρατηγικές μάρκετινγκ και επικεντρώνονται τόσο στη συλλογή όσο και στην πρόβλεψη των στάσεων του ευρύτερου κοινού προς τα προϊόντα τους [8]. Ακόμη, έχει βοηθήσει τους οργανισμούς να λάβουν κριτική σε πραγματικό χρόνο σχετικά με τις στρατηγικές μάρκετινγκ που ακολουθούν ή τις διαφημίσεις τους από τις αντιδράσεις του κοινού στα tweets και τις αναρτήσεις τους στα ιστολόγια. Για την κυκλοφορία ενός νέου προϊόντος μπορεί να τους δώσει άμεσα σχόλια σχετικά με την γνώμη του κοινού για αυτό. Μπορεί επίσης να μετρήσει κατά πόσο το logo της επωνυμίας τους αρέσει στους καταναλωτές ή όχι [2].

Πολλές είναι οι έρευνες που χρησιμοποιούν κριτικές πελατών. Μια από αυτές είναι εκείνη των Zhengjie Gao, Ao Feng, Xinyu Song και Xi Wu που πραγματοποιήθηκε το 2019 και χρησιμοποιήθηκαν τρία διαφορετικά σύνολα δεδομένων. Τα πρώτα δύο ήταν από το SemEval-2014 και αφορούσαν κριτικές εστιατορίων και φορητών υπολογιστών και το τρίτο ήταν ένα σύνολο από tweets. Σε αυτήν την έρευνα εστίασαν στην επεξεργασία της φυσικής γλώσσας με την βοήθεια των προεκπαιδευμένων μοντέλων BERT και στην ταξινόμηση συναισθημάτων με την βοήθεια αυτού, σημειώνοντας ότι σε μια πρόταση θα μπορούσαμε να έχουμε παραπάνω από ένα αντικρουόμενα συναισθήματα. [12].

Επίσης, στην έρευνα των Xin Li, Lidong Bing, Wenxuan Zhang και Wai Lam που έγινε και αυτή το 2019 στην οποία χρησιμοποίησαν ως σύνολο δεδομένων επεξεργασίας ένα από τα πιο δημοφιλή σύνολα δεδομένων το SemEval (Pontiki et al., 2014, 2015, 2016) όπου υπάρχουν κριτικές εστιατορίων και κριτικές που αφορούν τον τομέα των φορητών υπολογιστών. Δίνεται έμφαση στην υψηλή ακρίβεια του μοντέλου BERT [21].

Μια ακόμη έρευνα που αφορά τον τομέα του εμπορίου και των εταιριών είναι εκείνη των Alexander Rietzler, Sebastian Stabinger, Paul Opitz, Stefan Engl και δημοσιεύτηκε το 2019. Το σύνολο δεδομένων που χρησιμοποιήθηκε και σε αυτήν την έρευνα είναι το SemEval 2014 Task 4 και αφορά όπως ανέφερα και πριν τα εστιατόρια και τον τομέα φορητών υπολογιστών. Και τα δύο σύνολα δεδομένων έχουν μικρό αριθμό παραδειγμάτων. Αναφέρεται από τους ερευνητές ότι ταξινόμηση συναισθημάτων μπορεί να φανεί πολύ χρήσιμη στο ηλεκτρονικό εμπόριο τόσο για τις επιχειρήσεις όσο και για τους πελάτες. [28].

Επιπλέον, το άρθρο των Avinash Kumar, Pranjal Gupta, Raghunathan Balan, Lalita Bhanu Murthy Neti και Aruna Malapati δημοσιεύτηκε το 2021 και εστιάζει στην ημιεποπτευόμενη υβριδική προσέγγιση (Semi-Supervised Hybrid Approach) που βασίζεται στο BERT για την ταξινόμηση απόψεων και συναισθημάτων. Η ταξινόμηση αυτή σημειώνεται ότι μπορεί να βοηθήσει τους επιχειρηματίες να πάρουν τις κατάλληλες αποφάσεις ώστε να μεγαλώσουν την επιχείρηση τους. Το σύνολο δεδομένων που χρησιμοποίησαν αφορούσε αξιολογήσεις από την Yelp σχετικά με εστιατόρια και από την Amazon σχετικά με φορητούς υπολογιστές [19].

Ακόμη, οι Zixuan Ke, Hu Xu και Bing Liu ασχολήθηκαν επίσης με την ανάλυση συναισθημάτων στην έρευνα με τους το 2021 και το σύνολο δεδομένων το οποίο εξετάστηκε αποτελούταν από διαφορετικά datasets και περιείχε τις κριτικές 19 προϊόντων. Εστίασαν στην συνεχή εκμάθηση μιας αλληλουχίας εργασιών ώστε να επιτευχθεί η ταξινόμηση συναισθημάτων. Το ζητούμενο αυτής της εργασίας είναι να μεταφερθούν οι γνώσεις από τα προηγούμενα tasks στα επόμενα αλλά και να διατηρηθεί η απόδοση των προηγούμενων tasks χωρίς να ξεχαστούν [17].

Ακόμα μια έρευνα συναισθημάτων που αφορά τις κριτικές του κινεζικού ηλεκτρονικού εμπορίου έγινε το 2023 από τους Song Xie, Jingjing Cao, Zhou Wu, Kai Liu, Xiaohui Tao και Haoran Xie με τίτλο «Sentiment Analysis of Chinese E-commerce Reviews Based on BERT». Η τρέχουσα ανάλυση συναισθήματος ταξινομεί το συνολικό συναίσθημα των αξιολογήσεων ηλεκτρονικού εμπορίου χωρίς εκτεταμένη περιγραφή της οντότητας. Το σύνολο δεδομένων που χρησιμοποιήθηκε είναι αξιολογήσεις από το ηλεκτρονικό εμπόριο της εταιρίας “Taobao cosmetics” [35].

Ήταν πάντα σημαντικό για τους καταναλωτές να γνωρίζουν τη γνώμη των άλλων, όμως ειδικότερα στις μέρες μας, όλο και περισσότεροι είναι εκείνοι που επισκέπτονται διαδικτυακούς ιστότοπους κριτικών, ιστολόγια, φόρουμ, μέσα κοινωνικής δικτύωσης και ούτω καθεξής για να εκφράσουν τη γνώμη τους. Οι επισκέπτες αυτών των ιστοσελίδων μπορούν να γνωρίζουν τα πλεονεκτήματα και τα μειονεκτήματα ενός προϊόντος από τις εμπειρίες που μοιράζονται οι άνθρωποι στον διαδίκτυο, κάτι που μπορεί να είναι χρήσιμο για αυτούς ώστε να αποφασίσουν αν θα προβούν στην αγορά του ή όχι. Πολλοί είναι επίσης εκείνοι που ζητούν σε διαδικτυακά φόρουμς για συμβουλές [3].

Οι καταναλωτές λοιπόν, θα μπορούσαν να βοηθηθούν από έρευνες όπως αυτή των Manish Munikar, Sushil Shakya και Aakash Shrestha η οποία παρουσιάστηκε το 2019. Πραγματοποιήθηκε σε ένα σύνολο δεδομένων που αφορούσε κριτικές ταινιών και με την βοήθεια του μοντέλου BERT χώρισαν τα συναισθήματα σε πέντε κατηγορίες (πολύ αρνητικά, αρνητικά, ουδέτερα, θετικά και πολύ θετικά). Τόνισαν ότι η ταξινόμηση των συναισθημάτων σε κατηγορίες είναι μια σημαντική διαδικασία καθώς μας βοηθάει να κατανοήσουμε την αντίληψη των ανθρώπων πάνω σε ένα προϊόν, μια υπηρεσία ή ένα θέμα. Αξίζει να σημειωθεί ότι η ακρίβεια του μοντέλου έφτασε το 94.7% [25].

Στην ηλεκτρονική εφημερίδα «Journal of Big Data» δημοσιεύτηκε ένα άρθρο στο οποίο αναλύονταν δεδομένα τα οποία προήλθαν από τρία διαφορετικά σύνολα δεδομένων (το πρώτο αφορούσε tweets που σχετίζονταν με αεροπορικές εταιρίες, ενώ το δεύτερο και το τρίτο αφορούσαν tweets που σχετίζονται με την εταιρία “Apple”) το 2023. Σημειώνεται ότι με την ταχεία ανάπτυξη των μέσων κοινωνικής δικτύωσης αρκετοί άνθρωποι εκφράζουν την γνώμη τους μέσω αυτών και ότι η κατανόηση της κοινής γνώμης μπορεί να βοηθήσει τις επιχειρήσεις αλλά και τους καταναλωτές να λάβουν αποφάσεις. Σε αυτό συμβάλλει η ταξινόμηση συναισθημάτων. Στην συγκεκριμένη έρευνα τα συναισθήματα ταξινομήθηκαν σε τρείς κατηγορίες: θετικό αρνητικό και ουδέτερο. [31]

Υπάρχει ένας τεράστιος όγκος πληροφοριών διαθέσιμες στον Ιστό που μπορούν να βοηθήσουν άτομα και οργανισμούς στις διαδικασίες λήψης αποφάσεων, αλλά ταυτόχρονα παρουσιάζουν πολλές προκλήσεις καθώς οργανισμοί και άτομα προσπαθούν να αναλύσουν και

να κατανοήσουν τη συλλογική γνώμη των άλλων. Ως εκ τούτου, έχει προκύψει η ανάγκη ανάλυσης και κατανόησης αυτών των δεδομένων/κριτικών που υπάρχουν στο διαδίκτυο [3].

Ακόμη, με την ανάλυση συναισθήματος, μπορούμε να μάθουμε απόψεις που μπορεί να αφορούν ένα παγκόσμιο ζήτημα, να προσδιορίσουμε την καταλληλότητα βίντεο για παιδιά με βάση τα σχόλια, να προσδιορίσουμε κατά πόσο υπάρχει μεροληψία στις ειδήσεις κ.λπ. Η μαζική ζήτηση για πολιτική πληροφόρηση μπορεί να θεωρηθεί ένας ακόμα σημαντικός παράγοντας. Η εφαρμογή στο εμπόριο δεν είναι το μόνο κίνητρο πίσω από τα άτομα που αναζητούν ή εκφράζουν απόψεις στο διαδίκτυο.

Για παράδειγμα, στην ανάλυση συνομιλιών στο Twitter πριν από τις εκλογές του Ευρωπαϊκού Κοινοβουλίου του 2014, οι ερευνητές έχουν συλλέξει περισσότερα από 1,2 εκατομμύρια tweets σε τρεις γλώσσες (αγγλικά, γερμανικά και γαλλικά) κατά τη διάρκεια μιας περιόδου δύο εβδομάδων (1 Μαΐου έως 14 Μαΐου 2014). Επιπλέον, υπάρχουν διάλογοι χαμηλής έντασης στην αγγλική γλώσσα στο Twitter σχετικά με τους πέντε υποψηφίους που διαγωνίζονταν για την προεδρία της Ευρωπαϊκής Επιτροπής. Στην πολιτική, μπορούμε επίσης να αναλύσουμε τις τάσεις, να εντοπίσουμε ιδεολογικές προκαταλήψεις, να στοχεύσουμε σε ανάλογες διαφημίσεις/μηνύματα, να αξιολογήσουμε τις απόψεις του κοινού/ψηφοφόρων [12, 35].

Μία ακόμη έρευνα στην οποία μελετήθηκε ένα σύνολο δεδομένων το οποίο είναι ένας συνδυασμός από διαφορετικά tweets, είναι αυτή των Abayomi Bello, Sin Chun Ng και Man-Fai Leung και δημοσιεύτηκε το 2023. Είναι γνωστό ότι στο tweeter είναι πολλοί εκείνοι οι οποίοι αναγράφουν την γνώμη τους για αυτό το λόγο αποφάσισαν να χρησιμοποιήσουν σύνολο δεδομένων από εκεί και να πραγματοποιήσουν ταξινόμηση συναισθημάτων με το BERT. Επιπλέον, εξηγούν πώς μπορεί η ανάλυση αυτή να γίνει όσο το δυνατόν πιο ακριβής. Το ποσοστό ακριβείας που επιτεύχθηκε είναι το 93% [6].

Επιπλέον, η ανάλυση συναισθημάτων μπορεί να έχει εφαρμογή και σε οικονομικά-χρηματιστηριακά θέματα. Παράδειγμα αποτελεί η έρευνα των Mingzheng Li, Lei Chen, Jing Zhao, Qiang Li που δημοσιεύθηκε το 2020. Και σε αυτήν την εργασία χρησιμοποιείται το προεκπαιδευμένο μοντέλο BERT σε αξιολογήσεις Κινεζικών μετοχών. Σύμφωνα με την έρευνα αυτή, η συγκεκριμένη μέθοδος έχει μεγαλύτερη ακρίβεια και δίνει τα καλύτερα δυνατά αποτελέσματα σε σύγκριση με άλλες μεθόδους. Στο συγκεκριμένο σύνολο δεδομένων η ακρίβεια έφτασε το 88,09%. Τέτοιου είδους έρευνες μπορούν να βοηθήσουν σημαντικά στην ακρίβεια των προβλέψεων του χρηματιστηρίου [20].

Στην κοινωνιολογία, η διάδοση ιδεών μέσω ομάδων είναι κάτι πολύ σημαντικό, οι απόψεις που έχουν οι άνθρωποι και οι αντιδράσεις τους σε ιδέες σχετίζονται με την υιοθέτηση νέων ιδεών [12, 34].Στην ψυχολογία, η ανάλυση συναισθήματος παρέχει μια ενίσχυση των ψυχολογικών ερευνών με δεδομένα που αντλούνται από τη φυσική γλώσσα [12]. Για παράδειγμα, η παραδοσιακή έρευνα ψυχολογίας βασίζεται σε ερωτηματολόγια και ακαδημαϊκές συνεντεύξεις, όμως πολλοί ψυχολόγοι στρέφουν τώρα το βλέμμα τους στα διαδικτυακά μέσα και προσπαθούν να αναλύσουν τα δεδομένα των κοινωνικών δικτύων από τη σκοπιά της ψυχολογίας. Αναμφίβολα, αυτή η ενσωμάτωση της εμπνέει σθένος στον τομέα της ψυχολογίας. Για παράδειγμα μπορούν να ανιχνευθούν οι πάσχοντες από κατάθλιψη, μελετώντας τις δημοσιεύσεις τους στα μέσα κοινωνικής δικτύωσης [34].

Επίσης, από τη σκοπιά της δημόσιας ασφάλειας, κοινωνικοπολιτικά γεγονότα όπως η Αραβική Άνοιξη και οι ταραχές του Λονδίνου (2018) καταδεικνύουν έντονα τη σημασία της ανάλυσης συναισθημάτων για την κοινή ασφάλεια. Τόσο στην περίπτωση της πολιτικής, όσο και στην περίπτωση της κοινής ασφάλειας τα μέσα κοινωνικής δικτύωσης όπως το Twitter και το Facebook θεωρήθηκαν ότι συνέβαλαν σημαντικά στην διάδοση των γεγονότων. Η ανάλυση συναισθήματος μπορεί να βοηθήσει τις αρχές να ανακαλύψουν εκ των προτέρων αυτούς τους τύπους ευαίσθητων πληροφοριών. Αν γίνει αυτό, μια ενέργεια όπως το κλείσιμο των καναλιών επικοινωνίας του Διαδικτύου θα απέτρεπε τους υποστηρικτές της τρομοκρατίας από την πρόσβαση σε τέτοιες υπηρεσίες [38].

Μερικοί ακόμα παράγοντες οι οποίοι έχουν τεκμηριωθεί από έρευνες μέχρι σήμερα, που ωθούν την ανάπτυξη του συγκεκριμένου αντικειμένου στον ερευνητικό χώρο, περιλαμβάνουν, την ενίσχυση των μεθόδων μηχανικής μάθησης στην επεξεργασία φυσικής γλώσσας και την ανάκτηση πληροφοριών, την αύξηση του Παγκόσμιου Ιστού για την παροχή ηλεκτρονικών συνόλων δεδομένων τα οποία μπορούν να χρησιμοποιηθούν σε αλγορίθμους μηχανικής μάθησης και να συμβάλουν στην δημιουργία έξυπνων εφαρμογών [2].

Η έρευνα των Jianfei Yu και Jing Jiang έγινε το 2019 και χρησιμοποιήθηκε η μέθοδος BERT για την στοχευμένη ταξινόμηση συναισθημάτων τα οποία προήλθαν από πολυτροπικές πηγές. Οι περισσότερες έρευνες για τον προσδιορισμό συναισθημάτων εστιάζουν στο περιεχόμενο κειμένου αγνοώντας τις διαφορετικές πηγές δεδομένων όπως για παράδειγμα τις εικόνες και τα δεδομένα που είναι σε μορφή ήχου. Τα δεδομένα αυτά έχουν αυξηθεί κατά πολύ τα τελευταία χρόνια στο διαδίκτυο. Η συγκεκριμένη έρευνα εστίασε στην ταξινόμηση συναισθημάτων σε ένα σύνολο δεδομένων τα οποία προέρχονται από πολυτροπικές πηγές [37].

Επίσης, στην εργασία τους οι Lysimachos Maltoudoglou, Andreas Paisios και Harris Papadopoulos (2020), ασχολήθηκαν με την επεξεργασίας φυσικής γλώσσας για την ανάλυση συναισθημάτων σε κείμενο, εφαρμόζοντας την Επαγωγική Συμμορφική Πρόβλεψη (Inductive Conformal Prediction) σε ένα μοντέλο που βασίζεται σε μετασχηματιστές. Το σύνολο δεδομένων που μελετήθηκαν αφορούν 50000 κριτικές ταινιών [23].

Το 2021, οι Gati L. Martin, Medard E. Mswahili, Young-Seob Jeong τόνισαν ότι η γλώσσα Σουαχίλι είναι μια γλώσσα που έχει μείνει πίσω όσον αφορά έρευνες σε σύγκριση με άλλες γλώσσες. Για αυτό το λόγο αποφάσισαν να εφαρμόσουν το μοντέλο BERT σε ένα σύνολο δεδομένων σε Σουαχίλι. Το σύνολο δεδομένων αυτό δημιουργήθηκε από 8200 κριτικές και σχόλια σε διαφορετικές πλατφόρμες κοινωνικών μέσων και στο ISEAR dataset. Τα δεδομένα ταξινομήθηκαν είτε ως θετικά είτε ως αρνητικά. Το μοντέλο πέτυχε την καλύτερη ακρίβεια 87,59% [24].

Παρόμοια λογική έχει και η έρευνα των Mohammed M.Abdelgwad, Taysir Hassan A Soliman και Ahmed I.Taloba του 2023 στην οποία μελετάται η αραβική πτυχή της ταξινόμησης διαφορετικών συναισθημάτων με χρήση του BERT. Μια και οι περισσότερες μελέτες συναισθημάτων είναι στα αγγλικά σε σύγκριση με τον πολύ μικρό αριθμό μελετών που είναι στα Αραβικά, οι ερευνητές αποφάσισαν να πραγματοποιήσουν μια μελέτη χρησιμοποιώντας τρία διαφορετικά αραβικά σύνολα δεδομένων. Το πρώτο αναφερόταν σε κριτικές ξενοδοχείων, το δεύτερο σε κριτικές βιβλίων και το τελευταίο στις αραβικές ειδήσεις. Η ακρίβεια που πέτυχε το μοντέλο στις κριτικές ξενοδοχείων έφτασε το 89,51%, στις κριτικές βιβλίων 73,23% και στις αραβικές ειδήσεις το 85,73% [1].

.

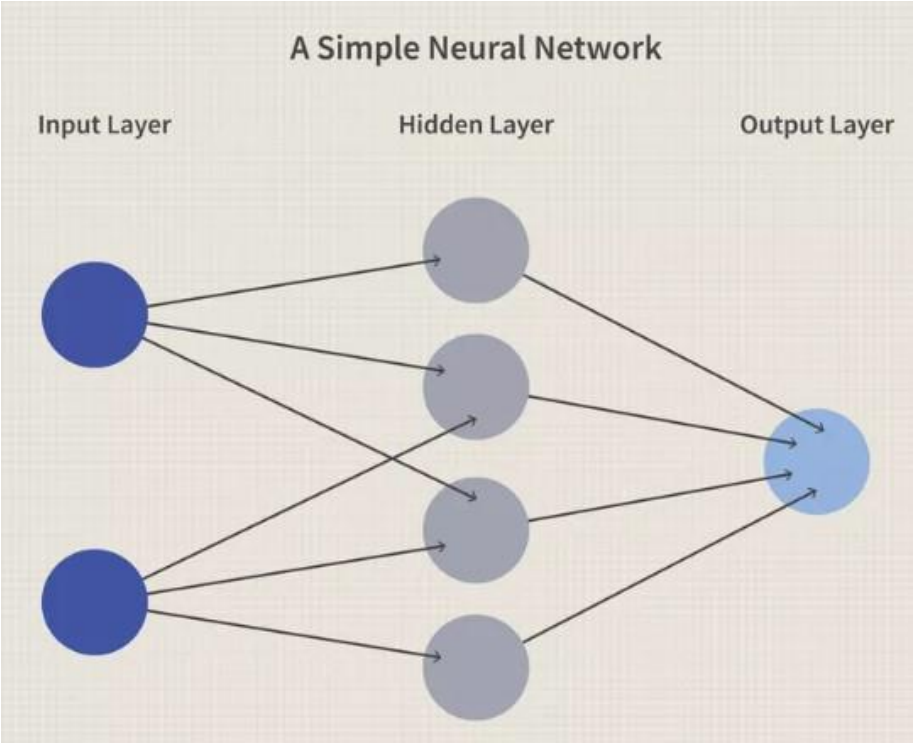
ΚΕΦΑΛΑΙΟ 2

ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

Τα νευρωνικά δίκτυα περιέχουν πολλά πυκνά διασυνδεδεμένα στοιχεία που ονομάζονται νευρώνες ή κόμβοι, τα οποία αποτελούν υπολογιστικά στοιχεία μη γραμμικής φύσης [38]. Τα νευρωνικά δίκτυα αντιπροσωπεύουν ένα εναλλακτικό υπολογιστικό παράδειγμα κατά το οποίο η λύση σε ένα πρόβλημα μαθαίνεται από ένα σύνολο παραδειγμάτων. Η έμπνευση για τα νευρωνικά δίκτυα προέρχεται αρχικά από μελέτες των μηχανισμών επεξεργασίας πληροφοριών στα βιολογικά νευρικά συστήματα, ιδιαίτερα στον ανθρώπινο εγκέφαλο. Τα βιολογικά νευρωνικά δίκτυα αποτελούν την κινητήρια δύναμη πίσω από μια μεγάλη έρευνα σε μοντέλα τεχνητών δικτύων, η οποία συμβάλλει στην επιθυμία να δημιουργηθούν καλύτερα συστήματα αναγνώρισης προτύπων και επεξεργασίας πληροφοριών. [7]

Ένα νευρωνικό δίκτυο μπορεί να θεωρηθεί ως μια μαθηματική συνάρτηση που μετατρέπει ένα σύνολο μεταβλητών εισόδου σε ένα σύνολο μεταβλητών εξόδου. Η ακριβής μορφή του μετασχηματισμού διέπεται από ένα σύνολο παραμέτρων που ονομάζονται βάρη (weights) των οποίων οι τιμές μπορούν να προσδιοριστούν με βάση ένα σύνολο παραδειγμάτων που αντιστοιχούν στην περίπτωση του προβλήματος που καλούμαστε να λύσουμε. Η διαδικασία προσδιορισμού των τιμών των παραμέτρων ονομάζεται συχνά μάθηση (learning) ή εκπαίδευση (training). Μόλις καθοριστούν τα βάρη, τα νέα δεδομένα μπορούν να υποβληθούν σε επεξεργασία από το νευρωνικό δίκτυο γρήγορα. Παραδείγματος χάριν, ένα πολυώνυμο μπορεί να θεωρηθεί ως μια απεικόνιση από μια ενιαία μεταβλητή εισόδου σε μια ενιαία μεταβλητή εξόδου. Οι συντελεστές στο πολυώνυμο είναι ανάλογοι με τα βάρη σε ένα νευρωνικό δίκτυο και ο προσδιορισμός αυτών των συντελεστών αντιστοιχεί στη διαδικασία εκπαίδευσης δικτύου.

Εκτός από την υψηλή ταχύτητα επεξεργασίας, τα νευρωνικά δίκτυα έχουν τη σημαντική ικανότητα να μαθαίνουν μια γενική λύση για ένα πρόβλημα από ένα σύνολο συγκεκριμένων παραδειγμάτων. Για πολλές εφαρμογές αυτό παρακάμπτει την ανάγκη ανάπτυξης ενός μοντέλου πρώτων αρχών (first-principles model) των βασικών διεργασιών, το οποίο συχνά μπορεί να αποδειχθεί δύσκολο ή αδύνατο να βρεθεί. Στο παρακάτω σχήμα φαίνεται ένα απλό νευρωνικό δίκτυο.



Σχήμα 1: Απλό νευρωνικό δίκτυο (Πηγή:[9])

Τα νευρωνικά δίκτυα χρησιμοποιούνται ευρέως από πολλούς ερευνητές σε πολλές διαφορετικές εφαρμογές όπως η ρομποτική, η αναγνώριση ομιλίας, η αναγνώριση ανθρώπινου προσώπου, οι ιατρικές εφαρμογές, οι κατασκευές καθώς και τα οικονομικά. Τα νευρωνικά δίκτυα παρουσιάζουν πολλά πλεονεκτήματα που τα κάνουν να χρησιμοποιούνται ευρέως σε πολλές διαφορετικές εφαρμογές. Αρχικά, τα νευρωνικά δίκτυα είναι ένα ισχυρό εργαλείο για την αναγνώριση μη γραμμικών συστημάτων (non-linear systems identification), επιπλέον, μπορούν να προσαρμοστούν αλλάζοντας τις παραμέτρους τους κι έτσι μπορούν εύκολα να χειριστούν ανακριβείς, ασαφείς και πιθανολογικές πληροφορίες. Αξίζει να σημειωθεί ότι τα νευρωνικά δίκτυα αντλούν την υπολογιστική τους ισχύ μέσω της δομής που έχουν και της ικανότητάς τους να μαθαίνουν και επομένως να γενικεύουν.

Αποδείχθηκε ότι τα νευρωνικά δίκτυα μπορούν να προσεγγίσουν οποιαδήποτε σύνθετη (μεγάλης κλίμακας) γραμμική ή μη γραμμική συνάρτηση με τα κατάλληλα δεδομένα εκπαίδευσης. Προσφέρουν επίσης χρήσιμες ιδιότητες και δυνατότητες όπως μη γραμμικότητα, οπτικοποίηση εισροών-εξόδων και προσαρμοστικότητα, προσέγγιση συναρτήσεων και γενίκευση. Τα νευρωνικά δίκτυα έχουν ένα είδος καθολικότητας. Επίσης, παίζουν πολύ σημαντικό ρόλο στην ανίχνευση σφαλμάτων, δεδομένου ότι μπορούν να παρέχουν και ένα μετασφαλματικό μοντέλο. Αυτό το μετασφαλματικό μοντέλο μπορεί να χρησιμοποιηθεί αποτελεσματικά για την απομόνωση και τον εντοπισμό των σφαλμάτων και, εάν είναι δυνατόν, για την αντιμετώπιση της αστοχίας.

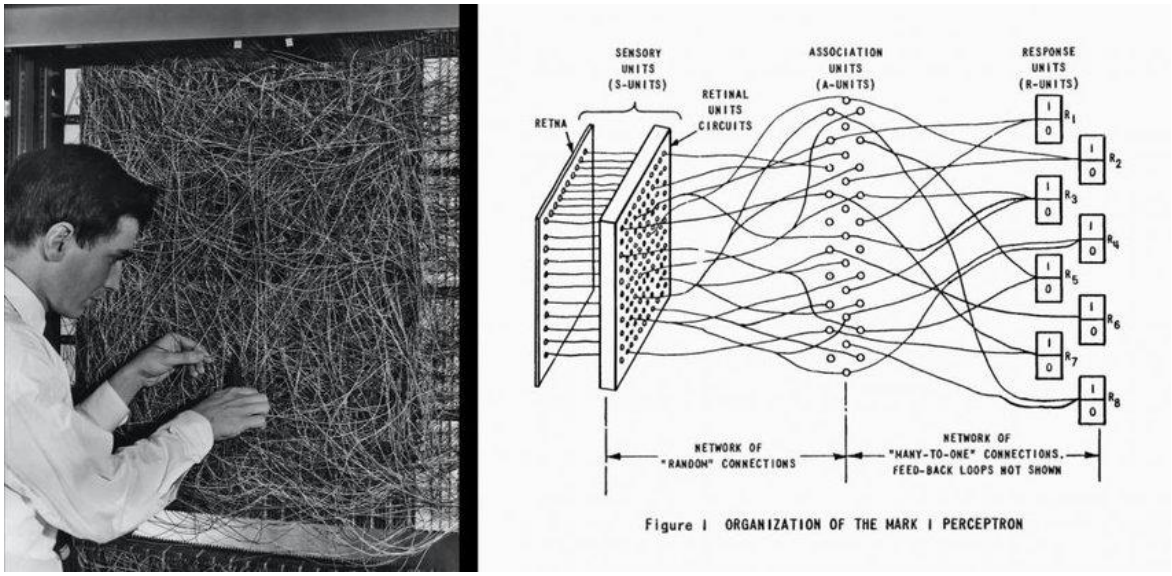
Τα κύρια μειονεκτήματα των νευρωνικών δικτύων πηγάζουν από την ανάγκη παροχής κατάλληλου συνόλου παραδειγμάτων δεδομένων για την εκπαίδευσή τους καθώς και τα πιθανά προβλήματα που μπορεί να προκύψουν εάν ένα νευρωνικό δίκτυο απαιτείται να επεκταθεί σε νέα δεδομένα εισόδου που διαφέρουν σημαντικά από εκείνα που αντιστοιχούν στα δεδομένα εκπαίδευσης. Σε πολλές πρακτικές εφαρμογές αυτά τα προβλήματα δεν τα συναντάμε, ενώ σε άλλες περιπτώσεις, υπάρχουν διάφορες τεχνικές που μπορούν να χρησιμοποιηθούν για να μετριάσουν τα προβλήματα.[30]

Τα πλεονεκτήματα και οι περιορισμοί των νευρωνικών δικτύων είναι συχνά συμπληρωματικά με εκείνα των συμβατικών τεχνικών επεξεργασίας δεδομένων. Σε γενικές γραμμές, τα νευρωνικά δίκτυα πρέπει να θεωρούνται ως πιθανοί υποψήφιοι για την επίλυση των προβλημάτων που έχουν μερικά ή όλα τα ακόλουθα χαρακτηριστικά: (i) υπάρχουν άφθονα δεδομένα για την εκπαίδευση του δικτύου. (ii) είναι δύσκολο να βρεθεί μια απλή λύση (iii) τα νέα δεδομένα πρέπει να υποβάλλονται σε επεξεργασία υψηλής ταχύτητας, είτε επειδή πρέπει να αναλυθεί μεγάλος όγκος δεδομένων, είτε λόγω κάποιου περιορισμού σε πραγματικό χρόνο, (iv) η μέθοδος επεξεργασίας των δεδομένων πρέπει να είναι ισχυρή. [7]

ΙΣΤΟΡΙΚΑ ΣΤΟΙΧΕΙΑ ΣΧΕΤΙΚΑ ΜΕ ΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ

Η προέλευση των νευρωνικών δικτύων ή του υπολογισμού με νευρωνικά δίκτυα (neural computing), βρίσκονται στη δεκαετία του 1940 με μια εργασία των McCulloch και Pitts. Έδειξαν ότι τα δίκτυα των μοντέλων νευρώνων είναι ικανά για καθολικούς υπολογισμούς, με άλλα λόγια μπορούν να μιμηθούν οποιαδήποτε υπολογιστική μηχανή γενικής χρήσης. Το επόμενο σημαντικό βήμα ήταν η δημοσίευση του βιβλίου “The Organization of Behavior” του Hebb το 1949, στο οποίο ο ίδιος πρότεινε έναν συγκεκριμένο μηχανισμό μάθησης στα βιολογικά νευρωνικά δίκτυα. Πρότεινε ότι η μάθηση γίνεται μέσα από τροποποιήσεις των δυνάμεων των συνδέσεων μεταξύ νευρώνων, έτσι ώστε εάν δύο νευρώνες τείνουν να ενεργοποιούνται μαζί τότε θα πρέπει να ενισχυθεί η μεταξύ τους σύναψη. Αυτός ο κανόνας μάθησης μπορεί να γίνει ποσοτικός και να διαμορφωθεί η βάση για τη μάθηση σε μερικά απλά μοντέλα νευρωνικών δικτύων.

