ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Διπλωματική Εργασία

Τίτλος εργασίας

Εντοπισμός ψευδών ειδήσεων σε ελληνικά και αγγλικά κείμενα με χρήση αλγορίθμων μηχανικής μάθησης

Συγγραφέας/είς

Ονοματεπώνυμο:

Θανάσης Τζιμούρτας

ΑΜ: 71345363

Επιβλέπων/ουσα:

Δρ. Παναγιώτα Τσελέντη

Ε.ΔΙ.Π.

Αθήνα, Φεβρουάριος 2022

UNIVERSITY OF WEST ATTICA

SCHOOL OF ENGINEERING

DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

Diploma Thesis

Title

Detecting fake news on Greek and English texts using machine learning algorithms

Student name and surname:

Thanasis Tzimourtas

Registration Number: 71345363

Supervisor:

Dr. Panagiota Tselenti

Athens, February 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Τίτλος εργασίας

Εντοπισμός ψευδών ειδήσεων σε ελληνικά και αγγλικά κείμενα με χρήση αλγορίθμων μηχανικής μάθησης

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η πτυχιακή/διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
	ΠΑΡΙΣ ΜΑΣΤΟΡΟΚΩΣΤΑΣ	**ΚΑΘΗΓΗΤΗΣ**	PARIS MASTORO KOSTAS Digitally signed by PARIS MASTOROKOSTAS Date: 2022.03.18 15:35:45 +02'00'
	ΓΕΩΡΓΙΟΣ ΜΠΑΡΔΗΣ	**ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ**	Georgios Bardis Digitally signed by Georgios Bardis Date: 2022.03.18 17:10:24 +02'00'
	ΠΑΝΑΓΙΩΤΑ ΤΣΕΛΕΝΤΗ	**ΕΔΙΠ**	Panagiota Tselenti Digitally signed by Panagiota Tselenti Date: 2022.03.08 14:43:09 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/η κάτωθι υπογεγραμμένος/η **Αθανάσιος Τζιμούρτας** του **Κωνσταντίνου**, με αριθμό μητρώου **71345363** φοιτητής/τρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής **Μηχανικών** του **Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών** δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο/Η Δηλών/ούσα

*** Ονοματεπώνυμο /Ιδιότητα**
(Υπογραφή)

Ψηφιακή Υπογραφή Επιβλέποντα

Detecting fake news on Greek and English texts using machine learning algorithms

By

Thanasis Tzimourtas

Περίληψη

Στην σημερινή εποχή, τόσο στα μέσα κοινωνικής δικτύωσης όσο και στα ειδησεογραφικά άρθρα εφημερίδων, υπάρχει μια σημαντική άνοδος των ψευδών ειδήσεων. Σε μια εποχή που τα μέσα κοινωνικής δικτύωσης έχουν γίνει αναπόσπαστο κομμάτι της καθημερινότητάς μας καθώς και για πολλούς μια σημαντική πηγή ειδήσεων, πρέπει να ελέγχεται η ορθότητα της πληροφορίας που αναρτιούνται σε αυτά. Το ίδιο ισχύει και για τα ειδησεογραφικά άρθρα που πλέον είναι διαθέσιμα στο διαδίκτυο και στα μέσα κοινωνικής δικτύωσης και οι άνθρωποι στρέφονται σε αυτά για να ενημερωθούν. Σε αυτή τη διπλωματική εργασία θα μελετήσουμε το πρόβλημα των ψευδών ειδήσεων εξετάζοντας κείμενα τόσο της αγγλικής όσο και της ελληνικής γλώσσας κάνοντας χρήση αλγορίθμων μηχανικής και βαθιάς μάθησης ώστε να μπορέσουμε να μελετήσουμε και να αξιολογήσουμε τις αποδόσεις των μοντέλων πάνω στο πρόβλημα των ψευδών ειδήσεων. Για το κομμάτι της αγγλικής γλώσσας έγινε χρήση του συνόλου δεδομένων PHEME που αποτελείται από tweets ενώ για το κομμάτι της ελληνικής γλώσσας επιλέχθηκε μια συλλογή δεδομένων που αποτελείται από ειδησεογραφικά άρθρα.

Abstract

Nowadays, both on social media and in newspaper news articles, there is a significant rise in fake news. At a time when social media has become an integral part of our daily lives as well as for many important news sources, the correctness of the information that is denied in them must be checked. The same goes for news articles that now exist on the internet and social media and people turn to them for their daily information. In this diploma we will study the problem of fake news by examining the texts of both English and Greek using machine and deep learning algorithms so that we can study and evaluate the performance of models on the problem of fake news. For the English language part, the PHEME data set consisting of tweets was used, while for the Greek language part, a data collection consisting of news articles was selected.

Contents

1. Introduction

Στην σύγχρονη εποχή που ζούμε, στην εποχή της πληροφορίας και της ταχύρρυθμης ζωής, όλο και περισσότερα δεδομένα μας κατακλύζουν κάθε λεπτό. Η εύκολη και γρήγορη ενημέρωση των γεγονότων που μας περικλείουν γίνεται μονόδρομος καθώς δεν μπορούμε να διαθέτουμε τον απαιτούμενο χρόνο για να ενημερωθούμε σωστά για τα συμβάντα που γίνονται γύρω μας και ανά τον κόσμο, καθώς πολύ γρήγορα το νέο γίνεται παλιό και αντικαθίσταται από μια νέα είδηση.

Η άνοδος των μέσων κοινωνικών δικτύωσης, μας σύστησε σε έναν καινούργιο κόσμο ψυχαγωγίας και ενημέρωσης αλλά μαζί του αναδύθηκε και ένα φαινόμενο, το οποίο μπορεί να μην είναι καινούργιο στο λεξιλόγιό μας, αλλά την τελευταία δεκαετία και με σημείο αναφοράς, των προεδρικών εκλογών των Ηνωμένων Πολιτειών το 2016 έφερε, στο προσκήνιο την ανεξέλεγκτη διάδοση των ψευδών ειδήσεων ή αλλιώς “fake news”. Μια μελέτη που έγινε πάνω στα άρθρα που δημοσιεύτηκαν κατά την διάρκεια της εκλογικής περιόδου [1], έδειξε ότι οι 20 κορυφαίες ψευδείς ειδήσεις που κυκλοφόρησαν, είχαν 8.711.000 διαμοιρασμούς (shares), αντιδράσεις (reactions) και σχόλια στο Facebook. Από την άλλη, οι 20 κορυφαίες πραγματικές ειδήσεις που δημοσιεύθηκαν την ίδια περίοδο, είχαν 7.367.000 διαμοιρασμούς (shares), αντιδράσεις (reactions) και σχόλια, νούμερο αρκετά χαμηλότερο από αυτό των παραπλανητικών άρθρων. Όπως καταλαβαίνουμε τα νούμερα είναι αρκετά ανησυχητικά αν λάβουμε υπόψιν μας τα ευρήματα της μελέτης [2], όπου αναφέρουν ότι το 51% των χρηστών της Αυστραλίας, το 46% των χρηστών της Ιταλίας, το 40% των χρηστών των Ηνωμένων Πολιτειών και το 36% των χρηστών του Ηνωμένου Βασιλείου που έχουν πρόσβαση στο διαδίκτυο, εξαρτώνται από τα μέσα κοινωνικής δικτύωσης για την ενημέρωσή τους. Βάσει των παραπάνω γίνεται αντιληπτό ότι ο περιορισμός των ειδήσεων με σκοπό την παραπλάνηση του κοινού είναι αρκετά σημαντικός καθώς όλο και περισσότεροι άνθρωποι στρέφονται στα μέσα κοινωνικής δικτύωσης, όπως το Facebook και το Twitter για να λάβουν την ενημέρωσή τους.

Η δημιουργία και η διάδοση ψευδών ειδήσεων δεν είναι καινούργιο φαινόμενο. Συγκεκριμένα σύμφωνα με το [3], το 1897 ο εκδότης της εφημερίδας New York States, William Randolph Hearst, μια εμβληματική φιγούρα της εποχής και απροκάλυπτα πολεμοκάπηλος, ήταν η αιτία που οι Ηνωμένες Πολιτείες ξεκίνησαν πόλεμο με την Ισπανία, λόγω μιας έκρηξης στο λιμάνι του USS Maine της Αβάνας, το οποίο η εφημερίδα του Hearst κατηγορούσε κατηγορηματικά τους Ισπανούς.

Πιο πρόσφατα, το 2003 πριν την εισβολή των Ηνωμένων Πολιτειών στο Ιράκ, οι New York Times δημοσίευσαν μια σειρά από άρθρα, εκ των οποίων ένα από αυτά μιλούσε για βιολογικά όπλα υπό την κατοχή των Ιρακινών δυνάμεων. Όπως συνεχίζει ο αρθρογράφος του [4], τα άρθρα της δημοσιογράφου Judith Miller’s περιείχαν παραπλανητικά στοιχεία, όπου αναφερόντουσαν σε όπλα μαζικής καταστροφής, τα οποία θα έθεταν τον αμερικάνικο λαό σε μεγάλο κίνδυνο. Όπως ανέφεραν αργότερα οι διοικητικοί υπάλληλοι της κυβέρνησης Bush τα άρθρα αυτά ήταν μια από τις αιτίες της εισβολής στο Ιράκ.

Ωστόσο οι ψευδείς ειδήσεις δεν αποτελούν μόνο απειλή για το δημοκρατικό πολίτευμα, αλλά μπορούν να απειλήσουν και τις οικονομίες μας. Πιο συγκεκριμένα το [5] αναφέρει ότι όταν παραπλανητικές ειδήσεις ισχυρίστηκαν ότι ο Barack Obama τραυματίστηκε σε έκρηξη, το αποτέλεσμα ήταν να χαθούν 130 δισεκατομμύρια δολάρια σε μετοχικές αξίες.

Όπως βλέπουμε οι ειδήσεις με σκοπό την παραπλάνηση της κοινής γνώμης τόσο σε θέματα πολιτικής όσο και σε θέματα επενδυτικών κινήσεων μπορούν να βλάψουν τις κοινωνίες μας. Τα μέσα κοινωνικής δικτύωσης επέτρεψαν τη διάδοση αυτών των ειδήσεων να πραγματοποιούνται ευκολότερα και χωρίς ελέγχους για το αντικείμενο διαπραγμάτευσής τους, διότι μέσω του

διαδικτύου ο καθένας μπορεί να ισχυριστεί ότι είναι δημοσιογραφική πηγή. Ωστόσο με την άνοδο της υπολογιστικής δύναμης των τελευταίων χρόνων, όλο ένα και περισσότεροι ερευνητές προσπαθούν να κατανοήσουν και να βοηθήσουν στην καταπολέμηση της διάδοσης των ψευδών ειδήσεων μέσω μοντέλων μηχανικής μάθησης.

Άλλο ένα κομβικό σημείο για την αντιμετώπιση των ψευδών ειδήσεων, είναι να ορίσουμε μια ορολογία που να ανταποκρίνεται στο πρόβλημα με σκοπό την καλύτερη μελέτη του. Ο όρος "fake news" ή αλλιώς ψευδείς ειδήσεις, είναι ένας αρκετά μη επαρκής όρος, καθώς υποδηλώνει ότι τα συγκεκριμένα άρθρα αποτελούν κατασκεύασμα αβάσιμων πληροφοριών. Ωστόσο τα ψευδή άρθρα δεν είναι μόνο τα κατασκευάσματα μυθοπλασίας με αβάσιμες πληροφορίες, όπως θα περίμενε κανείς από ένα μυθιστόρημα αλλά περιέχουν και πραγματικά γεγονότα. Εξάλλου όπως υπογραμμίζει και το [3] οι ψευδείς ειδήσεις έχουν σκοπό να μιμηθούν τα πραγματικά γεγονότα. Αρκετές ψευδείς ειδήσεις δεν χρειάζεται να περιέχουν και ψευδείς ισχυρισμούς για να κατηγοριοποιηθούν ως ψευδή, αλλά μπορούν να παρουσιάσουν επιλεκτικά μερική αλήθεια, αποκρύπτοντας βασικά κομμάτια γεγονότων ώστε να παραπλανήσουν τον αναγνώστη τους. Βάσει των παραπάνω αντιλαμβανόμαστε ότι η εύρεση μιας ορθής ορολογίας που να περιγράφει το φαινόμενο των ψευδών ειδήσεων είναι αναγκαίος.

Η ορολογία που μοιάζει να είναι πιο ταιριαστή και να ανταποκρίνεται στο φαινόμενο που έχει γίνει γνωστό την τελευταία δεκαετία ως "fake news" ή ψευδείς ειδήσεις, είναι "σκόπιμες παραπλανητικές ειδήσεις" καθώς στόχος αυτών των ειδήσεων είναι να "μεταμφιεστούν" και να παρουσιαστούν ως πραγματικές ειδησεογραφικές πληροφορίες με σκοπό να παραπλανήσουν τον αναγνώστη ή τον ακροατή για να εξυπηρετήσουν κάποια πολιτική ιδεολογία καθώς όπως θα δούμε κατά την διάρκεια αυτής της διπλωματικής εργασίας, ένα μεγάλο ποσοστό αυτών των άρθρων περιέχει πολιτικό χαρακτήρα και κατεύθυνση. Όπως αναφέρεται και από το [6] η Caroline Jack, μια ερευνήτρια κοινωνικών επιστήμων, ισχυρίζεται ότι η ορολογία "προβληματική πληροφορία", ανταποκρίνεται καλύτερα στα σημερινά δεδομένα. Ωστόσο για την διευκόλυνση μας στο υπόλοιπο αυτής της διπλωματικής εργασίας θα αναφερόμαστε στον φαινόμενο αυτό με την αρχικά ορισμένη ορολογία του.

Ένα κομβικό στοιχείο που πρέπει να οριστεί πριν η εργασία αρχίσει να επικεντρώνεται στα μοντέλα μηχανικής μάθησης για τον εντοπισμό τέτοιων "κακόβουλων" ειδήσεων, είναι να εξετάσουμε ποια άρθρα και πληροφορίες τοποθετούμε υπό τον ορισμό ψευδών ειδήσεων. Για παράδειγμα πρέπει να λαμβάνονται οι σατιρικές ειδήσεις ως ψευδείς; Αρκετοί ερευνητές όπως αναφέρει το [7] θεωρούν τέτοιου είδους ειδήσεις ως μια μορφή παραπλανητικών πληροφοριών και θα έπρεπε να χαρακτηρίζονται ως ψευδείς ειδήσεις, ενώ ταυτόχρονα υπάρχουν και αυτοί που διαφωνούν με αυτή την κατηγοριοποίηση. Ο ίδιος "διχασμός" ανάμεσα στους ερευνητές, παρατηρείται και για τα άρθρα που προωθούν τις φήμες, θεωρίες συνωμοσίας και τις "ακραίες" μορφές σάτιρας όπως οι φάρσες. Ωστόσο όπως σημειώνει και ο αρθρογράφος της πηγής [7] πρέπει να λαμβάνονται μόνο τα ειδησεογραφικά άρθρα που σκόπιμα διαστρεβλώνουν τα γεγονότα, με μερικές εξαιρέσεις όπως στην περίπτωση των σατιρικών άρθρων που λαμβάνονται λανθασμένα από το ευρύ κοινό ως σοβαρές ειδησεογραφικές ειδήσεις. Έτσι βλέπουμε ότι η ορολογία που δόθηκε στην προηγούμενη παράγραφο ανταποκρίνεται καλύτερα στο πρόβλημα που αυτή η εργασία έχει σκοπό να μελετήσει.

Όπως προαναφέρθηκε, στην σημερινή εποχή όπου τα μέσα κοινωνικής δικτύωσης έχουν αφομοιωθεί από εμάς για την διασκέδαση και την ενημέρωσή μας, η ανάγκη για έλεγχο ορθότητας των πηγών ενημέρωσής που βρίσκονται στο διαδίκτυο είναι καθοριστικός. Μια συμβατική προσέγγιση για τον περιορισμό των τροποποιημένων πληροφοριών που διαδίδονται, είναι να επιστρατευτούν επαγγελματίες δημοσιογράφοι ώστε να ελέγχουν την ορθότητα των άρθρων που διαμοιράζονται και αναρτιούνται στο διαδίκτυο, βάσει προηγούμενων επικυρωμένων ειδήσεων που θα έχουν ως γνώμονα για την διαδικασία αυτή [8]. Όπως συνειδητοποιούμε, τέτοιου είδους διαδικασία είναι αρκετά χρονοβόρα και ακριβή. Συγκεκριμένα η πηγή [8] δίνει ως παράδειγμα την

προσπάθεια της ειδησεογραφικής ιστοσελίδας “PolitiFact1” που επιστρατεύει τρεις (3) συντάκτες σε κάθε ειδησεογραφική πληροφορία που εξετάζεται, για να ελέγχεται η εγκυρότητά της δηλαδή αν οι πληροφορίες αποτελούν πραγματικές ή ψευδείς ειδήσεις. Βάσει των παραπάνω γίνεται κατανοητό ότι τέτοιου είδους μέθοδος δεν μπορεί να επιστρατευτεί μόνη της καθώς οι πληροφορίες που αναρτιούνται και διαμοιράζονται καθημερινά στο διαδίκτυο και στα μέσα κοινωνικής δικτύωσης είναι εξαιρετικά μεγάλος.

Σε αυτό το πρόβλημα έρχεται να βοηθήσει η διαδικασία της NLP (Natural Language Process) που με την βοήθεια των μοντέλων τόσο της μηχανικής μάθησης (machine learning) όσο και των μοντέλων βαθιάς μάθησης (deep learning), οι ερευνητές μπορούν να ανιχνεύουν τις ψευδείς ειδήσεις με το να αναλύουν το περιεχόμενο των άρθρων. Για να επιτευχθεί αυτό, πρέπει να δημιουργηθούν έμπιστες και σωστά μελετημένες βάσεις δεδομένων που θα περιέχουν τόσο πραγματικές όσο ψευδείς επιβεβαιωμένες ειδήσεις, ώστε να μπορούν τα μοντέλα που θα δημιουργηθούν να έχουν επαρκή δεδομένα, για να αναλύουν τα σημασιολογικά χαρακτηριστικά των άρθρων, ώστε να καταφέρουν να κατηγοριοποιήσουν με επιτυχία, όσο τον δυνατόν περισσότερα άρθρα με σκοπό να χρησιμοποιηθούν για να κατηγοριοποιήσουν νέα, μη ελεγμένες ειδήσεις.

Ωστόσο εκτός από βάσεις δεδομένων κομβικό σημείο είναι και η προ επεξεργασία των κειμένων που απαρτίζουν μια βάση δεδομένων καθώς και οι εύρεση σωστών features που θα μπορέσουν να βοηθήσουν τους αλγορίθμους μηχανικής μάθησης να εκπαιδευτούν με όσο το υψηλότερο ποσοστό επιτυχίας στην σωστή κατηγοριοποίηση, τόσο των εκπαιδευόμενων δεδομένων (training data) όσο και των δεδομένων προς εξέταση (testing data) είναι καθοριστικό.

Στο κεφάλαιο 2 της εργασίας θα αναπτυχθούν οι διεργασίες, προεπεξεργασίας των κειμένων ενώ στα κεφάλαια 3 και 4 θα αναπτυχθούν αναλυτικά τα μοντέλα μηχανικής και βαθιάς μάθησης αντίστοιχα, που θα χρησιμοποιηθούν στο πρακτικό κομμάτι της εργασίας. Στη συνέχεια, στο κεφάλαιο 5 θα αναλυθούν τα σύνολα δεδομένων που χρησιμοποιήθηκαν για τα αγγλικά και ελληνικά κείμενα, ενώ θα παρουσιαστούν και τα γραφήματα των επιδόσεων των μοντέλων πάνω σε αυτά. Επίσης στην υποενότητα 5.2.3 του κεφαλαίου 5 θα παρουσιαστούν και τα αποτελέσματα της εργασίας. Τέλος στην ενότητα 6 θα διατυπωθούν τα συμπεράσματα που πάρθηκαν από την μελέτη αυτών των μοντέλων στις συλλογές που επιλέχθηκαν για το πρακτικό κομμάτι αυτής της διπλωματικής εργασίας.

2. Text Preprocessing

2.1. Introduction

Πριν δώσουμε ένα κείμενο, στην περίπτωσή μας αυτό το κείμενο είναι ένα αγγλικό tweet ή ένα δημοσιογραφικό άρθρο από κάποια ελληνική εφημερίδα, σε κάποιο αλγόριθμο μηχανικής μάθησης, πρέπει πρώτα να το προ επεξεργαστούμε και να μετατρέψουμε τις λέξεις που το απαρτίζουν σε μορφή που τα μοντέλα να μπορούν να το επεξεργαστούν. Η προ επεξεργασία είναι ένα κομβικό σημείο στην διαδικασία της κατηγοριοποίησης κειμένων καθώς τα κείμενα που επιλέγονται για την εκπαίδευση των μοντέλων περιέχουν συχνά θόρυβο, δηλαδή σε επίπεδο λέξεων, πολλές λέξεις στο κείμενο δεν βοηθάνε τους αλγορίθμους μηχανικής μάθησης να κατατάξουν το εξεταζόμενο κείμενο στην επιθυμητή κλάση. Επίσης στόχος της προ επεξεργασίας των κειμένων είναι να ελαχιστοποιήσει το μέγεθος του λεξιλογίου, ομαδοποιώντας όρους στο ίδιο token αποδίδοντας την ίδια σημασιολογική ερμηνεία / βαρύτητα σε όλες τις μορφές της ίδιας λέξης που εντοπίζεται μέσα σε ένα έγγραφο ή μια συλλογή εγγράφων, που διαφορετικά δεν θα είχαν, διότι θα βρισκόντουσαν σε διαφορετική κλίση ή χρόνο, με διαφορετικό επίθημα ή πρόθεμα. Η διαδικασία αυτή είναι το πρώτο βήμα για να βοηθήσουμε τόσο τα μοντέλα μηχανικής μάθησης (Machine Learning), όσο και τα μοντέλα βαθιάς μάθησης (Deep learning) να συλλέξουν χρήσιμα σημασιολογικά στοιχεία για να μπορέσουν να κατηγοριοποιήσουν τόσο τα δεδομένα εκπαίδευσης (training data) όσο και τα δεδομένα εξάσκησης (testing data) στην σωστή κατηγορία (label) που ανήκουν.

Τέλος μέσα στην διαδικασία της προ επεξεργασίας των κειμένων εντάσσεται και η σωστή εύρεση των χαρακτηριστικών (features) που θα τροφοδοτούν με δεδομένα τους αλγορίθμους.

2.2. Text Cleaning and Pre-processing

Όπως προαναφέρθηκε πολλά κείμενα περιέχουν πολλές λέξεις όπως προθήματα (stop words), ορθογραφικά λάθη, καθώς ανάλογα με την κατηγορία που ανήκει ένα κείμενο, υπάρχουν συνήθως και λέξεις εξειδικευμένης σημασίας (αργκό). Επειδή δεν θέλουμε να επιβαρύνουμε την διαδικασία της ταξινόμησης με δεδομένα που όχι μόνο δεν θα μας βοηθήσουν να πάρουμε το επιθυμητό αποτέλεσμα αλλά πολλές φορές επιβαρύνουν τις επιδόσεις των αλγορίθμων, επιλέγουμε να αφαιρούμε από τα κείμενά αυτούς τους όρους από την διαδικασία συλλογείς όρων που θα μετατραπούν σε μορφή κατάλληλη για τα μοντέλα μηχανικής μάθησης.

2.2.1. Tokenization and Lowercasing

Για να προ επεξεργαστούμε ένα κείμενο πρέπει πρώτα να το μετατρέψουμε σε μία λίστα που θα περιέχει όλους τους όρους (tokens). Στα tokens συμπεριλαμβάνονται και τα σημεία στίξης που όπως είδαμε πρέπει να αφαιρούνται. Η διαδικασία lowercasing είναι η μετατροπή των λέξεων του κειμένου σε πεζούς χαρακτήρες. Πολλές φορές η διαδικασία lowercasing γίνεται παράλληλα με την διαδικασία tokenization. Μας ενδιαφέρει να μετατρέπουμε τους όρους tokens σε πεζούς χαρακτήρες ώστε να μην κατηγοριοποιούμε την ίδια λέξη πολλαπλές φορές στο λεξικό μας, αφού όταν θα περάσει στην διαδικασία μετατροπής της σε διάνυσμα, θα έχει την ίδια σημασιολογική ανάλυση με όλες τις διαφορετικές της παραδοχές.

2.2.2. Stemming

Κατά την επεξεργασία ενός εγγράφου μια λέξη μπορεί να εμφανιστεί με πολλές μορφές που η σημασιολογική τους αξία θα είναι ίδια για όλες της μορφές της. Η διαδικασία ή μέθοδος

stemming, είναι να εξερευνά και να βρίσκει την ρίζα της λέξης. Αυτή η διαδικασία προ επεξεργασίας στοχεύει στην αφαίρεση των διάφορων επιθημάτων για να μειώσει τον αριθμό των λέξεων, με το να αντιστοιχίζονται όλες οι διάφορες μορφές της λέξης στον ίδιο κορμό (stem), ώστε επιτευχθεί καλύτερος χρόνος και χώρος αποθήκευσης στην βάση του συστήματος. Για αυτό το στάδιο επεξεργασίας υπάρχουν πολλοί αλγόριθμοι που μπορούν να πραγματοποιήσουν stemming, ωστόσο ο πιο γνωστός είναι ο Porter stemmer [9].

2.2.3. Lemmatization

Η μέθοδος lemmatization είναι μια διαδικασία της NLP (Natural language processing) που αντικαθιστά το επίθημα μιας λέξης με την μια διαφορετική ή αφαιρεί το επίθημα μιας λέξης εντελώς για να πάρει τη βασική της μορφή (λήμμα) [10]. Ενώ φαίνεται ότι η μέθοδος stemming και lemmatization ενεργούν με παρόμοιο τρόπο, πρέπει να αναφερθούν και οι διαφορές τους για λόγους ευκρίνειας. Σύμφωνα με το [11] το stemming συνήθως αναφέρεται σε μια ακατέργαστη ευρετική διαδικασία που κόβει τα άκρα των λέξεων ενώ το lemmatization αναφέρεται στο να γίνεται παράλληλα σωστή χρήση λεξιλογίου και μορφολογικής ανάλυσης λέξεων, που στοχεύουν συνήθως να αφαιρέσουν μόνο τις καταλήξεις και όπως προαναφέρθηκε να επιστρέψουν την λέξη στην βασική της μορφή.

2.3. N-grams

Η πιο συνηθισμένη προσέγγιση κατά την προ επεξεργασία ενός κειμένου είναι να αντιμετωπίζουμε τις λέξεις του κειμένου ως μεμονωμένες οντότητες. Ωστόσο ορισμένες λέξεις έχουν καλύτερη σημασιολογική σαφήνεια όταν λαμβάνονται ως όροι εντός πλαισίων. Για παράδειγμα, η λέξη “εθνικό” έχει διαφορετικές ερμηνείες όταν χρησιμοποιείται στις εκφράσεις, “εθνική άμυνα” και “εθνικό χρέος”. Σύμφωνα με το [11], τα παραπάνω οδήγησαν σε μια πρακτική της συμπερίληψης n-gram από έγγραφα, όπου ένα n-gram είναι μια συνεχόμενη ακολουθία λέξεων που η σημασιολογική τους τιμή / αξία αποτυπώνεται σε έναν αριθμό / διάνυσμα ως μια οντότητα.

2.4. Bag of Words

Ένα μοντέλο bag-of-words (BOW), είναι μια απλοποιημένη αναπαράσταση ενός κειμένου από επιλεγμένα μέρη του εγγράφου, βάσει συγκεκριμένων κριτηρίων, όπως η συχνότητα των λέξεων [10]. Στην περίπτωση αυτή, δεν μας ενδιαφέρει σε ποιο σημείο του εγγράφου βρέθηκε η λέξη αλλά το πόσες φορές βρέθηκε μέσα στο κείμενο [12]. Για παράδειγμα, έχοντας την πρόταση, « Την επόμενη εβδομάδα έχω ρεπό », η διαδικασία bag of words, θα επέστρεφε, την πρόταση σε μορφή λίστας με το αναγνωριστικό της λέξεις, token και ένα αριθμό που θα έδειχνες πόσες φορές βρέθηκε μέσα στο κείμενο ([Την: 1, επόμενη: 1, εβδομάδα: 1, έχω: 1, ρεπό: 1]).

2.5. Term Frequency Inverse Document Frequency

Το Term Frequency–Inverse Document Frequency (TF-IDF), είναι μια στατιστική ανάλυση που αποκαλύπτει την σημασιολογική βαρύτητα μιας λέξης σε μια συλλογή εγγράφων [13]. Το TF- IDF χρησιμοποιείται συχνά ως συντελεστής στάθμισης στην ανάκτηση πληροφοριών και εξόρυξη κειμένου. Η τιμή του TF-IDF αυξάνεται αναλογικά με τον αριθμό των φορών που εμφανίζεται μια λέξη στο έγγραφο, αλλά αντισταθμίζεται από τη συχνότητα της λέξης στη συλλογή. Παρακάτω, βλέπουμε την μαθηματική έκφραση.

$$tf - idf_{t,d} = tf_{t,d} * idf_t$$

(2.1)

2.6. Word Embedding

Η μέθοδος word embedding είναι μια από τις πιο διαδεδομένες επιλογές για την αναπαράσταση του λεξιλογίου ενός εγγράφου ή μιας συλλογής εγγράφων. Είναι ικανή να αποδίδει την σημασιολογική και συντακτική ομοιότητα των λέξεων σε σχέση με τις υπόλοιπες λέξεις του κειμένου, σε αντίθεση με την απλή αναπαράσταση που τις παρουσιάζει η μέθοδος one hot vector representation που απλά αναπαριστά τις λέξεις σε διανύσματα των 0 και 1, ανάλογα με την θέση στην οποία εντοπίστηκε η λέξη μέσα στο κείμενο [14]. Όπως καταλαβαίνουμε αυτή η διαδικασία δεν μπορεί να αποδώσει κατάλληλα την σημασιολογική ανάλυση των λέξεων. Για τον λόγο αυτό χρειαζόμαστε λεξικά που να μπορούν να εντοπίζουν τις σημασιολογικές ομοιότητες των λέξεων μέσα στα έγγραφα. Αυτό επιτυγχάνεται με το να εκπαιδεύονται μοντέλα χωρίς επίβλεψη πάνω σε εκπαιδευόμενα έγγραφα ώστε να μπορέσουν να αποτυπώσουν τις λέξεις του κειμένου σε διανύσματα αριθμών.

Σύμφωνα με τον ερευνητή του [15], υπάρχουν τρία γνωστά μοντέλα για την διαδικασία word embedding και είναι: Latent semantic analysis (LSA), Word2Vectors (Word2vec) και Global Vectors (Glo Ve). Εμείς σε αυτή την εργασία θα χρησιμοποιήσουμε προ-εκπαιδευμένα λεξικά όπου οι λέξεις των αγγλικών και ελληνικών κειμένων θα αντιπροσωπεύονται από διανύσματα 300 διαστάσεων που θα περιέχουν χρήσιμα σημασιολογικά στοιχεία των λέξεων για τα μοντέλα βαθιάς μάθησης (deep learning). Στην περίπτωση των αγγλικών κειμένων το αρχείο που θα χρησιμοποιήσουμε έχει εκπαιδευτεί επάνω σε tweets με την χρήση του μοντέλου FastText ενώ για τα ελληνικά κείμενα έχει εκπαιδευτεί πάνω σε κείμενα της ελληνικής Wikipedia . Ωστόσο για λόγους κατανόησης και ευκρίνειας θα αναλύσουμε και τα τέσσερα μοντέλα.

- **Latent Semantic Analysis:** Το LSA είναι μια ισχυρή στατιστική τεχνική και βασίζεται σε δύο κύρια βήματα [15]. Το πρώτο βήμα είναι η κατασκευή ενός πίνακα που θα περιέχει όλους τους όρους των εγγράφων, έστω M. Το μέγεθος του πίνακα M είναι n*m, όπου οι σειρές αντιστοιχούν σε m όρους και οι στήλες αντιστοιχούν σε n έγγραφα, ενώ το M[i,j] αντιστοιχεί στην συχνότητα του όρους i στο έγγραφο j. Το δεύτερο βήμα είναι η αναπαράσταση / αποσύνθεση μοναδικής τιμής, όπου το M θα αποσυντεθεί, σύμφωνα με την παρακάτω εξίσωση, σε τρεις πίνακες U, V^t, που είναι δύο ορθογώνιοι πίνακες και στον διαγώνιο πίνακα S.

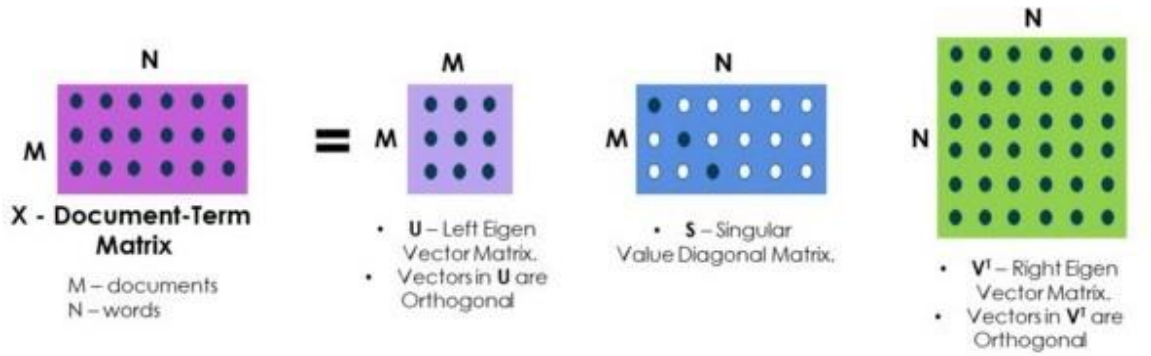
$$M = U * S * V^T$$

(2.2)

Τέλος, θα χρησιμοποιηθούν μόνο οι k μεγαλύτερες μοναδικές τιμές και οι αντίστοιχοι μοναδικοί φορείς τους από τους αντίστοιχους πίνακες U, V^t, προκειμένου να μειωθεί ο σημασιολογικός χώρος που αντιστοιχεί στο M_k.

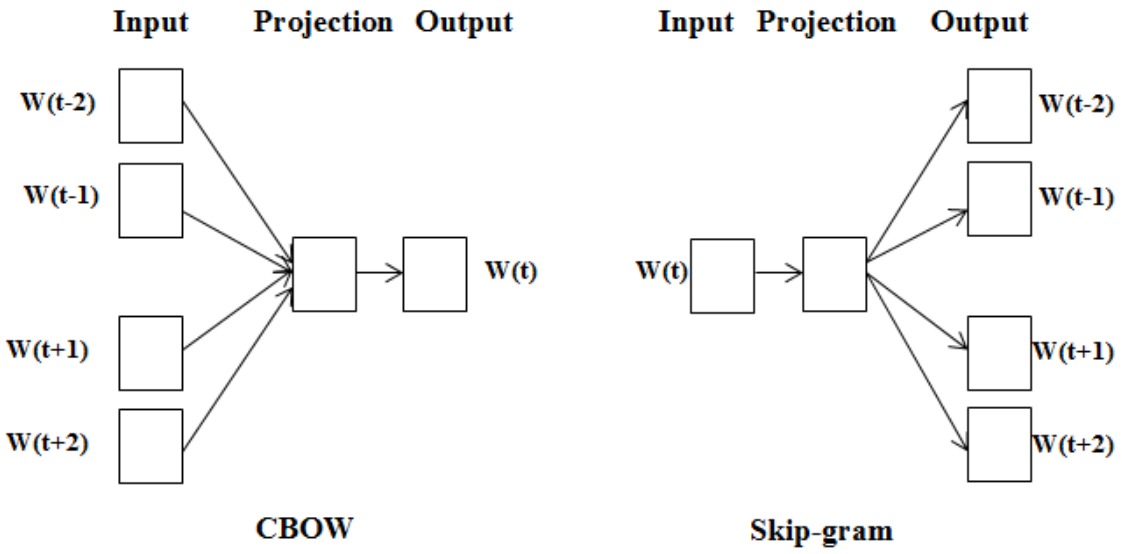
$$M_k = U_k * S_k * V_k^t$$

(2.3)

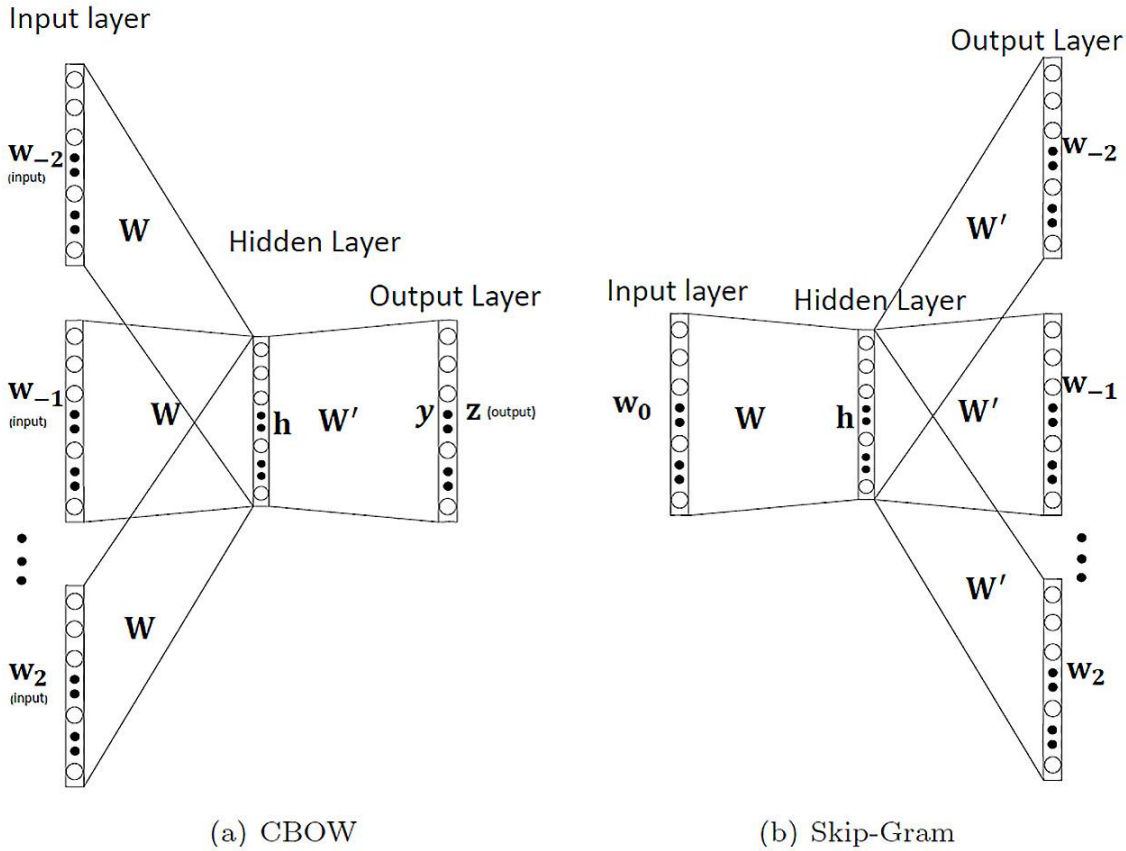


Εικόνα 2.1: Απεικόνιση του μαθηματικού μοντέλου του LSA από το [16]*, όπου το X είναι ο πίνακας όρου-εγγράφου που αποτελείται από όλες τις λέξεις που υπάρχουν σε καθένα από τα έγγραφα. Οι πίνακες μετατόπισης U και V είναι ορθοκανονικοί πίνακες όπου κάθε σειρά είναι ορθογώνια διανύσματα. Το S είναι ένας διαγώνιος πίνακας μονής τιμής με τις ιδιοτιμές του να υπάρχουν κατά μήκος της διαγώνιου. Ωστόσο ο V είναι ο πίνακας που είναι κεντρικός στη μοντελοποίηση.*

- **Word2Vectors:** Το μοντέλο Word2Vec για την διαδικασία του word embedding αναφέρθηκε για πρώτη φορά από τον Tomas Mikolov και την ομάδα του στο ερευνητικό άρθρο [17]. Βάσει του ερευνητικού άρθρου [18], για την μέθοδο word2vec, υπάρχουν δύο βασική αλγόριθμοι μηχανικής μάθησης που την απαρτίζουν, ο αλγόριθμος continuous bag of words (CBOW) και ο αλγόριθμος continuous skip-gram (CSG). Με χρήση του αλγορίθμου CBOW, προβλέπει την τρέχουσα λέξη με βάση του σημασιολογικού περιεχομένου του κειμένου στο οποίο βρίσκεται. Από την άλλη ο αλγόριθμος Skip-gram προβλέπει / υπολογίζει τις υπόλοιπες λέξεις του κειμένου βάση της τρέχουσας λέξης. Σε αντίθεση με το τυπικό μοντέλο bag of words, το continuous bag of words χρησιμοποιεί μια κατανεμημένη αναπαράσταση του περιεχομένου. Έτσι κάνοντας χρήσει αυτών των δύο αλγορίθμων, η μέθοδος WordVectors αποτυπώνει την ανάλυση των λέξεων.