Στα τέλη της δεκαετίας του 1950 δημιουργήθηκε το πρώτο νευρωνικό δίκτυο σε μορφή υλικού (hardware neural network). Το σύστημα αναπτύχθηκε από τον Rosenblatt. Το μοντέλο αυτό είναι γνωστό ως το «perceptron» και βασίστηκε στα μοντέλα νευρώνων των McCulloch-Pitts που προανέφερα. Είχε μια σειρά από φωτοϋποδοχείς που λειτουργούσαν ως εξωτερικές είσοδοι και χρησιμοποιήθηκαν συστοιχίες ποτενσιόμετρων για την παροχή προσαρμοστικών συναπτικών συνδέσεων οι οποίες θα μπορούσαν να διατηρήσουν μια ρύθμιση που έχει μάθει το δίκτυο.[14]



Εικόνα 1: Ο Frank Rosenblatt με το Mark I perceptron (αριστερά), και μια γραφική αναπαράσταση αυτού (δεξιά). (Πηγή [18])

Οι προσαρμογές στα ποτενσιόμετρα έγιναν χρησιμοποιώντας τον αλγόριθμο εκμάθησης «perceptron». Σε πολλές περιπτώσεις το perceptron μπορούσε να μάθει το νευρωνικό δίκτυο να διακρίνει μεταξύ χαρακτήρων ή σχημάτων που παρουσιάζονταν ως δεδομένα εισόδου σε μορφή εικόνων με pixel. Παρόμοια δίκτυα μελετήθηκαν επίσης από τον Widrow, ο οποίος ανέπτυξε το δίκτυο ADALINE (ADAptive Linear Element)» και η αντίστοιχη εκπαιδευτική διαδικασία που ονομάζεται κανόνας εκμάθησης Widrow-Hoff. Στη δεκαετία του 1960 παρατηρήθηκε μεγάλη ερευνητική δραστηριότητα στα νευρωνικά δίκτυα και πολλά από αυτά χαρακτηρίζονται από έλλειψη επιμέλειας. Κάτι που μερικές φορές καταλήγει σε υπερβολικούς ισχυρισμούς για τις δυνατότητες της τεχνολογίας. Παρά τις αρχικές επιτυχίες, η δυναμική στον τομέα άρχισε να μειώνεται προς τα τέλη της δεκαετίας του 1960 καθώς εμφανίστηκαν μια σειρά από δύσκολα προβλήματα που δεν μπορούσαν να επιλυθούν από τους τότε διαθέσιμους αλγόριθμους.

Επιπλέον, ο υπολογισμός με νευρωνικά δίκτυα δέχτηκε σφοδρή κριτική από τους υποστηρικτές του τομέα της Τεχνητής Νοημοσύνης (που προσπαθούσαν να διαμορφώσουν λύσεις στον τομέα της αναγνώρισης προτύπων και παρόμοια προβλήματα), με επίκεντρο το βιβλίο Perceptrons 1B του Minsky και Papert. Η κριτική τους επικεντρώθηκε σε μια κατηγορία προβλημάτων που ονομάζονται γραμμικά μη διαχωρίσιμα, τα οποία δεν μπορούσαν να επιλυθούν από δίκτυα όπως το perceptron και το ADALINE. Παρατηρήθηκε μια πτώση στον τομέα του υπολογισμού με νευρωνικά δίκτυα κατά τη διάρκεια της δεκαετίας του 1970, με αποτέλεσμα ελάχιστοι ερευνητές να παραμένουν ενεργοί. Μια δραματική αναζωπύρωση του ενδιαφέροντος για τα νευρωνικά δίκτυα ξεκίνησε στις αρχές της δεκαετίας του 1980 και οδηγήθηκε σε μεγάλο βαθμό από το έργο του φυσικού Hopfield, ο οποίος απέδειξε μια στενή σχέση μεταξύ μιας κατηγορίας μοντέλων νευρωνικών δικτύων και ορισμένων φυσικών συστημάτων γνωστών ως spin glasses.

Μια ακόμη σημαντική εξέλιξη ήταν η ανακάλυψη αλγορίθμων εκμάθησης, βασισμένων στην οπισθοδιάδοση σφαλμάτων (error backpropagation) που ξεπέρασε τους κύριους περιορισμούς προηγούμενων νευρωνικών δικτύων όπως το απλό perceptron. Κατά τη διάρκεια αυτής της περιόδου, πολλοί ερευνητές ανέπτυξαν ενδιαφέρον για τον υπολογισμό με νευρωνικά δίκτυα μέσω του βιβλίου Parallel Distributed Processing του Rumelhart. Ένας επιπλέον σημαντικός παράγοντας ήταν η ευρεία διαθεσιμότητα από τη δεκαετία του 1980 φθηνών ισχυρών υπολογιστών που δεν ήταν διαθέσιμοι 20 χρόνια νωρίτερα. Ο συνδυασμός αυτών των παραγόντων με την αποτυχία της Τεχνητής Νοημοσύνης να ανταποκριθεί σε πολλές από τις προσδοκίες των ερευνητών, οδήγησε σε έκρηξη του ενδιαφέροντος για την νευρωνική υπολογιστική [14] .

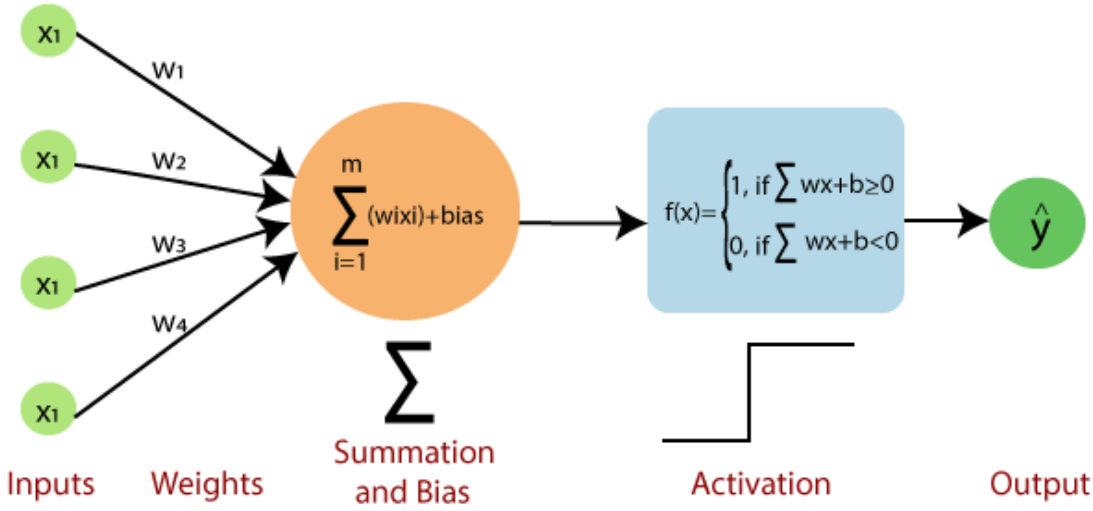
Επιπλέον, η πρόσφατη ανάκαμψη των νευρωνικών δικτύων προέρχεται και από τη βιομηχανία ηλεκτρονικών παιχνιδιών. Οι πολύπλοκες εικόνες και ο γρήγορος ρυθμός των

σημερινών βιντεοπαιχνιδιών απαιτούν υλικό που θα πρέπει να μπορεί να συμβαδίσει, και το αποτέλεσμα ήταν η μονάδα επεξεργασίας γραφικών (GPU), η οποία συγκεντρώνει χιλιάδες σχετικά απλούς πυρήνες επεξεργασίας σε ένα μόνο τσιπ. Οι ερευνητές δεν άργησαν να συνειδητοποιήσουν ότι η αρχιτεκτονική μιας GPU μοιάζει εντυπωσιακά με εκείνη ενός νευρωνικού δικτύου. Οι σύγχρονες GPU επέτρεψαν στα δίκτυα ενός επιπέδου της δεκαετίας του 1960 και στα δίκτυα δύο ή τριών επιπέδων της δεκαετίας του 1980 να εξελιχθούν στα σημερινά δίκτυα 10, 15, ακόμη και 50 επιπέδων (το "deep" στο "deep learning" αναφέρεται στο βάθος των επιπέδων του δικτύου). Επιπλέον, η βαθιά μάθηση είναι υπεύθυνη για τα συστήματα με τις καλύτερες επιδόσεις σχεδόν σε κάθε τομέα της έρευνας τεχνητής νοημοσύνης [14].

Οι αρχές της δεκαετίας του 1990 χαρακτηρίστηκαν από την εδραίωση των θεωρητικών θεμελίων του αντικειμένου, καθώς και από την εμφάνιση ευρέως διαδεδομένων επιτυχημένων εφαρμογών. Τα νευρωνικά δίκτυα μπορούν να βρεθούν τώρα ακόμη και σε ηλεκτρονικά είδη ευρείας κατανάλωσης και οικιακές συσκευές και έχουν εφαρμογές που ποικίλλουν [14].

ΤΥΠΟΙ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

- PERCEPTRON



Σχήμα 2: Σχήμα του Perceptron, (Πηγή [29])

Το perceptron είναι μια από τις απλούστερες αρχιτεκτονικές τεχνητών νευρωνικών δικτύων, που εφευρέθηκε το 1957 από τον Frank Rosenblatt. Όπως φαίνεται και στο σχήμα, το Perceptron αποτελείται από τέσσερα μέρη Βασίζεται σε έναν ελαφρώς διαφορετικό τεχνητό νευρώνα (όπως φαίνεται στο σχήμα 2) που ονομάζεται λογική μονάδα κατωφλίου (threshold logic unit) που εν συντομία λέγεται TLU, ή άλλες φορές γραμμική μονάδα κατωφλίου (linear threshold unit - LTU). Το επίπεδο εισόδου (input layer) μεταφέρει τα αρχικά δεδομένα στο σύστημα για περεταίρω επεξεργασία. Οι είσοδοι και έξοδοι είναι αριθμοί (αντί για δυαδικές τιμές) και κάθε σύνδεση εισόδου σχετίζεται με ένα βάρος (weight). Τα βάρη (weights) αντιπροσωπεύουν το πόσο ισχυρές είναι οι συνδέσεις μεταξύ των μονάδων. Η TLU υπολογίζει πρώτα μια γραμμική συνάρτηση των εισόδων της (δηλαδή το συνολικό άθροισμα των βαρών πολλαπλασιασμένα με τα δεδομένα εισόδου) [4]:

$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b = w^T x + b.$

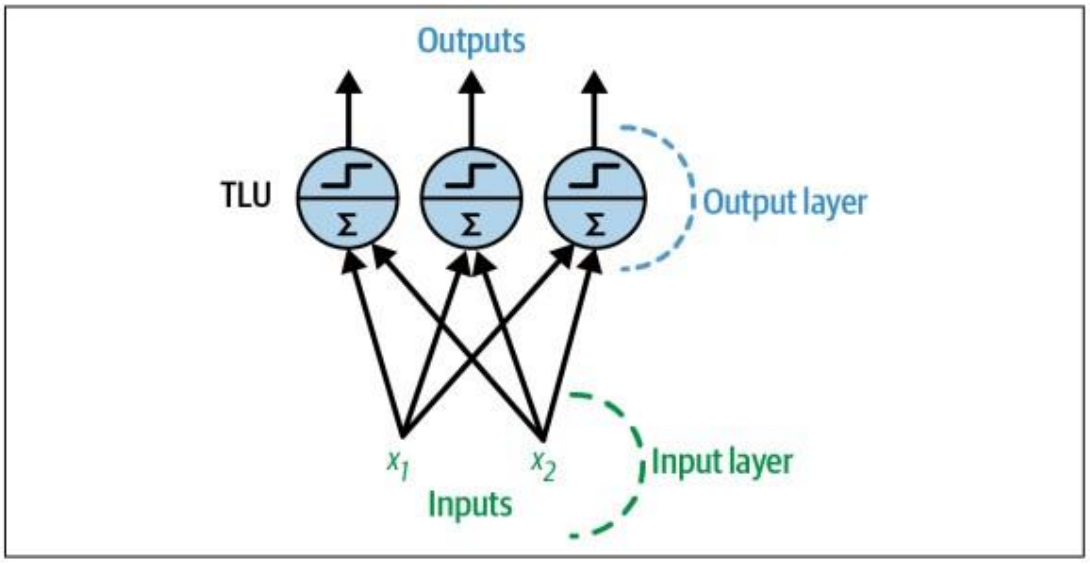
Το b είναι ο όρος πόλωσης (bias). Στη συνέχεια, εφαρμόζει μια συνάρτηση βήματος στο αποτέλεσμα:

$h_w(x) = \text{step}(z).$

Η πιο κοινή συνάρτηση βήματος που χρησιμοποιείται στα perceptron είναι η μοναδιαία βηματική συνάρτηση (Heaviside function)

heaviside (z) = $\begin{cases} 0 \ \alpha\nu \ z < 0 \\ 1 \ \alpha\nu \ z \geq 0 \end{cases}$

Μια μόνο TLU μπορεί να χρησιμοποιηθεί για μια απλή γραμμική δυαδική ταξινόμηση. Υπολογίζει μια γραμμική συνάρτηση των εισόδων, και αν το αποτέλεσμα υπερβαίνει ένα όριο, βγάζει την θετική κλάση. Διαφορετικά, βγάζει την αρνητική κλάση. Ένα perceptron μπορεί να αποτελείται από μια ή περισσότερες TLU οργανωμένες σε ένα μόνο επίπεδο, όπου η κάθε TLU συνδέεται με κάθε είσοδο. Ένα τέτοιο επίπεδο ονομάζεται πλήρως συνδεδεμένο επίπεδο (fully connected layer), ή πυκνό επίπεδο (dense layer). Οι είσοδοι αποτελούν το επίπεδο εισόδου. Για παράδειγμα, ένα perceptron με δύο διαφορετικές εισόδους και τρεις εξόδους παρουσιάζεται στο σχήμα 3.[4]



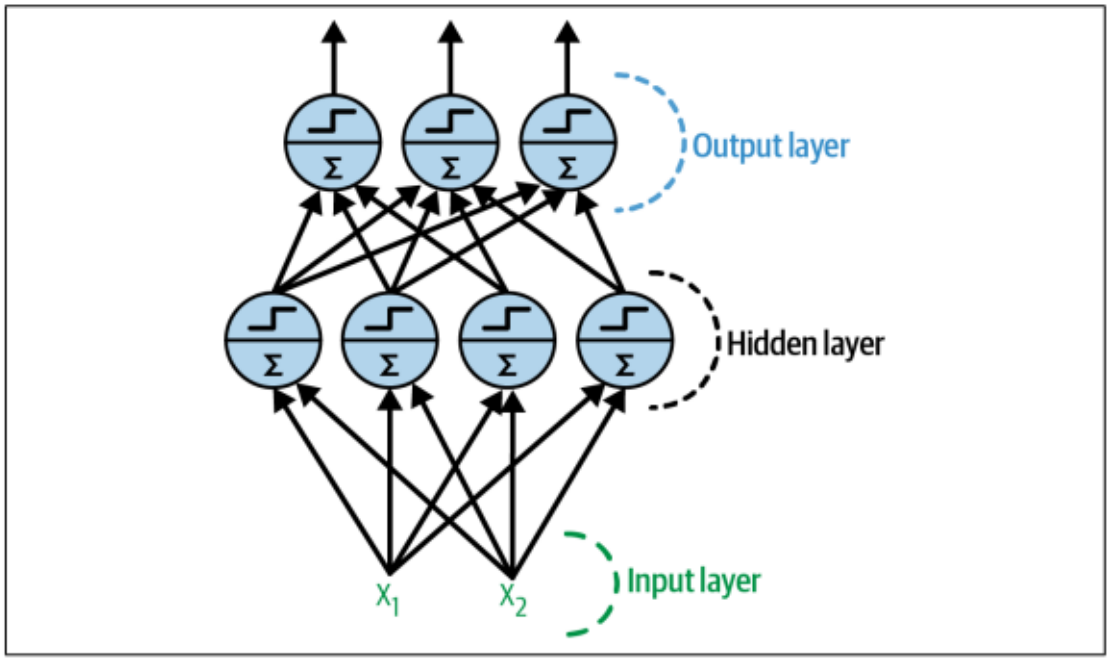
Σχήμα 3: Η αρχιτεκτονική ενός Perceptron με δύο εισόδους και τρείς νευρώνες εξόδου (Πηγή [4])

Αυτό το perceptron μπορεί να ταξινομήσει τις περιπτώσεις ταυτόχρονα σε τρεις διαφορετικές δυαδικές κλάσεις. Μπορεί επίσης να χρησιμοποιηθεί για πολυταξική (multiclass) ταξινόμηση. Η παρακάτω εξίσωση μπορεί να χρησιμοποιηθεί για να υπολογίσει τις εξόδους ενός επιπέδου τεχνητών νευρώνων για πολλές περιπτώσεις ταυτόχρονα.

$h_{W,b}(X) = \varphi(XW + b)$

Σε αυτή την εξίσωση το X αντιπροσωπεύει τον πίνακα των χαρακτηριστικών εισόδου. Έχει μία σειρά για την κάθε περίπτωση και μία στήλη για το κάθε χαρακτηριστικό. Ο πίνακας βαρών W περιέχει όλα τα βάρη. Έχει μία σειρά για κάθε είσοδο και μία στήλη για κάθε νευρώνα. Το διάνυσμα πόλωσης b περιέχει όλους τους όρους πόλωσης: ένας ανά νευρώνα. Η συνάρτηση φ ονομάζεται συνάρτηση ενεργοποίησης.[4]

Ένα πολυεπίπεδο Perceptron (Multilayer Perceptron - MLP) αποτελείται από ένα επίπεδο εισόδου, ένα ή περισσότερα επίπεδα TLU που ονομάζονται κρυφά επίπεδα (hidden layers) και ένα τελικό στρώμα TLU που ονομάζεται επίπεδο εξόδου (όπως φαίνεται και στο σχήμα 4). Τα επίπεδα τα οποία βρίσκονται κοντά στο επίπεδο εισόδου ονομάζονται συνήθως κατώτερα επίπεδα (lower layers) και αυτά που βρίσκονται κοντά στις εξόδους ονομάζονται συνήθως ανώτερα επίπεδα (upper layers).

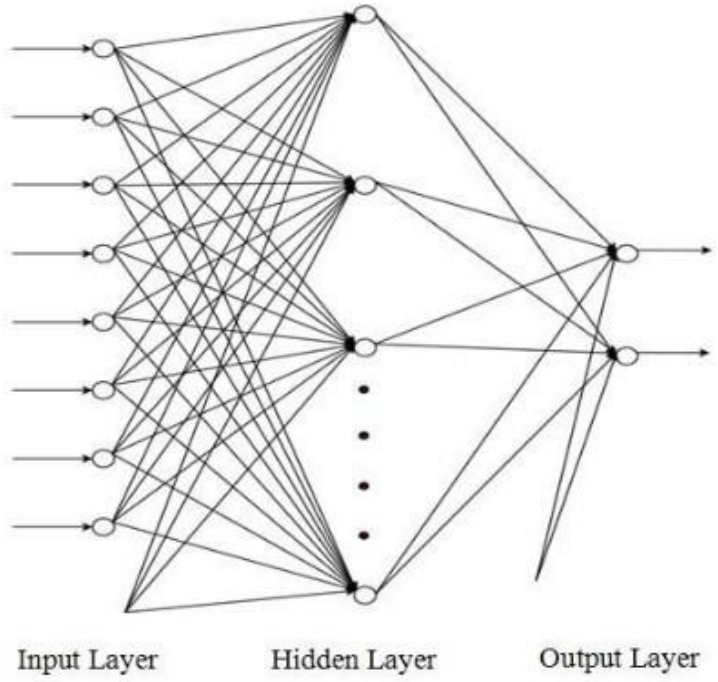


Σχήμα 4: Αρχιτεκτονική ενός πολυεπίπεδου perceptron με δύο εισόδους, ένα κρυφό επίπεδο τεσσάρων νευρώνων και τρεις νευρώνες εξόδου (Πηγή [4]).

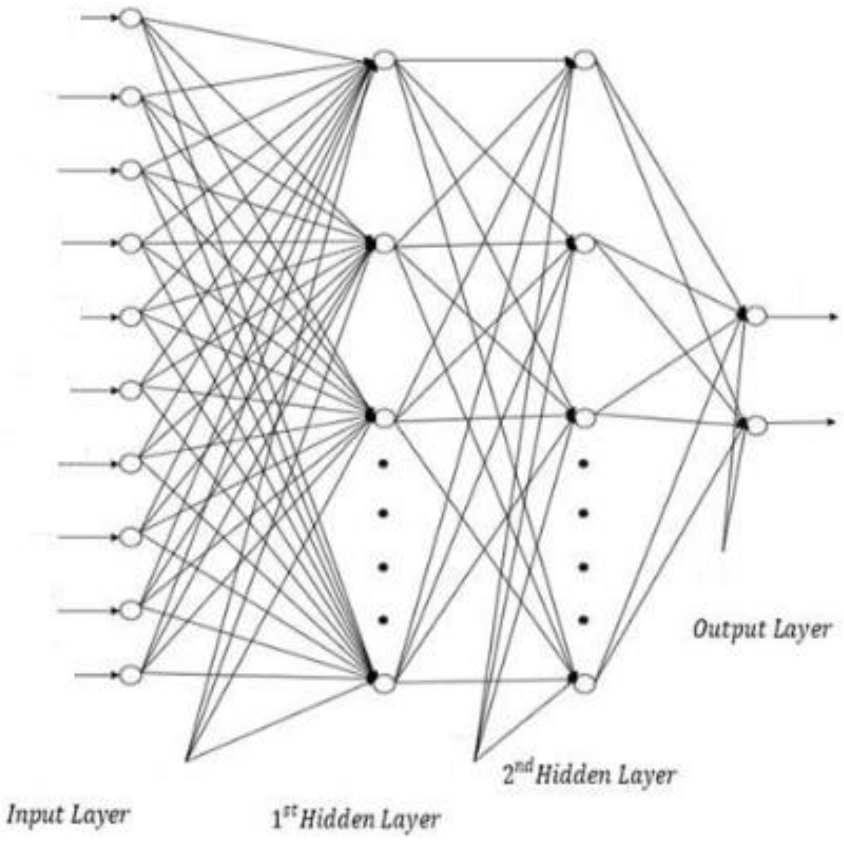
Η ροή των δεδομένων πηγαίνει μόνο προς μία κατεύθυνση (από τις εισόδους προς το εξόδους), επομένως αυτή η αρχιτεκτονική είναι ένα παράδειγμα ενός νευρωνικού δικτύου με πρόσθια τροφοδότηση (FNN).

- Νευρωνικά Δίκτυα με πρόσθια τροφοδότηση (Feedforward NN (FNN))

Τα νευρωνικά δίκτυα με πρόσθια τροφοδότηση είναι τα τεχνητά νευρωνικά δίκτυα στα οποία οι συνδέσεις μεταξύ των μονάδων δεν σχηματίζουν κύκλο. Αυτός ο τύπος νευρωνικών δικτύων ήταν ο πρώτος τύπος τεχνητού νευρωνικού δικτύου που εφευρέθηκε. Επιπλέον, είναι απλούστερα από τα αντίστοιχά τους, τα ανατροφοδοτούμενα νευρωνικά δίκτυα (recurrent neural networks RNN). Πήραν αυτήν την ονομασία, (Feedforward) επειδή οι πληροφορίες ταξιδεύουν μόνο προς τα εμπρός στο δίκτυο (χωρίς βρόχους), αρχικά μέσω των κόμβων εισόδου, μετά μέσω των κρυφών κόμβων (εάν υπάρχουν) και, τέλος, μέσω των κόμβων εξόδου. Τα νευρωνικά δίκτυα με πρόσθια τροφοδότηση μπορεί να είναι μονοεπίπεδα (Single-Layer) ή πολυεπίπεδα (Multilayer). Ο όρος «μονοεπίπεδο» αναφέρεται στο πλήθος των κρυφών επιπέδων του νευρωνικού δικτύου. Το επίπεδο εισόδου δεν προσμετράται καθώς δεν εκτελούνται υπολογισμοί εκεί. Η αρχιτεκτονική των πολυεπίπεδων δικτύων μπορεί να έχει παραπάνω από ένα κρυμμένα επίπεδα. Στα παρακάτω σχήματα απεικονίζονται σχηματικά τόσο τα μονοεπίπεδα όσο και τα πολυεπίπεδα δίκτυα [30].



Σχήμα 5: Μονοεπίπεδο νευρωνικό δίκτυο με πρόσθια τροφοδότηση (Πηγή [30])



Σχήμα 6: Πολυεπίπεδο νευρωνικό δίκτυο με πρόσθια τροφοδότηση (Πηγή [30])

Το κύριο πλεονέκτημα των νευρωνικών δικτύων με πρόσθια τροφοδότηση είναι ότι μπορούν να χρησιμοποιηθούν για δύσκολα έως πολύπλοκα προβλήματα. Επιπλέον, έχουν πολύ απλή

δομή σε σύγκριση με άλλους τύπους τεχνητών νευρωνικών δικτύων και δίνουν καλή απόδοση σε πολλές περιπτώσεις. Επίσης, αποτελούν ισχυρά νευρωνικά δίκτυα που μπορούν να εφαρμοστούν εύκολα και με επιτυχία σε διάφορους τομείς προβλημάτων. Ωστόσο, χρειάζονται μερικές φορές μεγάλο χρόνο εκπαίδευσης. Ακόμη, αυτός ο τύπος δικτύου απαιτεί μεγάλο αριθμό δεδομένων εισόδου για τη διαδικασία εκπαίδευσης [30].

Ας δούμε την διαδικασία εκπαίδευσης των νευρωνικών δικτύων με πρόσθια τροφοδότηση. Αν θεωρήσουμε ότι Y_j είναι η συνάρτηση που παράγεται στην έξοδο του νευρώνα j στο επίπεδο εξόδου από το ερέθισμα x(n), προκύπτει ότι το σήμα σφάλματος που παράγεται στην έξοδο του νευρώνα j είναι:

$e_j(n) = d_j(n) - y_j(n)$

όπου $j(n)$ είναι το iοστό στοιχείο του διανύσματος απάντησης d(n). Η ενέργεια του στιγμιαίου σφάλματος (instantaneous error energy) του νευρώνα j γίνεται γνωστή από τον τύπο:

$E_j(n)$ =12e_j2(n)

Αθροίζοντας τις συνεισφορές σφάλματος-ενέργειας (error-energy) όλων των νευρώνων στο επίπεδο εξόδου, η συνολική ενέργεια στιγμιαίου σφάλματος ολόκληρου του δικτύου εκφράζεται από τον τύπο:

$E(n)$ =$\sum_{j\in C} E_j$(n) =$\frac{1}{2}\sum_{j\in C} e_j^2\ (n)$

όπου το σύνολο C περιλαμβάνει όλους τους νευρώνες στο επίπεδο εξόδου. Με το δείγμα εκπαίδευσης που αποτελείται από N παραδείγματα, ο μέσος όρος ενέργειας σφάλματος στο δείγμα εκπαίδευσης ορίζεται από:

$Eav(N)$=$\frac{1}{N}$=$\sum_{n=1}^{N} E(n)$ =$\frac{1}{2N}\sum_{n=1}^{N}\ \sum_{j\in C}\ e_j^2$

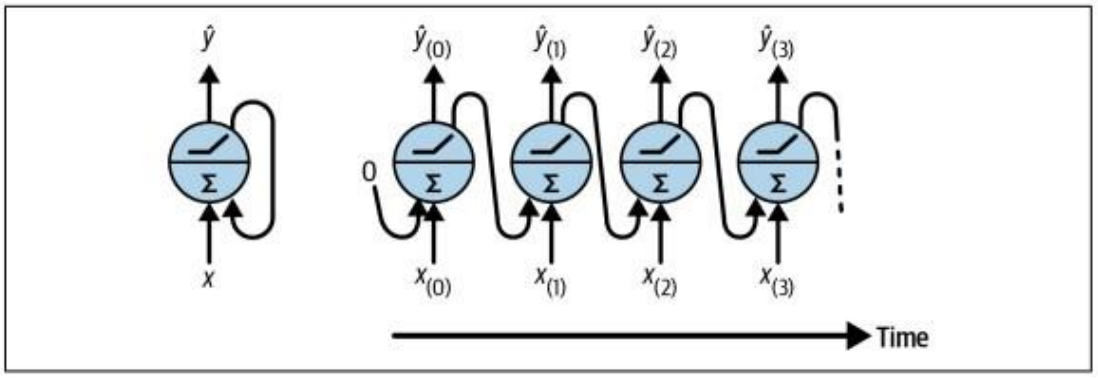
Παρουσιάζονται παρακάτω δύο διαφορετικές μέθοδοι για την εκπαίδευση του δικτύου: η εκμάθηση σε δέσμες (batch learning) και διαδικτυακή μάθηση (online learning). Στην εκμάθηση κατά δέσμες, οι προσαρμογές στα συναπτικά βάρη του νευρωνικού δικτύου πραγματοποιούνται μετά την παρουσίαση όλων των N παραδειγμάτων του δείγματος εκπαίδευσης που αποτελούν μια εποχή (epoch) εκπαίδευσης. Με άλλα λόγια, η συνάρτηση κόστους (cost function) για τη μάθηση κατά δέσμες ορίζεται από τη μέση ενέργεια σφάλματος Eav. Οι προσαρμογές στα συναπτικά βάρη του νευρωνικού δικτύου πραγματοποιούνται ανά εποχή (epoch by epoch).

Στη διαδικτυακή μέθοδο μάθησης, οι προσαρμογές των συναπτικών βαρών του νευρωνικού δικτύου πραγματοποιούνται ανά παράδειγμα. Η συνάρτηση κόστους που πρέπει να ελαχιστοποιηθεί, είναι επομένως η συνολική ενέργεια στιγμιαίου σφάλματος $E(n)$. Η on-line εκπαίδευση του δικτύου χρησιμοποιήθηκε για τον συντονισμό των παραμέτρων ελέγχου [30].

- Ανατροφοδοτούμενα νευρωνικά δίκτυα (Recurrent NN - RNN))

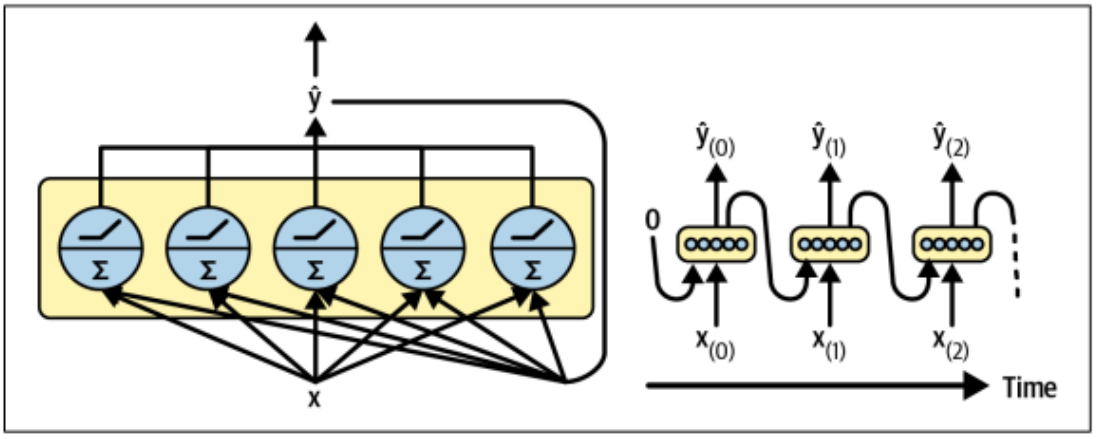
Ένα ανατροφοδοτούμενο νευρωνικό δίκτυο μοιάζει πολύ με ένα νευρωνικό δίκτυο με πρόσθια τροφοδότηση, με τη διαφορά ότι έχει επίσης συνδέσεις με κατεύθυνση προς τα πίσω. Τα ανατροφοδοτούμενα νευρωνικά δίκτυα διακρίνονται από ένα νευρωνικό δίκτυο τροφοδότησης το οποίο διαθέτει τουλάχιστον έναν βρόχο ανατροφοδότησης [4,30]. Ας δούμε το απλούστερο δυνατό RNN, που αποτελείται από έναν νευρώνα που δέχεται εισόδους, παράγει μια έξοδο, και στέλνει αυτή την έξοδο πίσω στον εαυτό του, όπως φαίνεται στο σχήμα 7 (αριστερά).

Σε κάθε χρονική στιγμή t, που ονομάζεται επίσης πλαίσιο (frame), αυτός ο επαναλαμβανόμενος νευρώνας λαμβάνει τις εισόδους $x_{(t)}$ καθώς και τη δική του έξοδο από το προηγούμενο χρονικό βήμα, $\hat{y}_{(t-1)}$. Δεδομένου ότι δεν υπάρχει προηγούμενη έξοδος στο πρώτο χρονικό βήμα, αυτή τίθεται στο 0. Μπορούμε να αναπαραστήσουμε αυτό το μικρό δίκτυο κατά τον άξονα του χρόνου, όπως φαίνεται στο σχήμα 7 (δεξιά). Αυτό ονομάζεται ξετύλιγμα (unrolling) του δικτύου μέσα στο χρόνο (είναι ο ίδιος επαναλαμβανόμενος νευρώνας που αναπαρίσταται μία φορά ανά χρονικό βήμα) [4].



Σχήμα 7: Ένας ανατροφοδοτούμενος νευρώνας (στα αριστερά), ο οποίος ξετυλίγεται μέσα στον χρόνο (δεξιά) (Πηγή [4])

Είναι εύκολο να δημιουργηθεί ένα στρώμα επαναλαμβανόμενων νευρώνων. Σε κάθε χρονικό βήμα t, κάθε νευρώνας λαμβάνει το διάνυσμα εισόδου $x_{(t)}$ και το διάνυσμα εξόδου από το προηγούμενο χρονικό βήμα $\hat{y}_{(t-1)}$, όπως φαίνεται παρακάτω στο σχήμα 8. Αξίζει να σημειωθεί ότι τόσο οι είσοδοι όσο και οι έξοδοι πλέον είναι διανύσματα (όταν υπήρχε μόνο ένας νευρώνας, η έξοδος ήταν κλιμακωτή (scalar).



Σχήμα 8: Ένα επίπεδο ανατροφοδοτούμενων νευρώνων (στα αριστερά) το οποίο ξετυλίγεται στο χρόνο (στα δεξιά) (Πηγή [4]).

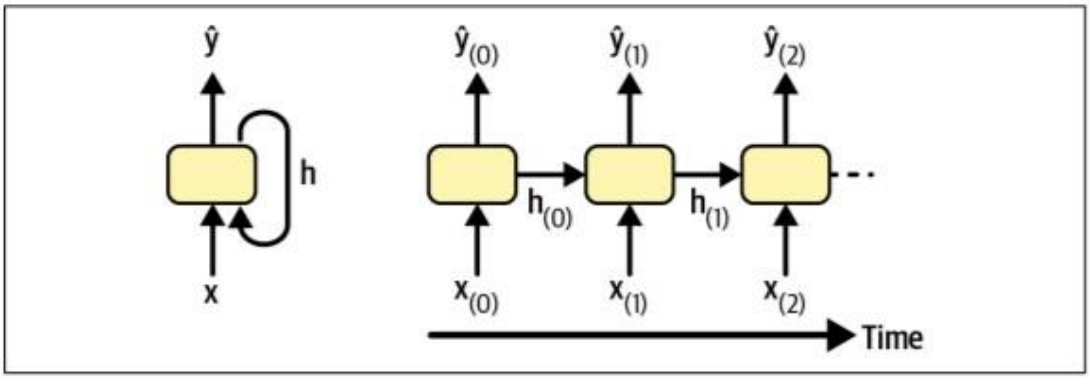
Κάθε επαναλαμβανόμενος νευρώνας έχει δύο σετ βαρών: ένα για τις εισόδους $x_{(t)}$ και το άλλο για τις εξόδους του προηγούμενου χρονικού βήματος, $\hat{y}_{(t-1)}$. Ας ονομάσουμε αυτά τα διανύσματα βάρους W_x και $W_{\hat{y}}$. Αν λάβουμε υπόψη ολόκληρο το επαναλαμβανόμενο επίπεδο αντί για έναν μόνο ανατροφοδοτούμενο νευρώνα, μπορούμε να τοποθετήσουμε όλα τα διανύσματα βάρους σε δύο πίνακες βάρους: Wx και Wŷ. Το διάνυσμα εξόδου ολόκληρου του ανατροφοδοτούμενου επιπέδου μπορεί στη συνέχεια να υπολογιστεί όπως φαίνεται στην εξίσωση που ακολουθεί, όπου b είναι το διάνυσμα πόλωσης και φ() είναι η συνάρτηση ενεργοποίησης (π.χ. ReLU).

$$\hat{y}_t = \varphi(W_x^T x_t + W_{\hat{y}}^T \hat{y}_{(t-1)} + b)$$

Εφόσον η έξοδος ενός ανατροφοδοτούμενου νευρώνα στο χρονικό βήμα t είναι συνάρτηση όλων των εισόδων από τα προηγούμενα χρονικά βήματα, θα μπορούσαμε να πούμε ότι έχει μια μορφή μνήμης. Ένα μέρος ενός νευρωνικού δικτύου που διατηρεί κάποια κατάσταση στα χρονικά βήματα ονομάζεται κύτταρο μνήμης (memory cell). Ένας μεμονωμένος ανατροφοδοτούμενος νευρώνας, ή ένα επίπεδο ανατροφοδοτούμενων νευρώνων, αποτελεί βασικό κύτταρο μνήμης, το οποίο μπορεί να μάθει μόνο μικρά μοτίβα (συνήθως περίπου 10 βημάτων, αλλά αυτό ποικίλλει ανάλογα με την εργασία). Η κατάσταση ενός κυττάρου στο χρονικό βήμα t, που συμβολίζεται h(t) (το "h" σημαίνει "κρυμμένο"(hidden)), είναι συνάρτηση μερικών από τις εισόδους στο συγκεκριμένο χρονικό βήμα και την κατάστασή του στο προηγούμενο χρονικό βήμα:

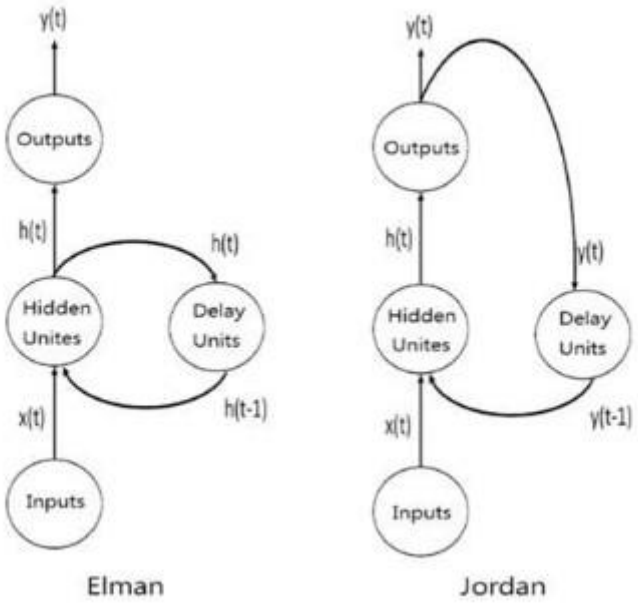
$h_{(t)} = f(x_{(t)}, h_{(t-1)})$

Η έξοδός του στο χρονικό βήμα t, που συμβολίζεται με $\hat{y}_{(t)}$, είναι επίσης συνάρτηση της προηγούμενης κατάστασης του και των τρεχουσών εισόδων. Στην περίπτωση των βασικών κυττάρων, η έξοδος είναι απλά ίση με την τρέχουσα κατάσταση, αλλά σε πιο σύνθετα κύτταρα αυτό δεν συμβαίνει πάντα, όπως φαίνεται στο παρακάτω σχήμα.[4]



Σχήμα 9: Η κρυφή κατάσταση ενός κελιού και η έξοδός του μπορεί να είναι διαφορετικές (Πηγή [4])

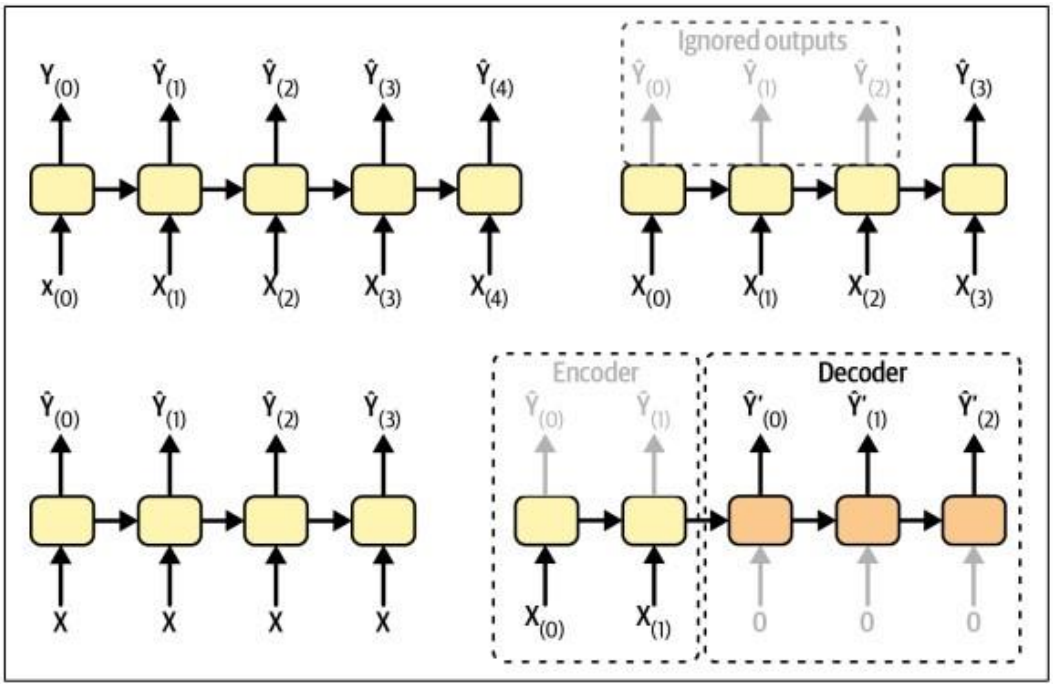
Οι δύο κύριες παραλλαγές που προτείνονται στη βιβλιογραφία [30] σχετικά με τα ανατροφοδοτούμενα νευρωνικά δίκτυα, που αποκαλούνται επίσης "απλά" RNN, είναι τα μοντέλα RNN του Elman και του Jordan. Και τα δύο νευρωνικά δίκτυα Elman και Jordan αποτελούνται από ένα επίπεδο εισόδου, ένα κρυφό επίπεδο (hidden layer), ένα επίπεδο καθυστέρησης (delay layer) και ένα επίπεδο εξόδου. Οι νευρώνες καθυστέρησης ενός νευρωνικού δικτύου Elman τροφοδοτούνται από το κρυφό επίπεδο, ενώ οι νευρώνες καθυστέρησης ενός νευρωνικού δικτύου Jordan τροφοδοτούνται από το επίπεδο εξόδου [30]. Η δομή του Elman και του Jordan παρουσιάζεται στο παρακάτω σχήμα:



Σχήμα 10: Στα αριστερά παρουσιάζεται η δομή του ανατροφοδοτούμενου νευρωνικού δικτύου του Elman και στα αριστερά του Jordan.[30]

Ένα ανατροφοδοτούμενο νευρωνικό δίκτυο μπορεί ταυτόχρονα να λάβει μια ακολουθία εισόδων και να παράγει μια ακολουθία εξόδων (όπως φαίνεται στο δίκτυο πάνω αριστερά στο σχήμα 11). Αυτός ο τύπος δικτύου αλληλουχίας σε αλληλουχία (sequence to sequence) χρησιμεύει για την πρόβλεψη χρονοσειρών, όπως π.χ. η ημερήσια κατανάλωση ενέργειας ενός σπιτιού. Το δίκτυο μπορεί να τροφοδοτηθεί επίσης από μια ακολουθία εισόδων και να αγνοήσει όλες τις εξόδους εκτός από την τελευταία (όπως φαίνεται στο δίκτυο πάνω δεξιά του σχήματος 11). Αυτό είναι ένα δίκτυο αλληλουχίας σε διάνυσμα (sequence to vector). Για παράδειγμα, θα μπορούσε να τροφοδοτηθεί το δίκτυο με μια ακολουθία λέξεων που αντιστοιχεί σε μια κριτική ταινίας και το δίκτυο θα έβγαζε μια ταξινόμηση συναισθήματος (π.χ. από 0 [μίσος] έως 1 [αγάπη]).

Επίσης, θα μπορούσε να τροφοδοτηθεί το δίκτυο με το ίδιο διάνυσμα εισόδου ξανά και ξανά σε κάθε χρονικό βήμα και έτσι να εξάγει μια ακολουθία (όπως φαίνεται κάτω αριστερά στο σχήμα 11). Αυτό είναι ένα δίκτυο διανύσματος σε αλληλουχία (vector to sequence). Για παράδειγμα, η είσοδος θα μπορούσε να έχει τη μορφή εικόνας και η έξοδος θα μπορούσε να είναι η λεζάντα για αυτήν την εικόνα. Τέλος, υπάρχει το δίκτυο αλληλουχίας σε διάνυσμα (sequence to vector), που ονομάζεται κωδικοποιητής (encoder) το οποίο ακολουθείται από ένα δίκτυο διανύσματος σε αλληλουχία (vector to sequence), που ονομάζεται αποκωδικοποιητής (decoder) (όπως φαίνεται στο σχήμα 11 κάτω δεξιά). Για παράδειγμα, αυτό θα μπορούσε να χρησιμοποιηθεί για τη μετάφραση μιας πρότασης από μια γλώσσα σε μια άλλη. Θα τροφοδοτούταν το δίκτυο με μια πρόταση σε μια γλώσσα, την οποία ο κωδικοποιητής θα μετατρέψει σε μια ενιαία διανυσματική αναπαράσταση και στη συνέχεια το ο αποκωδικοποιητής θα αποκωδικοποιήσει αυτό το διάνυσμα σε μια πρόταση σε άλλη γλώσσα.



Σχήμα 11: Sequence-to-sequence (πάνω αριστερά), sequence-to-vector (πάνω δεξιά), vector-to-sequence (κάτω αριστερά), and encoder–decoder (κάτω δεξιά) (Πηγή [4])

Τα ανατροφοδοτούμενα νευρωνικά δίκτυα αποτελούν ισχυρά υπολογιστικά εργαλεία τα οποία μπορούν να προσομοιώσουν μια οποιαδήποτε αλγοριθμική διαδικασία. Μπορούν να χρησιμοποιηθούν σε πολλές εφαρμογές και μοντέλα χρονικής επεξεργασίας (temporal processing models). Επίσης, διαθέτουν τη λεγόμενη καθολική ιδιότητα προσέγγισης (universal approximation property), δηλαδή είναι ικανά να προσεγγίζουν αυθαίρετα μη γραμμικά δυναμικά συστήματα με αυθαίρετη ακρίβεια, υλοποιώντας πολύπλοκες αντιστοιχίσεις από ακολουθίες εισόδου σε ακολουθίες εξόδου. Το μέγεθος των νευρωνικών δικτύων αυτών είναι σημαντικά

συμπαγές σε σύγκριση με τα δίκτυα πρόσθιας τροφοδότησης ώστε να μπορέσει να επιτευχθεί η ίδια ακρίβεια. Επιπλέον, τα δίκτυα αυτά χρησιμοποιούνται ευρέως σε εργασίες αλληλεπίδρασης ανθρώπου-ρομπότ [30].

Παρόλο που τα ανατροφοδοτούμενα νευρωνικά δίκτυα είναι απλά και ισχυρά μοντέλα, στην πράξη είναι δύσκολο να εκπαιδευτούν σωστά. Λόγω της μη γραμμικής φύσης των μονάδων ενεργοποίησης εξόδου και των στρατηγικών προσαρμογής των βαρών, η σταθερότητα του δικτύου είναι συχνά δύσκολο να εξακριβωθεί. Ακόμη, δεν μπορεί να επεξεργαστεί πολύ μεγάλες ακολουθίες εάν χρησιμοποιείται η tanh ή η relu ως συνάρτηση ενεργοποίησης [22]. Επιπλέον, υπάρχει το πρόβλημα μακροχρόνιας εξάρτησης (Long-term dependency problem), δηλαδή όταν επεξεργαζόμαστε ένα πολύ μεγάλο έγγραφο και προσπαθούμε να συνδέσουμε τους όρους που απέχουν πολύ μεταξύ τους, πρέπει να φροντίζουμε και να κωδικοποιούμε όλους τους άσχετους άλλους όρους μεταξύ αυτών των όρων [36].

Ένα ακόμη πρόβλημα αποτελεί το ότι είναι επιρρεπής σε προβλήματα εκρηκτικά αυξανόμενης ή εξαφανιζόμενης κλίσης (prone to exploding or vanishing gradient problems), δηλαδή όταν επεξεργαζόμαστε έγγραφα μεγάλου μήκους, η ενημέρωση των βαρών των πρώτων λέξεων είναι μεγάλη υπόθεση, γεγονός που καθιστά ένα μοντέλο μη εκπαιδεύσιμο λόγω προβλήματος εξαφανιζόμενης κλίσης. Ακόμη είναι δύσκολη η εφαρμογή παραλληλοποιήσιμης εκπαίδευσης (parallelizable training), η παραλληλοποίηση μπορεί να χωρίσει το κύριο πρόβλημα σε μικρότερα προβλήματα και έτσι μπορούν να εκτελούνται οι λύσεις ταυτόχρονα, όμως το ανατροφοδοτούμενο νευρωνικό δίκτυο ακολουθεί μια κλασική διαδοχική προσέγγιση. Κάθε στρώμα εξαρτάται έντονα από τα προηγούμενα στρώματα, γεγονός που καθιστά την παραλληλοποίηση αδύνατη. Επιπλέον, ο υπολογισμός είναι αργός αν η ακολουθία είναι μεγάλη. Ένα ανατροφοδοτούμενο νευρωνικό δίκτυο θα μπορούσε να είναι πολύ αποδοτικό για προβλήματα σύντομων κειμένων, όμως όταν επεξεργάζεται μεγαλύτερα έγγραφα η διαδικασία είναι πολύ αργή [36].

Παρακάτω θα δούμε δύο τρόπους εκπαίδευσης ενός ανατροφοδοτούμενου νευρωνικού δικτύου. Ο πρώτος τρόπος είναι η εκπαίδευση ανά εποχή (Epochwise training). Για μια δεδομένη εποχή, το νευρωνικό δίκτυο χρησιμοποιεί μια χρονική ακολουθία ζευγών εισόδου εξόδου και αρχίζει να τρέχει από κάποια αρχική κατάσταση μέχρι να φτάσει σε μια νέα κατάσταση, οπότε η εκπαίδευση διακόπτεται και το δίκτυο επαναφέρεται στην αρχική του κατάσταση για την επόμενη εποχή (epoch). Η αρχική κατάσταση δεν χρειάζεται να είναι η ίδια για κάθε εποχή εκπαίδευσης.

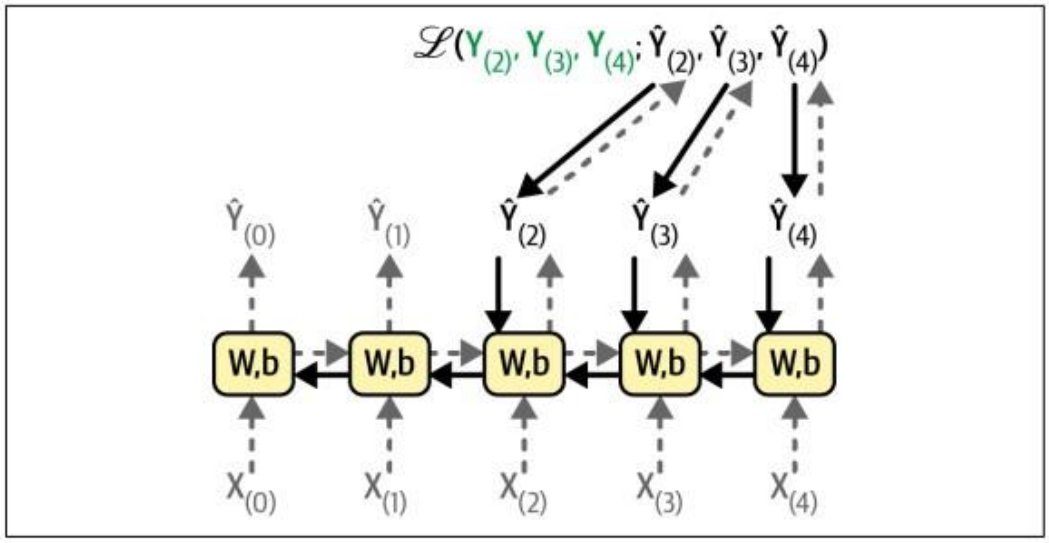
Ο δεύτερος τρόπος είναι η συνεχής εκπαίδευση (Continuous training). Αυτή η μέθοδος εκπαίδευσης είναι κατάλληλη για περιστάσεις όπου δεν υπάρχουν διαθέσιμες καταστάσεις επαναφοράς ή απαιτείται μάθηση μέσω διαδικτύου. Το χαρακτηριστικό γνώρισμα της συνεχούς εκπαίδευσης είναι ότι το δίκτυο μαθαίνει ενώ εκτελεί επεξεργασία σήματος (signal processing) . Με απλά λόγια, η διαδικασία μάθησης δεν σταματά ποτέ [30].

Για να εκπαιδευτεί ένα ανατροφοδοτούμενο νευρωνικό δίκτυο, το κόλπο είναι να το ξετυλίξουμε στο χρόνο και στη συνέχεια να χρησιμοποιήσουμε την κανονική οπισθοδιάδοση όπως φαίνεται στο παρακάτω σχήμα. Αυτή η στρατηγική ονομάζεται οπισθοδιάδοση στον χρόνο (backpropagation through time - BPTT). Ακριβώς όπως και στην κανονική οπισθοδιάδοση, υπάρχει ένα πρώτο πέρασμα προς τα εμπρός μέσω του ξετυλιγμένου δίκτυο (όπως φαίνεται από τα διακεκομμένα βέλη). Στη συνέχεια αξιολογείται η ακολουθία εξόδου χρησιμοποιώντας μια συνάρτηση απώλειας:

$L(Y(0), Y(1), ..., Y(T); \hat{Y}(0), \hat{Y}(1), ..., \hat{Y}(T))$

(όπου Y(i) είναι ο i-οστός στόχος, $\hat{Y}(i)$ είναι η i-οστή πρόβλεψη και T είναι το μέγιστο χρονικό βήμα). Αξίζει να σημειωθεί ότι αυτή η συνάρτηση απώλειας μπορεί να αγνοεί ορισμένες εξόδους. Για παράδειγμα, σε ένα RNN από ακολουθία σε διάνυσμα, όλες οι έξοδοι αγνοούνται εκτός από την τελευταία. Στο παρακάτω σχήμα, η συνάρτηση απώλειας υπολογίζεται με βάση τις τρεις τελευταίες εξόδους μόνο. Οι κλίσεις αυτής της συνάρτησης απώλειας διαδίδονται στη συνέχεια προς τα πίσω μέσω του ξετυλιγμένου δικτύου (που αναπαρίσταται από τα συμπαγή βέλη) [4].

Σε αυτό το παράδειγμα, δεδομένου ότι οι έξοδοι $\hat{Y}(0)$ και $\hat{Y}(1)$ δεν χρησιμοποιούνται για τον υπολογισμό των απωλειών, οι κλίσεις δεν ρέουν προς τα πίσω μέσω αυτών- ρέουν μόνο μέσω των $\hat{Y}(2)$, $\hat{Y}(3)$ και $\hat{Y}(4)$. Επιπλέον, δεδομένου ότι οι ίδιες παράμετροι W και b χρησιμοποιούνται σε κάθε χρονικό βήμα, οι κλίσεις τους θα ρυθμιστούν πολλές φορές κατά τη διάρκεια της οπισθοδιάδοσης. Μόλις ολοκληρωθεί η φάση οπισθοδιάδοσης και υπολογιστούν όλες οι κλίσεις, ο αλγόριθμος αυτός μπορεί να εκτελέσει ένα βήμα βαθμωτής καθόδου (gradient descent step) για την ενημέρωση των παραμέτρων.



Σχήμα 12: Οπισθοδιάδοση στον χρόνο (Πηγή [4])

Τα ανατροφοδοτούμενα νευρωνικά δίκτυα είναι πιο πολύπλοκα σε σύγκριση με τα νευρωνικά δίκτυα με πρόσθια τροφοδότηση. Το μαθηματικό μοντέλο των ανατροφοδοτούμενων είναι επίσης πιο περίπλοκο σε σύγκριση με τα δίκτυα με πρόσθια τροφοδότηση [30].