Εικόνα 2.2: Η κύρια διαφορά μεταξύ αυτών των δύο μεθόδων είναι ότι το CBOW χρησιμοποιεί το πλαίσιο για να προβλέψει μια λέξη-στόχο ενώ το skip-gram χρησιμοποιεί μια λέξη για να προβλέψει ένα πλαίσιο στόχου [19].



Εικόνα 2.3: Η εικόνα δείχνει πιο αναλυτικά τις δύο αρχιτεκτονικές των μοντέλων CBOW και Skip-Gram [20].

- **Global Vectors:** Η μέθοδος Global Vector δημοσιεύτηκε για πρώτη φορά στο επιστημονικό άρθρο [21]. Βασίζεται σε εμφανίσεις λέξεων μέσα σε μια συλλογή κειμένων και βασίζεται κυρίως σε δύο βασικά βήματα όπως αναφέρει και το [15].
    - **Κατασκευή πίνακα**: Το πρώτο βήμα είναι η κατασκευή ενός πίνακα συν-εμφανίσεων X από μια συλλογή κειμένων προς εκπαίδευση, όπου το $x_{i,j}$ είναι η συχνότητα εμφάνισης της λέξης / όρου i που συνυπάρχει με την λέξη j. Στην παρακάτω εξίσωση υπολογίζεται ο συνολικός αριθμός εμφανίσεων της λέξης i στην συλλογή εγγράφων. Το V που συναντάται μέσα στην μαθηματική έκφραση αντιστοιχεί στο μέγεθος της συλλογής.

$$\mathrm{x_i j} = \sum_{\mathrm{k}}^{\mathrm{v}} \mathrm{X}_{ik} \qquad (2.4)$$

    - **Παραγοντοποίηση:** Το δεύτερο βήμα είναι η παραγοντοποίηση του X, προκειμένου να ληφθούν τα διανύσματα. Σύμφωνα με το [15], το ερευνητικό άρθρο που δημοσίευσε την μέθοδο global vectors [21], έδειξε ότι σε σύγκριση με τις ακατέργαστες πιθανότητες, οι αναλογίες συμβάλουν στη μείωση του θορύβου προσδιορίζοντας τις σχετικές λέξεις από μη σχετικούς όρους του κειμένου. Για το λόγο αυτό, χρησιμοποιήσαν τον ακόλουθο μαθηματικό μοντέλο.

$$\mathrm{F}\left(\mathrm{w_i} - \mathrm{w_j}, \mathrm{w'_k}\right) = \frac{\mathrm{P_{ik}}}{\mathrm{P_{jk}}} \qquad (2.5)$$

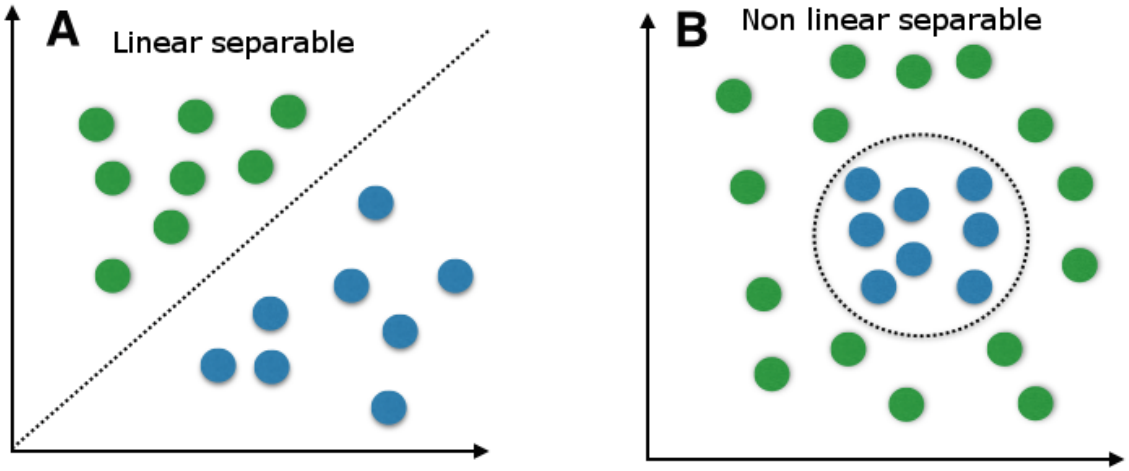
    Όπου το w_i, w_j και το w_k είναι τρεις διανυσματικές λέξεις, το $P_{ik} = \frac{X_{ik}}{X_i}$ είναι η πιθανότητα της λέξης k να εμφανιστεί μέσα στο περιεχόμενο της λέξης i και το $w_k{}'$ είναι σχετικά σημασιολογικά διανύσματα λέξης βάσει περιεχομένου στο οποίο βρίσκεται η λέξη.

- **FastText:** Όπως αναφέρει το [22], η εφαρμογή της μεθόδου embedding πάνω σε σπάνιες λέξεις μπορεί μερικές φορές να μην έχει τα επιθυμητά αποτελέσματα που θα είχαμε αν εφαρμόζαμε την μέθοδο σε συχνούς όρους κειμένων. Σε αυτό το πρόβλημα έρχεται βοηθήσει η μέθοδος FastText χρησιμοποιώντας πληροφορίες αποκλειστικά από λέξεις κλειδιά (subwords), δηλαδή από λέξεις που χρησιμοποιούν μερικά από τα γράμματα ενός θέματος για την σωστή αναπαράσταση των σπάνιων όρων. Για παράδειγμα η λέξη “meal” είναι subword της λέξης “Michael” καθώς χρησιμοποιεί τέσσερα από τα 7 γράμματα του ονόματος “Michael”. Η μέθοδος FastText βασίζεται στο μοντέλο skip-gram, όπου κάθε λέξη αναπαρίσταται ως bag of characters n-grams. Μια διανυσματική αναπαράσταση σχετίζεται με καθένα από τους χαρακτήρες των n-gram και από τον μέσο όρο αυτών των διανυσμάτων, δίνοντας έτσι την τελική αναπαράσταση της λέξης

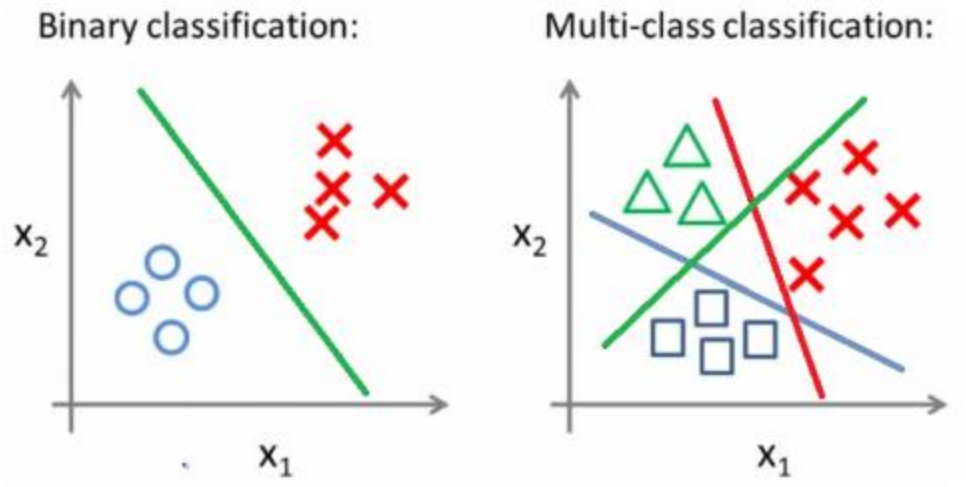
3. The Machine learning Architectures

3.1. Support Vector Machines (SVM)

Ο αλγόριθμος των support vector machines (SVM), αρχικά σχεδιάστηκε για εφαρμογές που διαχειρίζονταν δυαδικά προβλήματα [23], δηλαδή προβλήματα που τα δείγματα/δεδομένα τους, εισάγονται σε μια από τις δύο προκαθορισμένες κλάσεις ετικετών βάση κάποιων παραμέτρων. Τέτοιου είδους προβλήματα είναι η ανίχνευση ψευδών ειδήσεων , η ανίχνευση ανεπιθύμητων μηνυμάτων, ποιοτικός έλεγχος, ανίχνευση κακόβουλου λογισμικού. Στην Εικόνα 3.1 βλέπουμε ένα linear και ένα non-linear κατηγοριοποιητή που χρησιμοποιήθηκε για δισδιάστατη βάση (συνήθως σπάνια έχουμε δεδομένα που αναπαρίστανται σε τόσο λίγες διαστάσεις). Ωστόσο υπάρχουν και προβλήματα όπου τα εξεταζόμενα δεδομένα δεν μπορούν να περιοριστούν μόνο σε δύο προκαθορισμένες κλάσεις. Τέτοιου είδους προβλήματα είναι [24] η αναγνώριση χαρακτήρων, βιομετρική αναγνώριση και ασφάλεια, αναγνώριση προσώπου. Έτσι ανάλογα με το πρόβλημα, υπάρχουν και οι αντίστοιχες μέθοδοι και αλγόριθμοι που επιλέγονται για να το επιλύσουν. Στην Εικόνα 3.2 βλέπουμε την διαφορά μεταξύ της binary και της multi-class κατηγοριοποίησης.

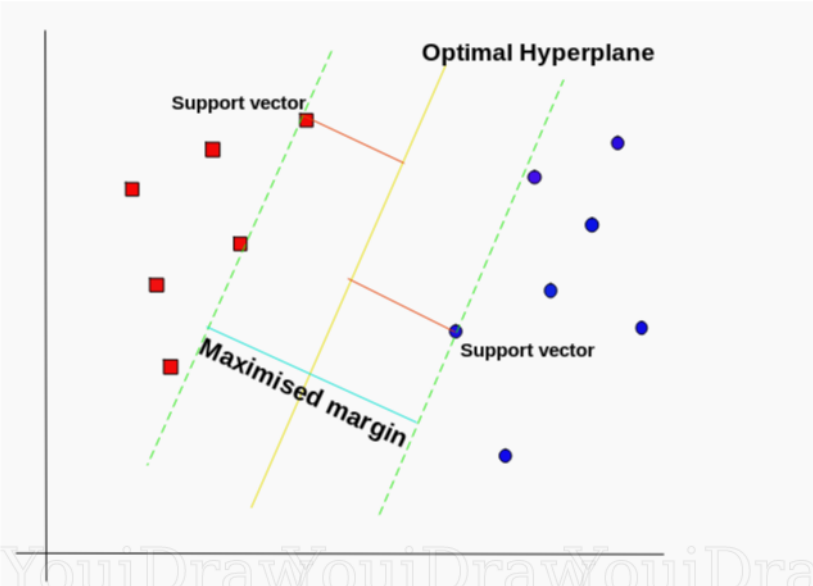


Εικόνα 3.1: Απεικόνιση Linear vs Non Linear Support Vector Machines [25].



Εικόνα 3.2: Απεικόνιση binary και multi-class classification για λόγους ευκρίνειας [26].

Όπως αναφέρεται στο [23] οι binary SVM είναι βασισμένοι πάνω στην αρχή της ελαχιστοποίησης του διορθωτικού κινδύνου από την υπολογιστική θεωρία (structural risk minimization principle from the computational learning theory). Τα binary SVM μοντέλα επιδιώκουν να διαχωρίζουν τα σημεία εκπαίδευσης σε δύο τάξεις και παίρνουν αποφάσεις για το που θα κατατάξουν τα δεδομένου που εξετάζονται βάση των βοηθητικών διανυσμάτων (support vectors) που επιλέγονται ως τα μόνα αποτελεσματικά στοιχεία για τα ζεύγη εκπαίδευσης του μοντέλου. Με τον όρο support vectors εννοούμε τα διανύσματα των σημείων που δημιουργούνται από τα πιο κοντινά σημεία και των δύο κλάσεων ως προς την γραμμή διαχωρισμού τους, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 3.3: Ο αλγόριθμος SVM έχει σχεδιαστεί με τέτοιο τρόπο ώστε να αναζητά σημεία στο γράφημα που βρίσκονται απευθείας στη διαχωριστική γραμμή που βρίσκεται πλησιέστερα. Αυτά τα σημεία ονομάζονται διανύσματα υποστήριξης. Στη συνέχεια, ο αλγόριθμος υπολογίζει την απόσταση μεταξύ των διανυσμάτων αναφοράς

και του επιπέδου διαίρεσης. Αυτή η απόσταση ονομάζεται κενό. Ο κύριος στόχος του αλγορίθμου είναι να μεγιστοποιήσει την απόσταση μεταξύ των δυο βοηθητικών διανυσμάτων.

Δεδομένου ενός συνόλου N γραμμικών διαχωρισμένων σημείων $S = x_i \in R \mid i = 1, 2, \ldots, N$, όπου κάθε x_i σημείο ανήκει σε μια από τις δύο κλάσεις, που ορίζονται μαθηματικά ως $y_i\ \varepsilon\ -1, +1$. Το διάνυσμα που διαχωρίζει τα στοιχεία που ανήκουν στο S στις 2 κλάσεις y_i ονομάζεται hyper-plane. Το διάνυσμα hyper-plane μπορεί να οριστεί σαν ένα ζεύγος τιμών τι μορφής (w,b) που ικανοποιεί την συνθήκη, διαδοχικά δημιουργούνται τα παρακάτω διανύσματα, παράλληλα στο προαναφερθέν διάνυσμα:

$$w * x + b = 0$$

$$x * x_i \geq 1, if\ y_i = +1$$

$$x * x_i \leq -1, if\ y_i = -1$$

(3.1)

Έτσι βάσει των παραπάνω ο στόχος ενός binary SVM αλγορίθμου είναι να βρει το πιο σωστό διαχωριστικό hyper-plane (OSH) που έχει το μέγιστο περιθώριο των δύο πλευρών διαχωρισμού. Όπως αναφέρει και το άρθρο[27], [28], κατά την διάρκεια της κατηγοριοποίησης, ο αλγόριθμος SVM τις περισσότερες φορές παίρνει αποφάσεις που βασισμένες στο optimal separating hyperplane - καλύτερο διαχωριστικό hyper-plane (OSH) αντί ολόκληρης της βάσης δεδομένων των στοιχείων εκπαίδευσης. Στην ουσία βρίσκει από ποια πλευρά του OSH εντοπίζεται το σημείο δοκιμής. Αυτή η ιδιότητα καθιστά το SVM πολύ ανταγωνιστικό σε σύγκριση με άλλους παραδοσιακούς αλγορίθμους αναγνώρισης, όσον αναφορά της αποτελεσματικότητας και ακρίβειας. Όπως αναφέρεται και στο [27], παρά τα πολλά πλεονεκτήματα των αλγορίθμων SVM, υπάρχουν κάποιες πολύ έντονες αδυναμίες μεταξύ των οποίων είναι: η επιλογή παραμέτρων, αλγοριθμική πολυπλοκότητα που επηρεάζει το χρόνο εκπαίδευσης του ταξινομητή σε μεγάλα σύνολα δεδομένων, ανάπτυξη βέλτιστων ταξινομητών για προβλήματα πολλαπλών τάξεων και την απόδοση των SVM σε μη ισορροπημένα δεδομένα.

Σε αυτή την διπλωματική εργασία το πρόβλημα που αντιμετωπίζουμε, ορίζεται ως binary πρόβλημα, οπότε η παραπάνω αρχιτεκτονική καλύπτει τις ανάγκες μας, αλλά για λόγους καλύτερης κατανόησης των μοντέλων SVM, στην παρακάτω παράγραφο θα αναλύσουμε και τα μοντέλα SVM που αντιμετωπίζουν προβλήματα με περισσότερες από δύο κλάσεις αντιστοίχισης.

3.2. Multi-Class Support Vector Machine

Όπως προαναφέρθηκε το μοντέλο των support vector machines (SVM) ξεκίνησε ως δυαδικός κατηγοριοποιητής αλλά στην συνέχεια λόγο των εκτενών σε αριθμό προβλημάτων που δεν περιοριζόντουσαν σε δύο κλάσεις, οι ερευνητές έστρεψαν την προσοχή τους στο να προσαρμόσουν τα support vector machines σε αυτά τα προβλήματα [24], [29].[27] Προκειμένου να εφαρμοστούν τα SVM σε τέτοια προβλήματα, είναι αναγκαίο να μετατραπούν αυτά τα προβλήματα σε πολλαπλά δυαδικά υποπροβλήματα. Για να επιτευχθεί αυτό υπάρχουν δύο τύποι αλγορίθμων βασισμένοι πάνω στα SVM.

One versus all:[29], [30] Η προσέγγιση των multiclass support vector machines είναι να προσεγγίσουν n δεδομένα της μορφής (xi , yi), και c κλάσεις έτσι ώστε να κατασκευάσουν j

δυαδικά SMV μοντέλα. Όταν εκπαιδεύεται το jth SVM μοντέλο, ορίζεται η κλάση j ως θετική και οι υπόλοιπες ως αρνητικές. Αν ο αριθμός των δεδομένων που εκπαιδεύονται σε κάθε κλάση είναι ισορροπημένος, αυτό το υποπρόβλημα είναι ένα μη ισορροπημένο δυαδικό πρόβλημα ταξινόμησης και μπορεί να αναπαρασταθεί,

$$\min_{\mathrm{wj}},_{\mathrm{bj}} = \left|\left|wj\right|\right|_2^2 + C\sum_{i=1}^{n}\xi_i^j$$

$$s.t\ w_j^T x_i + b_j \geq 1 - \xi_i^j, if\ y_i = j$$

$$w_j^T x_i + b_j \leq -1 + \xi_i^j, if\ y_i \neq j$$

(3.2)

One versus one: η μέθοδος / αλγόριθμος one versus one λύνει αυτό το πρόβλημα με την εκπαίδευση περισσότερων δυαδικών μοντέλων SVM.[29], [30] Σε αυτή την στρατηγική εκπαιδεύονται δυαδικά μοντέλα SVM για κάθε δύο τάξεις, οπότε υπάρχουν c(c − 1)/2 μοντέλα συνολικά. Για τις κλάσεις j και k, το μέγιστο περιθώριο μεταξύ τους είναι wjkxi + bjk = 0 και μπορεί να βρεθεί μέσω του προβλήματος που επιλύεται με την μαθηματική αναπαράσταση, όπως φαίνεται παρακάτω

$$\min_{\mathrm{wj_k}} = \frac{1}{2}\left|\left|wj\right|\right|_2^2 + C\sum_{i=1}^{n}\xi_i^{j_k}$$

$$w_{j_k}^T x_i + b_{j_k} \geq 1 - \xi_i^{j_k}, if\ y_i = j$$

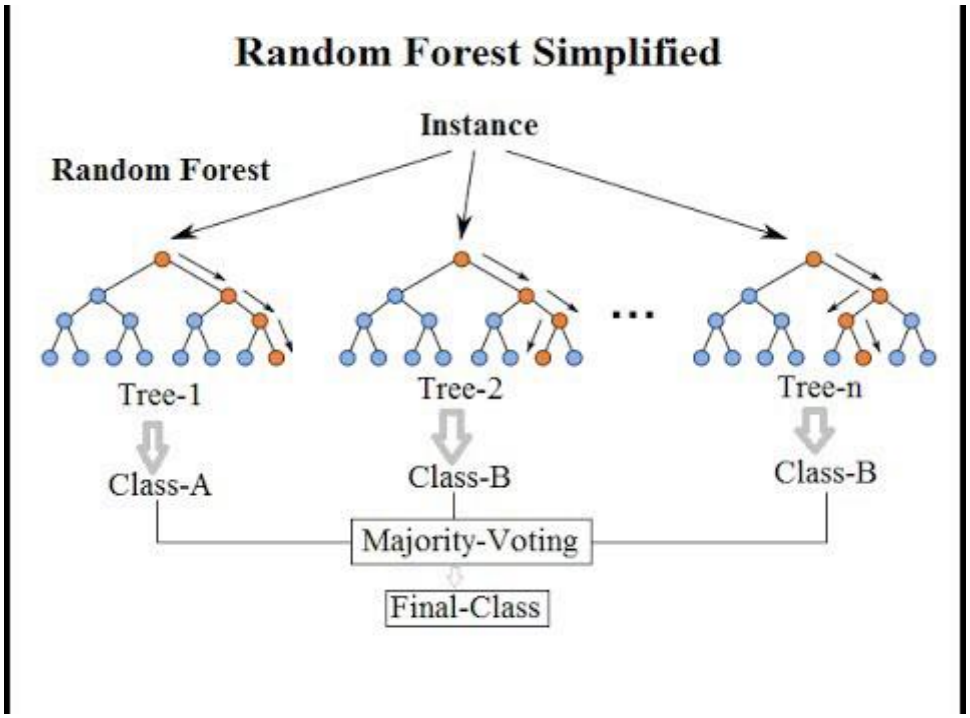
$$w_{j_k}^T x_i + b_{j_k} \leq -1 + \xi_i^{j_k}, if\ y_i \neq k$$

(3.3)

3.3. Random Forest

Όπως αναφέρεται στο [31] το Random Forest (RF) είναι μια μέθοδος μάθησης για την ταξινόμηση που λειτουργεί με την κατασκευή ενός πλήθους δέντρων αποφάσεων κατά το χρόνο εκπαίδευσης και την έξοδο της τάξης που είναι ο τρόπος των τάξεων των μεμονωμένων δέντρων. Το RF διορθώνει τη συνήθεια των δέντρων αποφάσεων να ταιριάζουν στο εκπαιδευτικό τους σύνολο. Όπως το SVM, το RF έχει επίσης αναγνωριστεί ως ένας από τους πιο ακριβείς ταξινομητές.