Το πρόβλημα της ταξινόμησης συναισθημάτων ανήκει στην κατηγορία πρόβλημα μοντελοποίησης ακολουθιών, επειδή μια ακολουθία λέξεων (word tokens) απεικονίζεται σε μια κατηγορία συναισθήματος (many to one mapping). Δηλαδή μια ακολουθία λέξεων συσχετίζεται με μια συγκεκριμένη κατηγορία συναισθήματος. Για να μοντελοποιήσουμε τις ακολουθίες, πρέπει να χειριστούμε ακολουθίες μεταβλητού μήκους, να παρακολουθήσουμε μακροπρόθεσμες εξαρτήσεις, να διατηρήσουμε τις πληροφορίες σχετικά με την σειρά καθώς και να χρησιμοποιήσουμε κοινές παραμέτρους σε όλη την ακολουθία. [5]

Επιπλέον, τα ανατροφοδοτούμενα νευρωνικά δίκτυα, έχουν καθιερωθεί σταθερά ως κύριες προσεγγίσεις στη μοντελοποίηση ακολουθιών και στα προβλήματα μεταγωγής (transduction problems) όπως η μοντελοποίηση γλώσσας (language modeling) και η μηχανική μετάφραση (machine translation). Πρόσφατα, έχουν επιτευχθεί σημαντικές βελτιώσεις στην υπολογιστική απόδοση μέσω τεχνασμάτων παραγοντοποίησης (factorization tricks) και του υπολογισμού υπό συνθήκες (conditional computation) (κάτι που βελτιώνει επίσης την απόδοση του μοντέλου). [32]

Οι μηχανισμοί προσοχής (attention mechanisms) έχουν γίνει αναπόσπαστο μέρος της μοντελοποίησης ακολουθιών και μοντέλων μεταγωγής σε πολλές εργασίες, επιτρέποντας τη μοντελοποίηση των εξαρτήσεων χωρίς να λαμβάνεται υπόψη η απόστασή τους στις ακολουθίες εισόδου ή εξόδου. Στις περισσότερες περιπτώσεις, τέτοιοι μηχανισμοί προσοχής χρησιμοποιούνται σε συνδυασμό με ένα ανατροφοδοτούμενο νευρωνικό δίκτυο. Σε αυτή την εργασία προτείνουμε τον Μετασχηματιστή, μια αρχιτεκτονική μοντέλου που αποφεύγει την επανάληψη και αντ' αυτού βασίζεται εξ ολοκλήρου σε έναν μηχανισμό προσοχής για να αντλεί παγκόσμιες εξαρτήσεις μεταξύ εισόδου και εξόδου [32].

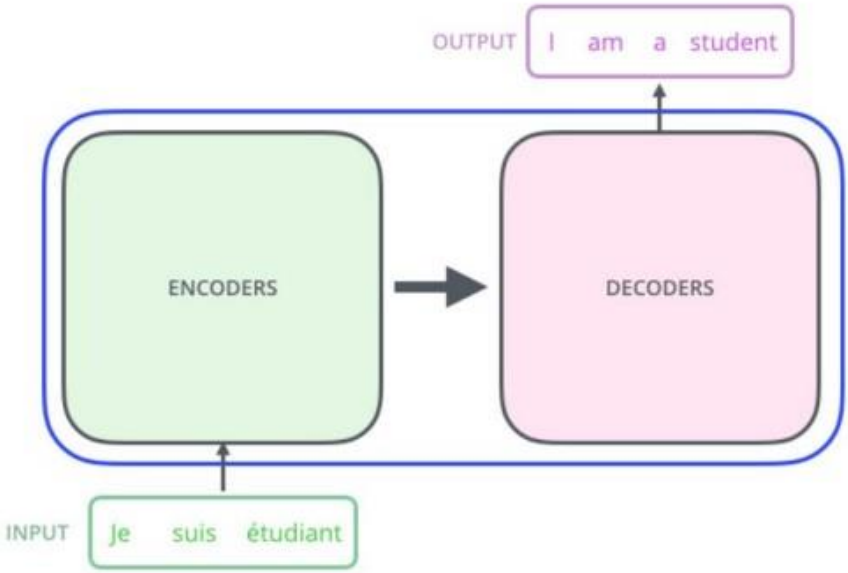
TRANSFORMER

Τα μοντέλα μετασχηματιστών (Transformer) έχουν προσελκύσει τεράστιο ενδιαφέρον λόγω της αποτελεσματικότητάς τους σε ένα τεράστιο φάσμα προβλημάτων επεξεργασίας φυσικής γλώσσας (NLP), από την ταξινόμηση κειμένων έως και τη δημιουργία κειμένων. Ο μηχανισμός προσοχής (attention mechanism) αποτελεί σημαντικό μέρος αυτών των μοντέλων και παίζει πολύ κρίσιμο ρόλο. Πριν από τα μοντέλα Transformer, ο μηχανισμός προσοχής προτάθηκε ως βοηθητικός μηχανισμός για τη βελτίωση των συμβατικών μοντέλων βαθιάς μάθησης όπως τα ανατροφοδοτούμενα νευρωνικά δίκτυα (RNN). Για να έχουμε μια καλή κατανόηση των Transformers και του αντίκτυπού τους στην επεξεργασία φυσικής γλώσσας, θα μελετήσουμε πρώτα τον μηχανισμό προσοχής [36].

Μια από τις πρώτες παραλλαγές του μηχανισμού προσοχής προτάθηκε από τον Bahdanau. Ο μηχανισμός αυτός βασίζεται στο γεγονός ότι τα μοντέλα που βασίζονται σε ανατροφοδοτούμενα νευρωνικά δίκτυα, αντιμετωπίζουν προβλήματα συμφόρησης πληροφοριών σε εργασίες όπως η μηχανική μετάφραση με τη χρήση νευρωνικών δικτύων (Neural Machine Translation)(σχήμα 13).[36]

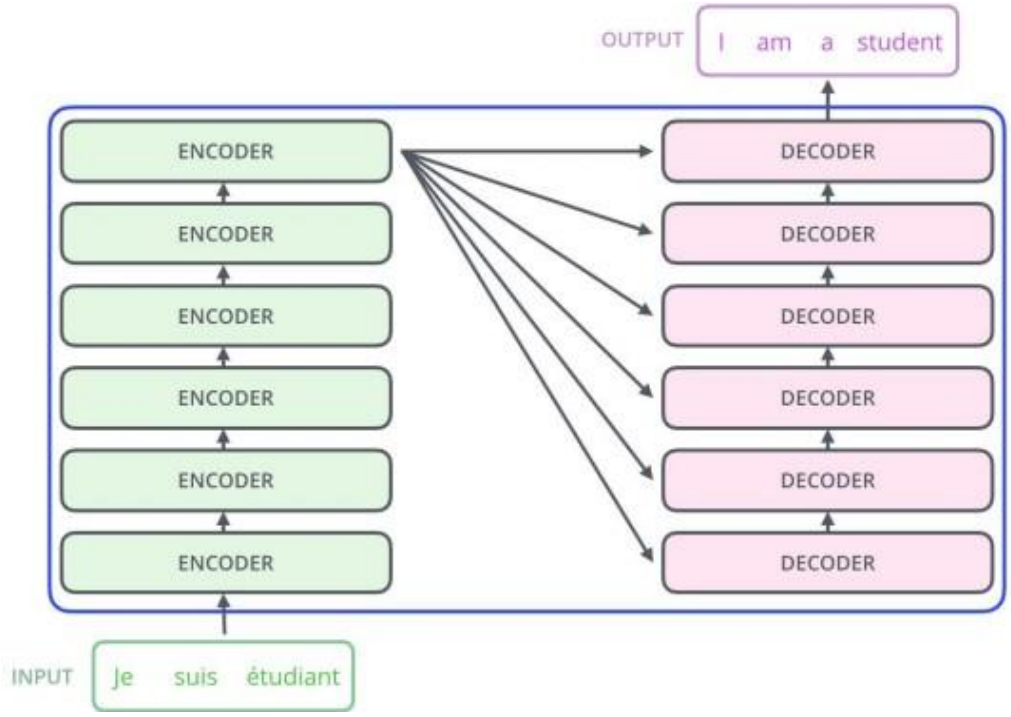
Σχήμα 13: Απεικόνιση μηχανικής μετάφρασης με τη χρήση νευρωνικών δικτύων (Πηγή [22])

Αυτά τα μοντέλα που βασίζονται σε αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή (encoder-decoder-based), λαμβάνουν την είσοδο με τη μορφή ενός token-id και την επεξεργάζονται με έναν ανατροφοδοτούμενο τρόπο (αυτή είναι δουλειά του κωδικοποιητή). Στη συνέχεια, η ενδιάμεση πληροφορία που έχει ήδη επεξεργαστεί τροφοδοτείται σε μια άλλη ανατροφοδοτούμενη μονάδα, τον αποκωδικοποιητή, για την εξαγωγή των αποτελεσμάτων [36].



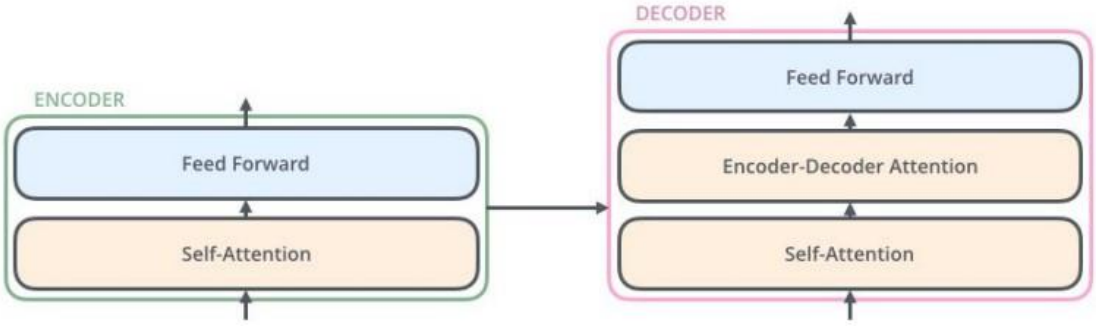
Σχήμα 14: Encoder-Decoder [22]

Ο συνολικός κωδικοποιητής είναι μια στοίβα κωδικοποιητών και ο συνολικός αποκωδικοποιητής είναι μια στοίβα αποκωδικοποιητών του ίδιου αριθμού. Οι κωδικοποιητές είναι όλοι πανομοιότυποι στη δομή, ωστόσο δεν μοιράζονται τα ίδια βάρη.



Σχήμα 15: Στοίβες encoder – decoder [22]

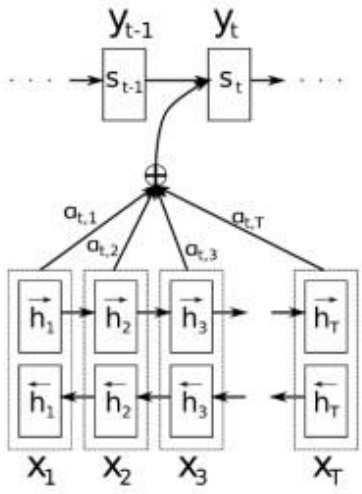
Κάθε κωδικοποιητής χωρίζεται σε δύο μέρη όπως φαίνεται στο παρακάτω σχήμα. Ένας κωδικοποιητής λαμβάνει μια λίστα διανυσμάτων ως είσοδο. Οι είσοδοι του κωδικοποιητή διέρχονται πρώτα από ένα στρώμα αυτοπροσοχής ώστε να επεξεργαστούν και στη συνέχεια οι έξοδοι του στρώματος αυτοπροσοχής τροφοδοτούνται σε ένα νευρωνικό δίκτυο με πρόσθια τροφοδότηση. Έπειτα, ο αποκωδικοποιητής διαθέτει ένα επιπλέον στρώμα προσοχής το οποίο τον βοηθάει να εστιάσει σε συγκεκριμένα σημεία της πρότασης εισόδου [22].



Σχήμα 16: Μέρη κωδικοποιητή – αποκωδικοποιητή (Πηγή [22])

Κάθε ενσωμάτωση λέξης (word embedding) της ακολουθίας εισόδου περνάει από κάθε ένα από τα δύο επίπεδα του κωδικοποιητή. Η μέγιστη ακολουθία εισόδου μπορεί να έχει μήκος έως 512 (αυτό μπορεί να είναι το μήκος της μεγαλύτερης πρότασης στο σύνολο δεδομένων εκπαίδευσης). Καθώς το μοντέλο επεξεργάζεται κάθε word token, η αυτοπροσοχή του επιτρέπει να εξετάζει άλλες θέσεις στην ακολουθία εισόδου για ενδείξεις που μπορούν να βοηθήσουν στην καλύτερη κωδικοποίηση αυτής της λέξης [22].

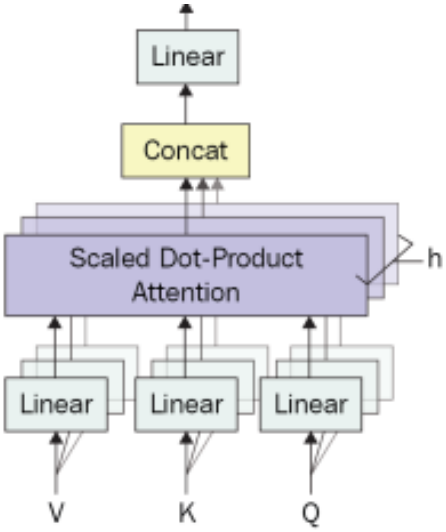
Ο Bahdanau πρότεινε έναν μηχανισμό προσοχής που χρησιμοποιεί βάρη (weights) στις ενδιάμεσες κρυφές τιμές. Αυτά τα βάρη ευθυγραμμίζουν το ποσό της προσοχής που πρέπει να δώσει ένα μοντέλο στις εισόδους σε κάθε βήμα αποκωδικοποίησης. Μια τέτοια καθοδήγηση βοηθά τα μοντέλα σε συγκεκριμένες εργασίες, όπως η νευρωνική μηχανική μετάφραση. Ένα διάγραμμα ενός τυπικού μηχανισμού προσοχής φαίνεται στο σχήμα 17:



Σχήμα 17: Μηχανισμός προσοχής [36]

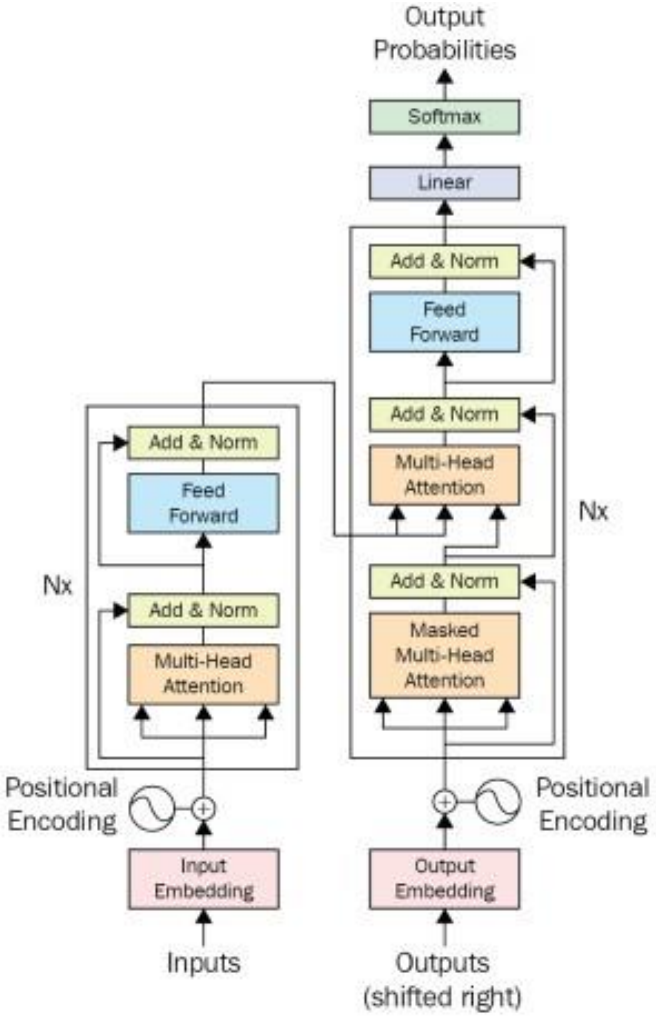
Δεδομένου ότι οι μηχανισμοί προσοχής δεν μπορούν να χρησιμοποιηθούν μόνο για την επεξεργασία φυσικής γλώσσας, χρησιμοποιούνται επίσης σε διάφορες περιπτώσεις και σε διάφορους τομείς, από την όραση υπολογιστών (computer vision) έως την αναγνώριση ομιλίας (speech recognition) [36].

Ο μηχανισμός προσοχής πολλαπλών κεφαλών (multi-head attention mechanism) που παρουσιάζεται στο ακόλουθο σχήμα αποτελεί θεμέλιο των transformers και είναι βασικό μέρος της αρχιτεκτονικής τους:



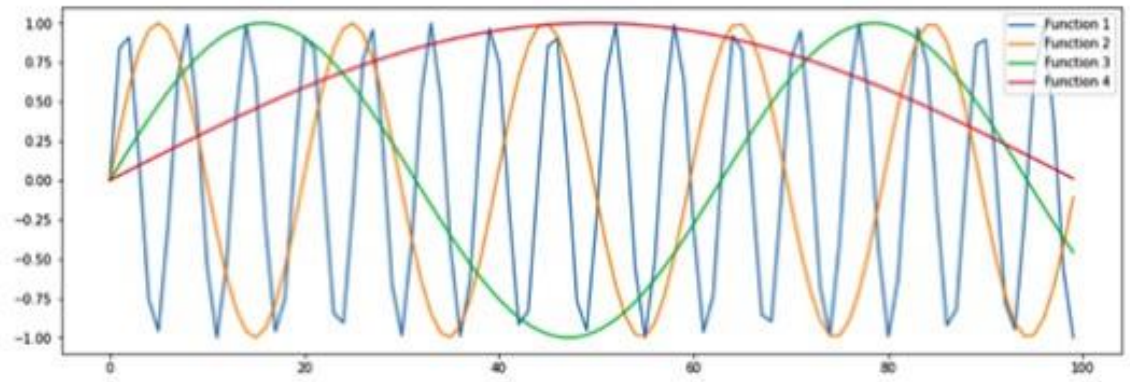
Σχήμα 18: Multi-head attention mechanism [36]

Ο μηχανισμός προσοχής “scaled dot-product” μοιάζει πολύ με τον μηχανισμό αυτοπροσοχής, με τη διαφορά ότι χρησιμοποιεί έναν συντελεστή κλιμάκωσης (scaling factor). Το κομμάτι πολλαπλών κεφαλών (multi-head), από την άλλη πλευρά, εξασφαλίζει ότι το μοντέλο είναι ικανό να εξετάζει διάφορες πτυχές των δεδομένων εισόδου σε όλα τα επίπεδα. Τα μοντέλα transformer παρακολουθούν τις επισημάνσεις του κωδικοποιητή (encoder) και τις κρυφές τιμές από τα προηγούμενα επίπεδα. Η αρχιτεκτονική του μοντέλου Transformer δεν έχει μια ανατροφοδοτούμενη ροή βήμα προς βήμα, αντίθετα, χρησιμοποιεί κωδικοποίηση θέσης προκειμένου να έχει πληροφορίες σχετικά με τη θέση κάθε συμβόλου στην ακολουθία εισόδου. Οι συνυφασμένες τιμές των ενσωματώσεων (embeddings) (τυχαία αρχικοποιημένες) και οι σταθερές τιμές των κωδικοποιημένων θέσεων είναι η είσοδος που τροφοδοτείται στα επίπεδα στο πρώτο μέρος του κωδικοποιητή και διαδίδεται μέσω της αρχιτεκτονικής, όπως απεικονίζεται στο ακόλουθο σχήμα:



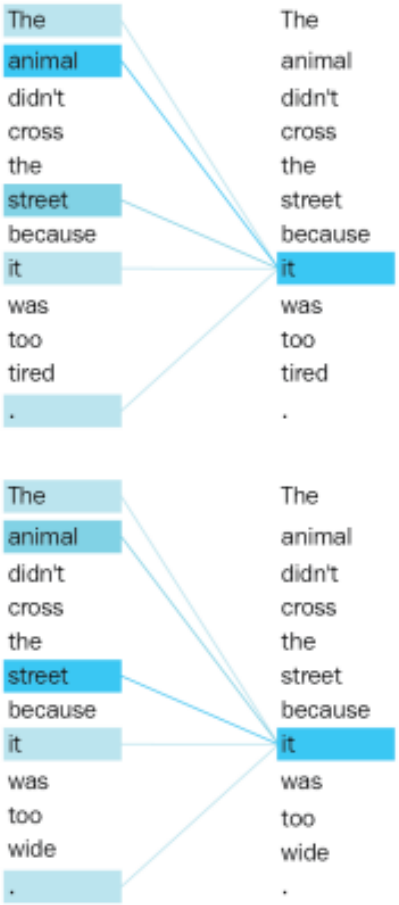
Σχήμα 19: Ένας Transformer (Πηγή [36])

Οι πληροφορίες θέσης λαμβάνονται με την αξιολόγηση ημιτονοειδών και συνημιτονοειδών κυμάτων σε διαφορετικές συχνότητες. Ένα παράδειγμα κωδικοποίησης θέσης απεικονίζεται στο ακόλουθο στιγμιότυπο:



Σχήμα 20: Κωδικοποίηση θέσης (Πηγή [36])

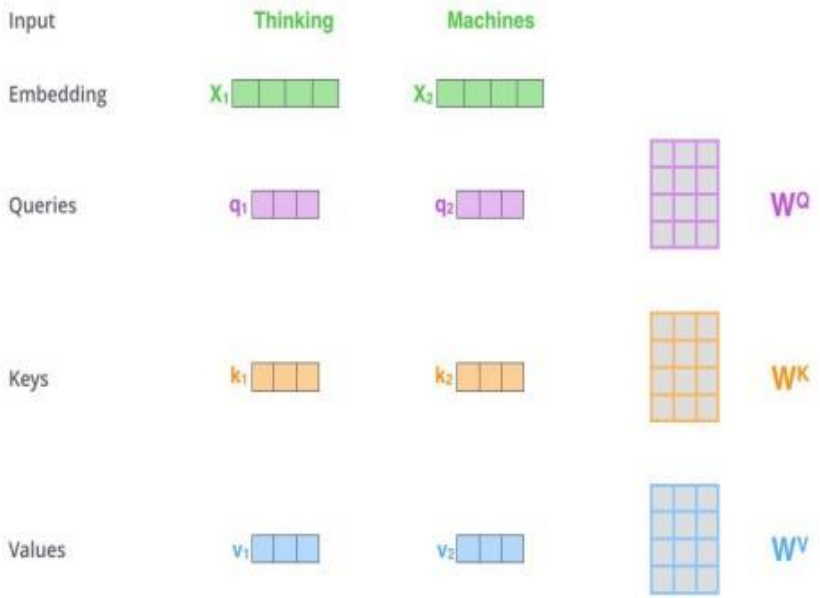
Ένα καλό παράδειγμα επιδόσεων στην αρχιτεκτονική Transformer και στον μηχανισμό προσοχής “dot-product” δίνεται στο ακόλουθο σχήμα:



Σχήμα 21: Απεικόνιση προσοχής των Transformers (Πηγή [36])

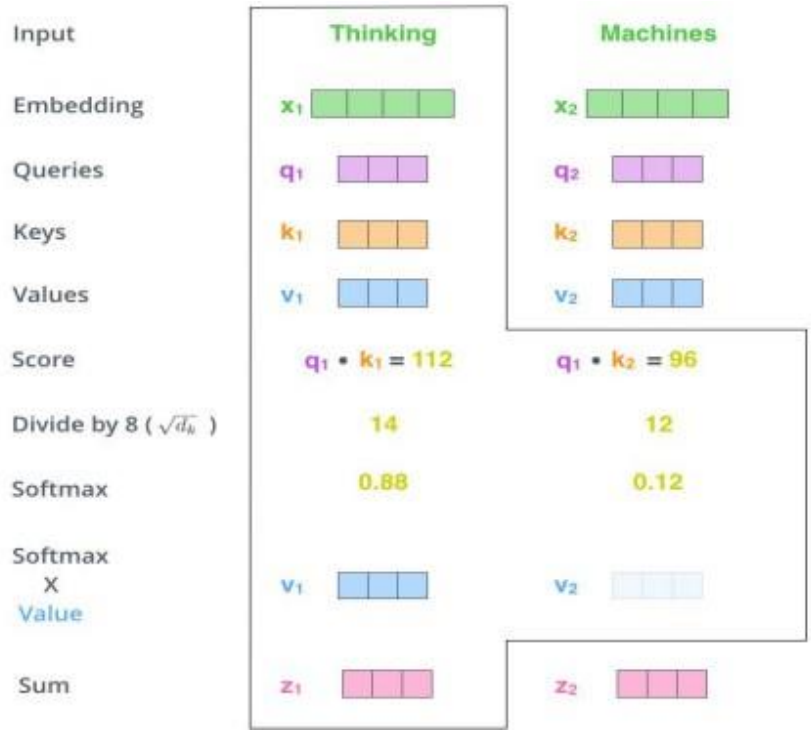
Η λέξη it αναφέρεται σε διαφορετικές οντότητες και σε διαφορετικά συμφραζόμενα, όπως φαίνεται από το προηγούμενο σχήμα [36].

Πριν αρχίσουμε να ασχολούμαστε με τους μηχανισμούς προσοχής του dot-product, ας δούμε με μεγαλύτερη λεπτομέρεια το πώς λειτουργεί ο μηχανισμός αυτοπροσοχής. Αρχικά, για κάθε word embedding που εισάγεται δημιουργούνται τρία διανύσματα. Έτσι, για κάθε λέξη, δημιουργείται ένα διάνυσμα ερώτησης (query vector), ένα διάνυσμα κλειδί (key vector) και ένα διάνυσμα τιμής (value vector). Αυτά τα διανύσματα προκύπτουν από τον πολλαπλασιασμό των word embeddings με τρεις εκπαιδεύσιμους πίνακες [36,22]. Μια σχηματική απεικόνιση της παραπάνω διαδικασίας φαίνεται στο παρακάτω σχήμα:



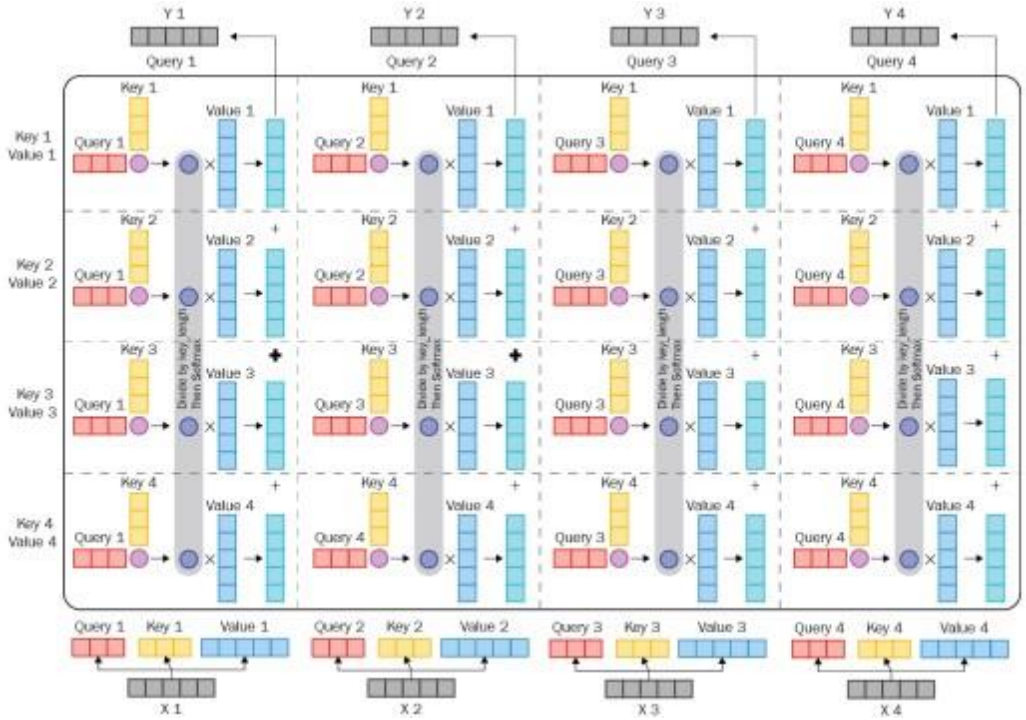
Σχήμα 22: Διανύσματα που δημιουργούνται για κάθε word embedding (Πηγή [22])

Στην συνέχεια, υπολογίζεται ένα «σκορ» για το κάθε word embedding. Αυτό υπολογίζεται από την τετραγωνική ρίζα του γινομένου του διανύσματος ερώτησης (query vector) με το διάνυσμα κλειδί (key vector) της αντίστοιχης λέξης που βαθμολογούμε. Κατόπιν, πραγματοποιείται διαίρεση των βαθμολογιών με την τετραγωνική ρίζα της διάστασης των διανυσμάτων κλειδιών. Αυτό οδηγεί σε πιο σταθερές κλίσεις. Ύστερα, το αποτέλεσμα περνάει από μιας λειτουργία softmax, η οποία καθορίζει σε ποιο βαθμό θα δίνεται προσοχή σε κάθε λέξη. Το επόμενο βήμα είναι ο πολλαπλασιασμός κάθε διανύσματος τιμής με την τιμή που έχει προκύψει από την softmax. Τέλος, γίνεται άθροισμα των σταθμισμένων τιμών των διανυσμάτων. Αυτό παράγει την έξοδο του στρώματος αυτοπροσοχής. Το διάνυσμα που προκύπτει είναι αυτό που μπορούμε να στείλουμε στο νευρωνικό δίκτυο με πρόσθια τροφοδότηση [22]. Μια σχηματική απεικόνιση των βημάτων αυτών δίνεται παρακάτω.