Πιο αναλυτικά ο αλγόριθμος Random Forest δημοσιεύθηκε επίσημα το 2001 από τον Leo Breiman στο ερευνητικό άρθρο [32]. Ο αλγόριθμος αυτός όπως αναφέρει το [33] ανήκει στην «οικογένεια» των μεθόδων που λαμβάνουν ένα δέντρο αποφάσεων ως μεμονωμένη πρόβλεψη, βασιζόμενοι στις μεθόδους Bagging Randomizing και Random Subspace excuse boosting. Συνδυάζει τις έννοιες των τυχαίων υποδιαστημάτων και του “bagging”. Ο αλγόριθμος decision tree forest εκπαιδεύεται σε δέντρα πολλαπλών αποφάσεων με βάση ελαφρώς διαφορετικά υποσύνολα δεδομένων.



Εικόνα 3.4: Απεικόνιση λειτουργίας του μοντέλου Random Forest [34].

Σύμφωνα με το [33], ο random forest αλγόριθμος είναι ένας από τους καλύτερους μεταξύ των αλγορίθμων ταξινόμησης, διότι είναι ικανός να ταξινομήσει μεγάλες ποσότητες δεδομένων με ακρίβεια. Πρόκειται για μια μέθοδο μάθησης για ταξινόμηση που χρησιμοποιεί αναδρομή (regression) όπου κατασκευάζει μια σειρά δένδρων αποφάσεων κατά την διάρκεια της εκπαίδευσής του και προσδιορίζει την τελική κλάση από μεμονωμένα δέντρα.

3.4. Naive Bayes

Ο ταξινομητής Naive Bayes είναι ένας πιθανοτικός αλγόριθμος που βασίζεται στο θεώρημα του Bayes. Σε αυτόν το αλγόριθμο κατηγοριοποίησης κειμένου, η παρουσία ή η απουσία μιας λέξης σε ένα κείμενο καθορίζει το αποτέλεσμα της πρόβλεψης [35]. Με άλλα λόγια, σε κάθε επεξεργασμένο όρο εκχωρείται μια πιθανότητα ότι ανήκει σε μια συγκεκριμένη κατηγορία. Αυτή η πιθανότητα υπολογίζεται από τις εμφανίσεις του όρου στα εκπαιδευτικά έγγραφα όπου οι κατηγορίες είναι ήδη γνωστές. Όταν υπολογίζονται όλες αυτές οι πιθανότητες, ένα νέο έγγραφο μπορεί να ταξινομηθεί σύμφωνα με το άθροισμα των πιθανοτήτων για κάθε κατηγορία κάθε όρου που εμφανίζεται στο έγγραφο.

Πιο αναλυτικά, για ένα κείμενο d (document), από όλες τις κλάσεις c ∈ C ο ταξινομητής επιστρέφει την κλάση c’ που έχει την μέγιστη πιθανότητα του δεδομένου εγγράφου να ανήκει σε αυτή [12].

$$c' = argmax_{c \in C} P(c \mid d) \qquad (3.4)$$

Σύμφωνα με το [17], δεδομένου ενός συνόλου r διανυσμάτων εγγράφων D = $\{d_1, \ldots, d_r\}$, ταξινομημένα μαζί με ένα σύνολο C κατηγοριών / κλάσεις q, C = $\{c_1, \ldots, c_q\}$, ο ταξινομητής

Naive Bayes εκτιμά τις πιθανότητες κάθε κλάσης c_k που έχει ένα έγγραφο d_j όπως περιγράφεται από την παρακάτω εξίσωση:

$$P(C_k|d_j) = \frac{P(c_k)P(d_j|c_k)}{P(d_j)}$$

(3.5)

Συνεχίζοντας το [12], αναφέρει ότι υπολογίζουμε την πιο πιθανή κλάση c' έχοντας κάποιο έγγραφο d και επιλέγοντας την κλάση που έχει το υψηλότερο γινόμενο ανάμεσα από δύο πιθανότητες, την προηγούμενη πιθανότητα την κλάσης P(c) και την πιθανότητα P(c|d) όπως αυτή απεικονίζεται στην παρακάτω σχέση:

$$c' = argmax_{c \in C}\, P(c|d)\, P(c)$$

(3.6)

Όπου το $P(c|d)$ είναι η πιθανότητα του εγγράφου να ανήκει στην κλάση με το υψηλότερο γινόμενο και το $P(c)$ είναι η προηγούμενη πιθανότητα της κλάσης.

Ωστόσο το [12] τονίζει ότι χωρίς κάποιες απλοποιήσεις στην σχέση 9 που απεικονίζεται πιο πάνω, είναι ακόμα πολύ δύσκολο να υπολογιστούν οι πιθανότητες των κλάσεων. Για αυτό τον λόγο ο κατηγοριοποιητής Naive Bayes κάνει δύο ακόμα απλοποιήσεις στον αλγόριθμό του:

- Η πρώτη απλοποίηση είναι ότι κάνει χρήση της μεθόδου bag of words που συζητήθηκε σε προηγούμενο κεφάλαιο, δηλαδή ότι παίρνει μόνο τις φορές εμφάνισης του κάθε όρου μέσα στο κείμενο και όχι και την θέση στην οποία εντοπίστηκε. Έτσι τα χαρακτηριστικά τη σχέσης 10 (f_1, f_2, …, f_n) συμβολίζουν μόνο την «ταυτότητα» των λέξεων και όχι την θέση.
- Η δεύτερη απλοποίηση ονομάζεται, υπόθεση του Naive Bayes και αναφέρεται στην υπόθεση ότι η πιθανότητες $P(f_i|c)$ είναι ανεξάρτητες από την δοσμένη κλάση c και ως εκ τούτου μπορούν να πολλαπλασιαστούν ως «αφελείς» με τον εξής τρόπο:

$$P(f_1, f_2, \dots, f_n|\, c) = P(f_1|c) * P(f_2|c) * \dots * P(f_n|c)$$

(3.7)

Έτσι συνοψίζοντας με βάσει τα παραπάνω, η τελική εξίσωση με την οποία ο αλγόριθμος / ταξινομητής Naive Bayes αποφασίζει την τελική κλάση είναι:

$$class = argmax_{c \in C}\, P(c) \prod_{f \in F} P(f|c)$$

(3.8)

Ωστόσο σύμφωνα με το βιβλίο [12], είναι αρκετά συνηθισμένο προκειμένου να εφαρμόσουμε τον πιθανοτικό αλγόριθμο Naive Bayes στην διαδικασία κατηγοριοποίησης κειμένου θα πρέπει να συμπεριλάβουμε και τις θέσεις στις οποίες εντοπίστηκαν οι λέξεις του κειμένου, εφαρμόζοντας ένα αρχείο με όλες τις θέσεις των λέξεων που υπάρχουν σε ένα έγγραφο (σε αυτή την διπλωματική εργασία δεν θα χρησιμοποιήσουμε αυτή την προσέγγιση αλλά αναφέρθηκε για λόγους ευκρίνειας). Έτσι η εξίσωση 10, αλλάζει ελάχιστά και γίνεται:

$$class = argmax_{c \in C}\, P(c) \prod_{i \in positions} P(w_i|c)$$

(3.9)

Για αν αποφευχθεί η υπερχείλιση και για να αυξηθεί η ταχύτητα του αλγορίθμου η εξίσωση 11 μετατρέπεται στην γενική μορφή:

$$class = argmax_{c \in C} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

(3.10)

3.5. K-Nearest Neighbor

Ο αλγόριθμος K-Neared Neighbor (KNN) σύμφωνα με το [36] είναι μια μη παραμετρική τεχνική που χρησιμοποιείται για την ταξινόμηση κειμένων και όχι μόνο. Επιπρόσθετα το ερευνητικό κείμενο [37], αναφέρει ότι ο κατηγοριοποιητής KNN είναι μια απλή αλλά αποτελεσματική μέθοδος κατηγοριοποίησης κειμένου με τρία αρκετά σοβαρά ελαττώματα:

- Η πολυπλοκότητα των υπολογιστικών δειγμάτων της ομοιότητας είναι αρκετά μεγάλη.
- Η απόδοση του αλγορίθμου επηρεάζεται αρκετά εύκολα από ένα μόνο δείγμα εκπαίδευσης, το οποίο θα μπορούσε να περιείχε θόρυβο.
- Ο K-Neared Neighbor (KNN) δεν δημιουργεί μοντέλο ταξινόμησης. Ως αποτέλεσμα, δεν ταιριάζει καλά σε εφαρμογές, όπου τα δεδομένα τους ενημερώνονται δυναμικά και απαιτείται η ανάλυσή τους σε πραγματικό χρόνο

Ωστόσο στην περίπτωση της ανάλυσης των tweets σε δύο κλάσεις (real, fake) και έχοντας πεπερασμένο αριθμών κειμένων μέσα στην βάση δεδομένων μας, το τελευταίο πρόβλημα δεν επιβαρύνει τον αλγόριθμό (KNN).

Σύμφωνα με το [37], η βασική ιδέα του του (KNN) είναι: με δεδομένο ένα έγγραφο δοκιμής (test document) x, βρίσκει τους κοντινότερους «γείτονες» μεταξύ όλων των εγγράφων εκπαίδευσης και βαθμολογεί τις υποψήφιες κατηγορίες με βάση την κατηγορία των γειτόνων. Η ομοιότητα του εγγράφου x και κάθε γειτονικού εγγράφου είναι η βαθμολογία της κατηγορίας του γειτονικού εγγράφου.

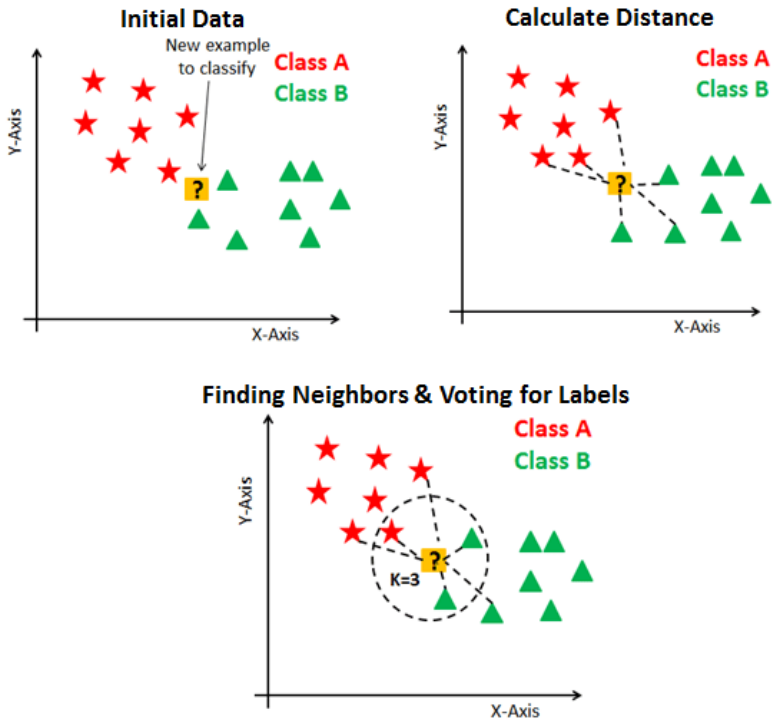
Εάν πολλά από τα πλησιέστερα γειτονικά έγγραφα ανήκουν στην ίδια κατηγορία, τότε το άθροισμα την βαθμολογία αυτής της κατηγορίας είναι η βαθμολογία ομοιότητας της κατηγορίας σε σχέση με το δοκιμαστικό έγγραφο. Με την ταξινόμηση των βαθμολογιών των υποψήφιων κατηγοριών, το σύστημα εκχωρεί την υποψήφια κατηγορία με την υψηλότερη βαθμολογία στο δοκιμαστικό / εξεταζόμενο έγγραφο x. Ο κανόνας με τον οποίο ο αλγόριθμος K-Neared Neighbor (KNN) παίρνει μια απόφαση μπορεί να αναπαρασταθεί ως:

$$f(x) = argmax_j Score\left(x, C_j\right) = \sum_{d \in kNN} sim(x, d_i) y\left(d_i, C_j\right)$$

(3.11)

Όπου το $f(x)$ είναι η ετικέτα που αποδίδεται στο εξεταζόμενο έγγραφο x, το $Score(x, C_j)$ είναι βαθμολογία της υποψήφιας κατηγορίας C_j σε σχέση με το x, το $sim(x, d_i)$ είναι η ομοιότητα του κειμένου x σε σχέση με τα δεδομένα προς εκπαίδευσης (training documents) d_i, το $y\left(d_i, C_j\right) \in \{0,1\}$ είναι η τιμή δυαδικής κατηγορίας του εκπαιδευόμενου εγγράφου d_i σε σχέση με το C_j (το y = 1, υποδεικνύει ότι το έγγραφο d_i είναι μέρος της κατηγορίας C_j και ανάλογα με y = 0, υποδεικνύει ότι το έγγραφο d_i δεν είναι μέρος της κατηγορίας C_j.

Αυτή η προσέγγιση όπως προαναφέρθηκε, είναι μια αρκετά αποτελεσματική και εύκολη στην εφαρμογής της, χωρίς να περιέχει παραμέτρους.



Εικόνα 3.5: Απεικόνιση λειτουργίας του μοντέλου KNN. Η εικόνα πάρθηκε από το [38].

4. The Deep Learning Architectures

Οι παραδοσιακές τεχνικές ταξινόμησης κειμένου έχουν διάφορα πλεονεκτήματα, αλλά εξακολουθούν να έχουν πολλές ελλείψεις [39]. Προκειμένου να βελτιωθεί η ακρίβεια της ταξινόμησης προτείνεται η δομή μη παραδοσιακών αλγορίθμων ταξινόμησης, αυτή των μοντέλων βαθιάς μάθησης.

Σε αυτή την εργασία θα αναπτυχθούν τρείς κατηγορίες νευρωνικών δικτύων, των feed forward networks (π.χ. CNN), recurrent networks (π.χ. RNN) και μιας καινούργιας αρχιτεκτονικής νευρωνικών δικτύων, αυτής των Transformer neural networks όπου πάνω σε αυτή την αρχιτεκτονική στηρίχτηκε και το μοντέλο BERT (Bidirectional Encoder Representations from Transformers) της Google το 2018.

4.1. Feed-Forward Neural Networks

Πριν μιλήσουμε για τα συνελικτικά νευρωνικά δίκτυα πρέπει πρώτα να αναφερθούμε στην γενικότερη κατηγορία των Feed-Forward Neural Networks ώστε να μπορέσουμε στην συνέχεια να έχουμε καλύτερη αντίληψη για την αρχιτεκτονική των Convolutional Neural Networks.

Πρώτα από όλα τα Τεχνητά Νευρωνικά Δίκτυα (ANN) είναι υπολογιστικά συστήματα επεξεργασίας των οποίων η λειτουργία εμπνέεται σε μεγάλο βαθμό από τον τρόπο λειτουργίας των βιολογικών νευρικών συστημάτων (όπως ο ανθρώπινος εγκέφαλος). Στην περίπτωση των ANN κάθε νευρώνας έχει βαθμωτές εισόδους και εξόδους, με κάθε είσοδο να έχει και ένα σχετικό βάρος.

Ένας τεχνητός νευρώνας πολλαπλασιάζει κάθε είσοδο με το βάρος του και στη συνέχεια τα αθροίζει, εφαρμόζει μια μη γραμμική συνάρτηση στο αποτέλεσμα και την μεταδίδει στην έξοδό του. Τέλος οι νευρώνες που συνδέονται μεταξύ τους, σχηματίζουν ένα δίκτυο όπου η έξοδος ενός νευρώνα τροφοδοτεί τις εισόδους ενός ή περισσοτέρων νευρώνων όπως φαίνεται και στην παρακάτω Εικόνα 4.1.

Όπως βλέπουμε τα βελάκια αναπαριστούν τα βάρη όπου συνήθως είναι διανύσματα n-διαστάσεων, των εισόδων και εξόδων κάθε τεχνητού νευρώνα. Σε αντίθεση με το Input Layer που είναι η εκκίνηση του τεχνητού νευρωνικού δικτύου και του Output Layer που είναι το τέλος του, δηλαδή το τελικό αποτέλεσμα πιθανοτήτων σε μορφή δεκαδική τιμής που έδωσαν τα νευρωνικά δίκτυα, ιδιαίτερο ενδιαφέρον παρουσιάζουν το/τα Hidden Layer/s όπου κάνουν χρήση μη γραμμικών συναρτήσεων που εφαρμόζονται στην τιμή του νευρώνα πριν αυτή μεταβιβαστεί στην έξοδό του ώστε να μπορέσει να γίνει είτε η είσοδος ως τιμή εκκίνησης στο επόμενο νευρώνα του δικτύου, είτε να είναι η τελική τιμή βάρους για το τελικό αποτέλεσμα.