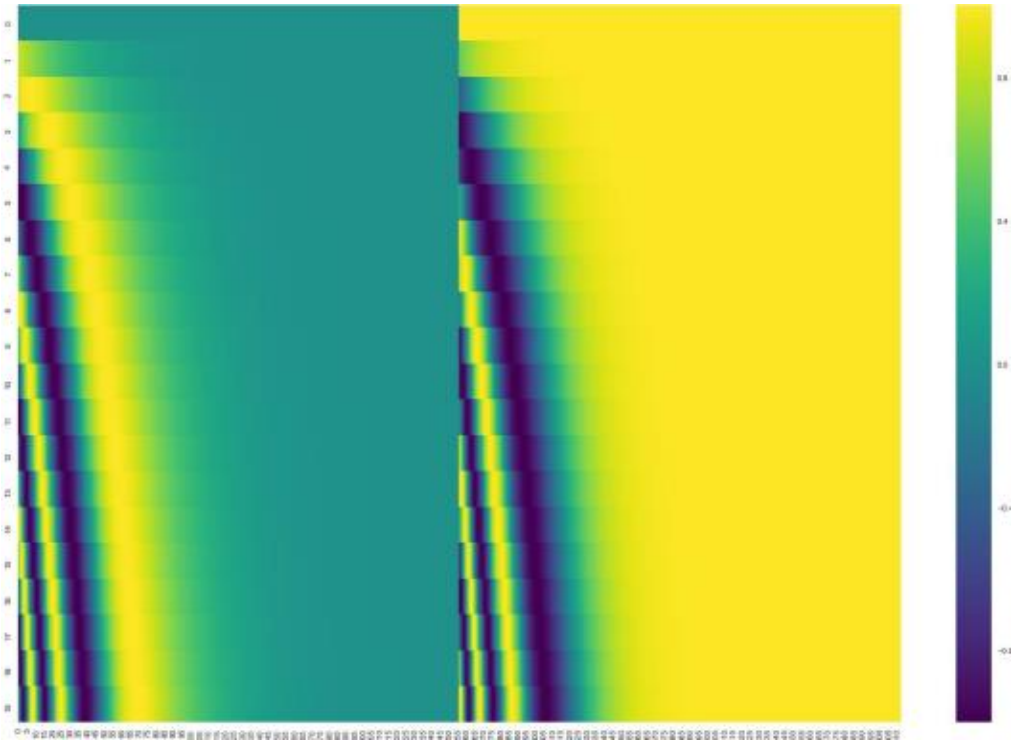
Σχήμα 23: Βήματα μηχανισμού αυτοπροσοχής (Πηγή [22])

Κάτι σημαντικό που επιτυγχάνεται με τη χρήση μιας αρχιτεκτονικής Transformer είναι ο παραλληλισμός. Τα συμβατικά ανατροφοδοτούμενα μοντέλα ακολουθιών δεν έχουν τέτοιες δυνατότητες επειδή επεξεργάζονται την είσοδο ξεχωριστά για κάθε σύμβολο (token by token). Από την άλλη πλευρά, τα επίπεδα με πρόσθια τροφοδότηση (feedforward) επιταχύνονται λίγο περισσότερο επειδή ο πολλαπλασιασμός ενός πίνακα είναι πολύ πιο γρήγορος από αυτόν μιας ανατροφοδοτούμενης μονάδας. Οι στοίβες των επιπέδων προσοχής πολλαπλών κεφαλών (Stacks of multi-head attention layers) αποκτούν καλύτερη κατανόηση των σύνθετων προτάσεων. Ένα οπτικό παράδειγμα ενός μηχανισμού προσοχής πολλαπλών κεφαλών παρουσιάζεται στο ακόλουθο σχήμα [36].



Σχήμα 24: Μηχανισμός προσοχής πολλαπλών κεφαλών (Πηγή [36])

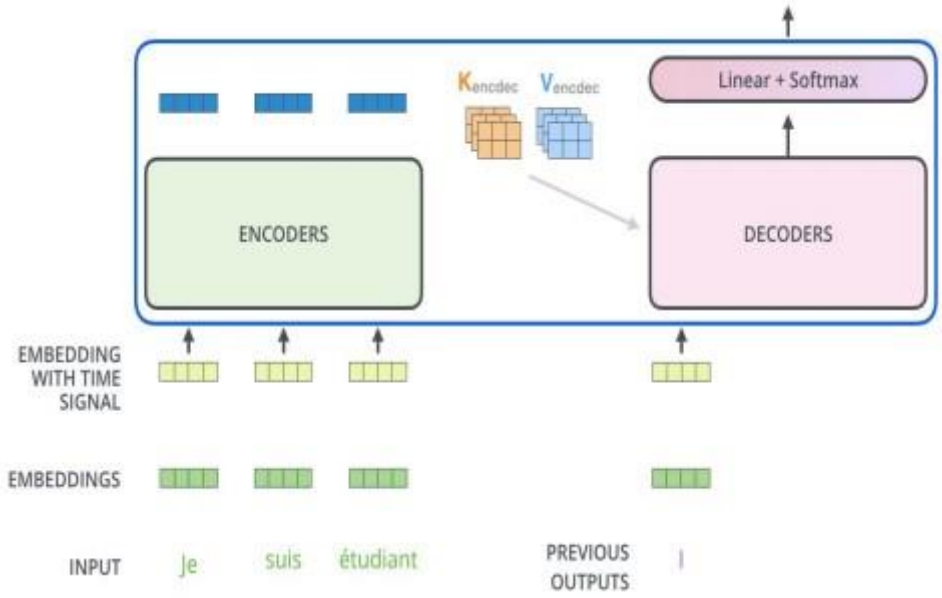
Κάτι το οποίο λείπει είναι ένας τρόπος να ληφθεί υπόψη η σειρά των λέξεων στην ακολουθία εισόδου. Για να δώσουμε στο μοντέλο μια αίσθηση της σειράς των λέξεων, προσθέτουμε διανύσματα κωδικοποίηση θέσης οι τιμές των οποίων ακολουθούν ένα συγκεκριμένο μοτίβο. Ένα πραγματικό παράδειγμα κωδικοποίησης θέσης για 20 λέξεις (γραμμές) με μέγεθος ενσωμάτωσης 512 (στήλες) φαίνεται στο παρακάτω σχήμα. Μπορούμε να δούμε ότι εμφανίζεται χωρισμένο στη μέση κατά μήκος του κέντρου. Αυτό οφείλεται στο γεγονός ότι οι τιμές του αριστερού μισού παράγονται από μια συνάρτηση ημιτόνου, ενώ το δεξιό μισό παράγεται από μια συνάρτηση συνημίτονου. Στη συνέχεια συνενώνονται για να σχηματίσουν κάθε ένα από τα διανύσματα κωδικοποίησης θέσης [22].



Σχήμα 25: Απεικόνιση κωδικοποίησης θέσης (Πηγή [22])

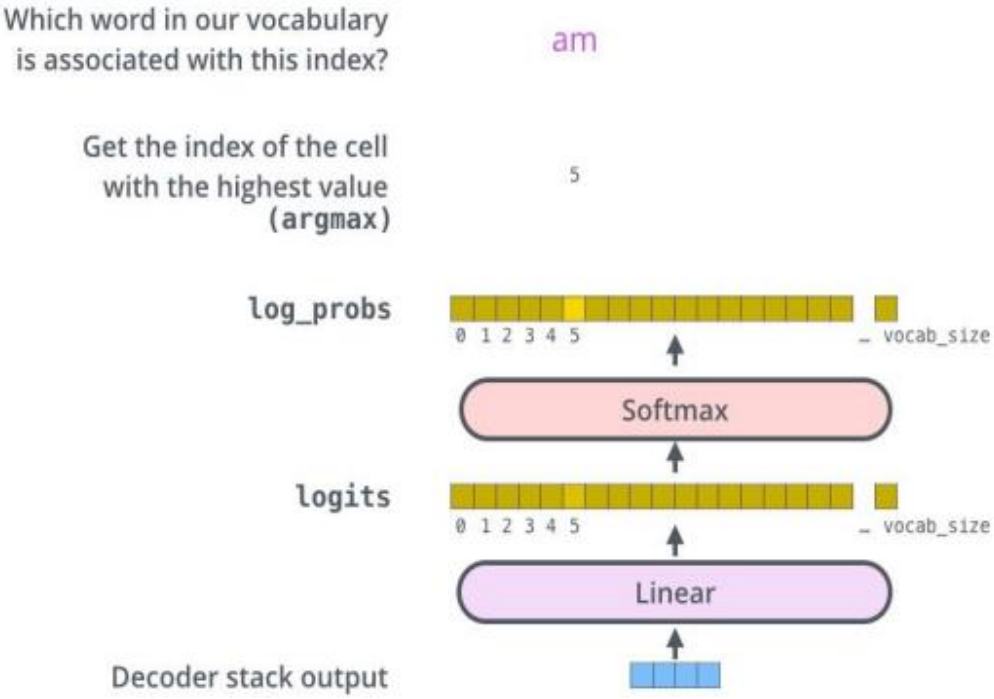
Στην πλευρά του αποκωδικοποιητή του μηχανισμού προσοχής, χρησιμοποιείται μια πολύ παρόμοια μέθοδος με αυτή του κωδικοποιητή με μικρές τροποποιήσεις [36]. Ο κωδικοποιητής ξεκινά με την επεξεργασία της ακολουθίας εισόδου. Η έξοδος του κορυφαίου κωδικοποιητή μετατρέπεται στη συνέχεια σε ένα σύνολο πινάκων προσοχής K (key) και V (value). Αυτοί οι πίνακες πρόκειται να χρησιμοποιηθούν από κάθε αποκωδικοποιητή, γεγονός που βοηθά τον αποκωδικοποιητή να εστιάσει στα κατάλληλα σημεία της ακολουθίας εισόδου. Στα επόμενα βήματα επαναλαμβάνεται η διαδικασία έως ότου επιτευχθεί ένα <EOS> token. Η έξοδος κάθε βήματος τροφοδοτείται στον κάτω αποκωδικοποιητή στο επόμενο χρονικό βήμα. Και όπως ακριβώς γίνεται με τις εισόδους του κωδικοποιητή, έτσι και στον αποκωδικοποιητή ενσωματώνουμε και προσθέτουμε κωδικοποίηση θέσης στις εισόδους του για να υποδείξουμε τη θέση κάθε λέξης [22].

Στον αποκωδικοποιητή, το επίπεδο αυτοπροσοχής επιτρέπεται να παρακολουθεί μόνο τις προηγούμενες θέσεις στην ακολουθία εξόδου (αποκρύπτοντας τις μελλοντικές θέσεις πριν από το βήμα softmax στον υπολογισμό της αυτοπροσοχής). Το επίπεδο «Encoder-Decoder Attention» λειτουργεί ακριβώς όπως η αυτοπροσοχή πολλαπλών κεφαλών, με τη διαφορά ότι δημιουργεί τον πίνακα ερωτημάτων (Queries) του από το στρώμα που βρίσκεται από κάτω του και παίρνει τον πίνακα Κλειδιών (Keys) και Αξιών (Values) από την έξοδο της στοίβας του κωδικοποιητή. Η διαδικασία που περιεγράφηκε απεικονίζεται στο παρακάτω σχήμα.



Σχήμα 26: Αποκωδικοποιητής (Πηγή [22])

Το γραμμικό επίπεδο (Linear layer) είναι ένα απλό πλήρως συνδεδεμένο νευρωνικό δίκτυο που προβάλλει το διάνυσμα που παράγεται από τη στοίβα των αποκωδικοποιητών σε ένα διάνυσμα logit (dim=vocal_size). Στη συνέχεια, το στρώμα softmax μετατρέπει το διάνυσμα logit σε πιθανότητες. Το κελί με την υψηλότερη πιθανότητα επιλέγεται και η λέξη που σχετίζεται με αυτό παράγεται ως έξοδος για αυτό το χρονικό βήμα [22].

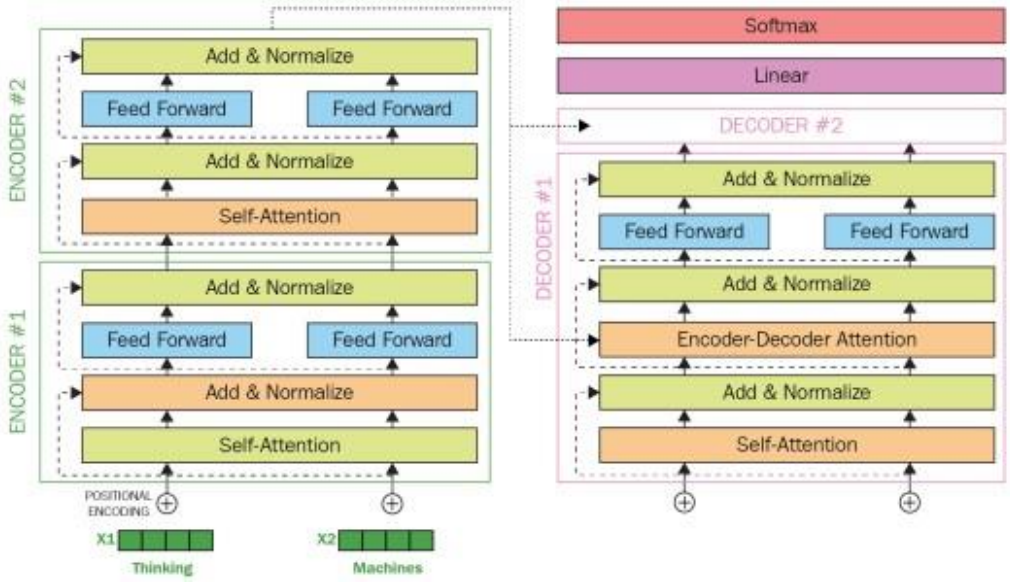


Σχήμα 27: Επίπεδα linear και softmax του αποκωδικοποιητή (Πηγή [22])

Αυτή η αρχιτεκτονική (όπως βλέπουμε και στο παρακάτω σχήμα) έχει εισόδους και εξόδους. Οι είσοδοι χρησιμεύουν τόσο στην εκπαίδευση όσο και στην εξαγωγή συμπερασμάτων, ενώ οι έξοδοι χρησιμοποιούνται μόνο στην εκπαίδευση και στην εξαγωγή συμπερασμάτων, τα οποία παράγονται από το μοντέλο. Ο λόγος για τον οποίο δεν χρησιμοποιούμε τις προβλέψεις του μοντέλου στην εξαγωγή συμπερασμάτων είναι για να αποτρέψουμε το μοντέλο από τα λάθη. Παραδείγματος χάριν, ας υποθέσουμε ότι έχουμε ένα μοντέλο νευρωνικής μετάφρασης που προσπαθεί να μεταφράσει μια πρόταση από τα αγγλικά στα γαλλικά. Σε κάθε βήμα, κάνει μια

πρόβλεψη για μια λέξη και χρησιμοποιεί αυτή την προβλεπόμενη λέξη για να προβλέψει την επόμενη. Αν όμως κάνει λάθος σε κάποιο βήμα, όλες οι επόμενες προβλέψεις θα είναι επίσης λανθασμένες. Για να αποτρέψουμε το μοντέλο από το να κάνει τέτοιο λάθος, παρέχουμε εμείς τις σωστές λέξεις [36].

Ένα οπτικό παράδειγμα ενός μοντέλου Transformer δίνεται στο ακόλουθο σχήμα όπου φαίνεται ένα μοντέλο με δύο κωδικοποιητές και δύο επίπεδα αποκωδικοποιητών. Το Add & Normalize από αυτό το σχήμα προσθέτει και κανονικοποιεί την είσοδο που λαμβάνει από το Feedforward επίπεδο:



Σχήμα 28: Μοντέλο μετασχηματιστή (Πηγή [36])

Μια άλλη σημαντική βελτίωση που χρησιμοποιείται από μια αρχιτεκτονική βασισμένη σε αυτή του transformer βασίζεται σε ένα σχήμα συμπίεσης κειμένου για την αποφυγή tokens που δεν έχουν ειδωθεί στην πλευρά της εισόδου. Αυτή η προσέγγιση, η οποία πραγματοποιείται με τη χρήση διαφορετικών μεθόδων, όπως η κωδικοποίηση ζεύγους byte (byte-pair encoding) και η κωδικοποίηση μέρους μια πρότασης (sentence-piece encoding), βελτιώνει τις επιδόσεις ενός transformer στην αντιμετώπιση αθέατων tokens. Καθοδηγεί επίσης το μοντέλο όταν συναντά tokens με παρόμοια μορφή. Τέτοια tokens ήταν αθέατα στο παρελθόν και σπάνια χρησιμοποιούνταν στην εκπαίδευση, όμως παρόλα αυτά, μπορεί να εμφανιστεί ένα συμπέρασμα. Σε ορισμένες περιπτώσεις, κάποια από αυτά λαμβάνονται υπόψιν στην εκπαίδευση, ιδιαίτερα στην περίπτωση πλούσιων γλωσσών όπως η Τουρκική, η Γερμανική, η Τσεχική και η Λετονική [36].

Τα μοντέλα που βασίζονται σε transformers έχουν αρκετά κοινά χαρακτηριστικά, για παράδειγμα, όλα βασίζονται σε αυτή την αρχική αρχιτεκτονική με διαφορές ως προς τα βήματα που χρησιμοποιούν και δεν χρησιμοποιούν. Σε ορισμένες περιπτώσεις γίνονται μικρές διαφοροποιήσεις, όπως για παράδειγμα, πραγματοποιούνται βελτιώσεις στον μηχανισμό προσοχής πολλαπλών κεφαλών.

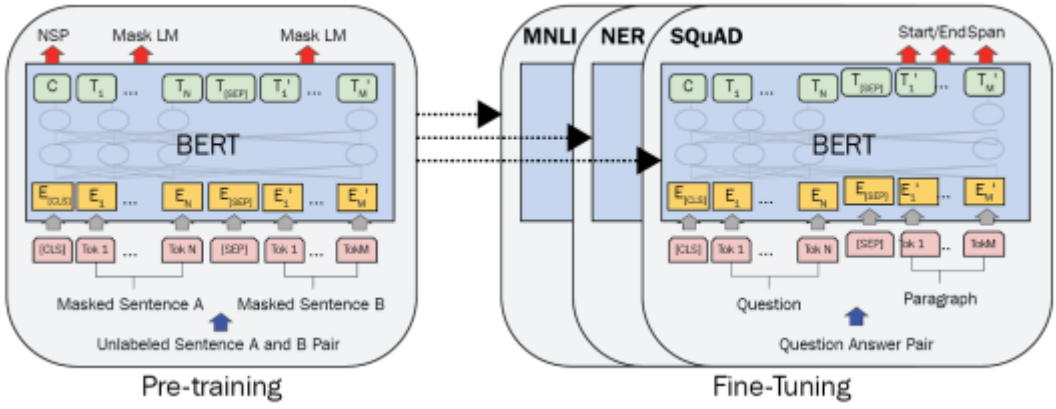
Η μεταβιβαστική μάθηση (transfer learning) είναι ένας τομέας της Τεχνητής Νοημοσύνης και της μηχανικής μάθησης που στοχεύει στο να καταστήσει τα μοντέλα επαναχρησιμοποιήσιμα για διαφορετικές εργασίες, για παράδειγμα, ένα μοντέλο που εκπαιδεύεται σε μια δεδομένη εργασία, είναι επαναχρησιμοποιήσιμο (με μικρές ρυθμίσεις) σε μια διαφορετική εργασία. Στον τομέα της επεξεργασίας φυσικής γλώσσας αυτό είναι εφικτό με τη χρήση αρχιτεκτονικών που μοιάζουν με transformers και μπορούν να αποτυπώσουν την κατανόηση της γλώσσας μέσω της γλωσσικής μοντελοποίησης. Τέτοια μοντέλα ονομάζονται γλωσσικά μοντέλα (language models) και παρέχουν ένα μοντέλο για τη γλώσσα στην οποία έχουν εκπαιδευτεί. Η μεταβιβαστική μάθηση δεν είναι μια νέα τεχνική και έχει χρησιμοποιηθεί σε διάφορους τομείς, όπως η όραση υπολογιστών [36].

ΤΟ ΜΟΝΤΕΛΟ BERT

Η μεταβιβαστική μάθηση στην επεξεργασία φυσικής γλώσσας με τη χρήση μοντέλων transformer είναι επίσης δυνατή, επειδή αυτά τα μοντέλα μπορούν να μάθουν μια γλώσσα από μόνα τους χωρίς να έχουν επισημανθεί δεδομένα. Η μοντελοποίηση γλώσσας είναι μια εργασία που χρησιμοποιείται για την εκπαίδευση των βαρών τα οποία μπορούν να μεταφερθούν (transferable weights) για διάφορα προβλήματα. Το masked language modeling είναι μία από τις μεθόδους που χρησιμοποιούνται ώστε να μάθει το μοντέλο μια γλώσσα από μόνο του. Με δεδομένη μια πιθανότητα, κάθε λέξη καλύπτεται (is masked) και αντικαθίσταται με ένα ειδικό token, όπως το [MASK]. Το γλωσσικό μοντέλο (στην περίπτωσή μας ένα μοντέλο βασισμένο στην αρχιτεκτονική Transformer) πρέπει να προβλέψει τις masked λέξεις. Δίνεται δηλαδή μια ολόκληρη πρόταση και η έξοδος του μοντέλου πρέπει να είναι η ίδια πρόταση με συμπληρωμένες τις masked λέξεις.

Ένα από τα πρώτα μοντέλα που χρησιμοποίησαν την αρχιτεκτονική Transformer για γλωσσική μοντελοποίηση είναι το BERT, το οποίο βασίζεται στο τμήμα του κωδικοποιητή της αρχιτεκτονικής Transformer. Το masked language modeling επιτυγχάνεται από το BERT με τη χρήση της ίδιας μεθόδου που περιγράφεται πριν και μετά την εκπαίδευση ενός γλωσσικού μοντέλου. Το BERT είναι ένα μεταβιβάσιμο γλωσσικό μοντέλο για διάφορες εργασίες επεξεργασίας φυσικής γλώσσας, όπως η ταξινόμηση συμβόλων, η ταξινόμηση ακολουθιών ή ακόμη και η απάντηση ερωτήσεων.

Κάθε μία από αυτές τις εργασίες που προαναφέρθηκαν είναι μια εργασία βελτιστοποίησης για το BERT αφού έχει ήδη εκπαιδευτεί ένα γλωσσικό μοντέλο. Το BERT είναι περισσότερο γνωστό για τα χαρακτηριστικά του που βασίζονται στο βασικό μοντέλο του κωδικοποιητή της αρχιτεκτονικής Transformer και, μεταβάλλοντας κάποια από αυτά τα χαρακτηριστικά, προτείνονται διάφορες εκδοχές του. Η ενσωμάτωση με βάση τα συμφραζόμενα (contextual embedding) επιτρέπει στο μοντέλο να έχει τη σωστή σημασία κάθε λέξης με βάση το πλαίσιο στο οποίο δίνεται - για παράδειγμα, η λέξη Cold μπορεί να έχει διαφορετική σημασία σε δύο διαφορετικές προτάσεις όπως Cold-hearted killer και Cold weather. Ο αριθμός των επιπέδων του κωδικοποιητή, το μέγεθος της εισόδου, το μέγεθος της ενσωμάτωσης εξόδου (output embedding) και ο αριθμός των μηχανισμών προσοχής πολλαπλών κεφαλών είναι τα βασικά του χαρακτηριστικά, όπως απεικονίζεται στο ακόλουθο σχήμα:



Σχήμα 29: Διαδικασίες προ-εκπαίδευσης και τελικής ρύθμισης για το BERT (Πηγή [36])

Η φάση προ-εκπαίδευσης αποτελείται επίσης από έναν άλλο στόχο, γνωστό ως πρόβλεψη της επόμενης πρότασης. Όπως γνωρίζουμε, κάθε έγγραφο αποτελείται από προτάσεις που διαδέχονται η μία την άλλη, και ένα άλλο σημαντικό μέρος της εκπαίδευσης για να κατανοήσει ένα μοντέλο τη γλώσσα είναι να κατανοήσει τις σχέσεις των προτάσεων μεταξύ τους, δηλαδή αν σχετίζονται ή όχι. Για την επίτευξη αυτών των καθηκόντων, το BERT εισήγαγε ειδικά tokens όπως τα [CLS] και [SEP]. Ένα token [CLS] είναι ένα token που χρησιμοποιείται ως token έναρξης για όλες τις εργασίες και περιέχει όλες τις πληροφορίες σχετικά με την πρόταση.

Είναι επίσης χρήσιμο για την αξιολόγηση του νοήματος μιας πρότασης ή για τη κατανόηση της σημασιολογίας της - για παράδειγμα, όταν χρησιμοποιείται ένα μοντέλο Siamese BERT, η σύγκριση αυτών των δύο token [CLS] για διαφορετικές προτάσεις με ένα μέτρο όπως η ομοιότητα συνημίτονου είναι πολύ χρήσιμη. Από την άλλη πλευρά, το [SEP] χρησιμοποιείται για τη διάκριση μεταξύ δύο προτάσεων και χρησιμοποιείται μόνο για το διαχωρισμό δύο προτάσεων. Μετά την προ-εκπαίδευση, εάν κάποιος στοχεύει να προβεί στην τελική ρύθμιση του BERT σε μια εργασία

ταξινόμησης ακολουθιών, όπως η ανάλυση συναισθήματος, η οποία είναι μια εργασία ταξινόμησης ακολουθιών, θα χρησιμοποιήσει έναν ταξινομητή πάνω στην ενσωμάτωση εξόδου του [CLS]. Είναι επίσης αξιοσημείωτο ότι όλα τα μοντέλα μεταβιβαστικής μάθησης μπορούν να παγώσουν κατά τη διάρκεια της τελικής ρύθμισης ή να απελευθερωθούν (όταν παγώνουν βλέπουμε όλα τα βάρη και τις προκαταλήψεις μέσα στο μοντέλο ως σταθερές και σταματάμε την εκπαίδευση σε αυτά). Στο παράδειγμα της ανάλυσης συναισθήματος, θα εκπαιδευτεί μόνο ο ταξινομητής και όχι το μοντέλο, εάν είναι παγωμένο [36].

ΚΕΦΑΛΑΙΟ 3

ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΕ

Το σύνολο δεδομένων το οποίο χρησιμοποιήθηκε για ανάλυση συναισθήματος σε αυτήν την εργασία είναι το “GoEmotions: A Dataset of Fine-Grained Emotions”. Το GoEmotions ένα σύνολο δεδομένων το οποίο αποτελείται από ανθρώπινους σχολιασμούς 58k σχολίων του Reddit που εξάγονται από δημοφιλή αγγλόφωνα subreddits και επισημαίνονται με 27 κατηγορίες συναισθήματος. Αποτελεί το μεγαλύτερο σύνολο δεδομένων συναισθήματος στην αγγλική γλώσσα μέχρι σήμερα. Σε αντίθεση με τα έξι βασικά συναισθήματα (χαρά, λύπη, θυμός, φόβος, έκπληξη και αηδία), τα οποία περιλαμβάνουν μόνο ένα θετικό συναίσθημα (χαρά), η ταξινόμηση του συνόλου αυτού περιλαμβάνει 12 θετικές, 11 αρνητικές, 4 διφορούμενες κατηγορίες συναισθήματος και 1 ουδέτερη, καθιστώντας την, ευρέως κατάλληλη για εργασίες κατανόησης συνομιλιών που απαιτούν λεπτή διαφοροποίηση μεταξύ των εκφράσεων συναισθήματος.

Στόχος των δημιουργών ήταν να φτιαχτεί ένα μεγάλο σύνολο δεδομένων, επικεντρωμένο σε δεδομένα συνομιλιών, όπου το συναίσθημα αποτελεί κρίσιμο στοιχείο της επικοινωνίας. Επειδή η πλατφόρμα Reddit προσφέρει έναν μεγάλο, δημόσια διαθέσιμο όγκο περιεχομένου που περιλαμβάνει συνομιλίες μεταξύ χρηστών, αποτελεί πολύτιμο πόρο για την ανάλυση συναισθήματος. Έτσι, κατασκευάστηκε το GoEmotions χρησιμοποιώντας σχόλια του Reddit από το 2005 (έναρξη του Reddit) έως τον Ιανουάριο του 2019, τα οποία προέρχονται από subreddits με τουλάχιστον 10k σχόλια, εξαιρουμένων αυτών που έχουν διαγραφεί και των μη αγγλικών σχολίων.

Για να μπορέσουν να δημιουργήσουν ευρέως αντιπροσωπευτικά μοντέλα συναισθήματος, εφάρμοσαν μέτρα για τον έλεγχο των δεδομένων ώστε να διασφαλίσουν ότι το σύνολο δεδομένων δεν ενισχύει γλωσσικές προκαταλήψεις. Αυτό ήταν ιδιαίτερα σημαντικό καθώς οι χρήστες του Reddit συνήθως είναι νεαροί άνδρες, κάτι που δεν αντικατοπτρίζει έναν παγκοσμίως ποικιλόμορφο πληθυσμό. Η πλατφόρμα έχει επίσης μια κλίση προς την τοξική, προσβλητική γλώσσα. Για να αντιμετωπιστούν αυτές οι ανησυχίες, εντοπίστηκαν τα επιβλαβή σχόλια χρησιμοποιώντας προκαθορισμένους όρους για προσβλητικό/ενήλικο και χυδαίο περιεχόμενο, καθώς και για την ταυτότητα και τη θρησκεία, οι οποίοι χρησιμοποιήθηκαν για το φιλτράρισμα και τη συγκάλυψη των δεδομένων. Επεξεργάστηκαν επιπλέον τα δεδομένα ώστε να μειωθούν οι βωμολοχίες, να περιοριστεί το μήκος του κειμένου και να εξισορροπηθούν τα συναισθήματα τα οποία αντιπροσωπεύονται. Για την αποφυγή της υπερεκπροσώπηση των δημοφιλών subreddit και για να μπορέσει να διασφαλιστεί ότι τα σχόλια αντικατοπτρίζουν και τα λιγότερο ενεργά subreddit, εξισορροπήθηκαν επίσης τα δεδομένα μεταξύ των κοινοτήτων subreddit.

Δημιουργήθηκε μια ταξινόμηση που επιδιώκει να μεγιστοποιήσει τρεις στόχους: (1) να παρέχει τη μεγαλύτερη δυνατή κάλυψη των συναισθημάτων που εκφράζονται στα δεδομένα Reddit, (2) να παρέχει τη μεγαλύτερη δυνατή κάλυψη των τύπων συναισθηματικών εκφράσεων και (3) να περιορίσει τον συνολικό αριθμό των συναισθημάτων και την επικάλυψη μεταξύ τους. Μια τέτοια ταξινόμηση επιτρέπει την κατανόηση των συναισθημάτων με βάση τα δεδομένα σε λεπτομερές επίπεδο, ενώ παράλληλα αντιμετωπίζει την πιθανή σπανιότητα των δεδομένων για ορισμένα συναισθήματα.