Πιο αναλυτικά σύμφωνα με το [39], η πιο απλούστερη μορφή νευρωνικού δικτύου είναι το perceptron που απεικονίζεται στην Εικόνα 4.2, όπου είναι μια γραμμική συνάρτηση με είσοδο:

$$ANN_{\text{Perceptro}}(x) = xW + b$$
$$x \in R^{d_{input}}, W \in R^{d_{input} * d_{output}}, b \in R^{d_{output}} \quad (4.1)$$

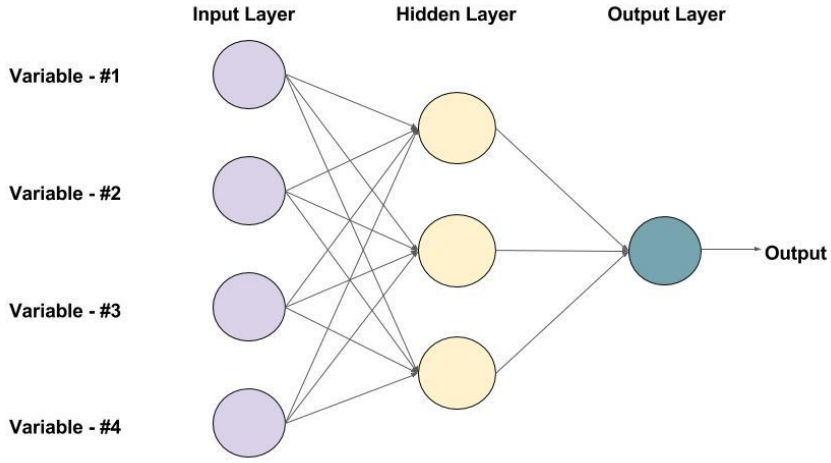
όπου W είναι το weight matrix και b είναι μια bias μεταβλητή. Όπως συνεχίζει να αναφέρει το άρθρο, για να προχωρήσουμε πέρα από τις γραμμικές συναρτήσεις όπως αυτού του απλού

νευρωνικού δικτύου (perceptron), πρέπει να εισάγουμε ένα μη γραμμικό κρυφό στρώμα (hidden layer). Ένα νευρωνικό δίκτυο τροφοδοσίας (feed-forward neural network) με ένα κρυφό στρώμα θα έχει την παρακάτω μαθηματική αναπαράσταση:

$$ANN_{WithHiddenLayer}(x) = g(xW^1 + b^1)W^2 + b^2$$
$$x \in R^{d_{input}}, W^1 \in R^{d_{input}*d_1}, b^1 \in R^{d_1}, W^2 \in R^{d_1*d_2}, b^2 \in R^{d_2}$$

(4.2)

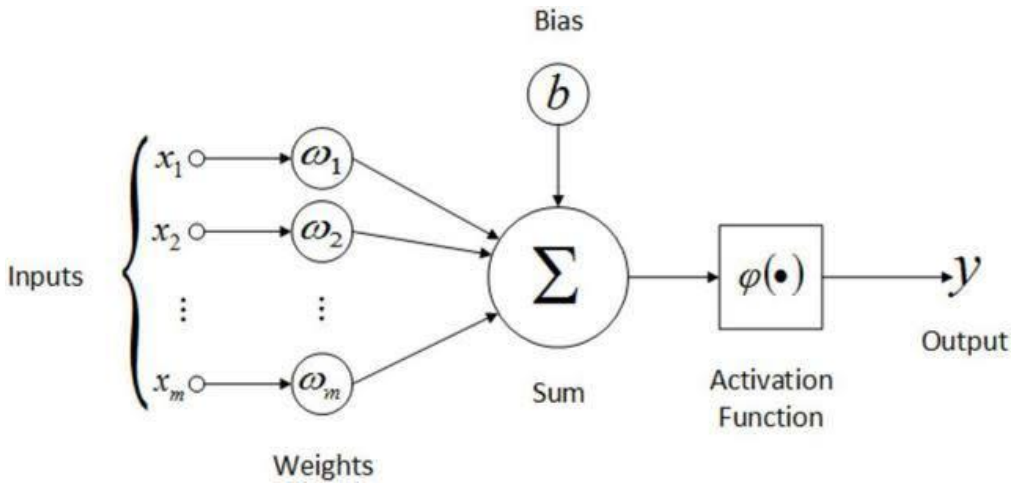
Τα W^1 και b^1 είναι ένας πίνακας και ένας όρος πόλωσης για τον πρώτο γραμμικό μετασχηματισμό της εισόδου, το g είναι μια μη γραμμική συνάρτηση που εφαρμόζεται βάσει στοιχείων που ονομάζεται επίσης και συνάρτηση ενεργοποίησης. Τέλος τα W^2 και b^2, είναι οι αντίστοιχοι όροι για τον δεύτερο γραμμικό μετασχηματισμό του νευρωνικού δικτύου.



Εικόνα 4.1: Είναι ένα κατευθυνόμενο άκυκλο γράφημα που σημαίνει ότι δεν υπάρχουν συνδέσεις ανάδρασης ή βρόχοι στο δίκτυο. Έχει ένα επίπεδο εισόδου, ένα επίπεδο εξόδου και ένα κρυφό στρώμα. Γενικά, μπορεί να υπάρχουν πολλά κρυφά επίπεδα. Κάθε κόμβος στο επίπεδο είναι ένας Νευρώνας, ο οποίος μπορεί να θεωρηθεί ως η βασική μονάδα επεξεργασίας ενός νευρωνικού δικτύου [40].

Αναλύοντας την παραπάνω μαθηματική έκφραση για την καλύτερη κατανόηση, το $xW^1 + b^1$ είναι ένας γραμμικός μετασχηματισμός της εισόδου x από d_{input} σε d_1 διαστάσεις. Το g εφαρμόζεται στην συνέχεια σε καθεμιά από τις διαστάσεις d_1 ενώ ο πίνακας W^2 μαζί με το διάνυσμα πόλωσης b_2, χρησιμοποιούνται στη συνέχεια για να μετατρέψουν το αποτέλεσμα στο διάνυσμα εξόδου διαστάσεων d_2. Η μη γραμμική συνάρτηση ενεργοποίησης g όπως αναφέρει το άρθρο [39], έχει κρίσιμο ρόλο στη ικανότητα του δικτύου να αναπαριστά σύνθετες συναρτήσεις. Χωρίς την μη γραμμικότητα της μεταβλητής g, το νευρονικό δίκτυο μπορεί να αναπαραστήσει μόνο γραμμικούς μετασχηματισμούς της εισόδου.

Σε αυτό το σημείο είναι καλό να αναφερθεί ότι δεν υπάρχει κάποιο όριο στα πόσα κρυφά στρώματα θα έχει ένα τεχνικό νευρωνικό δίκτυο καθώς συνήθως αυτό καθορίζεται από την περιπλοκότητα και το μέγεθος της εισόδου, ωστόσο όπως αναφέρει και το [41], ακόμη και ένα μόνο κρυφό στρώμα στο νευρονικό δίκτυο με μη πολυωνυμική εξίσωση για λειτουργία ενεργοποίησης, όπως για παράδειγμα η sigmoid, μπορεί να έχει πολύ καλά αποτελέσματα για το τεχνικό δίκτυό μας.



Εικόνα 4.2: Το Perceptron είναι ένας δυαδικός αλγόριθμος γραμμικής ταξινόμησης. Είναι ένας supervised τύπος μηχανικής μάθησης και η απλούστερη μορφή νευρωνικού δικτύου. Είναι μια βάση για νευρωνικά δίκτυα. Η εικόνα είναι από το το [42].

4.1.1. Non-Linear Activation Functions

Όπως προαναφέραμε πιο πάνω κάθε νευρώνας στο τεχνητό νευρωνικό δίκτυο, εφαρμόζει μια μη γραμμική συνάρτηση στο αποτέλεσμα των βαρών του πριν το μεταθέσει στον επόμενο νευρώνα του δικτύου. Σε αυτή την παράγραφο θα δούμε μερικές από τις πιο γνωστές μη γραμμικές συναρτήσεις ενεργοποίησης.

- **Sigmoid**: Η συνάρτηση αυτή είναι σχήματος "S", μετατρέποντας κάθε τιμή x στο πεδίο [0,1]. Επομένως, εάν η είσοδος στη συνάρτηση είναι, είτε ένας πολύ μεγάλος αρνητικός αριθμός είτε ένας πολύ μεγάλος θετικός αριθμός, η είσοδος είναι πάντα μεταξύ του 0 και 1. Η μαθηματική έκφραση της συνάρτησης φαίνεται στην παρακάτω εξίσωση.

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \qquad (4.3)$$

- **Hyperbolic Tangent (tanh):** Μοιάζει πολύ με την συνάρτηση Sigmoid που προαναφέρθηκε και μάλιστα έχει το ίδιο σχήμα, δηλαδή έχει και αυτή σχήμα που θυμίζει "S". Αυτή η συνάρτηση παίρνει οποιαδήποτε πραγματική τιμή ως είσοδο και εξάγει τιμές

στην περιοχή -1 έως +1, δηλαδή το πεδίο της σε αντίθεση με τις sigmoid είναι [-1,+1]. Όσο μεγαλύτερη είναι η είσοδος, τόσο πιο κοντά η τιμή της εξόδου θα είναι στο 1.0, ενώ όσο πιο μικρότερη είναι η είσοδος τόσο πιο κοντά η έξοδος θα είναι στο -1.0.

$$\tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

(4.4)

- **Rectifier (ReLU):** Η συγκεκριμένη συνάρτηση ενεργοποίησης είναι ίσως η πιο συχνά εφαρμοζόμενη συνάρτηση για τους νευρώνες στα κρυφά επίπεδα του τεχνητού δικτύου [43]. Είναι συνηθισμένη η χρήση της, επειδή είναι τόσο απλή στην εφαρμογή της όσο και αποτελεσματική για την υπέρβαση των περιορισμών που έχουν οι συναρτήσεις Sigmoid και Hyperbolic Tarngent (tanh). Πιο συγκεκριμένα είναι λιγότερο επιρρεπής σε εξαφανιζόμενες κλίσεις (vanishing gradients) που εμποδίζουν την εκπαίδευση των μοντέλων deep learning. Η συνάρτηση rectifier (ReLU) υπολογίζεται ως εξής:

$$\mathrm{ReLu(x)} = \max(0, x)$$

(4.5)

Αυτό σημαίνει ότι εάν η τιμή της εισόδου x είναι αρνητική τότε επιστρέφεται η τιμή 0.0, διαφορετικά επιστρέφεται η τιμή x.

4.1.2. Loss Functions

Οι συναρτήσεις απώλειας (loss functions) είναι ένα από τα πιο σημαντικά στοιχεία των τεχνητών νευρωνικών δικτύων, κυρίως κατά την εκπαίδευση τους. Οι συναρτήσεις απώλειας υπολογίζουν κατά πόσο μακριά η εκτιμώμενη τιμή που έδωσε το τεχνητό νευρωνικό δίκτυο απέχει από την πραγματική τιμή των δεδομένων. Σε αυτή την διπλωματική εργασία θα χρησιμοποιήσουμε την Binary Cross Entropy συνάρτηση καθώς έχουμε ένα binary classification πρόβλημα, ωστόσο για λόγους ευκρίνειας θα αναφερθούν και οι συναρτήσεις απώλειας: Binary Hinge, Log και Mean Squared Error.

- **Binary Hinge:** Για προβλήματα δυαδική ταξινόμησης, η έξοδος του δικτύου είναι μια βαθμωτή $\hat{y}$ και η προβλεπόμενη έξοδος y είναι μέσα σε πεδίο [+1,-1]. Μια ταξινόμηση θεωρείται σωστή εάν $y * \hat{y} > 0$, που σημαίνει ότι η μεταβλητή y και $\hat{y}$ έχουν το ίδιο πρόσημο. Η απώλεια άρθρωσης (Hinge Loss), είναι γνωστή και ως απώλεια SVM καθώς και δημιουργήθηκε για να υπολογίζει το μέγιστο περιθώριο από το hyperplane των κλάσεων και ορίζεται [39]:

$$\text{LossHinge}_{\text{binary}}(\hat{y}, y) = \max(0, 1 - y * \hat{y})$$

(4.6)

Η απώλεια είναι 0 όταν y και ŷ έχουν το ίδιο πρόσημο και $|\hat{y}| \geq 1$. Διαφορετικά η απώλεια είναι γραμμική.

- **Log Loss**: Η Log loss συνάρτηση είναι μια κοινή παραλλαγή της απώλειας άρθρωσης (Hinge Loss), η οποία μπορεί να θεωρηθεί ως μια "ήπια" έκδοση της απώλειας άρθρωσης με άπειρο περιθώριο.

$$LogLoss(\hat{y}, y) = \log(1 + \exp(-(\hat{y}_t - \hat{y}_k))$$

(4.7)

- **Binary Cross Entropy (BCE):** Αυτή είναι η πιο συνηθισμένη συνάρτηση απώλειας που χρησιμοποιείται για προβλήματα ταξινόμησης που έχουν δύο κλάσεις. Στην προκειμένη περίπτωση ο όρος "Entropy" έχει στατιστική ερμηνεία και είναι το μέτρο της τυχαιότητας στις πληροφορίες που υποβάλλονται σε επεξεργασία, ενώ η ορολογία cross-entropy είναι ένα μέτρο της διαφοράς της τυχαιότητας μεταξύ δύο τυχαίων μεταβλητών. Εάν η απόκλιση της προβλεπόμενης πιθανότητας από την πραγματική ετικέτα αυξηθεί, η απώλεια cross entropy αυξάνεται. Βάση αυτού, η πρόβλεψη μιας πιθανότητας 0.01 όταν η πραγματική ετικέτα είναι 1, θα είχε ως αποτέλεσμα μια υψηλή τιμή απώλειας.

$$\text{BCE} = -\sum_{i=1}^{N} y_i \log\big(h_\theta(x_i)\big) + (1 - y_i) \log(1 - h_\theta(x_i))$$

(4.8)

Όπου το y_i είναι η πραγματική ετικέτα και το $h_\theta(x_i)$, είναι η προβλεπόμενη τιμή.

- **Mean Squared Error:** Το μέσο τετραγωνικό σφάλμα (Mean Squared Error), είναι ο μέσος όρος των διαφορών στο τετράφωνο μεταξύ των πραγματικών και των προβλεπόμενων τιμών του μοντέλου. Για ένα σημείο δεδομένων Y_i και την ανάλογη προβλεπόμενη τιμή του $\hat{Y}_i$, όπου n είναι ο συνολικός αριθμός σημείων δεδομένων στο σύνολο δεδομένων, το μέσο τετραγωνικό σφάλμα ορίζεται ως:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \big(Y_i - \hat{Y}_i\big)^2$$

(4.9)

4.1.3. Convolutional Neural Networks

Τα συνελικτικά νευρωνικά δίκτυα (CNN) είναι μια από τις πιο εντυπωσιακές αρχιτεκτονικές τεχνητών νευρωνικών δικτύων ANN (Artificial Neural Networks). Τα CNN χρησιμοποιούνται κυρίως για την επίλυση δύσκολών εργασιών αναγνώρισης προτύπων βάσει εικόνας λόγω της ακριβής αλλά και απλής αρχιτεκτονικής. Ωστόσο σύμφωνα με το [44], τα συνελικτικά δίκτυα εφαρμόζονται για δεκαετίες σε ακολουθίες, όπως για παράδειγμα στην αναγνώριση ομιλίας στις δεκαετίες του 80 και 90. Τα ConvNets εφαρμόστηκαν στη συνέχεια σε εργασίες NLP, όπως στην επισήμανση μέρους του λόγου και στην σημασιολογική επισήμανση των όρων του κειμένου. Πιο πρόσφατα, τα συνελικτικά νευρωνικά δίκτυα εφαρμόστηκαν στην ταξινόμηση των προτάσεων και στην ταξινόμηση κειμένων.

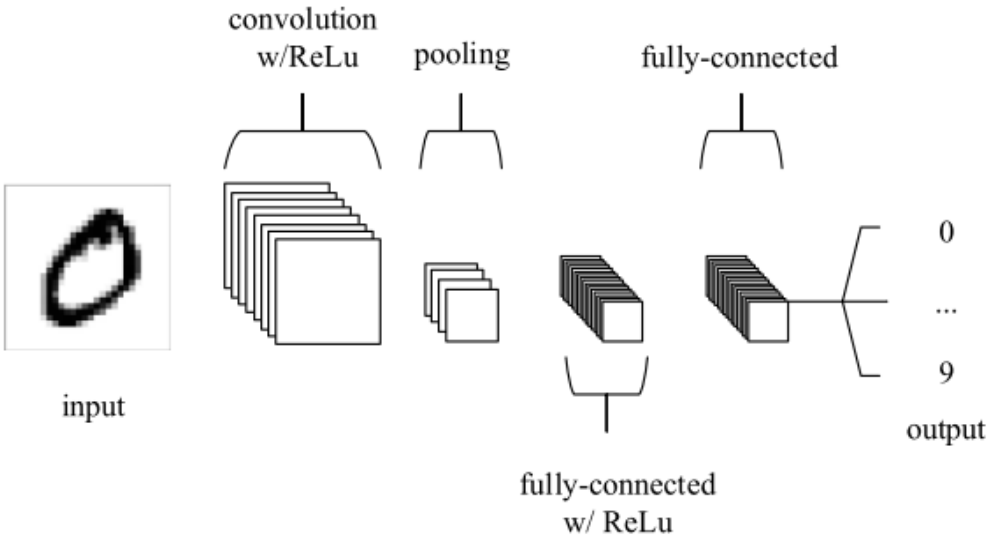
Τα CNNs έρχονται σε αναλογία με τα παραδοσιακά ANN, καθώς αποτελούνται από νευρώνες που αυτόβελτιστοποιούνται μέσω της μάθησης. Κάθε νευρώνας θα εξακολουθεί να λαμβάνει μια είσοδο και να εκτελεί μια λειτουργία όπως και τόσα ANN. Από τα διανύσματα ακατέργαστης εικόνας εισαγωγής έως την τελική έξοδο της βαθμολογίας κλάσης, ολόκληρο το δίκτυο θα εξακολουθεί να εκφράζει μια μοναδική συνάρτηση αντίληψης βαθμολογίας (το βάρος). Το τελευταίο επίπεδο θα περιέχει συναρτήσεις απώλειας. Η μόνη αξιοσημείωτη διαφορά μεταξύ των CNN και των παραδοσιακών ANN είναι ότι τα CNN’s χρησιμοποιούνται κυρίως στον τομέα της αναγνώρισης προτύπων στις εικόνες. Αυτό μας επιτρέπει να κωδικοποιήσουμε ειδικά χαρακτηριστικά εικόνας στην αρχιτεκτονική, καθιστώντας το δίκτυο πιο κατάλληλο για εργασίες που εστιάζουν στην εικόνα - μειώνοντας παράλληλα τις παραμέτρους που απαιτούνται για τη ρύθμιση του μοντέλου.

Σε αυτή την διπλωματική εργασία θα χρησιμοποιήσουμε τα συνελικτικά νευρωνικά δίκτυα για να κατηγοριοποιήσουμε κείμενα στην σωστή κλάση (fake, true), ωστόσο πριν δούμε τα CNN για εργασίες NLP θα αναφερθούμε πρώτα στην αρχιτεκτονική τους που λαμβάνουν ως είσοδο εικόνες.

4.1.3.1. Convolutional Neural Networks architecture

Τα CNN εστιάζουν κυρίως στη βάση ότι η είσοδος θα αποτελείται από εικόνες. Αυτό εστιάζει την αρχιτεκτονική που θα διαμορφωθεί έτσι ώστε να ταιριάζει καλύτερα στην ανάγκη αντιμετώπισης του συγκεκριμένου τύπου δεδομένων. Μία από τις βασικές διαφορές είναι ότι τα στρώματα στο CNN αποτελούνται από νευρώνες οργανωμένους σε τρεις διαστάσεις (dimensions), τη χωρική διάσταση (spatial) της εισόδου (height και width) και το depth (βάθος). Το βάθος δεν αναφέρεται στον συνολικό αριθμό επιπέδων εντός του ANN, αλλά στην τρίτη διάσταση ενός τόμου ενεργοποίησης. Σε αντίθεση με το τυπικά ANN’s, οι νευρώνες σε οποιοδήποτε δεδομένο στρώμα θα συνδεθούν μόνο σε μια μικρή περιοχή του στρώματος που προηγείται.

Τα CNN αποτελούνται από τρεις τύπους επιπέδων. Αυτά είναι συνελικτικά στρώματα (convolutional layers), συγκεντρωτικά στρώματα (pooling layers) και πλήρως συνδεδεμένα στρώματα (fully-connected layers). Όταν αυτά τα επίπεδα στοιβάζονται, έχει δημιουργηθεί μια αρχιτεκτονική CNN.



Εικόνα 4.3: Μια κοινή μορφή αρχιτεκτονικής CNN στην οποία τα συνελικτικά στρώματα στοιβάζονται μεταξύ συνεχόμενων activation functions ReLus, πριν περάσουν μέσα από το επίπεδο συγκέντρωσης [45].

- **Συνελικτικό επίπεδο (convolutional layer)**: Όπως υποδηλώνει το όνομα, το συνελικτικό επίπεδο παίζει ζωτικό ρόλο στον τρόπο λειτουργίας των CNN μοντέλων. Οι παράμετροι επιπέδων επικεντρώνονται στη χρήση kernels τους οποίους μαθαίνουν. Αυτά τα kernels είναι συνήθως μικρά σε χωρική διάσταση, αλλά απλώνονται σε όλο το βάθος της εισόδου. Όταν τα δεδομένα χτυπήσουν ένα συνεκτικό στρώμα, το στρώμα εμπλέκει κάθε φίλτρο σε όλη τη χωρική διάσταση της εισόδου για να παράγει έναν δυσδιάστατο χάρτη ενεργοποίησης. Καθώς περνάμε μέσα από την είσοδο, το κλιμακωτό προϊόν υπολογίζεται για κάθε τιμή σε αυτόν τον kernel. Έτσι, το δίκτυο θα μάθει τα kernels που «πυροδοτούν» όταν βλέπουν ένα συγκεκριμένο χαρακτηριστικό σε μια δεδομένη χωρική θέση της εισόδου. Αυτά είναι συνήθως γνωστά ως ενεργοποιήσεις. Κάθε kernel θα έχει έναν αντίστοιχο χάρτη ενεργοποίησης, ο οποίος θα στοιβάζεται κατά μήκος της διάστασης βάθους για να σχηματίσει τον πλήρη όγκο εξόδου από το συμβατικό στρώμα. Τα συνελικτικά στρώματα μπορούν επίσης να μειώσουν σημαντικά την πολυπλοκότητα του μοντέλου μέσω της βελτιστοποίησης της απόδοσής του. Αυτά βελτιστοποιούνται μέσω τριών υπερπαραμέτρων, του **depth**, του **stride** και του **zero-padding**.
    - **Depth:** Η μείωση αυτού του υπερπαραμέτρου μπορεί να ελαχιστοποιήσει σημαντικά τον συνολικό αριθμό νευρώνων του δικτύου, αλλά μπορεί επίσης να μειώσει σημαντικά τις δυνατότητες αναγνώρισης προτύπων του μοντέλου.
    - **Stride:** θέτουμε το βάθος γύρω από τη χωρική διάσταση της εισόδου για να τοποθετήσουμε το δεκτικό πεδίο.
    - **Zero-padding:** είναι η απλή διαδικασία γεμίσματος του περιγράμματος της εισόδου, και είναι μια αποτελεσματική μέθοδος για να δοθεί περαιτέρω έλεγχος ως προς τη διάσταση των τόμων εξόδου.