Ο καθορισμός της ταξινόμησης ήταν μια επαναληπτική διαδικασία για τον καθορισμό και την τελειοποίηση των κατηγοριών των ετικετών συναισθημάτων. Κατά τη διάρκεια των σταδίων επισήμανσης των δεδομένων, εξετάστηκαν συνολικά 56 κατηγορίες συναισθημάτων. Από αυτό το δείγμα, εντοπίστηκαν και αφαιρέθηκαν τα συναισθήματα που επιλέγονταν ελάχιστα από τους βαθμολογητές και είχαν χαμηλή συμφωνία μεταξύ των βαθμολογητών λόγω ομοιότητας με άλλα συναισθήματα ή επειδή ήταν δύσκολο να εντοπιστούν στο κείμενο. Προστέθηκαν επίσης συναισθήματα που προτείνονταν συχνά από τους βαθμολογητές και εκπροσωπούνταν καλά στα δεδομένα μέσω αυτών. Τέλος, βελτιώθηκαν τα ονόματα των κατηγοριών συναισθημάτων για να μεγιστοποιηθεί η ερμηνευσιμότητα, οδηγώντας σε υψηλή συμφωνία μεταξύ των βαθμολογητών, με το 94% των παραδειγμάτων να έχουν τουλάχιστον δύο βαθμολογητές που συμφωνούν σε τουλάχιστον μια ετικέτα συναισθήματος.

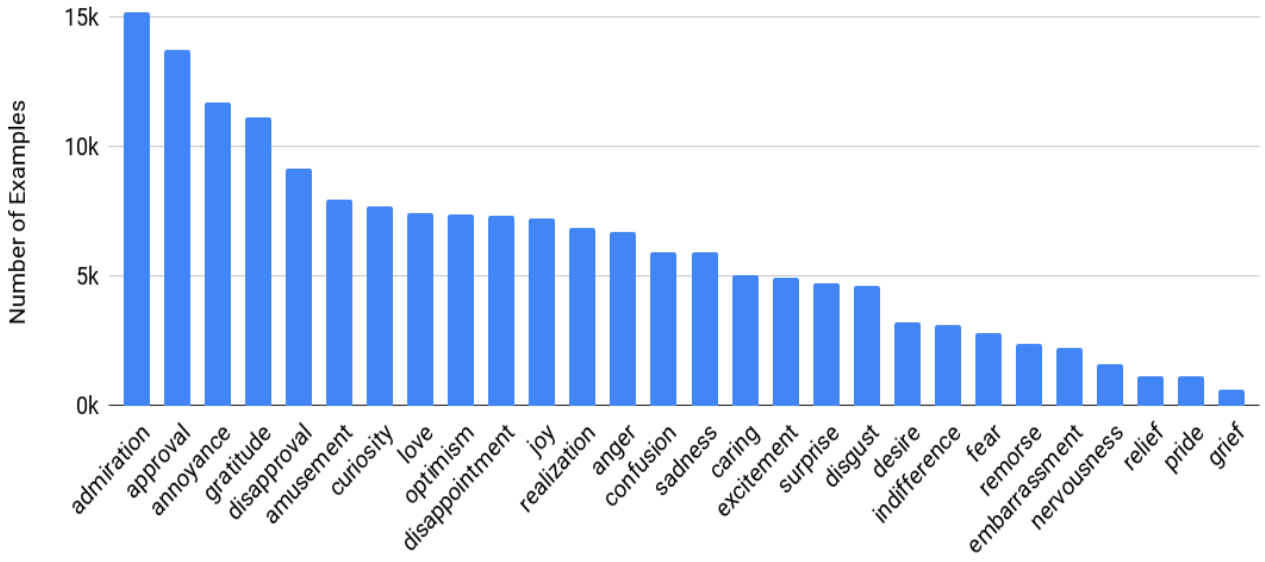
Το δημοσιευμένο σύνολο δεδομένων GoEmotions περιλαμβάνει την ταξινόμηση που παρουσιάζεται παρακάτω και συλλέχθηκε μέσα από τον τελικό γύρο επισήμανσης των δεδομένων, όπου τόσο η ταξινόμηση όσο και τα πρότυπα αξιολόγησης ήταν προκαθορισμένα και σταθερά.

Positive		Negative		Ambiguous
admiration 👋	joy 😃	anger 😡	grief 😢	confusion 😕
amusement 😂	love ❤️	annoyance 😒	nervousness 😬	curiosity 🤔
approval 👍	optimism 🤞	disappointment	remorse 😔	realization 💡
caring 🤗	pride 😌	disapproval 👎	sadness 😞	surprise 😲
desire 😍	relief 😅	disgust 🤮		
excitement 🤩		embarrassment 😳		
gratitude 🙏		fear 😨		

Πίνακας 1: Ταξινόμηση του GoEmotions, περιλαμβάνει 28 κατηγορίες συναισθημάτων συμπεριλαμβανομένης της «ουδέτερης» (Πηγή [10])

Όπως βλέπουμε και στον πίνακα, στην κατηγορία των θετικών συναισθημάτων έχουν ενταχθεί τα συναισθήματα θαυμασμός, διασκέδαση, έγκριση, στοργικότητα, επιθυμία, ενθουσιασμός, ευγνωμοσύνη, ευτυχία, αγάπη, αισιοδοξία, περηφάνια, ανακούφιση. Στα αρνητικά συναισθήματα υπάρχουν ο θυμός, η ενόχληση, η απογοήτευση, η αποδοκιμασία, η αηδία, η ντροπή, ο φόβος, η θλίψη, το άγχος, οι ενοχές και η στενοχώρια. Στην κατηγορία των αμφίβολων συναισθημάτων ανήκουν η σύγχυση, η περιέργεια η συνειδητοποίηση και η έκπληξη.

Τα συναισθήματα δεν κατανέμονται ομοιόμορφα στο σύνολο δεδομένων GoEmotions. Είναι σημαντικό ότι η υψηλή συχνότητα των θετικών συναισθημάτων ενισχύει το κίνητρο για μια πιο ποικιλόμορφη ταξινόμηση συναισθήματος από αυτή που προσφέρουν τα κανονικά έξι βασικά συναισθήματα.



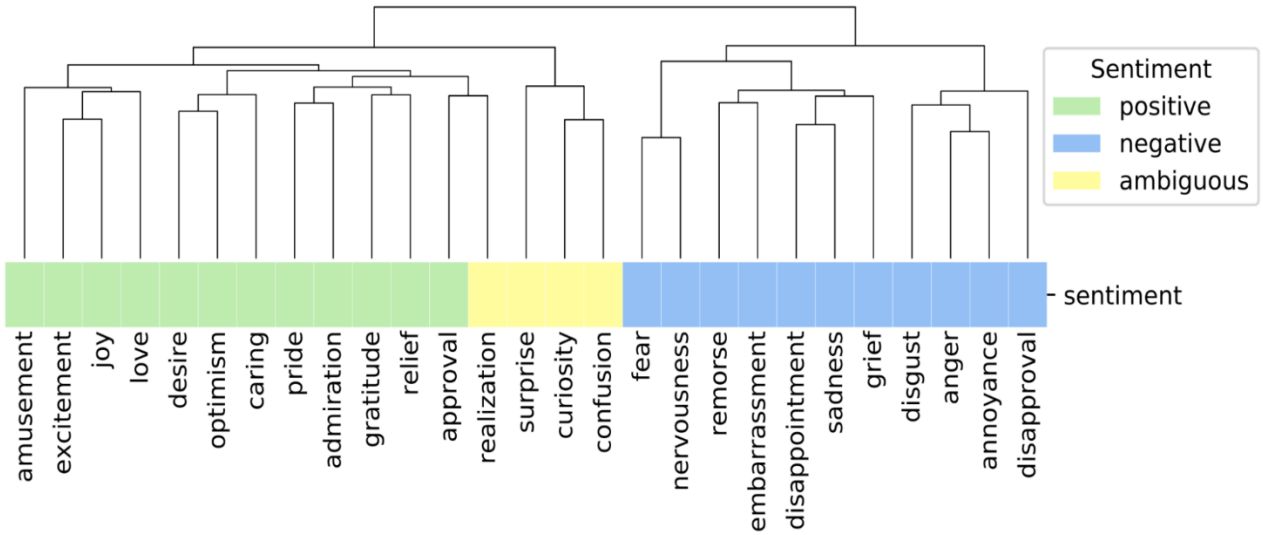
Σχήμα 30: Διάγραμμα συχνότητας εμφάνισης συναισθημάτων (Πηγή[10])

Για να επικυρώσουν οι δημιουργοί ότι οι ταξινομικές τους επιλογές ταιριάζουν με τα υποκείμενα δεδομένα, διεξήγαγαν ανάλυση κύριων διατηρούμενων συνιστωσών (principal preserved component analysis - PPCA), μια μέθοδο που χρησιμοποιείται για τη σύγκριση δύο συνόλων δεδομένων με την εξαγωγή γραμμικών συνδυασμών κρίσεων συναισθημάτων που

παρουσιάζουν τη μεγαλύτερη κοινή μεταβλητότητα σε δύο σύνολα βαθμολογητών. Ως εκ τούτου, μας βοηθά να αποκαλύψουμε τις διαστάσεις των συναισθημάτων που έχουν υψηλή συμφωνία μεταξύ των βαθμολογητών. Η ανάλυση αυτή χρησιμοποιήθηκε στο παρελθόν για την κατανόηση των κύριων διαστάσεων της αναγνώρισης συναισθημάτων σε βίντεο και ομιλία και τη χρησιμοποίησαν και στην προκειμένη περίπτωση για την κατανόηση των κύριων διαστάσεων των συναισθημάτων σε κείμενο.

Αυτό που διαπιστώθηκε είναι ότι κάθε συνιστώσα είναι σημαντική (με τιμές $p < 1,5e-6$ για όλες τις πτυχές), γεγονός που υποδηλώνει ότι κάθε συναίσθημα αποτυπώνει ένα μοναδικό μέρος των δεδομένων. Αυτό δεν είναι ασήμαντο, δεδομένου ότι σε προηγούμενες εργασίες σχετικά με την αναγνώριση συναισθημάτων στον λόγο, μόνο 12 από τις 30 πτυχές των συναισθημάτων βρέθηκαν σημαντικές.

Εξετάστηκε επίσης η ομαδοποίηση των καθορισμένων συναισθημάτων με βάση τις συσχετίσεις μεταξύ των κρίσεων των βαθμολογητών. Με αυτή την προσέγγιση, δύο συναισθήματα θα ομαδοποιηθούν όταν επιλέγονται συχνά από κοινού από τους βαθμολογητές. Αυτό που διαπιστώνεται είναι ότι τα συναισθήματα που σχετίζονται ως προς το γενικό συναίσθημά τους (αρνητικά, θετικά και διφορούμενα) συγκεντρώνονται μαζί. Για παράδειγμα, εάν ένας βαθμολογητής επιλέξει τον «ενθουσιασμό» ως ετικέτα για ένα συγκεκριμένο σχόλιο, είναι πιο πιθανό ένας άλλος βαθμολογητής να επιλέξει ένα συναφές συναίσθημα, όπως η «χαρά», παρά τον «φόβο». Ίσως αποτελεί έκπληξη το γεγονός ότι όλα τα διφορούμενα συναισθήματα συγκεντρώθηκαν σε ομάδες και συσχετίστηκαν πιο στενά με τα θετικά συναισθήματα.



Σχήμα 31: Συσχέτιση συναισθημάτων (Πηγή[10])

Γενικά, η βάση GoEmotions παρέχει ένα μεγάλο σύνολο δεδομένων για την πρόβλεψη συναισθήματος σε λεπτομερές επίπεδο. Η ανάλυσή που περιεγράφηκε παραπάνω καταδεικνύει την αξιοπιστία των επισημάνσεων και την υψηλή κάλυψη των συναισθημάτων που εκφράζονται στα σχόλια του Reddit. Για αυτό το λόγο το GoEmotions αποτελεί πολύτιμο πόρο για τους ερευνητές που ασχολούνται με τα συναισθήματα στο λόγο και επιτρέπει στους επαγγελματίες να δημιουργήσουν εφαρμογές με γνώμονα τα συναισθήματα [10].

ΚΕΦΑΛΑΙΟ 4

ΑΠΟΤΕΛΕΣΜΑΤΑ ΑΝΑΛΥΣΗΣ

Η εκπαίδευση των δεδομένων καθώς και ο έλεγχος των δεδομένων έγινε αξιοποιώντας της μέθοδο του τυχαίου ποσοστιαίου διαχωρισμού (random percentage split method). Σε αυτήν την μέθοδο, το 80% των δεδομένων χρησιμοποιήθηκε για εκπαίδευση και το 20% για δοκιμή.

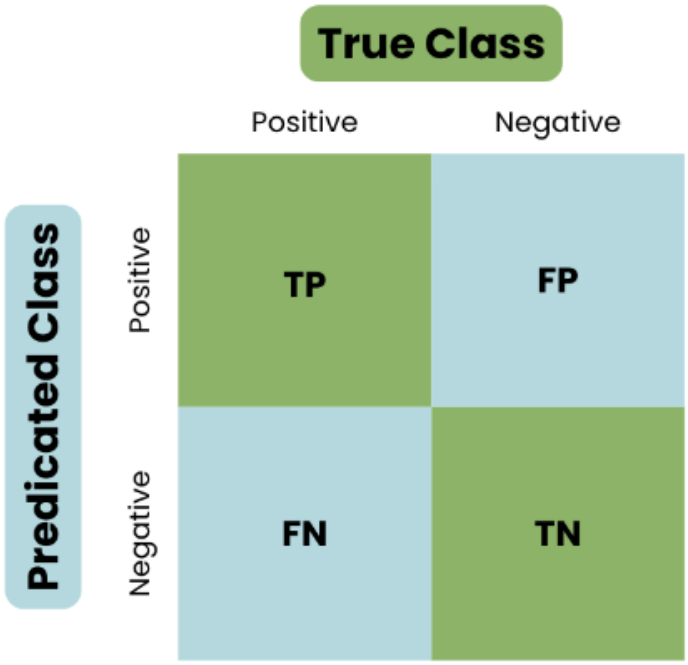
Σημαντικό ρόλο για την αξιολόγηση της αποτελεσματικότητας μιας ταξινόμησης παίζουν τα μέτρα αξιολόγησης. Τα μέτρα αυτά μπορούν να κατηγοριοποιηθούν σε τρεις τύπους, οι οποίοι είναι τα μέτρα κατωφλιού (threshold metrics), τα μέτρα πιθανότητας (probability metrics) και τα μέτρα κατάταξης (ranking metrics). Κάθε ένας από αυτούς τους τύπους μέτρων αξιολογεί τον ταξινομητή με διαφορετικούς στόχους. Στην πράξη, τα μέτρα κατωφλίου και τα μέτρα κατάταξης ήταν τα πιο συνηθισμένα μέτρα που χρησιμοποιήθηκαν από τους ερευνητές για τη μέτρηση της απόδοσης των ταξινομητών. Γενικά αυτοί οι τύποι μέτρων μπορούν να χρησιμοποιηθούν σε διαφορετικές εφαρμογές αξιολόγησης [16].

Τα μέτρα αξιολόγησης χρησιμοποιήθηκαν για την αξιολόγηση της ικανότητας γενίκευσης του εκπαιδευμένου ταξινομητή. Σε αυτή την περίπτωση, τα μέτρα αυτά χρησιμοποιούνται για τη μέτρηση και τη σύνοψη της ποιότητας του εκπαιδευμένου ταξινομητή όταν δοκιμάζεται με τα αδημοσίευτα δεδομένα. Η ακρίβεια (accuracy) ή το ποσοστό σφάλματος (error rate) είναι κάποια από τα πιο κοινά μέτρα που χρησιμοποιούνται από πολλούς ερευνητές για την αξιολόγηση της ικανότητας γενίκευσης των ταξινομητών. Μέσω της ακρίβειας, ο εκπαιδευμένος ταξινομητής αξιολογείται με βάση τη συνολική ορθότητα, η οποία φαίνεται στο σύνολο των περιπτώσεων που προβλέπονται σωστά από τον εκπαιδευμένο ταξινομητή όταν δοκιμάζεται σε δεδομένα που δεν έχει δει κατά τη διάρκεια της εκπαίδευσης [16].

Επιπλέον, τα μέτρα αξιολόγησης χρησιμοποιήθηκαν κατά τη διαδικασία επιλογής μοντέλων. Σε αυτή την περίπτωση, το έργο των μέτρων αξιολόγησης είναι να προσδιοριστεί ο καλύτερος ταξινομητής μεταξύ διαφορετικών τύπων εκπαιδευμένων ταξινομητών, οι οποίοι εστιάζουν στην καλύτερη μελλοντική απόδοση (βέλτιστο μοντέλο) όταν δοκιμάζονται σε δεδομένα που δεν ήταν διαθέσιμα κατά τη φάση της εκπαίδευσης. Ακόμη, τα συγκεκριμένα μέτρα αξιολόγησης παρείχαν τις απαραίτητες συναρτήσεις απώλειας οι οποίες χρησιμοποιήθηκαν για τον προσδιορισμό των βέλτιστων διαμορφώσεων βαρών για το κάθε νευρωνικό μοντέλο κατά τη διαδικασία της εκπαίδευσης. Για παράδειγμα, το μέτρο της ακρίβειας μπορεί να χρησιμοποιηθεί για την κατάταξη των πιθανών λύσεων και την επιλογή της καλύτερης λύσης από αυτές που παράγονται από έναν συγκεκριμένο αλγόριθμο ταξινόμησης. Μόνο η καλύτερη λύση που ταυτοποιεί το αντίστοιχο βέλτιστο μοντέλο θα χρησιμοποιηθεί για την ταξινόμηση των δεδομένων ελέγχου. Αξίζει να σημειωθεί ότι οι συναρτησιακές μορφές των μετρικών απώλειας που μπορούν να χρησιμοποιηθούν κατά τη διάρκεια της εκπαίδευσης θα πρέπει να είναι συνεχείς συναρτήσεις διαφορίσιμες σε όλο το πεδίο ορισμού τους [16].

Στα περισσότερα προβλήματα ταξινόμησης, τα μέτρα αξιολόγησης χρησιμοποιούνται και στο στάδιο της εκπαίδευσης αλλά και στο στάδιο της δοκιμής. Στο στάδιο της εκπαίδευσης, τα μέτρα αξιολόγησης χρησιμοποιούνται για τη βελτιστοποίηση του αλγορίθμου ταξινόμησης. Με άλλα λόγια, χρησιμοποιούνται για τη διάκριση και την επιλογή της βέλτιστης λύσης που μπορεί να παράγει ακριβέστερη πρόβλεψη της μελλοντικής αξιολόγησης ενός συγκεκριμένου ταξινομητή. Κατόπιν, στο στάδιο της δοκιμής, τα μέτρα αυτά χρησιμοποιούνται ως αξιολογητές για τη μέτρηση της αποτελεσματικότητας του ταξινομητή όταν δοκιμάζεται σε δεδομένα ελέγχου [16].

Για προβλήματα δυαδικής ταξινόμησης, η αξιολόγηση διάκρισης της καλύτερης (βέλτιστης) λύσης κατά την εκπαίδευση ταξινόμησης μπορεί να οριστεί με βάση τον πίνακα σύγχυσης (confusion matrix) [16]. Ο πίνακας σύγχυσης (επίσης γνωστός ως πίνακας ενδεχομένων ή πίνακας σφαλμάτων) είναι μια διάταξη πίνακα που επιτρέπει την οπτικοποίηση της απόδοσης ενός αλγορίθμου εποπτευόμενης μάθησης. Κάθε στήλη του πίνακα αντιπροσωπεύει τις τιμές σε μια προβλεπόμενη κλάση, ενώ κάθε γραμμή αντιπροσωπεύει τις τιμές σε μια πραγματική κλάση. Όλες οι σωστές προβλέψεις βρίσκονται κατά μήκος της κύριας διαγωνίου του πίνακα, έτσι ώστε τα σφάλματα να μπορούν εύκολα να απεικονιστούν από οποιεσδήποτε μη μηδενικές τιμές εκτός της κύριας διαγωνίου. Για να έχει ένας ταξινομητής καλή ακρίβεια, ιδανικά οι περισσότερες κλάσεις θα πρέπει να αντιπροσωπεύονται κατά μήκος της κύριας διαγωνίου του πίνακα σύγχυσης. Παρακάτω φαίνεται ένα παράδειγμα ενός πίνακα σύγχυσης [27].



Πίνακας 2: Παράδειγμα πίνακα σύγχυσης (Πηγή[26])

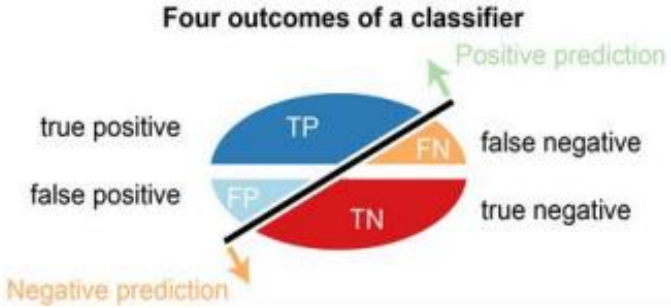
Οι εκτιμήσεις των πιθανοτήτων των μοντέλων ταξινόμησης προκύπτουν από τις εκφράσεις TP, TN, FP, FN, οι οποίες υπάρχουν στον πίνακα σύγχυσης.

TP (True Positive): Η κατηγορία δεδομένων είναι True Positive (TP) όταν προβλέπεται από το μοντέλο ένα θετικό αποτέλεσμα και η πραγματική κατηγορία είναι η ίδια.

FP (False Positive) : Η κατηγορία δεδομένων είναι False Positive (FP) όταν προβλέπεται ένα θετικό αποτέλεσμα και η πραγματική κατηγορία είναι διαφορετική. Αυτό το σενάριο είναι γνωστό ως σφάλμα τύπου 1 (Type 1 Error).

FN (False Negative): Η κατηγορία δεδομένων είναι False Negative (FN) όταν προβλέπεται αρνητικό αποτέλεσμα και η πραγματική κατηγορία είναι διαφορετική. Αυτό το σενάριο είναι γνωστό ως σφάλμα τύπου 2 (Type 2 Error).

TN (True Negative): Η κατηγορία δεδομένων στον πίνακα σύγχυσης είναι True Negative (TN) όταν προβλέπεται ένα αρνητικό αποτέλεσμα και η πραγματική κατηγορία είναι η ίδια [33]. Τα αποτελέσματα της δυαδικής ταξινόμησης παρουσιάζονται στο παρακάτω σχήμα.



Σχήμα 32: Σχηματική αναπαράσταση των τεσσάρων δυαδικών αποτελεσμάτων ενός ταξινομητή (Πηγή [33])

Ένα ακόμα αξιοσημείωτο μέτρο είναι η ακρίβεια (accuracy) η οποία μετρά την αναλογία των σωστών προβλέψεων δια του συνολικού αριθμού των περιπτώσεων που αξιολογήθηκαν [16]. Η καλύτερη τιμή της ακρίβειας είναι 1,0 και η χειρότερη είναι 0,00 [33].

Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$

Ένα άλλο μέτρο είναι και η ανάκληση (recall) η οποία μετριέται διαιρώντας τον συνολικό αριθμό των True Positive αποτελεσμάτων με τον συνολικό αριθμό των πραγματικών θετικών αποτελεσμάτων όπως φαίνεται στην παρακάτω εξίσωση [15]. Αλλιώς μπορεί να ονομάζεται και TP rate ή sensitivity.

Recall = $\frac{TP}{TP+FN}$

Εκτός από το TP rate υπάρχει και το FP rate (False Positive Rate). Το ποσοστό ψευδώς θετικών προβλέψεων υπολογίζεται ως ο αριθμός των ψευδώς θετικών προβλέψεων (FP) διαιρεμένος με τον συνολικό αριθμό των αρνητικών (TN+FP). Το καλύτερο ποσοστό ψευδώς θετικών προβλέψεων είναι 0,0 και το χειρότερο είναι 1,0 [33].

FPR = $\frac{FP}{TN+FP}$

Η αξιοπιστία (precision) χρησιμοποιείται για τη μέτρηση των θετικών μοτίβων που προβλέπονται σωστά από τα συνολικά προβλεπόμενα πρότυπα σε μια θετική κλάση [15]. Η καλύτερη ακρίβεια αυτού του μέτρου είναι το 1,0 και η χειρότερη το 0.0 [33].

Precision = $\frac{TP}{TP+FP}$

Η εξειδίκευση (specificity) ή αλλιώς True Negative rate, μετριέται από τον συνολικό αριθμό των πραγματικών αρνητικών διαιρούμενο με τον συνολικό αριθμό των πραγματικών αρνητικών, όπως φαίνεται και στην εξίσωση. Το μέτρο αυτό χρησιμοποιείται για τη μέτρηση του κλάσματος των αρνητικών μοτίβων που ταξινομούνται σωστά [16]. Η καλύτερη εξειδίκευση είναι 1,0 και η χειρότερη 0,0 [33].

Specificity = $\frac{TN}{TN+FP}$

Το μέτρο F (F score) είναι ένα ενιαίο μέτρο που αποτελεί τον αρμονικό μέσο όρο της αξιοπιστίας (precision) και της ανάκλησης (recall).

F score = $\frac{2*Recall*Precision}{Recall+Precision}$

Το σφάλμα λανθασμένης ταξινόμησης μετρά τον λόγο των λανθασμένων προβλέψεων προς τον συνολικό αριθμό των περιπτώσεων που αξιολογήθηκαν. Δίνεται από τον παρακάτω τύπο:

Error Rate = $\frac{FP+FN}{TP+FP+TN+FN}$

Συντελεστής συσχέτισης Matthews (Matthews Correlation Coefficient - MCC) είναι η συσχέτιση μεταξύ των προβλεπόμενων κλάσεων και των αντίστοιχων πραγματικών. Υπολογίζεται με βάση τις τιμές από τον πίνακα σύγχυσης.

MCC = $\frac{TP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Το MCC θεωρείται γενικά ένα ισορροπημένο μέτρο, το οποίο μπορεί να χρησιμοποιηθεί ακόμη και σε περιπτώσεις όπου αναδύεται το πρόβλημα της ταξικής ανισορροπίας [33].

Το μέτρο «Γεωμετρικός μέσος» (Geometric-mean) χρησιμοποιείται για τη μεγιστοποίηση του tp rate και του tn rate, διατηρώντας ταυτόχρονα και τα δύο σχετικά ισορροπημένα. Ο τύπος του είναι ο εξής:

Geometric - mean = $\sqrt{TP*TN}$

Η μέση ακρίβεια (Averaged Accuracy) περιγράφει τη μέση αποτελεσματικότητα όλων των τάξεων και δίνεται από τον τύπο:

Averaged Accuracy = $\frac{\sum_{i=1}^{l}\frac{TPi+TNi}{TPi+FNi+FPi}}{l}$

Το μέσο ποσοστό σφάλματος (Average error rate) βρίσκει το μέσο ποσοστό σφάλματος όλων των κλάσεων και δίνεται από τον τύπο:

Averaged Error Rate = $\frac{\sum_{i=1}^{l}\frac{FPi+FNi}{TPi+FNi+FPi}}{l}$

Η μέση αξιοπιστία (Averaged Precision) υπολογίζει το μέσο όρο της αξιοπιστίας ανά κατηγορία. Ο τύπος είναι ο εξής:

Averaged Precision = $\frac{\sum_{i=1}^{l}\frac{TPi}{TPi+FPi}}{l}$

Επίσης υπάρχει η μέση ανάκληση (Averaged Recall) ανά τάξη και δίνεται από τον τύπο:

Averaged Recall = $\frac{\sum_{i=1}^{l}\frac{TPi}{TPi+FNi}}{l}$

Ένα μέτρο ακόμα είναι ο μέσος όρος του μέτρου F (Averaged F-Measure)

Averaged F-Measure = $\frac{2*PRECISIONm*RECALLm}{PRECISIONm+RECALLm}$

Σε αυτόν τον τύπο το m σημαίνει macro-averaging.

Όπως προκύπτει από πληθώρα μελετών η ακρίβεια είναι το πιο συχνά χρησιμοποιούμενο μέτρο αξιολόγησης στην πράξη είτε για δυαδικά είτε για προβλήματα ταξινόμησης πολλαπλών κλάσεων. Μέσω της ακρίβειας αξιολογείται η ποιότητα της παραγόμενης λύσης με βάση το ποσοστό των σωστών προβλέψεων επί του συνόλου των περιπτώσεων. Συμπληρωματικό μέτρο της ακρίβειας αποτελεί το ποσοστό σφάλματος που αξιολογεί την παραγόμενη λύση με βάση το ποσοστό των εσφαλμένων προβλέψεων. Και τα δύο αυτά μέτρα χρησιμοποιούνται συνήθως από τους ερευνητές στην πράξη για τη διάκριση και την επιλογή της βέλτιστης λύσης [16].

Τα πλεονεκτήματα της ακρίβειας και του ποσοστού σφάλματος είναι ότι αυτά τα μέτρα είναι εύκολο να υπολογιστούν με μικρότερη πολυπλοκότητα, έχουν εφαρμογή σε προβλήματα πολλαπλών κατηγοριών και πολλαπλών ετικετών, και είναι εύκολα κατανοητά από τον άνθρωπο. Όπως επισημαίνεται από πολλές μελέτες, το μέτρο της ακρίβειας έχει περιορισμούς στις διαδικασίες αξιολόγησης και διάκρισης. Ένας από τους κύριους περιορισμούς της ακρίβειας είναι ότι παράγει λιγότερο διακριτές τιμές. Κατά συνέπεια, οδηγεί σε μικρότερη διακριτική ικανότητα κατά την επιλογή και τον προσδιορισμό του βέλτιστου ταξινομητή. Επιπλέον, η ακρίβεια είναι επίσης λιγότερο ευνοϊκή προς τις περιπτώσεις της μειονοτικής κλάσης [16].