  Έχουν επίσης αναπτυχθεί μέθοδοι για να μειωθεί σημαντικά ο συνολικός αριθμός παραμέτρων εντός του συνεκτικού στρώματος. Η **parameter sharing** λειτουργεί με την

υπόθεση ότι εάν ένα χαρακτηριστικό μιας περιοχής είναι χρήσιμο για υπολογισμό σε μια καθορισμένη χωρική περιοχή, τότε είναι πιθανό να είναι χρήσιμο σε μια άλλη περιοχή. Εάν περιορίσουμε κάθε μεμονωμένο χάρτη ενεργοποίησης εντός του όγκου εξόδου στα ίδια βάρη και μεροληψία, τότε θα δούμε μια τεράστια μείωση του αριθμού των παραμέτρων που παράγονται από το συνεκτικό στρώμα.

- **Συγκεντρωτικό επίπεδο (pooling layer):** Τα συγκεντρωτικά στρώματα στοχεύουν στη σταδιακή μείωση της διαστατικότητας της αναπαράστασης, και συνεπώς στη μείωση περαιτέρω του αριθμού των παραμέτρων και της υπολογιστικής πολυπλοκότητας του μοντέλου. Το pooling layer λειτουργεί σε κάθε χάρτη ενεργοποίησης στην είσοδο και κλιμακώνει τη διάστασή του, χρησιμοποιώντας τη λειτουργία "MAX". Στα περισσότερα CNN μοντέλα, αυτά έρχονται με τη μορφή στρώσεων μέγιστης συγκέντρωσης με πυρήνες διαστάσεων 2 × 2 που εφαρμόζονται με ένα βήμα 2 κατά μήκος των χωρικών διαστάσεων της εισόδου. Αυτό κλιμακώνει το χάρτη ενεργοποίησης στο 25% του αρχικού μεγέθους - διατηρώντας παράλληλα τον όγκο του βάθους στο τυπικό του μέγεθος.
    - **Overlapping pooling** μπορεί να χρησιμοποιηθεί, όπου το βήμα είναι ρυθμισμένο σε 2 με μέγεθος kernel ρυθμισμένο σε 3. Λόγω της καταστροφικής φύσης της συγκέντρωσης, το μέγεθος πυρήνα πάνω από 3 συνήθως θα μειώσει σημαντικά την απόδοση του μοντέλου.
    - **General pooling** στρώματα αποτελούνται από ομαδοποιημένους νευρώνες που είναι σε θέση να εκτελούν πλήθος κοινών λειτουργιών, συμπεριλαμβανομένης της ομαλοποίησης L1 / L2 και της μέσης συγκέντρωσης.
- **Κανονικοποίηση παρτίδας (Batch normalization):** Σύμφωνα με το [46], η κανονικοποίηση παρτίδας χρησιμοποιείται για την αντιμετώπιση των ζητημάτων που σχετίζονται με την εσωτερική μετατόπιση συνδιακύμανσης στα χαρακτηριστικά που λαμβάνονται για το τελικό αποτέλεσμα (feature-map). Η εσωτερική ολίσθηση συνδιακύμανσης είναι μια αλλαγή στην κατανομή των τιμών των κρυφών μονάδων, η οποία επιβραδύνει τη σύγκληση (αναγκάζοντας τον ρυθμό εκμάθησης σε μικρότερο εύρος τιμών) και απαιτεί προσεκτική αρχικοποίηση των παραμέτρων. Με πιο απλά λόγια παίρνει τιμές που έχουν μεγάλες διαφορές μεταξύ τους και τις κανονικοποιεί σε ένα μικρότερο εύρος τιμών όπως για παράδειγμα το [0,1]. Το Bench normalization για έναν μετασχηματισμένο χάρτη χαρακτήρων F_l^k, περιγράφεται στην παρακάτω μαθηματική αναπαράσταση:

$$\mathrm{N}_l^k = \frac{F_l^k - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

(4.10)

Στην παραπάνω εξίσωση, το N_l^k αναπαριστά τις κανονικοποιημένες τιμές των χρήσιμων χαρακτήρων, η τιμή F_l^k είναι η είσοδος feature-map, το μ_B και το σ_B^2 απεικονίζουν τον μέσο όρο κα την διακύμανση αντίστοιχα, ενός feature-map για μια μικρή παρτίδα (batch). Τέλος προκειμένου να αποφευχθεί η διαίρεση με το μηδέν, προστίθεται ο όρος ε για αριθμητική σταθερότητα.

- **Επίπεδο εγκατάλειψης (Dropout layer):** Το Dropout εισάγει τακτοποίηση εντός του δικτύου, η οποία τελικά βελτιώνει την γενίκευση παρακάμπτοντας τυχαία ορισμένες τιμές του τεχντητού νευρωνικού δικτύου. Στα νευρωνικά δίκτυα, πολλαπλές συνδέσεις που μαθαίνουν μια μη γραμμική σχέση μερικές φορές προσαρμόζονται γεγονός που προκαλεί υπερπροσαρμογή [46]. Αυτό έχει ως αποτέλεσμα του τεχνητό νευρωνικό δίκτυο να μην μαθαίνει να κατηγοριοποιεί αλλά να λειτουργεί σαν να αποστηθίζει συσχετίσεις που όταν του δοθούν τα εξεταζόμενα αρχεία για να τα κατηγοριοποιήσει μόνο του αστοχεί και δεν έχει τα επιθυμητά αποτελέσματα. Έτσι με αυτή την τυχαία απόρριψη ορισμένων συνδέσεων εντός του δικτύου, παράγει αρκετές αραιωμένες αρχιτεκτονικές δικτύου ενώ τέλος επιλέγεται ένα αντιπροσωπευτικό δίκτυο με μικρά βάρη.
- **Πλήρως συνδεδεμένο επίπεδο:** Το πλήρως συνδεδεμένο στρώμα περιέχει νευρώνες οι οποίοι συνδέονται άμεσα με τους νευρώνες στα δύο γειτονικά στρώματα, χωρίς να συνδέονται με στρώματα μέσα τους.

Τα CNN μοντέλα είναι εξαιρετικά ισχυροί αλγόριθμοι μηχανικής μάθησης, ωστόσο μπορεί να είναι τρομερά απαιτητικά όσων αφορά του πόρους τους. Ένα παράδειγμα αυτού του προβλήματος θα μπορούσε να είναι το φιλτράρισμα μιας μεγάλης εικόνας. Τα συνελικτικά νευρωνικά δίκτυα διαφέρουν από άλλες μορφές τεχνητού νευρικού δικτύου, διότι αντί να επικεντρωθούν στο σύνολο του προβλήματος, αξιοποιείται η γνώση για τον συγκεκριμένο τύπο εισόδου. Αυτό με τη σειρά του επιτρέπει τη δημιουργία μιας πολύ απλούστερης αρχιτεκτονικής δικτύου.

4.1.3.2. Convolutional Neural Networks for text classification

Σύμφωνα με το [47], στην περίπτωση που έχουμε ως είσοδο κείμενα και όχι εικόνες όπως είδαμε πιο πάνω, το μοντέλο επεξεργάζεται το κείμενο βάσει των παρακάτω βημάτων:

1. Μονοδιάστατα φίλτρα περιέλιξης χρησιμοποιούνται ως ανιχνευτές ngram, με κάθε φίλτρο να ειδικεύεται σε μια στενά συνδεδεμένη "οικογένεια" ngram.
2. Το max-pooling επίπεδο, με την πάροδο του χρόνου εξάγει τα σχετικά ngram για την λήψη μιας απόφασης.
3. Το υπόλοιπο τεχνητό νευρωνικό δίκτυο ταξινομεί το κείμενο βάση των παραπάνω πληροφοριών.

Στη ουσία αντί να έχουμε 2D εισόδους όπως στην περίπτωση των εικόνων, έχουμε 1D εισόδους. Πιο αναλυτικά θεωρούμε ότι κάθε λέξη αναπαρίσταται ως ένα ενσωματωμένο διάνυσμα (embedding vector), πάνω στο οποίο εφαρμόζεται ένα ενιαίο συνελικτικό στρώμα με n φίλτρα όπου παράγουν ένα διάνυσμα n διαστάσεων για κάθε ngram του εγγράφου. Τα διανύσματα συνδυάζονται κάνοντας χρήση max-pooling που ακολουθείται από μια συνάρτηση ενεργοποίησης ReLU. Τέλος το αποτέλεσμα περνά στη συνέχεια σε ένα γραμμικό στρώμα για την τελική ταξινόμηση. Σύμφωνα με το [47] η προαναφερθείσα τεχνική θεωρείται η πιο απλή αρχιτεκτονική CNN για κατηγοριοποίηση κειμένου.

Πιο συγκεκριμένα για ένα κείμενο με n-λέξεων που μπαίνει ως είσοδο στο τεχνητό νευρωνικό δίκτυο με βάρη $w_1, w_2, \dots, w_n$, ενσωματώνουμε κάθε σύμβολο ως διάνυσμα d διαστάσεων, με αποτέλεσμα τα διανύσματα των λέξεων $w_1, w_2, \dots, w_n \in R^d$. Ο πίνακας που προκύπτει από αυτή την διαδικασία είναι ένας πίνακας d*n διαστάσεων, που στη συνέχεια τροφοδοτείται σε ένα συνελικτικό επίπεδο του τεχνητού δικτύου όπου περνάμε ένα ολισθώμενο παράθυρο πάνω από το κείμενο. Για κάθε μήκος λέξεων, ngram:

$$u_i = [w_i, \dots, w_{i+l-1}] \in R^{d*l}, 0 \le i \le n - l$$

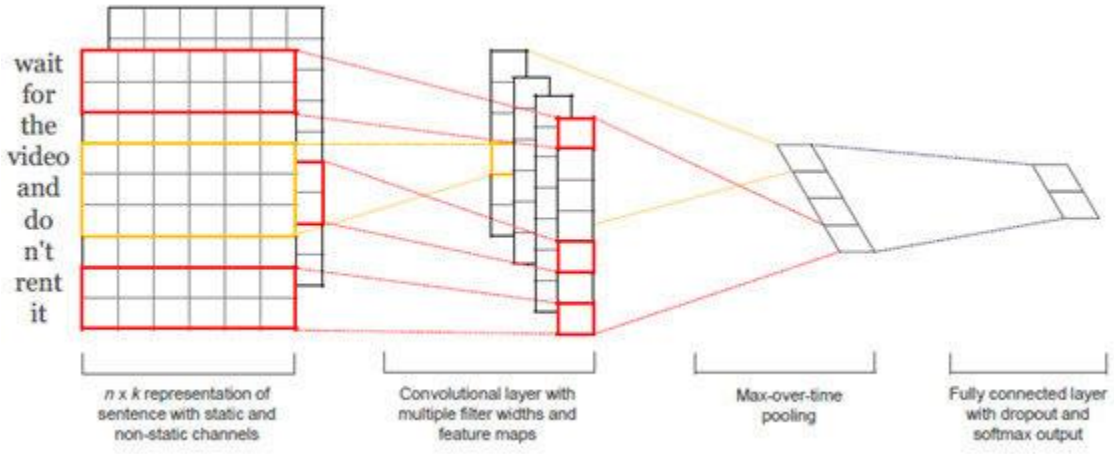
(4.11)

Για κάθε φίλτρο $f_j \in R^{d*l}$, υπολογίζουμε (u_i, f_j). Η συνέλιξη έχει ως αποτέλεσμα τον πίνακα $F \in R^{n*m}$. Η εφαρμογή της max-pooling κατά μήκος της διάστασης του ngram, έχει ως αποτέλεσμα $p \in R^m$, το οποίο τροφοδοτείται στην μη γραμμική συνάρτηση ενεργοποίησης ReLU που έχουμε προαναφέρει. Τέλος ένα γραμμικό πλήρως συνδεδεμένο επίπεδο $w \in R^{n*m}$ παράγει την κατανομή στις κατηγορίες ταξινόμησης από τις οποίες εξάγεται η ισχυρότερη κλάση. Αναλυτικά έχουμε τα παρακάτω:

1. $u_u = [w_i, \dots, w_{i+l-1}]$
2. $F_{ij} = (u_i, f_j)$
3. $p_j = ReLU(\max_i F_{ij})$
4. $output = softmax(W_p)$

(4.12)

Στην πράξη, χρησιμοποιούμε πολλαπλά μεγέθη παραθύρων $l \in L$ με το $L \subsetneq N$, χρησιμοποιώντας πολλαπλά στρώματα συνέλιξης παράλληλα μεταξύ τους και συνενώνοντας τα διανύσματα p^l που προκύπτουν.

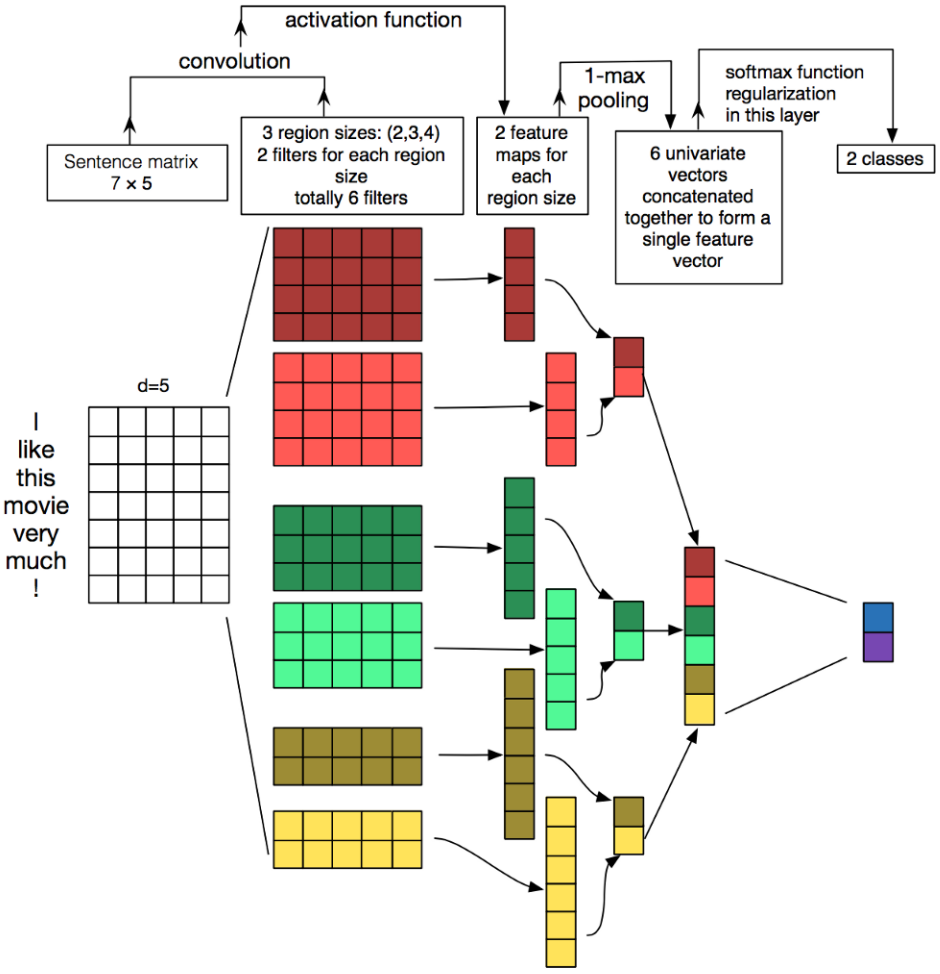


Εικόνα 4.4: Ένα παράδειγμα CNN αρχιτεκτονικής για την επεξεργασία φυσικής γλώσσας (Natural Language Processing). Η εικόνα πάρθηκε από το [48].

Στο πειραματικό κομμάτι της εργασία, κάνουμε χρήση μιας αρχιτεκτονικής βάσει του [49]. Το μοντέλο αυτό που φαίνεται και στην παρακάτω εικόνα περιλαμβάνει τρία επίπεδα. Το πρώτο επίπεδο είναι το Embedding layer, που μετατρέπει τις λέξεις των κειμένων στα αντίστοιχα διανύσματα n διαστάσεων. Το δεύτερο είναι το επίπεδο τις συνέλιξης στο οποίο λαμβάνει χώρα η κύρια επεξεργασία του μοντέλου, όπου τα προκαθορισμένα φίλτρα κυλούν πάνω από τον πίνακα

προτάσεων και μειώνουν την πολυπλοκότητα των διαστάσεων των λέξεων, ενώ το τρίτο επίπεδο αυτού του μοντέλου είναι μια συνάρτηση softmax.

Σε αυτή την αρχιτεκτονική ορίζονται φίλτρα διαφορετικών μεγεθών που στην ουσία προσομοιάζουν την διαδικασία των ngrams. Θα χρησιμοποιήσουμε αυτή την αρχιτεκτονική πάνω στην βάση των αγγλικών tweets αλλά και πάνω στην βάση των ελληνικών ψευδών ειδήσεων, ώστε να δούμε αν μπορεί το μοντέλο που προαναφέρθηκε να εντοπίσει κοινά στοιχεία ngrams που θα μας δώσουν να καταλάβουμε αν υπάρχουν κάποιες φράσεις που επαναλαμβάνονται μέσα στα κείμενα τόσο των ψευδών όσο και των αληθών ειδήσεων.



Εικόνα 4.5: Ορίζουμε διαφορετικά μεγέθη φίλτρων ώστε να προσομοιαστεί η διαδικασία των ngrams μέσα στο μοντέλο CNN. Η εικόνα χρησιμοποιήθηκε από το [49].

4.2. Recurrent Neural Networks

Τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN), είναι μια κατηγορία νευρωνικών δικτύων που επιτρέπουν προηγούμενες εξόδους να χρησιμοποιηθούν ως είσοδοι ενώ έχουν κρυφές καταστάσεις.

Όταν τα δεδομένα που έχουμε ως είσοδο είναι γλωσσολογικά δεδομένα είναι πολύ συνηθισμένο να εργαζόμαστε με ακολουθίες, όπως για παράδειγμα είναι:

1. Λέξεις
2. Προτάσεις
3. Έγγραφα

Όπως αναφέρει και το [50], τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) είναι ένα είδος feed-forward neural networks, που έχουν μια επαναλαμβανόμενη κρυφή κατάσταση, που ενεργοποιείται από τις προηγούμενες καταστάσεις σε μια συγκεκριμένη χρονική στιγμή. Έτσι τα RNN μπορούν να μοντελοποιήσουν δυναμικά τις πληροφορίες συμφραζομένων από τα κείμενα που έχουν να επεξεργαστούν, ενώ μπορούν να χειριστούν ακολουθίες μεταβλητού μήκους. Αυτό συμβαίνει διότι σε αντίθεση με άλλα νευρωνικά δίκτυα που όλες οι είσοδοι είναι ανεξάρτητες μεταξύ τους, στην αρχιτεκτονική των RNN όλες οι είσοδοι σχετίζονται μεταξύ τους.