Εκτός από την ακρίβεια, το μέτρο F και ο γεωμετρικός μέσος αναφέρθηκαν επίσης ως αποτελεσματικά μέτρα για διαχώριση και αποδίδουν καλύτερα από την ακρίβεια στη βελτιστοποίηση του ταξινομητή για προβλήματα δυαδικής ταξινόμησης. Αντίθετα, τα υπόλοιπα μέτρα που προαναφέρθηκαν είναι δεν θεωρούνται κατάλληλα για την επιλογή της βέλτιστης λύσης, Για τη διάκριση και την επιλογή της βέλτιστης λύσης κατά τη διάρκεια της εκπαίδευσης ταξινόμησης, η αντιστάθμιση της σημαντικότητας μεταξύ των κλάσεων είναι απαραίτητη για να εξασφαλιστεί ότι κάθε κλάση αντιπροσωπεύεται από το χαρακτηριστικό της πρωτότυπο (ή τα χαρακτηριστικά της πρωτότυπα). Η βέλτιστη επιλεγμένη λύση αποδεικνύεται ανώφελη εάν καμία από τις περιπτώσεις κλάσεων μειοψηφίας δεν ήταν σε θέση να προβλεφθεί σωστά από τον/τους επιλεγμένο/ους αντιπρόσωπο/ους (πρωτότυπο) ή να επιλεγεί ως αντιπρόσωπος/οι (π.χ. εάν χρησιμοποιείται η τυχαία μέθοδος επιλογής πρωτοτύπων) κατά τη διάρκεια της εκπαίδευσης ταξινόμησης [16].

‘Ένα σημαντικό μέτρο είναι και το μέσο απόλυτο σφάλμα (Mean Absolute Error – MAE). Είναι η μέση τιμή των απόλυτων τιμών των μεμονωμένων σφαλμάτων πρόβλεψης όλων των περιπτώσεων στο σύνολο δοκιμών. Κάθε σφάλμα πρόβλεψης είναι η διαφορά μεταξύ της πραγματικής τιμής και της προβλεπόμενης τιμής για την περίπτωση. Το μέσο απόλυτο σφάλμα (MAE) ενός μεμονωμένου μοντέλου υπολογίζεται από τον τύπο [33]:

MAE = $\frac{1}{n}\sum_{j=1}^{n}|P_{(ij)}\sum_{j=1}^{n}|P_{ij}$-$T_j||$

όπου P(ij) είναι η τιμή που προβλέπει το ατομικό μοντέλο i για την εγγραφή j και Tj είναι η τιμή-στόχος για την εγγραφή j. Για μια τέλεια πρόβλεψη, P(ij) = Tj και Ei = 0. Έτσι, ο δείκτης MAE κυμαίνεται από 0 έως άπειρο και το 0 αντιστοιχεί στο ιδανικό [33].

Το σχετικό απόλυτο σφάλμα (Relative absolute error - RAE) είναι το συνολικό απόλυτο σφάλμα και κανονικοποιείται διαιρώντας με το συνολικό απόλυτο σφάλμα του απλού εκτιμητή. Το σχετικό απόλυτο σφάλμα RAE ενός μεμονωμένου μοντέλου δίνεται από την εξίσωση [33]:

RAE = $\frac{\sum_{j=1}^{n}|P_{(ij)}-T_j|}{\sum_{j=1}^{n}|T_j-\bar{T}|}$

Όπου P (ij) είναι η τιμή που προβλέπει το μεμονωμένο μοντέλο i για την εγγραφή j, Tj είναι η τιμή-στόχος για την εγγραφή j και το $\bar{T}$ δίνεται από τον τύπο [32]:

$\bar{T} = \frac{1}{n}\sum_{j=1}^{n}T_j$

Για μια τέλεια πρόβλεψη, ο μετρητής είναι 0 και RAE = 0. Έτσι, ο δείκτης RAE κυμαίνεται από 0 έως άπειρο και το 0 αντιστοιχεί στο ιδανικό. Ένα καλό μοντέλο πρόβλεψης παράγει έναν σχεδόν μηδενικό δείκτη [33].

Το ριζικό σχετικό τετραγωνικό σφάλμα (Root relative squared error - RRSE) μειώνει το σφάλμα στις ίδιες διαστάσεις με το προβλεπόμενο μέγεθος. Το σχετικό τετραγωνικό σφάλμα είναι το συνολικό τετραγωνικό σφάλμα διαιρούμενο με το συνολικό τετραγωνικό σφάλμα ενός απλού προβλεπτή. Η ρίζα του σχετικού τετραγωνικού σφάλματος RRSE ενός μεμονωμένου μοντέλου j υπολογίζεται από τον τύπο:

$$\text{RRSE} = \sqrt{\frac{\sum_{j=1}^{n} |P_{(ij)} - T_j|}{\sum_{j=1}^{n} |T_j - \bar{T}|}}$$

Όπου P(ij) είναι η τιμή που προβλέπει το μεμονωμένο μοντέλο i για την εγγραφή j. Για τέλεια πρόβλεψη, ο μετρητής είναι ίσος με 0 και RRSE = 0. Ο δείκτης RRSE κυμαίνεται από 0 έως άπειρο και το 0 αντιστοιχεί στο ιδανικό [33].

Ένα ακόμη μέτρο είναι και το μέσο τετραγωνικό σφάλμα (Mean Square Error - MSE). Γενικά, αυτό το μέτρο μετρά τη διαφορά μεταξύ των προβλεπόμενων λύσεων και των επιθυμητών λύσεων και ορίζεται ως εξής:

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(P_j - A_j)^2$$

όπου Pj είναι η προβλεπόμενη τιμή της περίπτωσης j, Aj είναι η πραγματική τιμή της περίπτωσης j και n είναι ο συνολικός αριθμός των περιπτώσεων.

Παρομοίως με την ακρίβεια, ο κύριος περιορισμός του μέσου τετραγωνικού σφάλματος είναι ότι δεν παρέχει πληροφορίες για την αντιστάθμιση μεταξύ των δεδομένων των κλάσεων. Αυτό μπορεί να οδηγήσει τη διαδικασία διάκρισης στην επιλογή της μη βέλτιστης λύσης. Επιπλέον, εξαρτάται πραγματικά από τη διαδικασία αρχικοποίησης των βαρών. Σε πρόβλημα εξαιρετικά ανισόρροπων κλάσεων, εάν τα αρχικά βάρη δεν έχουν επιλεγεί σωστά (δηλαδή δεν υπάρχει αρχικό βάρος για την αντιπροσώπευση των δεδομένων της μειονοτικής κλάσης), αυτό μπορεί να οδηγήσει τη διαδικασία διάκρισης να καταλήξει σε μη βέλτιστη λύση λόγω έλλειψης πληροφοριών για τα δεδομένα της μειονοτικής κλάσης [16].

Το ριζικό μέσο τετραγωνικό σφάλμα (Root mean squared error – RMSE) προκύπτει λαμβάνοντας την τετραγωνική ρίζα του σχετικού τετραγωνικού σφάλματος, το σφάλμα μειώνεται στις ίδιες διαστάσεις με το προβλεπόμενο μέγεθος. Η ρίζα του μέσου τετραγωνικού σφάλματος (RMSE) ενός μεμονωμένου μοντέλου και υπολογίζεται από τον τύπο [33]:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(P_j - A_j)^2}$$

Ακόμη ένα μέτρο, ονομάζεται «Area under the ROC Curve» (ROC=Receiver operating characteristic), δηλαδή η περιοχή κάτω από την καμπύλη χαρακτηριστικής λειτουργίας δέκτη. Αποτελεί ένα από τα δημοφιλή μέτρα κατάταξης. Σε αντίθεση με τα μέτρα κατωφλίου και πιθανότητας, η τιμή AUC αντικατοπτρίζει τη συνολική απόδοση κατάταξης ενός ταξινομητή. Για πρόβλημα δύο κλάσεων, η τιμή AUC μπορεί να υπολογιστεί ως εξής:

$$\text{AUC} = \frac{S_p - n_p\,(n_n+1)/2}{n_p\,n_n}$$

όπου, S_p είναι το άθροισμα όλων των θετικών παραδειγμάτων που κατατάσσονται, ενώ n_p και n_n υποδηλώνουν τον αριθμό των θετικών και αρνητικών παραδειγμάτων αντίστοιχα. Η AUC έχει αποδειχθεί θεωρητικά και εμπειρικά καλύτερη από το μέτρο της ακρίβειας για την αξιολόγηση της απόδοσης του ταξινομητή και τη διάκριση μιας βέλτιστης λύσης κατά την εκπαίδευση ταξινόμησης. Παρόλο που η απόδοση του AUC ήταν εξαιρετική για τις διαδικασίες αξιολόγησης και διάκρισης, το υπολογιστικό κόστος της AUC είναι υψηλό, ειδικά για τη διάκριση ενός όγκου παραγόμενων λύσεων προβλημάτων πολλαπλών κατηγοριών [16].

Η βελτιστοποιημένη αξιοπιστία (Optimized Precision – OP) ανήκει στον τύπο υβριδικών μέτρων κατωφλίου (hybrid threshold metrics) και έχει προταθεί ως διακριτικός παράγοντας για τη δημιουργία ενός βελτιστοποιημένου ταξινομητή εύρεσης. Αυτό το μέτρο είναι ένας συνδυασμός των μέτρων ακρίβειας, ευαισθησίας και ειδικότητας (accuracy, sensitivity and specificity). Τα μέτρα ευαισθησίας και ειδικότητας χρησιμοποιήθηκαν για τη σταθεροποίηση και τη βελτιστοποίηση της απόδοσης ακρίβειας κατά την αντιμετώπιση προβλημάτων ανισοβαρών κατηγοριών. Το μέτρο OP μπορεί να οριστεί ως εξής

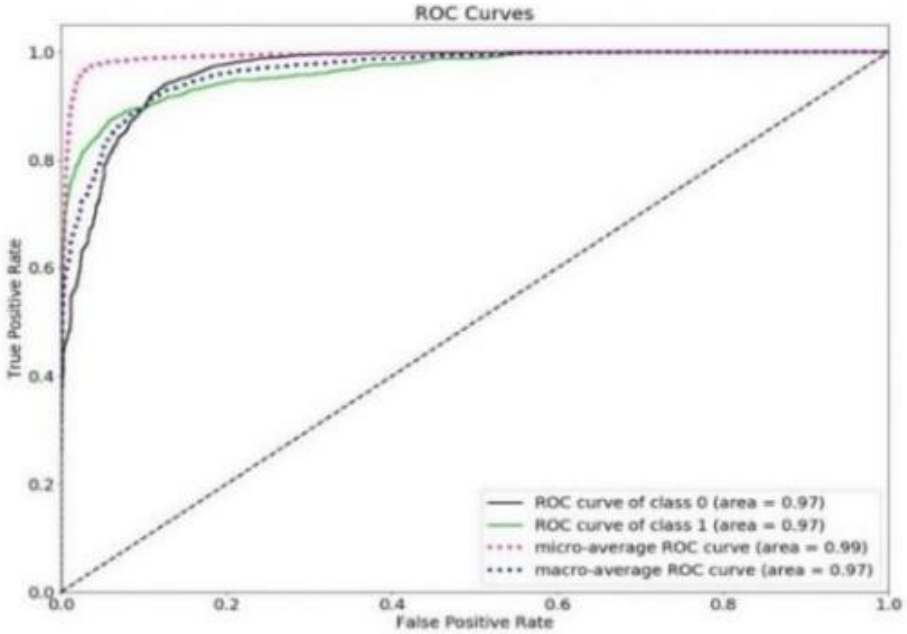
$$\text{OP} = \text{acc} - \frac{|sp - sn|}{sp + sn}$$

όπου acc είναι η τιμή της ακρίβειας, ενώ sp και sn δηλώνουν τις τιμές εξειδίκευσης (specificity) και ευαισθησίας (sensitivity) αντίστοιχα.

Ο κύριος περιορισμός αυτού του υβριδικού μέτρου είναι ότι περιορίζεται μόνο στη διάκριση και αξιολόγηση των προβλημάτων δυαδικής ταξινόμησης. Στο σύνολο δεδομένων του πραγματικού κόσμου, τα διαθέσιμα δεδομένα δεν περιορίζονται σε προβλήματα δύο κατηγοριών. Πολλά σύνολα δεδομένων περιλαμβάνουν περισσότερες από δύο κλάσεις [16].

Εκτός από τα μέτρα που προαναφέρθηκαν, έχουν προταθεί και μέτρα βασισμένα σε γραφικές παραστάσεις, τα οποία είναι αποτελεσματικότερα από την ακρίβεια για την αξιολόγηση της απόδοσης των ταξινομητών. Αυτά τα μέτρα είναι σε θέση να απεικονίζουν τις αντισταθμίσεις μεταξύ διαφορετικών προοπτικών αξιολόγησης που επιτρέπουν πλουσιότερη ανάλυση των αποτελεσμάτων. Παρόλο που τα μέτρα αυτά είναι καλύτερα από την ακρίβεια ή το ποσοστό σφάλματος, η παραγωγή τους με βάση τη γραφική απεικόνιση τα περιορίζει. Παραδείγματα από αυτά τα μέτρα είναι η καμπύλη λειτουργίας δέκτη (Receiver Operating Curve - ROC), η Bayesian Receiver Operating Characteristic (B-ROC), η καμπύλη αξιοπιστίας-ανάκλησης (Precision-Recall Curve), η καμπύλη κόστους (cost curve), το διάγραμμα ανύψωσης (lift chart) και το διάγραμμα βαθμονόμησης (calibration plot) να χρησιμοποιούνται ως διακριτικοί παράγοντες [16].

Η καμπύλη ROC (Receiver Operating Characteristic) είναι ένα γράφημα που απεικονίζει τη σύνδεση μεταξύ του True Positive Rate και του False Positive Rate (Όπως φαίνεται και στο παρακάτω σχήμα). Για κάθε κατώφλι (threshold), υπολογίζουμε το True Positive Rate και το False Positive Rate και τα απεικονίζουμε σε ένα γράφημα. Όσο υψηλότερο είναι το True Positive Rate και όσο χαμηλότερο είναι το False Positive Rate για κάθε κατώφλι, τόσο το καλύτερο. Η περιοχή κάτω από την καμπύλη ROC ονομάζεται βαθμολογία ROC AUC, ένας αριθμός που καθορίζει πόσο καλή είναι η καμπύλη ROC.

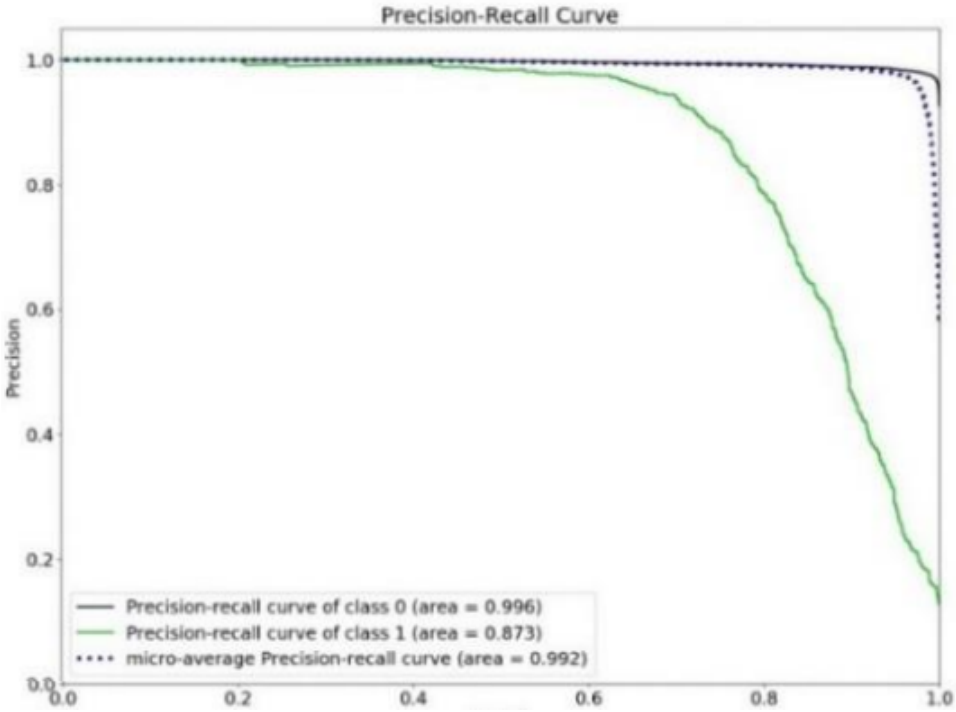


Σχήμα 33: Καμπύλη ROC (Πηγή [33])

Η βαθμολογία ROC AUC δείχνει πόσο καλό είναι το μοντέλο στην κατάταξη των προβλέψεων. Δείχνει την πιθανότητα μια τυχαία επιλεγμένη θετική περίπτωση να κατατάσσεται υψηλότερα από μια τυχαία αρνητική περίπτωση [33].

Η Περιοχή PRC Precision-Recall Curve (Περιοχή καμπύλης αξιοπιστίας-ανάκλησης) είναι ένας αριθμός που περιγράφει τις δυνατότητες του μοντέλου. Η τιμή του PR AUC είναι ο μέσος όρος των βαθμολογιών αξιοπιστίας που υπολογίζονται για κάθε κατώφλι υπενθύμισης (reminder threshold). Η καμπύλη PRC προκύπτει από το συνδυασμό της θετικής αξίας πρόβλεψης (Positive Predictive Value) και του αληθούς θετικού ποσοστού (True Positive Rate) όπως φαίνεται και στο παρακάτω σχήμα. Για κάθε κατώφλι υπολογίζονται η Θετική Αξία Πρόβλεψης και το Ποσοστό των Αληθινών Θετικών και απεικονίζεται το αντίστοιχο σημείο της γραφικής παράστασης. Αυτό το μέτρο και το προηγούμενο δεν είναι ανεξάρτητα. Για το λόγο αυτό γίνεται ένας συνδυασμός μεταξύ

τους. Μια καλή καμπύλη PRC έχει υψηλότερη AUC. Η έρευνα έχει δείξει ότι η PRC είναι γραφικά πιο πληροφοριακή από τις γραφικές παραστάσεις ROC κατά την εκτίμηση δυαδικών ταξινομητών σε μη ισορροπημένα σύνολα [33].



Σχήμα 34: Καμπύλη αξιοπιστίας-ανάκλησης (Πηγή [33])

Επιπλέον, υπάρχουν και μερικά ακόμη μέτρα που έχουν σχεδιαστεί ειδικά για έναν συγκεκριμένο αλγόριθμο ταξινόμησης. Το κέρδος πληροφορίας και τα μέτρα εντροπίας είναι δύο τύποι μέτρων πιθανότητας που έχουν χρησιμοποιηθεί για την αξιολόγηση της χρησιμότητας των χαρακτηριστικών των δεδομένων στη δημιουργία των βελτιστοποιημένων ταξινομητών δέντρων απόφασης (optimized decision tree classifiers). Λόγω του συγκεκριμένου σκοπού, αυτά τα μέτρα δεν είναι κατάλληλα για τη διάκριση και την επιλογή της βέλτιστης λύσης [16].

Στον παρακάτω πίνακα φαίνονται οι επιδόσεις του μοντέλου BERT, στο σύνολο δεδομένων GoEmotions, το οποίο όπως βλέπουμε φτάνει μέσο F1-score 0,46. Το μοντέλο αυτό επιτυγχάνει τις καλύτερες επιδόσεις σε συναισθήματα όπως η ευγνωμοσύνη (0.86), η διασκέδαση (0.8) και η αγάπη (0.78). Επιπλέον, επιτυγχάνει το χαμηλότερο F1-score για το πένθος (0), την ανακούφιση (0.15) και τη συνειδητοποίηση (0.21), τα οποία είναι τα συναισθήματα με τη χαμηλότερη συχνότητα. Διαπιστώνουμε ότι τα λιγότερο συχνά συναισθήματα τείνουν να συγχέονται από το μοντέλο με τα συχνότερα συναισθήματα που σχετίζονται ως προς το συναίσθημα και την ένταση (π.χ. το πένθος με τη θλίψη, η υπερηφάνεια με το θαυμασμό, η νευρικότητα με το φόβο). [11] Στον πίνακα φαίνονται επίσης οι τιμές της αξιοπιστίας και της ανάκλησης.

Emotion	Precision	Recall	F1
admiration	0.53	0.83	0.65
amusement	0.70	0.94	0.80
anger	0.36	0.66	0.47
annoyance	0.24	0.63	0.34
approval	0.26	0.57	0.36
caring	0.30	0.56	0.39
confusion	0.24	0.76	0.37
curiosity	0.40	0.84	0.54
desire	0.43	0.59	0.49
disappointment	0.19	0.52	0.28
disapproval	0.29	0.61	0.39
disgust	0.34	0.66	0.45
embarrassment	0.39	0.49	0.43
excitement	0.26	0.52	0.34
fear	0.46	0.85	0.60
gratitude	0.79	0.95	0.86
grief	0.00	0.00	0.00
joy	0.39	0.73	0.51
love	0.68	0.92	0.78
nervousness	0.28	0.48	0.35
neutral	0.56	0.84	0.68
optimism	0.41	0.69	0.51
pride	0.67	0.25	0.36
realization	0.16	0.29	0.21
relief	0.50	0.09	0.15
remorse	0.53	0.88	0.66
sadness	0.38	0.71	0.49
surprise	0.40	0.66	0.50
macro-average	**0.40**	**0.63**	**0.46**
std	**0.18**	**0.24**	**0.19**

Πίνακας 3: Αποτελέσματα της ταξινόμησης του συνόλου GoEmotions [11]

Οι παρακάτω πίνακες 4 και 5 παρουσιάζουν τα αποτελέσματα για ένα μοντέλο ομαδοποίησης συναισθήματος (sentiment - grouped model) (F1-score = 0,69) και ένα μοντέλο ομαδοποίησης Ekman (Ekman - grouped model) (F1-score = 0,64), αντίστοιχα.

Sentiment	Precision	Recall	F1
ambiguous	0.54	0.66	0.60
negative	0.65	0.76	0.70
neutral	0.64	0.69	0.67
positive	0.78	0.87	0.82
macro-average	**0.65**	**0.74**	**0.69**
std	**0.09**	**0.10**	**0.09**

Πίνακας 4: Αποτελέσματα μοντέλου ομαδοποίησης συναισθήματος [11]

Στον παραπάνω πίνακα έγινε μια κατηγοριοποίηση των συναισθημάτων σε 4 κατηγορίες (θετική, αρνητική, διφορούμενη και ουδέτερη).

Ekman Emotion	Precision	Recall	F1
anger	0.50	0.65	0.57
disgust	0.52	0.53	0.53
fear	0.61	0.76	0.68
joy	0.77	0.88	0.82
neutral	0.66	0.67	0.66
sadness	0.56	0.62	0.59
surprise	0.53	0.70	0.61
macro-average	**0.59**	**0.69**	**0.64**
std	**0.10**	**0.11**	**0.10**

Πίνακας 5 : Αποτελέσματα μοντέλου ομαδοποίησης Ekman [11]

Ο Ekman χώρισε τα συναισθήματα σε 6 βασικές κατηγορίες οι οποίες είναι η χαρά, ο θυμός, ο φόβος, η θλίψη, η αηδία και η έκπληξη. Στην ταξινόμηση αυτή χρησιμοποιήθηκε επίσης η κατηγορία ουδέτερο. Η κατηγορία θυμός αντιστοιχεί σε συναισθήματα όπως ο θυμός, η ενόχληση και η αποδοκιμασία), η κατηγορία αηδία αντιστοιχεί στο συναίσθημα αηδία), η κατηγορία φόβος αντιστοιχεί στα συναισθήματα φόβος και νευρικότητα, η κατηγορία χαρά περιλαμβάνει όλα τα θετικά συναισθήματα, η κατηγορία θλίψη αντιστοιχεί στη θλίψη, την απογοήτευση, την αμηχανία, το πένθος και τις τύψεις) και στην κατηγορία έκπληξη υπάρχουν όλα τα διφορούμενα συναισθήματα [11].

Η σημαντική αύξηση των επιδόσεων κατά τη μετάβαση από την πλήρη ταξινόμηση στην ταξινόμηση επιπέδου Ekman υποδεικνύει ότι αυτή η ομαδοποίηση μετριάζει τη σύγχυση μεταξύ των κατηγοριών κατώτερου επιπέδου της εσωτερικής ομάδας (inner-group lower-level categories). Τα αποτελέσματα αυτά αφήνουν πολλά περιθώρια βελτίωσης, αναδεικνύοντας ότι το έργο αυτό δεν έχει ακόμη αντιμετωπιστεί πλήρως από τα τρέχοντα σύγχρονα μοντέλα κατανόησης φυσικής γλώσσας (Natural Language Understanding - NLU) [11].

ΚΕΦΑΛΑΙΟ 5

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΗ ΕΡΕΥΝΑ

Σε αυτήν την εργασία φαίνεται πώς μπορεί να χρησιμοποιηθεί το προεκπαιδευμένο μοντέλο BERT ώστε να πραγματοποιηθεί ανάλυση συναισθήματος στο σύνολο δεδομένων GoEmotions. Το μοντέλο αυτό είναι βασισμένο στην αρχιτεκτονική των μετασχηματιστών (Transformer) για γλωσσική μοντελοποίηση. Οι μετασχηματιστές, έχουν προσελκύσει τεράστιο ενδιαφέρον λόγω της αποτελεσματικότητάς τους σε ένα τεράστιο φάσμα προβλημάτων επεξεργασίας φυσικής γλώσσας.

Η ανάλυση συναισθήματος μπορεί να φανεί χρήσιμη σε πολλαπλούς τομείς όπως τα οικονομικά, το εμπόριο, η κοινωνιολογία και η πολιτική. Επιπλέον μπορεί να βοηθήσει εταιρείες να βελτιώσουν τα προϊόντα ή τις υπηρεσίες που προσφέρουν μέσω των αξιολογήσεων των καταναλωτών. Για αυτό είναι σημαντική η συνεχής βελτίωση των μοντέλων που χρησιμοποιούνται για αυτήν όπως το BERT ώστε να έχουμε πιο ακριβή αποτελέσματα . Αξίζει να σημειωθεί πως πρόσφατα, υπάρχει ένας μεγάλος όγκος εργασιών που χρησιμοποιεί το μοντέλο BERT όχι μόνο για ανάλυση συναισθήματος αλλά και για διάφορες εργασίες πάνω στην επεξεργασία φυσικής γλώσσας, όπως η ταξινόμηση κειμένων, η απάντηση ερωτήσεων, η περίληψη και πολλές ακόμη εργασίες.

Από τα αποτελέσματα που είδαμε στο προηγούμενο κεφάλαιο καταλαβαίνουμε ότι με τους υπάρχοντες περιορισμένους πόρους για την επισήμανση πρόσθετων δεδομένων που διατίθενται για ταξινόμηση συναισθημάτων για εξειδικευμένους τομείς, το σύνολο δεδομένων που χρησιμοποιήθηκε μπορεί να παρέχει βασική κατανόηση των συναισθημάτων και να συμβάλλει στην αύξηση της ακρίβειας του μοντέλου. ωστόσο, τα αποτελέσματα υποδεικνύουν πολλά περιθώρια για μελλοντική βελτίωση [11]

Κάτι ακόμη που αξίζει να ειπωθεί είναι πως το μοντέλο BERT έχει φανεί ότι έχει μεγάλο πλεονέκτημά της στην κατηγοριοποίηση κειμένου σε πολλές εργασίες επεξεργασίας φυσικής γλώσσας, συμπεριλαμβανομένης της ταξινόμησης συναισθήματος κυρίως σε επίπεδο κειμένου. Αρκετές έρευνες που χρησιμοποίησαν το BERT έχουν πραγματοποιήσει αξιοσημείωτες βελτιώσεις στο μοντέλο αυτό, ορισμένες με απλές ιδέες και άλλες με μια πιο σύνθετη δομή. Μια άλλη παρατήρηση είναι ότι η μέση ακρίβεια ταξινόμησης έχει φτάσει σε υψηλά ποσοστά αλλά εξακολουθούν να υπάρχουν ορισμένες κατηγορίες δεδομένων, για τις οποίες το τρέχον μοντέλο δεν μπορεί να δώσει ικανοποιητική λύση. Η ακρίβεια ταξινόμησης των ουδέτερων περιπτώσεων είναι πολύ χαμηλότερη από εκείνη των περιπτώσεων με σαφή πόλωση, ενώ οι περιπτώσεις με μικτή πόλωση συναισθήματος προς διαφορετικές πτυχές (hard data) είναι ακόμη πιο δύσκολο να επεξεργαστούν. Ο ακριβής προσδιορισμός τέτοιων περιπτώσεων απαιτεί μεγάλο όγκο δεδομένων εκπαίδευσης, μαζί με εμπεριστατωμένη ανάλυση για την εξαγωγή χρήσιμων προτύπων [12]. Για αυτόν τον λόγο η προσπάθεια για την βελτιστοποίηση του μοντέλου είναι πολύ σημαντική.

Μελλοντικές επεκτάσεις της παρούσας εργασίας μπορούν να αντιμετωπίσουν το πρόβλημα της ταξινόμησης συναισθήματος μεταξύ πολλών κλάσεων διακρίνοντας αρχικά μεταξύ των ευρύτερων κατηγοριών συναισθήματος που υπάρχουν σε ένα συγκεκριμένο σύνολο δεδομένων. Θα μπορούσε δηλαδή να εκπαιδεύσει κανείς ένα ευρύτερο σύνολο ταξινομητών που διακρίνει αρχικά μεταξύ των γενικότερων κατηγοριών συναισθήματος και στη συνέχεια εξειδικεύει τις αποφάσεις του μεταξύ των υποκλάσεων που συνθέτουν την κάθε ευρύτερη κατηγορία. Με τη συγκεκριμένη στρατηγική αναμένεται η αύξηση της συνολικής ακρίβειας του συστήματος καθώς οι επιμέρους ταξινομητές καλούνται να διακρίνουν μεταξύ ενός αρκετά περιορισμένου συνόλου πιθανών κλάσεων.

ΠΑΡΑΡΤΗΜΑ 1

Ο κώδικας που χρησιμοποιήθηκε για την ανάλυση συναισθήματος φαίνεται παρακάτω:

```
# ------------------------------------------------------------------------
# This script file provides fundamental computational functinality for performing
# sentiment classificaion on the Musk Tweets Hourly dataset. The sentiment
# classifier to be built will be utilizing the BERT model which will be used for
# tokenization and word-embedding purposes. The final model will be trained on
# the go emotions database.
# ------------------------------------------------------------------------

# Import required Python libraries.
import openpyxl
import pandas as pd
import numpy as np
import re
import os
from collections import defaultdict
from tqdm import tqdm
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, Dataset, random_split
from transformers import BertTokenizer, BertModel
import matplotlib.pyplot as plt
import sys
# ------------------------------------------------------------------------
#                          COMMAND LINE ARGUMENTS:
# ------------------------------------------------------------------------
# Get the number of command line arguments given.
n_args = len(sys.argv)
# Command Line Argument 1: set the percentage of data to be kept from each
#                          distinct class.
if n_args > 1:
    retain_percent = float(sys.argv[1])
    print(f'retain_percent:{retain_percent}')
else:
    retain_percent = 0.01


# ------------------------------------------------------------------------
#                          FUNCTION DEFINITIONS:
# ------------------------------------------------------------------------
# This method provides fundamental text cleaning functionality.
def clean_text(text):
    # Remove emoticons
    emoticon_pattern = re.compile(u'('
        u'\ud83d[\ude00-\ude4f]|'  # emoticons
        u'\ud83c[\udf00-\uffff]|'  # symbols & pictographs (part 1)
```

```
        u'\ud83d[\u0000-\uddff]|'  # symbols & pictographs (part 2)
        u'\ud83d[\ude80-\udeff]|'  # transport & map symbols
        u'\ud83c[\udde0-\uddff]'   # flags (iOS)
        u')+', flags=re.UNICODE)
    text = emoticon_pattern.sub(r'', text)

    # Remove URLs
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)

    # Remove user @ references and '#' from tweet
    text = re.sub(r'\@\w+|\#','', text)

    # Remove punctuations
    text = re.sub(r'[^\w\s]', '', text)

    # Lowercase the words
    text = text.lower()
    return text
# ----------------------------------------------------------------------------
# This method will be used to plot the absolute frequency histogram for the
# sequence lengths.
def plot_histogram(data,bin_centers,xlabel,ylabel,title):
    # Calculate the bin edges by shifting the bin centers.
    bin_edges = np.array(bin_centers)-0.5
    # Append the edge point for the last bin.
    bin_edges = np.append(bin_edges,bin_centers[-1]+0.5)
    plt.hist(data,bins=bin_edges)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    plt.title(title)
    plt.grid(True)
    plt.show()

# ----------------------------------------------------------------------------
# This method provides fundamental parsing functionality for a given excel
# file. The output returned is a list of strings where each string contains
# a comma separated character array with the cell contents for each row of the
# original excel file.
def parse_excel(file_path):
    # Initialize the list of strings to be returned by this function.
    lines = []
    # Load the workbook and select the active sheet.
    workbook = openpyxl.load_workbook(file_path)
    sheet = workbook.active
```

```
    # Iterate over the various rows of the file and read each cell as text.
    for row in sheet.iter_rows():
        # Initialize the current line container to the empty string.
        line_text = ""
        # Iterate over the contents of the current row.
        for cell in row:
            # Convert the content of each cell to string so as to be appended
            # to the current row.
            cell_value = str(cell.value) if cell.value is not None else ""
            line_text += cell_value + ","
        # Append current line string to the lines list.
        lines.append(line_text)
    return lines
# ------------------------------------------------------------------------
#                          CLASS DEFINITIONS:
# ------------------------------------------------------------------------
# Define the Custom Dataset Class.
# ------------------------------------------------------------------------
class SentimentDataset(Dataset):

    # Class constructor.
    def __init__(self,texts,labels,tokenizer,max_length):
        self.texts = [clean_text(text) for text in texts]
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    # Implement the custom len() behaviour for the objects of this class.
    def __len__(self):
        return len(self.texts)

    # Implement the class method that allows slicing within the contents of the
    # custom dataset class.
    def __getitem__(self,idx):
        text = self.texts[idx]
        if len(self.labels)>0:
            label = self.labels[idx]
        # ------------------------------------------------------------------
        # Encoding text technical details:
        # ------------------------------------------------------------------
        # This is a method provided by the BERT tokenizer loaded from the
        # Transformers library. It takes a text string and converts it into
        # input IDs and attention masks necessary for BERT.
        # ------------------------------------------------------------------
        # Input Parameters:
```

```
        # ------------------------------------------------------------------
        # text: The text to be encoded. This is the individual text string
        #       from the dataset that needs sentiment analysis.
        # add_special_tokens = True: This tells the tokenizer to add special
        #                            tokens like [CLS] at the beginning and
        #                            [SEP] at the end of each text string.
        #                            These tokens are important for BERT as
        #                            [CLS] is often used for classification
        #                            tasks, and [SEP] is a separator token,
        #                            useful especially when dealing with two
        #                            text strings (like in question-answering
        #                            models.
        # ------------------------------------------------------------------
        # max_length: This sets the maximum length of the tokenized input
        #             sequence. If the text is longer than this, it will be
        #             truncated to max_length. This value is typically set to
        #             the maximum length the model can accept (e.g., 512 tokens
        #             for BERT).
        # ------------------------------------------------------------------
        # return_token_type_ids = False: This is specific to certain BERT tasks
        #                                that requires differentiating between
        #                                multiple input sequences like questing
        #                                answering tasks. For simple classification
        #                                tasks this value should be set to False.
        # ------------------------------------------------------------------
        # padding = 'max_length': This ensures that all encoded sequences are
        #                         padded to the same length (max_length). In
        #                         case a sequence is shorter than max_length,
        #                         it will be padded with zeros.
        # ------------------------------------------------------------------
        # return_attention_mask = True: This directive instructs the tokenizer
        #                               to generate and return attention masks,
        #                               which tell the model which tokens should
        #                               be attended to and which should not
        #                               (e.g. padding tokens)
        # ------------------------------------------------------------------
        # return_tensors = 'pt': This specifies that the returned tensors should
        #                        PyTorch tensors.
        # ------------------------------------------------------------------
        # trancation = True: This argument ensures that if a text string is
        #                    longer than the max_length, it will be truncated
        #                    to fit.
        # ------------------------------------------------------------------
```

```
        # Output Parameters:
        # ------------------------------------------------------------------
        # input_ids: These are the token ids for each token in the text.They
        #            constitute the input for the BERT model.
        # ------------------------------------------------------------------
        # attention_mask: This is a mask of 1s and 0s indicating which tokens
        #                 are actual words and which are padding.The BERT
        #                 model utilizes this information to know which parts
        #                 of the input it should pay attention to and which
        #                 parts should be ignored.
        # ------------------------------------------------------------------

        # Encoding text.
        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens = True,
            max_length = self.max_length,
            return_token_type_ids = False,
            padding = 'max_length',
            return_attention_mask = True,
            return_tensors = 'pt',
            truncation = True
            )

        # Create the dictionary object which will be returned as the item for
        # the requested position.
        # Check whether text sentiment labels are not provided:
        if len(self.labels)==0:
            item = {
                'text' : text,
                'input_ids' : encoding['input_ids'].flatten(),
                'attention_mask' : encoding['attention_mask'].flatten()
                }
        else:
            item = {
                'text' : text,
                'input_ids' : encoding['input_ids'].flatten(),
                'attention_mask' : encoding['attention_mask'].flatten(),
                'label' : torch.tensor(label,dtype=torch.long)
                }
        return item
# ----------------------------------------------------------------------------

# Define an additional function that will preserve a given percentage of data
# instances per class in order to reduce the overall size of the dataset in
# order to reduce the computation time required to process it.This function
# will return the data indices to be retained keeping an equal amount of
```

```
# patterns from each distinct class.
def balanced_dataset_reduction(dataset,percentage):
    # Initialize a dictionary object that will be used for grouping the data
    # points indices per class.
    indices_by_class = defaultdict(list)

    # defaultdict: is a class provided by the collections module in Python.
    # It works like a regular dictionary but with one key difference: it
    # automatically creates new items for keys that have not been encountered
    # yet. When you try to access a key that doesn't exist, instead of raising a
    # a KeyError, defaultdict creates the key and initializes it to a default
    # value that you specify.

    # list: in this case is the default value type specified for the
    # defaultdict. It means that when a new key is accessed in indices_by_class
    # that doesn't exist yet, a new empty list will be created as its value.

    # So, indices_by_class = defaultdict(list) creates a defaultdict where the
    # default value type for new keys is an empty list. This is useful for
    # collecting indices corresponding to different class labels in a dataset,
    # as you can simply append indices to the list associated with each class
    # label without having to check if the key exists first.

    # Loop through the various datapoints stored in the dataset object by
    # keeping both the actual data and the corresponding indices.
    for idx,data in enumerate(dataset):
        # Acquire the class label for the currently processed data instance.
        label = data["label"].item()
        indices_by_class[label].append(idx)

    # Shuffle and select indices from each class.
    # Initialize the list container that will store the data indices that will
    # be retained.
    filtered_indices = []
    # Loop through the value entries for each key in the dictionary.
    for indices in indices_by_class.values():
        # Convert indices to numpy array to increase execution speed.
        indices = np.array(indices)
        np.random.shuffle(indices)
        # Compute the number of indices to be kept.
        retain_number = int(len(indices)*percentage)
        filtered_indices.extend(indices[:retain_number])
    return filtered_indices
# ----------------------------------------------------------------------------
```

```
# Define the Sentiment Classifier Class.
# ---------------------------------------------------------------------------
class SentimentClassifier(nn.Module):

    # Class constructor.
    def __init__(self,bert_model,n_classes,dropout_percent=0.3):
        # Call the super class constructor.
        super(SentimentClassifier,self).__init__()
        self.bert = bert_model
        # Freeze the parameters of the BERT model.
        for param in self.bert.parameters():
            param.requires_grad = False
        self.drop = nn.Dropout(p=dropout_percent)
        self.out = nn.Linear(self.bert.config.hidden_size,n_classes)

    # Define the function that describes the forward pass of information within
    # the network. Mind that the network module is being fed with the input ids
    # and the attention mask provided by the BERT tokenizer for each text.
    def forward(self,input_ids,attention_mask):
        _, pooled_output = self.bert(
            input_ids=input_ids,
            attention_mask=attention_mask,
            return_dict=False
            )
        # When return_dict=False, the BERT model returns a tuple of two elements:
        # [1]: The hidden state of all layers of the neural model.
        # [2]: The pooled output, which is the representation of the input
        #      sequence after being processed by the model.
        # By setting return_dict=False, we are explicitly instructing the BERT
        # model to return a tuple of outputs instead of a dictionary. This is
        # often done for compatibility with older versions of the Hugging Face
        # Transformers library or for custom model implementations that expect
        # tuples as outputs. If return_dict=True, the BERT model would return a
        # dictionary containing various outputs, such as the hidden states,
        # pooled output, and other intermediate outputs.
        output = self.drop(pooled_output)
        return self.out(output)

# ---------------------------------------------------------------------------
#                          INITIALIZATION TASKS:
# ---------------------------------------------------------------------------

# Read the individual data files into dataframe objects.
df1 = pd.read_csv("data/goemotions_1.csv")
df2 = pd.read_csv("data/goemotions_2.csv")
```

```
df3 = pd.read_csv("data/goemotions_3.csv")

# Concatenate the individual dataframes into a single dataframe object.
df = pd.concat([df1,df2,df3],ignore_index=True)

# Delete the constituent dataframes.
del df1,df2,df3

# Filter all examples that have been marked as very unclear.
unclear = df["example_very_unclear"]==True
df = df[~unclear]

# Drop columns that are not required for the sentiment analysis task.
dropped_columns = ["id","author","subreddit","link_id","parent_id",
                   "created_utc","rater_id","example_very_unclear"]
df.drop(dropped_columns,axis="columns",inplace=True)

# Isolate the list of string texts.
texts = list(df["text"])

# Compute the sequence length distribution for the cleaned texts.
cleaned_texts = [clean_text(text) for text in texts]
# Generate the list of lengh per sequence.
L = [len(clean_text) for clean_text in cleaned_texts]
```

```
# Convert the list to numpy array.
NL = np.array(L)
# Compute the minimum and maximum sequence length.
Lmin,Lmax = NL.min(),NL.max()
# Compute the mean and the standard deviation of the sequence lengths.
Lmean,Lstd = NL.mean(),NL.std()
# Set the bin centers.
bin_centers = [l for l in range(Lmin,Lmax+1)]
# Plot the histogram.
plot_histogram(L,bin_centers,"Sequence Length","Absolute Frequency",
               "Sequence Length Distribution")


# Isolate the dataframe of labels and covert it to a numpy array.
labels = df.iloc[:,1:].to_numpy()
# Convert the one-hot vector encoded labels into a list of integer ids
# indicating the corresponding sentiment category.
labels = list(np.argmax(labels,axis=1))
# Get the number of classes existing within the ground truth dataset.
classes_num = len(set(labels))
# Get the names of the various classes.
ClassNames = list(df)[1:] # The first entry is omitted since it is the text.
# Compute the distribution distribution of data instances per label.
Labels = np.array(labels)
```

```
Labels_min = Labels.min()
Labels_max = Labels.max()
Labels_bins = np.arange(Labels_min,Labels_max+2)
LabelsHist,_ = np.histogram(Labels,Labels_bins)
# Set the bin centers for the labels.
Labels_centers = [lc for lc in range(Labels_min,Labels_max+1)]
# Plot the histogram.
plot_histogram(Labels,Labels_centers,"Sentiment","Absolute Frequency",
               "Sentiment Class Distribution")
# Create a dictionary storing the frequency of occurence for each distinct
# sentiment class.
ClassNamesHist = dict(zip(ClassNames,LabelsHist))

# Compute the class priors.
class_priors = LabelsHist / LabelsHist.sum()

# Set the location of the dataset that will be exclusively used for testing
# purposes and for which no ground truth labels are available.
excel_file_path = 'data/Musk_HOURLY.xlsx'
# Read the excel lines.
excel_lines = parse_excel(excel_file_path)
# Read the text of each tweet which is stored in the excel file.
# Assuming that the textual content of each line is a comma separated string,
# we need to keep the first substring.
excel_texts = [line.split(",")[0] for line in excel_lines]
# Drop the first entry since it corresponds to the content of the header line.
excel_texts = excel_texts[1:]
# Additionally, we need to define a lambda function which returns the last
# non-empty string in a list of strings.
last_non_empty = lambda lst: next((s for s in reversed(lst) if s),None)
# Thus, we can acquire the date string for each tweet knowing that it is the
# last piece of information which is stored in each excel line.
excel_dates = [last_non_empty(line.split(",")) for line in excel_lines]
# Drop the first entry since it corresponds to the content of the header line.
excel_dates = excel_dates[1:]

# ------------------------------------------------------------------------
#                    LOAD BERT TOKENIZER AND BERT MODEL:
# ------------------------------------------------------------------------

# Set the name of the BERT model to be utilized.
model_name = 'bert-base-uncased'
# Instantiate the BERT tokenizer.
tokenizer = BertTokenizer.from_pretrained(model_name)
# Instantiate the BERT model.
bert_model = BertModel.from_pretrained(model_name)

```

```
# ------------------------------------------------------------------------
#                           DATA PREPARATION:
# ------------------------------------------------------------------------

# Set the maximum length for the input sequences to be 3 standard deviations
# away from the average sequence length.
max_length = int(Lmean+3*Lstd)+1

# Set the batch size to be used during the training process.
batch_size = 2

# Create the ground truth dataset instance of the SentimentDataset class.
ground_truth_dataset = SentimentDataset(texts,labels,tokenizer,max_length)


# Filter the ground truth dataset per class according to the prespecified
# percentage value per class if the respective percentage is not set to None.
# In case it is not None, generate the reduced version of the complete dataset.
if retain_percent is not None:
    filtered_indices = balanced_dataset_reduction(ground_truth_dataset,retain_percent)
    ground_truth_dataset = torch.utils.data.Subset(ground_truth_dataset,filtered_indices)

# Create the dataset instance for the unseen set of tweets.
unseen_dataset = SentimentDataset(excel_texts,[],tokenizer,max_length)

# Set the percentage of the available data that will be used for training.
train_percent = 0.8

# Set the train and the test sizes.
train_size = int(train_percent * len(ground_truth_dataset))
test_size = len(ground_truth_dataset) - train_size

# Split the ground truth dataset into training and testing subsets.
train_dataset, test_dataset = random_split(
    ground_truth_dataset, [train_size,test_size])

# Instantiate the training and testing data loaders.
train_loader = DataLoader(train_dataset,batch_size=batch_size,shuffle=True)
test_loader = DataLoader(test_dataset,batch_size=batch_size)


# ------------------------------------------------------------------------
#                           MODEL EVALUATION PROCESS:
# ------------------------------------------------------------------------

def evaluate_model(model,data_loader,device,return_probabilities=False):
    # Set the model evaluation environment.
    model.eval()
```

```
    # Initialize variables counting the total and correctly classified text
    # instances.
    total,correct = 0,0
    # Initialize the list containing the output probabilities for each text
    # instance.
    all_probabilities = []

    # Indicate that no gradient-based updating of the model weight-vector will
    # be performed during this process.
    with torch.no_grad():
        for batch in tqdm(data_loader):
            # Acquire the information that the bert tokenizer associated with
            # each textual input.
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["label"].to(device)
            # Acquire the network output for each instance in the batch.
            outputs = model(input_ids=input_ids,attention_mask=attention_mask)
            # Convert the network primitive output to probabilities.
            probabilities = torch.nn.functional.softmax(outputs,dim=1)
            # Get the predicted class index by determining which component of
            # the probability vector contains the maximum value.
            _,predicted = torch.max(probabilities,1)
            # If the corresponding input argument indicates that probability
            # vectors should also be returned, then perform the necessary
            # computation.
            if return_probabilities:
                all_probabilities.extend(probabilities.cpu().numpy())
            # Accumulate the total number of text instances that have been
            # classified so far.
            total += labels.size(0)
            # Accumulate the total number of text instances that have been
            # correctly classified.
            correct += (predicted==labels).sum().item()

    # Compute the total accuracy of the model on the subset of text instances
    # stored in the data loader.
    accuracy = correct / total

    return (accuracy,np.array(all_probabilities)) if return_probabilities else accuracy

# ------------------------------------------------------------------------------
#                           MODEL TRAINING PROCESS:
# ------------------------------------------------------------------------------

def train_model(model,train_loader,test_loader,optimizer,loss_fn,epochs,
                checkpoint_path=None,save_every_batches=None):
```

```
504
505     # Load the state variables from the last training session.
506     checkpoint = torch.load(checkpoint_path)
507     # Load the last state of the neural model.
508     model.load_state_dict(checkpoint["model_state_dict"])
509     # Load the last state of the optimizer.
510     optimizer.load_state_dict(checkpoint["optimizer_state_dict"])
511     # Set the starting epoch of the training process to be the last
512     # training epoch of the previous session.
513     start_epoch = checkpoint["epoch"]
514     # Load the last batch processed during the previous training session.
515     batch_count = checkpoint["batch_count"]
516     # Load the train accuracy achieved by the model so far.
517     train_accuracy = checkpoint['train_accuracy']
518     # Load the test accuracy achieved by the model so far.
519     test_accuracy = checkpoint["test_accuracy"]
520
521     # Actual Model Training Process
522
523     # Loop through the remaining training epochs.
524     for epoch in range(start_epoch,epochs):
525         # Set the model training environment.
526         model.train()
```

```
527         # Loop through the various batches.
528         for batch_idx,batch in enumerate(tqdm(train_loader,
529                                               desc=f"Epoch: {epoch+1} / {epochs} train_acc: {train_accuracy} test_acc: {test_accuracy}")):
530             # Skip batches until the last processed batch is reached.
531             if batch_idx < batch_count:
532                 continue
533             # Clear gradients.
534             optimizer.zero_grad()
535             # Load the necessary information from the current batch.
536             input_ids = batch["input_ids"].to(device)
537             attention_mask = batch["attention_mask"].to(device)
538             labels = batch["label"].to(device)
539             # Compute the output of the model and the associated loss by
540             # performing the forward pass of information.
541             outputs = model(input_ids=input_ids,attention_mask=attention_mask)
542             loss = loss_fn(outputs,labels)
543             # Optimize model parameters by performing the backward pass of
544             # information.
545             loss.backward()
546             optimizer.step()
547             # Increase the number of batches processed so far.
548             batch_count += 1
549             # Save checkpoint if specified and if the currently processed batch
```

```
            # is an integer multiple of the save_every_batches input arguments.
            # Thus, a training checkpoint will be saved after every given amount
            # of batches has been processed.
            if save_every_batches is not None and batch_count % save_every_batches==0:
                torch.save({
                    'epoch':epoch,
                    'batch_count':batch_count,
                    'model_state_dict':model.state_dict(),
                    'optimizer_state_dict':optimizer.state_dict(),
                    'train_accuracy':train_accuracy,
                    'test_accuracy':test_accuracy
                    },checkpoint_path)

        # After the termination of each training epoch the batch_count variable
        # must be reset to zero so that the next training epoch does not skip
        # all the batches that have been completed by the previous session.
        batch_count = 0
        # Evaluate the train accuracy of the model after each epoch.
        train_accuracy = evaluate_model(model,train_loader,device)
        # Evaluate the model on the test set after each epoch.
        test_accuracy = evaluate_model(model,test_loader,device)
        # Save final checkpoint after each training epoch has been completed.
        torch.save({
            'epoch':epoch,
            'batch_count':batch_count,
            'model_state_dict':model.state_dict(),
            'optimizer_state_dict':optimizer.state_dict(),
            'train_accuracy':train_accuracy,
            'test_accuracy':test_accuracy
        },checkpoint_path)

    # Report the termination of the training process.
    print("Training process completed")

# ----------------------------------------------------------------------------
#    CLASSIFIER, LOSS FUNCTION, OPTIMIZER AND CHECK POINT INITIALIZATION:
# ----------------------------------------------------------------------------
# Set the device on which training will be performed.
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = SentimentClassifier(bert_model,classes_num)
optimizer = torch.optim.Adam(model.parameters(),lr=2e-5)
# Set the class weights to be inversly proportinal to the class priors.
class_weights = 1 / class_priors
# Convert class weights to tensor
weight = torch.tensor(class_weights, dtype=torch.float).to(device)
loss_fn = nn.CrossEntropyLoss(weight=weight)
# Set the number of training epochs.
```

```
epochs = 10
model = model.to(device)

# Method .to(device) is a PyTorch method used to move tensors or models to a
# specified device. The device argument specifies where the tensors or model
# should be moved. It could be either a CPU or a GPU. In the context of deep
# learning, moving models to GPUs can significantly speed up computation due to
# their parallel processing capabilities.

# Define the name of the checkpoint directory
checkpoint_directory = "checkpoints"
# Define the name of the file that will actually store the checkpoint dictionary.
checkpoint_filename = "model_checkpoint.pth"
# Create the full path for the checkpoint directory within the current directory
checkpoint_path = os.path.join(checkpoint_directory,checkpoint_filename)
# Set the number of batches within each training epoch after which a checkpoint
# entry will be saved.
batches_save_period = 5
# Create the checkpoint directory and the corresponding file in case it does
# not exist. In that case, create the file and save the initial state of the
# training process.
if not os.path.exists(checkpoint_directory):
    os.makedirs(checkpoint_directory)
    f = open(checkpoint_path,"w")
    f.close()
    # Set the initial state variables of the training process.
    epoch = 0
    batch_count=0
    model_state_dict = model.state_dict()
    optimizer_state_dict = optimizer.state_dict()
    train_accuracy = 0
    test_accuracy = 0
    torch.save({
        'epoch':epoch,
        'batch_count':batch_count,
        'model_state_dict':model.state_dict(),
        'optimizer_state_dict':optimizer.state_dict(),
        'train_accuracy':train_accuracy,
        'test_accuracy':test_accuracy
        },checkpoint_path)

# Call the model training method.
train_model(model,train_loader,test_loader,optimizer,loss_fn,epochs,
            checkpoint_path=checkpoint_path,save_every_batches=batches_save_period)
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Abdelgwad, Mohammed M., Taysir Hassan A. Soliman, and Ahmed I. Taloba. "Arabic aspect sentiment polarity classification using BERT." *Journal of Big Data* 9.1 (2022): 115.
2. Agarwal, Basant, et al. "Machine learning approach for sentiment analysis." *Prominent feature extraction for sentiment analysis* (2016): 21-45.
3. Agarwal, Basant, et al. "Sentiment analysis using common-sense and context information." *Computational intelligence and neuroscience* 2015 (2015): 30-30.
4. Aurélien Géron, “Hands on machine learning with Scikit-Learn, Keras & TensorFlow”, O’Reilly (2023)
5. Ava Soleimany, “Deep Sequence Modeling MIT 6.S191” (2019)
6. Bello, Abayomi, Sin-Chun Ng, and Man-Fai Leung. "A BERT framework to sentiment analysis of tweets." *Sensors* 23.1 (2023): 506.
7. Bishop, Chris M. "Neural networks and their applications." *Review of scientific instruments* 65.6 (1994): 1803-1832.
8. Cambria, Erik, et al., eds. *A practical guide to sentiment analysis*. Vol. 5. Cham: Springer International Publishing, 2017.
9. Chen James, “What Is a Neural Network?”, Investopedia (2024)
10. Dana Alon, Jeongwoo Ko, GoEmotions: A Dataset for Fine-Grained Emotion Classification, Google Research, 2021
11. Demszky, Dorottya, et al. "GoEmotions: A dataset of fine-grained emotions." *arXiv preprint arXiv:2005.00547* (2020).
12. Gao, Zhengjie, et al. "Target-dependent sentiment classification with BERT." *Ieee Access* 7 (2019): 154290-154299.
13. Godsay, Manasee. "The process of sentiment analysis: a study." *International Journal of Computer Applications* 126.7 (2015): 26-30.
14. Gurney, Kevin. *An introduction to neural networks*. CRC press, 2018.
15. Hardesty Larry, “Explained: Neural networks”, MIT News on campus and around the world (2017)
16. Hossin, Mohammad, and Md Nasir Sulaiman. "A review on evaluation metrics for data classification evaluations." International journal of data mining & knowledge management process 5.2 (2015): 1.
17. Ke, Zixuan, Hu Xu, and Bing Liu. "Adapting BERT for continual learning of a sequence of aspect sentiment classification tasks." *arXiv preprint arXiv:2112.03271* (2021).
18. Kolambage Nival Chamara, Design, Development, and Implementation of a Machine Learning-based Predictive Modelling Tool to Accurately Predict Thalassemia Carrier state using Full Blood Count Indices and Haemoglobin Variants, ResearchGate, 2020
19. Kumar, Avinash, et al. "BERT based semi-supervised hybrid approach for aspect and sentiment classification." *Neural Processing Letters* 53 (2021): 4207-4224.
20. Li, Mingzheng, et al. "Sentiment analysis of Chinese stock reviews based on BERT model." *Applied Intelligence* 51 (2021): 5016-5024.
21. Li, Xin, et al. "Exploiting BERT for end-to-end aspect-based sentiment analysis." *arXiv preprint arXiv:1910.00883* (2019).
22. Liyan Tang, “Trasnformer” presentation
23. Maltoudoglou, Lysimachos, Andreas Paisios, and Harris Papadopoulos. "BERT-based conformal predictor for sentiment analysis." *Conformal and Probabilistic Prediction and Applications*. PMLR, 2020.
24. Martin, Gati L., Medard E. Mswahili, and Young-Seob Jeong. "Sentiment classification in swahili language using multilingual bert." *arXiv preprint arXiv:2104.09006* (2021).
25. Munikar, Manish, Sushil Shakya, and Aakash Shrestha. "Fine-grained sentiment classification using BERT." *2019 Artificial Intelligence for Transforming Business and Society (AITB)*. Vol. 1. IEEE, 2019.
26. Nisha Arya Ahmed, “What is A Confusion Matrix in Machine Learning? The Model Evaluation Tool Explained”, “Datacamp”, 2023.
27. Patro, V. Mohan, and M. Ranjan Patra. "Augmenting weighted average with confusion matrix to enhance classification accuracy." *Transactions on Machine Learning and Artificial Intelligence* 2.4 (2014): 77-91.

28. Rietzler, Alexander, et al. "Adapt or get left behind: Domain adaptation through BERT language model finetuning for aspect-target sentiment classification." *arXiv preprint arXiv:1908.11860* (2019).
29. Sasirekha Cota, “Deep learning basics — Part 2 — Perceptron”, Medium, 2023
30. Sharkawy, Abdel-Nasser. "Principle of neural network and its main types." *Journal of Advances in Applied & Computational Mathematics* 7 (2020): 8-19.
31. Talaat, Amira Samy. "Sentiment analysis classification system using hybrid BERT models." *Journal of Big Data* 10.1 (2023): 110.
32. Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).
33. Vujović, Ž. "Classification model evaluation metrics." *International Journal of Advanced Computer Science and Applications* 12.6 (2021): 599-606.
34. Wang, Xinyu, et al. "A depression detection model based on sentiment analysis in micro-blog social network." *Trends and Applications in Knowledge Discovery and Data Mining: PAKDD 2013 International Workshops: DMApps, DANTH, QIMIE, BDM, CDA, CloudSD, Gold Coast, QLD, Australia, April 14-17, 2013, Revised Selected Papers 17*. Springer Berlin Heidelberg, 2013.
35. Xie, Song, et al. "Sentiment analysis of chinese e-commerce reviews based on bert." *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*. Vol. 1. IEEE, 2020.
36. Yıldırım, Savaş, and Meysam Asgari-Chenaghlu. *Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques*. Packt Publishing Ltd, 2021.
37. Yu, Jianfei, and Jing Jiang. "Adapting BERT for target-oriented multimodal sentiment classification." IJCAI, 2019.
38. Yue, Lin, et al. "A survey of sentiment analysis in social media." *Knowledge and Information Systems* 60 (2019): 617-663.
39. Zahner, Daniel A., and Evangelia Micheli-Tzanakou. "Artificial neural networks: definitions, methods, applications." *Supervised and Unsupervised Pattern Recognition*. CRC Press, 2017. 61-78.