Τα Recurrent Neural Networks έχουν αρκετά πλεονεκτήματα αλλά όπως και κάθε άλλη αρχιτεκτονική είτε machine learning είτε deep learning έχουν και κάποια μειονεκτήματα. Τα πλεονεκτήματά τους είναι, όπως τα ονομάζει η σελίδα του Stanford [51]:

- Δυνατότητα επεξεργασίας εισόδου οποιουδήποτε μήκους
- Το μέγεθος του μοντέλου δεν αυξάνεται με το μέγεθος της εισόδου
- Η διαδικασία επεξεργασίας του μοντέλου λαμβάνει υπόψη προηγούμενες πληροφορίες
- Τα βάρη μοιράζονται καθόλη την διάρκεια λειτουργίας του

Τα μειονεκτήματά του είναι:

- Ο υπολογισμός είναι αργός
- Δυσκολία πρόσβασης σε πληροφορίες από πολύ παλιότερες εισόδους
- Δεν είναι δυνατή η εξέταση μελλοντικών δεδομένων για την τρέχουσα κατάσταση

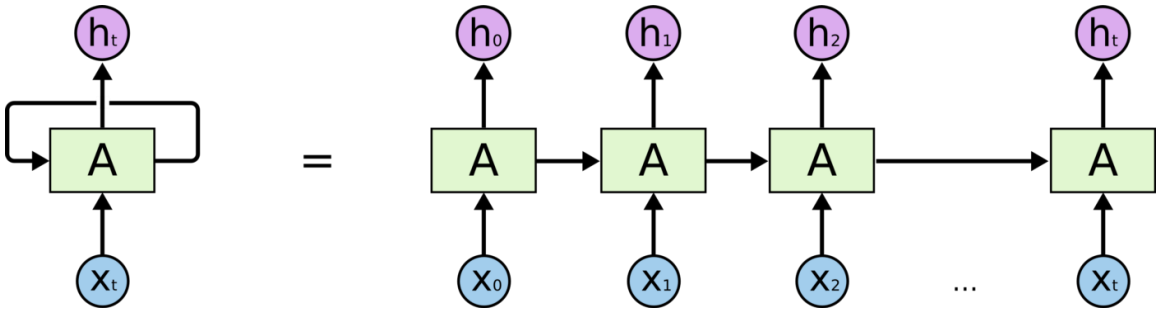
4.2.1. Recurrent Neural Network architecture

Σύμφωνα με το [39] τα RNN παίρνουν ως είσοδο μια διατεταγμένη λίστα διανυσμάτων $x_1, x_2, x_3, \dots, x_n$, μαζί με ένα διάνυσμα αρχικής κατάστασης s_0 και επιστρέφει μια διατεταγμένη λίστα διανυσμάτων καταστάσεων $s_1, s_2, s_3, \dots, s_n$ καθώς και μια ταξινομημένη λίστα με διανύσματα εξόδου $y_1, y_2, y_3, \dots, y_n$. Ένα διάνυσμα εξόδου y_i είναι συνάρτηση του αντίστοιχου διανύσματος της κατάστασης s_i. Τα διανύσματα εισόδου x_i παρουσιάζονται στο RNN με διαδοχικό τρόπο και το διάνυσμα κατάστασης s_i και το διάνυσμα εξόδου y_i αντιπροσωπεύουν την κατάσταση του RNN μετά το πέρας των εισόδων $s_{1:i}$. Το διάνυσμα εξόδου y_i χρησιμοποιείται στη συνέχεια για περαιτέρω πρόβλεψη.

Πιο απλά όπως φαίνεται και στην εικόνα 2, αρχικά παίρνει το x_0 από την ακολουθία εισόδου και μετά βγάζει την έξοδο h_0 που μαζί με το x_1 είναι η είσοδος για το επόμενο βήμα. Ομοίως, το h_1 από το επόμενο είναι η είσοδος με το x_2 για το επόμενο βήμα και ούτω καθεξής. Με αυτό τον τρόπο συνεχίζει να θυμάται το πλαίσιο των συμφραζόμενων πληροφοριών κατά την εκπαίδευσή του. Η μαθηματική έκφραση για μια κατάσταση h_t είναι:

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

(4.13)



Εικόνα 4.6: Βασική αρχιτεκτονική των επαναλαμβανόμενων νευρωνικών δικτύων. Το παραπάνω σχήμα δείχνει ένα RNN που ξεδιπλώνεται σε ένα πλήρες δίκτυο. Με το ξεδίπλωμα εννοούμε απλώς ότι επαναλαμβάνουμε την ίδια δομή στρώματος δικτύου για ολόκληρη την ακολουθία. Η εικόνα χρησιμοποιήθηκε από το [52]

Τέλος εφαρμόζεται μια non-linear activation συνάρτηση όπως για παράδειγμα sigmoid ή tanh ή ReLU.

4.2.2. Vanishing Gradient Problem

Κατά την εκπαίδευση ενός RNN μοντέλου κάνουμε χρήση της μεθόδου Backpropagation-through-time (Θα μιλήσουμε περισσότερο σε επόμενη παράγραφο για το Backpropagation-through-time), ώστε με κάθε κλίση της προς τα πίσω στο χρόνο να γίνεται η σωστή ενημέρωση των βαρών στο δίκτυο. Εάν η επίδραση του προηγούμενου στρώματος στο τρέχον στρώμα είναι μικρή, τότε η gradient τιμή θα είναι μικρή και το αντίστροφο. Εάν η gradient τιμή του προηγούμενου στρώματος είναι μικρή, τότε η επίδρασή της τιμή για το τρέχον στρώμα του νευρωνικού δικτύου θα είναι ακόμα μικρότερη. Αυτό κάνει τις gradient τιμές να "συρρικνώνονται" εκθετικά καθώς διασχίζουμε το μοντέλο προς τα πίσω.

Μικρότερη gradient τιμή, σημαίνει ότι δεν θα επηρεάσει την ενημέρωση των βαρών στο δίκτυο. Εξαιτίας αυτού, το δίκτυο δεν μαθαίνει την επίδραση τον προηγούμενων εισόδων, προκαλώντας έτσι το πρόβλημα της βραχυπρόθεσμης μνήμης.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

(4.14)

Όπου το W είναι το βάρος, το h είναι το μεμονωμένο κρυφό διάνυσμα, το W_{hh} το βάρος στην προηγούμενη κρυφή κατάσταση, το W_{hx} το βάρος στην τρέχουσα κατάσταση εισόδου ενώ όπως έχουμε δει και σε προηγούμενο κεφάλαιο, η συνάρτηση tanh είναι η συνάρτηση ενεργοποίησης.

Το κύριο πρόβλημα που συναντάται σε αυτή την περίπτωση είναι ότι είναι πολύ δύσκολο για το RNN να μάθει να διατηρεί πληροφορίες σε πολλά χρονικά βήματα. Στην αρχική "έκδοση" RNN, η κρυφή κατάσταση του δικτύου ξαναγράφεται συνεχώς.

Για να ξεπεραστεί αυτό το πρόβλημα δημιουργήθηκαν δύο εξειδικευμένες εκδόσεις των Recurrent Neural Networks. Η πρώτή είναι η GRU (Gated Recurrent Unit) και η δεύτερη είναι η LSTM (Long Short Term Memory). Ας υποθέσουμε ότι υπάρχουν οι ακόλουθες δύο προτάσεις μέσα στην βάση των αγγλικών tweets. Η πρώτη είναι "The vaccine was dangerous to my immune system" και η δεύτερη είναι " The vaccines were not properly tasted …" . Αν χρειαστεί να προβλέψουμε την λέξη "was" / "were" το δίκτυο πρέπει να θυμάται την αρχική λέξη "vaccine" / "vaccines". Έτσι τα μοντέλα LSTM και GRU, κάνουν χρήση κελιών μνήμης για να αποθηκεύσουν την τιμή ενεργοποίησης των προηγούμενων λέξεων στις μεγάλες ακολουθίες.

4.2.3. Gated Recurrent Unit

Η Gated Recurrent Unit (GRU), είναι ένα είδος αρχιτεκτονικής RNN και έχει γίνει η κύρια δομή των RNN προς το παρών. Η GRU αντιμετωπίζει το πρόβλημα του vanishing gradient, χρησιμοποιώντας έναν μηχανισμό πύλης που παρακολουθεί την κατάσταση των ακολουθιών χωρίς να χρησιμοποιεί ξεχωριστά κελιά μνήμης.

Υπάρχουν δύο τύπων πυλών στην αρχιτεκτονική GRU, η πύλη επαναφοράς r_t και η πύλη ενημέρωσης z_t που μαζί ελέγχουν τον τρόπο ενημέρωση των πληροφοριών των κάθε καταστάσεων. Σύμφωνα με το [50], η μαθηματική έκφραση μιας GRU μονάδας υπολογίζεται:

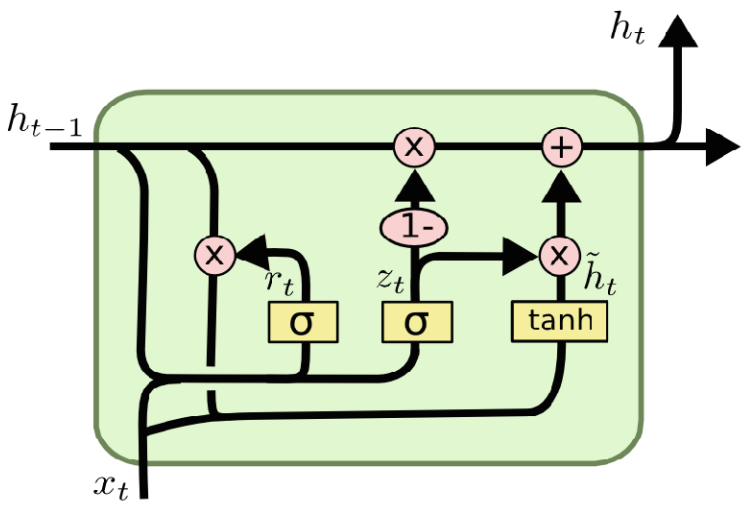
$$r_t = \sigma(W_r x_r + U_r h_{t-1} + b_r)$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z$$

$$\hat{\mathrm{h}}_t = \tanh(W_h x_t + r_t \otimes (\mathrm{U_h} h_{t-1}) + b_h$$

$$h_t = (1 - z_t) \otimes \mathrm{h(t-1)} + \mathrm{z_t} \otimes \hat{\mathrm{h}}$$

(4.15)

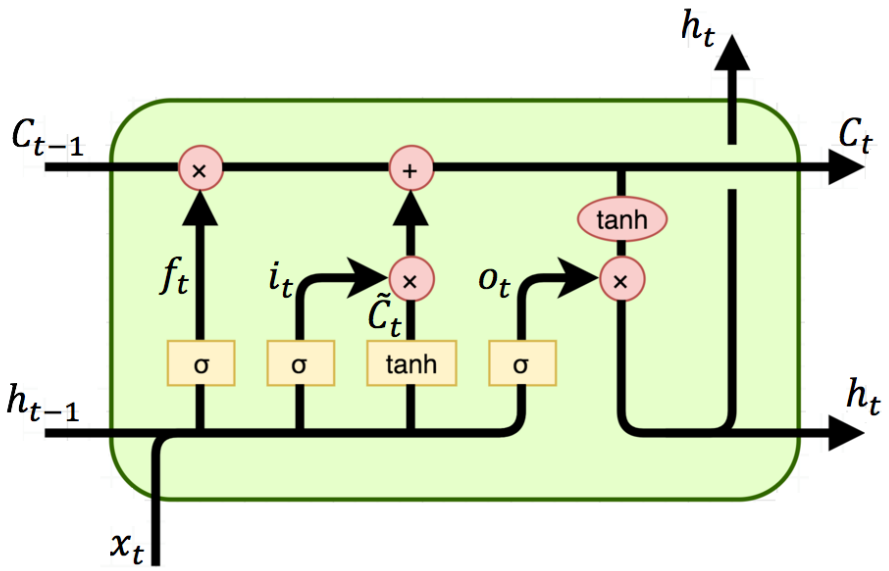


Εικόνα 4.7: Βασική αρχιτεκτονική GRU cell από το [53]

Όπου το z_t είναι η πύλη ενημέρωσης, το r_t είναι η πύλη επαναφοράς, το h_{t-1} είναι μια προηγούμενη κατάσταση, το $\hat{h}_t$ είναι η υποψήφια κατάσταση την χρονική στιγμή t, το x_t είναι το διάνυσμα ακολουθίας τη χρονική στιγμή t , η $\sigma(.)$ και tanh(.), είναι η sigmoid συνάρτηση ενεργοποίησης και hyperbolic tangent συνάρτηση ενεργοποίησης αντίστοιχα και τα b_z, b_h, b_r είναι bias όροι. Τέλος ο τελεστής $\otimes$ υποδηλώνει πολλαπλασιασμό βάσει στοιχείων.

4.2.4. Long Short-Term Memory

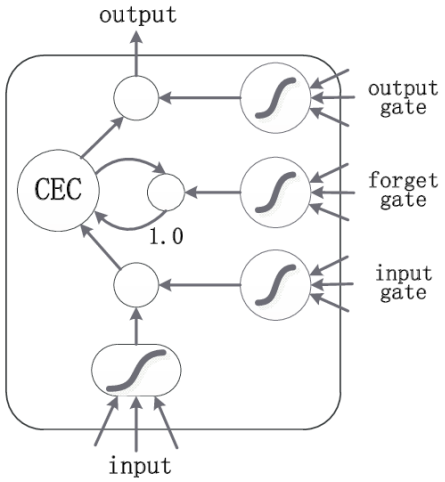
Η αρχιτεκτονική Long Short-Term Memory (LSTM) σύμφωνα με το [54] μπορεί να μάθει πώς να γεφυρώνει ελάχιστες χρονικές καθυστερήσεις άνω των 1000 διακριτών χρονικών βημάτων. Τα LSTM χρησιμοποιούν constant error carousels (CEC), τα οποία επιβάλλουν μια σταθερή ροή σφάλματος εντός ειδικών κελιών. Η πρόσβαση στα κελιά γίνεται με πολλαπλασιαστικές μονάδες gate units, οι οποίες μαθαίνουν πότε πρέπει να χορηγούν πρόσβαση.



Εικόνα 4.8: Βασική αρχιτεκτονική LSTM cell από το [53]

4.2.4.1. Constant Error Carousel

Σύμφωνα με το [55], το constant error carousel (CEC) μετριάζει το πρόβλημα vanishing (exploding) της κλίσης καθώς η backward ροή του τοπικού σφάλματος παραμένει σταθερή εντός του CEC χωρίς να αυξάνεται η να μειώνεται, όσο δεν υπάρχει νέα είσοδος δεδομένων ή εξωτερικό σήμα σφάλματος. Η εκπαίδευση αυτού του μοντέλου βασίζεται σε μια τροποποίηση των αλγορίθμων backpropagation trough time και real-time recurrent learning.



Εικόνα 4.9: Ένα κελί αποτελείται από μια μονάδα κυψέλης και τρεις πύλες (είσοδος, λήθη και έξοδος). Οι πύλες χρησιμοποιούν μια συνάρτηση σιγμοειδούς ενεργοποίησης και η κατάσταση εισόδου και κυψέλης συνήθως μετασχηματίζονται από το tanh, μια άλλη συνάρτηση ενεργοποίησης. Ο μηχανισμός πύλης μπορεί να κρατήσει πληροφορίες για μεγάλες διάρκειες, ωστόσο τα LSTM δεν διαθέτουν πύλη λήψεως και αντ' αυτού προσθέτουν μια αμετάβλητη κατάσταση κυψέλης (π.χ. μια επαναλαμβανόμενη σύνδεση με σταθερό βάρος 1). Αυτή η προσθήκη

ονομάζεται Carousel Constant Error επειδή λύνει το εκπαιδευτικό πρόβλημα της εξαφάνισης και της έκρηξης κλίσεων [56]

4.2.4.2. Memory Blocks

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο, κατά την έλλειψη νέων εισόδων στο κελί γνωρίζουμε ότι η backward ροή του CEC παραμένει σταθερή. Ωστόσο, ως μέρος ενός νευρωνικού δικτύου, το CEC δεν συνδέεται μόνο με τον εαυτό του, αλλά και με άλλες μονάδες του νευρωνικού δικτύου. Όπως αναφέρει και το [54], πρέπει να λάβουμε υπόψη αυτά τα πρόσθετα σταθμισμένα inputs και outputs.

Οι εισερχόμενες συνδέσεις στον νευρώνα u μπορεί να έχουν αντικρουόμενα σήματα ενημέρωσης βάρους επειδή το ίδιο βάρος χρησιμοποιείται για την αποθήκευση και την παράβλεψη εισόδων. Για σταθμισμένες συνδέσεις εξόδου από ον νευρώνα u, τα ίδια βάρη μπορούν να χρησιμοποιηθούν τόσο για την ανάκτηση των περιεχομένων του u όσο και για την αποτροπή της ροής εξόδου του u σε άλλους νευρώνες στο δίκτυο.

4.2.5. Backpropagation Through Time on Recurrent models

Ο στόχος των αλγορίθμων επαναλαμβανόμενης (recurrent) μάθησης είναι να βελτιστοποιήσουν ένα παραμετρικό δυναμικό σύστημα, έτσι ώστε η ακολουθία εξόδου ή οι προβλέψεις του μοντέλου να είναι όσο το δυνατόν πιο κοντά σε κάποια ακολουθία στόχο γνωστή και ως priori [57]. Δεδομένου ενός δυναμικού συστήματος με μια κατάσταση s, x εισόδους, με παράμετρο θ και συνάρτηση μετάβασης F έχουμε την παρακάτω μαθηματική έκφραση που μας την δίνει το παραπάνω paper:

$$s_{t+1} = F(x_{t+1}, s_t, \theta)$$

(4.16)

Ο στόχος είναι να βρεθεί ένα θ ελαχιστοποιώντας μια συνολική απώλεια σε σχέση με τις εξόδους στόχους o_t^* σε κάθε χρονική στιγμή του μοντέλου,

$$\mathcal{L}_T = \sum_{t=1}^{T} \ell_t = \sum_{t=1}^{T} \ell(s_t, o_t^*)$$

(4.17)

Η αναδρομική διάδοση στο χρόνο (Backpropagation through time) υπολογίζει την διαβάθμιση (gradient) ξεδιπλώνοντας το δυναμικό σύστημα/μοντέλο μέσα στο χρόνο και διαδίδοντας πίσω μέσα από αυτό να βρίσκει τα κατάλληλα βάρη, με κάθε χρονικό βήμα να αντιστοιχεί σε ένα επίπεδο (layer) του νευρωνικού δικτύου. Η αναδρομική διάδοση στο χρόνο

αποσυνθέτει τη διαβάθμιση ως άθροισμα, στα χρονικά βήματα t, της επίδρασης μιας αλλαγής παραμέτρου τη στιγμή t σε όλες τι επόμενες απώλειες.

$$\frac{\partial \mathcal{L}_T}{\partial \theta} = \sum_{t=1}^{T} \delta \ell_t \frac{\partial F}{\partial \theta}(x_t, s_{t-1}, \theta)$$

(4.18)

Όπως αναφέρει το [57], τυπικά όπου το $\delta \ell_t \coloneqq \frac{\partial \mathcal{L}_T}{\partial s_t}$ υπολογίζεται προς τα πίσω επαναληπτικά σύμφωνα με την εξίσωση backpropagation

$$\begin{cases} \delta \ell_T = \dfrac{\partial \ell}{\partial \sigma}(s_T, o_T^*) \\ \delta \ell_t = \delta \ell_{t+1} \dfrac{\partial F}{\partial s}(x_{t+1}, s_t, \theta) + \dfrac{\partial \ell}{\partial s}(s_T, o_T^*) \end{cases}$$

(4.19)

Δυστυχώς, η αναδρομική διάδοση στο χρόνο απαιτεί την επεξεργασία της πλήρους ακολουθίας τόσο προς τα εμπρός όσο και προς τα πίσω. Αυτό απαιτεί την διατήρηση του πλήρους ξεδιπλωμένου δικτύου. Αυτό δεν είναι πρακτικό όταν επεξεργαζόμαστε πολύ μεγάλες ακολουθίες σε μεγάλα δίκτυα, διότι η επεξεργασία ολόκληρης της ακολουθίας σε κάθε βήμα διαβάθμισης επιβραδύνει τη μάθηση του τεχνητού νευρωνικού δικτύου.

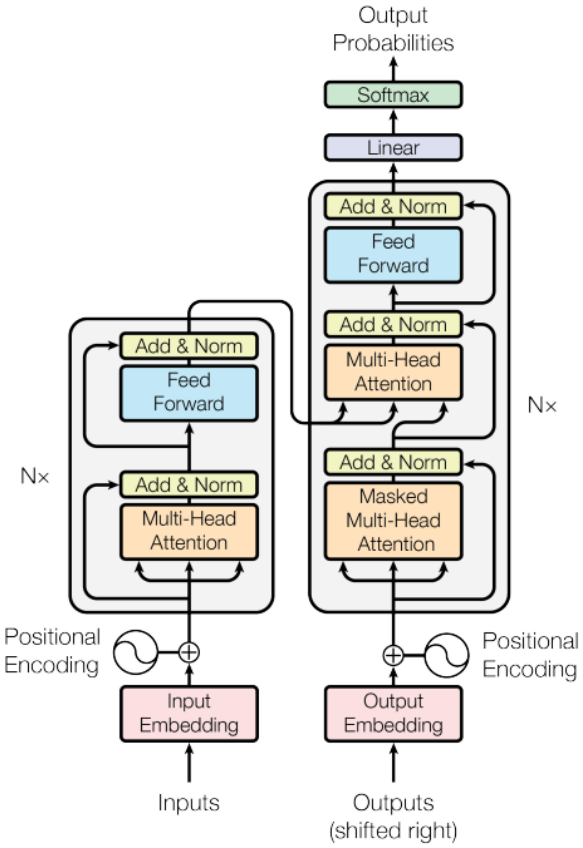
Πρακτικά, αυτό μετριάζεται με την περικοπή των ροών βαθμίδωσης μετά από έναν καθορισμένο αριθμό χρονικών βημάτων, ή ισοδύναμα, με τον διαχωρισμό της ακολουθίας εισόδου σε υποακολουθίες σταθερού μήκους. Αυτός ο αλγόριθμος αναφέρεται ως Truncated BPTT. Με μήκος περικοπής L<T, οι αντίστοιχες εξισώσεις απλώς ρίχνουν τον επαναλαμβανόμενο όρο $\delta \ell_{t+1} \frac{\partial F}{\partial s}(x_{t+1}, s_t, \theta)$ κάθε L βήματα [57].

4.3. Transformer Neural Networks

Η αρχιτεκτονική των Transformer νευρωνικών δικτύων, λειτουργεί βασιζόμενη σε έναν μηχανισμό αυτοπροσοχής (ενδοπροσοχή), αφαιρώντας όλες τις επαναλαμβανόμενες λειτουργίες που βρίσκονται στην προηγούμενη προσέγγιση (RNN). Ωστόσο, με την απουσία επανάληψης, η κωδικοποίηση της θέσης, προστίθεται στις ενσωματώσεις εισόδου και εξόδου. Ομοίως, όπως το χρονικό βήμα σε ένα επαναλαμβανόμενο δίκτυο, οι πληροφορίες θέσης παρέχουν στο δίκτυο του μετασχηματιστή τη σειρά των ακολουθιών εισόδου και εξόδου [58].

Πιο αναλυτικά, τα περισσότερα μοντέλα μεταγωγής νευρωνικής ακολουθίας έχουν δομή κωδικοποιητή-αποκωδικοποιητή. Εδώ, ο κωδικοποιητής αντιστοιχίζει μια ακολουθία εισόδου αναπαραστάσεων συμβόλων $(x_1, \ldots, x_n)$ σε μια ακολουθία συνεχών αναπαραστάσεων z =

$(z_1, \ldots, z_n)$. Με δεδομένο το z, ο αποκωδικοποιητής δημιουργεί στη συνέχεια μια ακολουθία εξόδου $(y_1, \ldots, y_m)$ συμβόλων για ένα στοιχείο κάθε φορά. Σε κάθε βήμα το μοντέλο είναι αυτόματα παλινδρομικό, καταναλώνοντας τα σύμβολα που δημιουργήθηκαν προηγουμένως ως πρόσθετη είσοδο κατά τη δημιουργία του επόμενου. Ο μετασχηματιστής ακολουθεί αυτή τη συνολική αρχιτεκτονική χρησιμοποιώντας στοιβαγμένα επίπεδα αυτοπροσοχής, πλήρως συνδεδεμένα στρώματα τόσο για τον κωδικοποιητή όσο και για τον αποκωδικοποιητή [59].



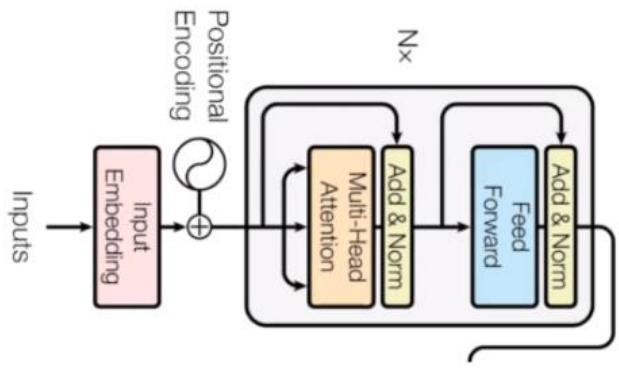
Εικόνα 4.10: Αρχιτεκτονική των Transformer Neural Networks.

4.3.1. The architecture

4.3.1.1. Encoder Block

Ο κωδικοποιητής αποτελείται από μια στοίβα N = 6 πανομοιότυπων επιπέδων. Κάθε στρώμα έχει δύο υποστρώματα. Το πρώτο είναι ένας μηχανισμός αυτοπροσοχής πολλών κεφαλών και ο δεύτερος είναι ένα απλό, πλήρως συνδεδεμένο δίκτυο προώθησης τροφοδοσίας με γνώμονα τη θέση. Η έξοδος κάθε υποστρώματος είναι LayerNorm(x + Sublayer(x)), όπου το Sublayer(x) είναι η συνάρτηση που υλοποιείται από το ίδιο το υποστρώμα. Για να διευκολυνθούν αυτές οι

υπολειπόμενες συνδέσεις, όλα τα υποστρώματα του μοντέλου, καθώς και τα στρώματα ενσωμάτωσης, παράγουν εξόδους διάστασης dmodel = 512 [59].



Εικόνα 4.11: Ο κωδικοποιητής από την αρχιτεκτονική των Transformer Neural Networks.

Multi-Head Attention

Αυτό το επίπεδο, εστιάζει στο μέρος του πόσο σχετική είναι μια συγκεκριμένη λέξη με άλλες λέξεις αυτής της πρότασης. Αναπαρίσταται ως διάνυσμα προσοχής. Για κάθε λέξη, μπορούμε να δημιουργήσουμε ένα διάνυσμα προσοχής, το οποίο αποτυπώνει τη σχέση μεταξύ των λέξεων σε αυτήν την πρόταση.

Feed Forward Network

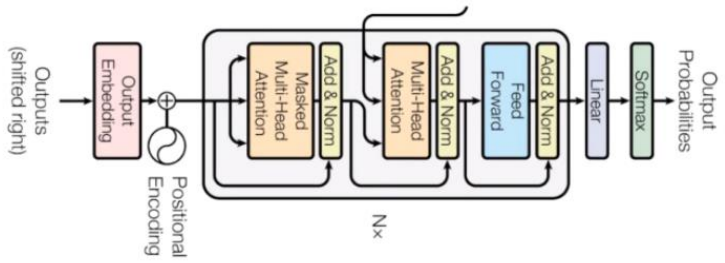
Το δεύτερο κομμάτι του encoder block είναι το Feed Forward Neural Network. Αυτό είναι ένα απλό νευρωνικό δίκτυο τροφοδοσίας που εφαρμόζεται σε κάθε διάνυσμα προσοχής, ο κύριος σκοπός του είναι να μετατρέψει τα διανύσματα προσοχής σε μια μορφή που είναι αποδεκτή από το επόμενο επίπεδο κωδικοποιητή ή αποκωδικοποιητή. Το Feed Forward Network δέχεται ένα διανύσματα προσοχής κάθε φορά.

Σε αντίθεση με την περίπτωση του RNN, εδώ καθένα από αυτά τα διανύσματα προσοχής είναι ανεξάρτητο το ένα από το άλλο. Έτσι, η παραλληλοποίηση μπορεί να εφαρμοστεί εδώ, και αυτό κάνει τη διαφορά.

Έτσι μπορούμε να περάσουμε όλες τις λέξεις ταυτόχρονα στο μπλοκ κωδικοποιητή και να πάρουμε το σύνολο των κωδικοποιημένων διανυσμάτων για κάθε λέξη ταυτόχρονα.

4.3.1.2. Decoder

Σύμφωνα με το [59], ο αποκωδικοποιητής αποτελείται επίσης από μια στοίβα N = 6 πανομοιότυπων στρωμάτων. Εκτός από τα δύο υποστρώματα σε κάθε επίπεδο κωδικοποιητή, ο αποκωδικοποιητής εισάγει ένα τρίτο υπόστρωμα, το οποίο εκτελεί προσοχή πολλαπλών κεφαλών (multi-head attention) στην έξοδο της στοίβας κωδικοποιητή. Παρόμοια με τον κωδικοποιητή, χρησιμοποιούμε συνδέσεις γύρω από κάθε ένα από τα υποστρώματα, ακολουθούμενες από κανονικοποίηση στρώματος. Τροποποιούμε επίσης το υποστρώμα αυτοπροσοχής στη στοίβα αποκωδικοποιητή για να αποτρέψουμε την παρακολούθηση θέσεων σε επόμενες θέσεις. Αυτή η κάλυψη, σε συνδυασμό με το γεγονός ότι οι ενσωματώσεις εξόδου αντισταθμίζονται κατά μία θέση, διασφαλίζει ότι οι προβλέψεις για τη θέση i μπορούν να εξαρτώνται μόνο από τις γνωστές εξόδους σε θέσεις μικρότερες από i.



Εικόνα 4.12: : Ο αποκωδικοποιητής από την αρχιτεκτονική των Transformer Neural Networks.

Masked Multi-Head Attention

Αυτό το υποστρώμα λαμβάνει την προηγούμενη έξοδο της στοίβας του αποκωδικοποιητή, την αυξάνει με πληροφορίες θέσης και εφαρμόζει την αυτοπροσοχή πολλών κεφαλών πάνω του. Ενώ ο κωδικοποιητής έχει σχεδιαστεί για να παρακολουθεί όλες τις λέξεις στην ακολουθία εισαγωγής, ανεξάρτητα από τη θέση τους στην ακολουθία, ο αποκωδικοποιητής τροποποιείται για να παρακολουθεί μόνο τις προηγούμενες λέξεις. Ως εκ τούτου, η πρόβλεψη για μια λέξη στη θέση i, μπορεί να εξαρτάται μόνο από τις γνωστές εξόδους για τις λέξεις που εμφανίζονται πριν από αυτήν στην ακολουθία. Στον μηχανισμό προσοχής πολλαπλών κεφαλών (ο οποίος υλοποιεί πολλαπλές, μεμονωμένες συναρτήσεις προσοχής παράλληλα), αυτό επιτυγχάνεται με την εισαγωγή μιας μάσκας στις τιμές που παράγονται από τον κλιμακούμενο πολλαπλασιασμό των πινάκων Q και K που προκύπτουν από το τον κωδικοποιητή και πιο συγκεκριμένα από το Multi-Head Attention επίπεδό του.

Multi-Head Attention

Αυτό το επίπεδο υλοποιεί έναν μηχανισμό αυτοπροσοχής πολλαπλών κεφαλών, ο οποίος είναι παρόμοιος με αυτόν που εφαρμόζεται στο πρώτο υποστρώμα του κωδικοποιητή. Από την πλευρά του αποκωδικοποιητή, αυτός ο μηχανισμός πολλαπλών κεφαλών λαμβάνει τα ερωτήματα από το

προηγούμενο υποστρώμα του αποκωδικοποιητή, και τα κλειδιά και τις τιμές από την έξοδο του κωδικοποιητή. Αυτό επιτρέπει στον αποκωδικοποιητή να παρακολουθεί όλες τις λέξεις στην ακολουθία εισόδου.

Feed Forward Network

Τέλος το τελευταίο επίπεδο υλοποιεί ένα πλήρως συνδεδεμένο δίκτυο προώθησης τροφοδοσίας, το οποίο είναι παρόμοιο με αυτό που εφαρμόζεται στο δεύτερο υποστρώμα του κωδικοποιητή.

4.3.2. BERT

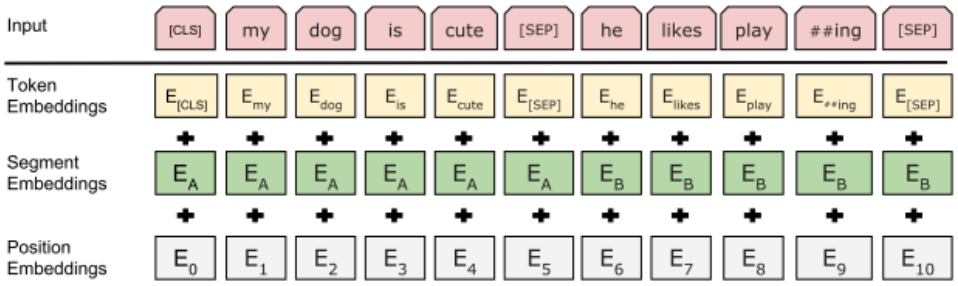
Πρώτα από όλα, η συντομογραφία του μοντέλου BERT σημαίνει Bidirectional Encoder Representations from Transformers και η αρχιτεκτονική του είναι, όπως υποδηλώνει και το όνομά του βασισμένη στα Transformer Neural Networks. Το μοντέλο BERT που παρουσιάστηκε για πρώτη φόρα το 2018 από την Google στο paper [60] κάνει χρήση δύο προ εκπαιδευμένων ενεργειών, μοντελοποίηση μάσκας γλώσσας ή αλλιώς Masked Language Modeling (MLM) στα αγγλικά και πρόβλεψη επόμενης πρότασης (Next Sentence Prediction). Οι δημιουργοί του μοντέλου, χρησιμοποίησαν τα BookCorpus που περιέχει 800 εκατομμύρια λέξεις και την αγγλική Wikipedia που περιέχει 2.500 εκατομμύρια λέξεις ως συλλογές για να προεκπαιδεύσουν το μοντέλο. Ωστόσο για το κομμάτι των ελληνικών θα χρησιμοποιήσουμε το Greek BERT που έχει εκπαιδευτεί πάνω στο ελληνικό κομμάτι της Wikipedia, στα ελληνικά κείμενα της European Parliament Proceedings Parallel Corpus και στις ελληνικές συλλογές κειμένων OSCAR και Common Crawl.

Η αρχιτεκτονική του μοντέλου του BERT είναι ένας πολυεπίπεδος αμφίδρομος κωδικοποιητής Transformer που βασίζεται στην αρχιτεκτονική των Transformers που είδαμε πιο πάνω, δηλαδή έχει έναν κωδικοποιητή και έναν αποκωδικοποιητή.

4.3.3. BERT Representations

Πριν προχωρήσουμε στη μεθοδολογία της προεκπαίδευσης του μοντέλου, πρέπει πρώτα να αναφέρουμε τις αναπαραστάσεις εισόδου /εξόδου που χρησιμοποιούνται στο BERT όπως αυτές αναφέρονται στο [60] και απεικονίζονται στην Εικόνα 4.13.

- **[CLS]**: Αυτό το token ονομάζεται token ταξινόμησης και χρησιμοποιείται στην αρχή μιας ακολουθίας.
- **[SEP]**: Αυτό το token υποδεικνύει τον διαχωρισμό 2 ακολουθιών, δηλαδή λειτουργεί ως οριοθέτης.
- **[MASK]**: Χρησιμοποιείται για να υποδείξει το κρυφό token στην εργασία MLM που πραγματοποιεί το μοντέλο κατά την διάρκεια εκπαίδευσής του.
- **Segment Embeddings:** Οι ενσωματώσεις τμημάτων χρησιμοποιούνται για να υποδείξουν την ακολουθία στην οποία ανήκει ένα token.



Εικόνα 4.13: Αναπαράσταση εισόδου BERT. Οι ενσωματώσεις εισόδου είναι το άθροισμα των ενσωματώσεων token, οι ενσωματώσεις τμηματοποίησης και οι ενσωματώσεις θέσης.

4.3.4. BERT Pre-Training

Σύμφωνα με το [60], για την προεκπαίδευση του μοντέλου BERT δεν χρησιμοποιήθηκαν παραδοσιακά μοντέλα γλώσσας από αριστερά προς τα δεξιά ή από δεξιά προς τα αριστερά. Για αυτό το κομμάτι χρησιμοποιήθηκαν οι εργασίες χωρίς επίβλεψη, Masked Language Model και η Next Sentence Prediction.

4.3.4.1. Masked Language Model (MLM)

Σε αυτό το κομμάτι του μοντέλου BERT, το 15% των tokens από κάθε ακολουθία καλύπτεται τυχαία (αντικαταστάθηκε με το διακριτικό [MASK]). Το μοντέλο εκπαιδεύεται να προβλέπει αυτά τα tokens χρησιμοποιώντας όλα τα άλλα tokens της ακολουθίας. Ωστόσο, η εργασία fine-tuning δεν πρόκειται σε καμία περίπτωση να δει το token [MASK] στην είσοδό του. Έτσι, για να προσαρμόσει το μοντέλο αυτές τις περιπτώσεις, το 80% των περιπτώσεων, τα tokens του καλύπτονται κατα 15%, 10% των περιπτώσεων, τα tokens αντικαθίστανται κατά 15% με τυχαία tokens και το 10% των περιπτώσεων μένουν ως έχουν, δηλαδή ανέγγιχτα.

Για παράδειγμα αν έχουμε την πρόταση «μου αρέσουν τα μήλα», κατά 80% των περιπτώσεων ο αλγόριθμος θα αντικαταστήσει με το token [MASK] ένα token και η παραπάνω πρόταση θα γίνει «μου αρέσουν τα [MASK]». Κατά 10% των περιπτώσεων ο αλγόριθμος θα αντικαταστήσει το token [MASK] με μια τυχαία λέξη και η πρόταση που δόθηκε ως παράδειγμα θα γίνει «μου αρέσουν τα αχλάδια». Τέλος κατά 10% των περιπτώσεων ο αλγόριθμος θα κρατήσει όπως έχει την πρόταση. Ο σκοπός αυτού είναι να προκαταλάβει την αναπαράσταση προς την πραγματική παρατηρούμενη λέξη [MASK].

4.3.4.2. Next Sentence Prediction (NSP)

Σε αυτό το κομμάτι του μοντέλου, υπάρχουν δύο ακολουθίες εισόδου (που χωρίζονται χρησιμοποιώντας το διακριτικό [SEP] και χρησιμοποιούνται ενσωματώσεις τμημάτων). Η Next Sentence Prediction (NSP), είναι μια διεργασία δυαδικής ταξινόμησης που περιλαμβάνει

πρόβλεψη για να πούμε εάν η δεύτερη πρόταση διαδέχεται την πρώτη πρόταση μέσα στο κείμενο σωστά. Για αυτό, το 50% των φορών, η επόμενη πρόταση χρησιμοποιείται σωστά ως η επόμενη πρόταση και στο 50% των φορών, μια τυχαία πρόταση λαμβάνεται από το σώμα για εκπαίδευση. Αυτό διασφαλίζει ότι το μοντέλο προσαρμόζεται στην εκπαίδευση σε πολλαπλές ακολουθίες.

Για παράδειγμα με είσοδο την ακολουθία «[CLS] πήγα να πάρω μήλα από τον μανάβη της [MASK] μου [SEP] δεν είχε καθόλου [MASK] και αναγκάστηκα να πάω αλλού [SEP]» έχουμε την ταμπέλα **IsNext** που υποδηλώνει ότι η ακολουθία «δεν είχε καθόλου [MASK] και αναγκάστηκα να πάω αλλού [SEP]» διαδέχεται την πρώτη ακολουθία του κειμένου, δηλαδή την «[CLS] πήγα να πάρω μήλα από τον μανάβη της [MASK] μου [SEP]». Ωστόσο όπως αναφέρθηκε το 50% των περιπτώσεων ο αλγόριθμος χρησιμοποιεί μια τυχαία πρόταση για να διαδεχθεί την πρώτη ακολουθία του κειμένου. Έτσι η ακολουθία «[CLS] πήγα να πάρω μήλα από τον μανάβη της [MASK] μου [SEP]» συμπληρώνεται με την παρακάτω τυχαία ακολουθία «το νερό δεν [MASK] κάτι είχε μέσα [SEP]» και γίνεται «[CLS] πήγα να πάρω μήλα από τον μανάβη της [MASK] μου [SEP] το νερό δεν [MASK] κάτι είχε μέσα [SEP]» ενώ τοποθετείται η ταμπέλα **NotNext**.

5. Methodology

5.1. Datasets

Πριν προχωρήσουμε στη μελέτη των αλγορίθμων ταξινόμησης που εξετάζουμε σε αυτή την διπλωματική εργασία, πρέπει πρώτα να μιλήσουμε για τις βάσεις δεδομένων τόσο των αγγλικών, όσο και των ελληνικών χαρακτήρων που χρησιμοποιήσαμε για να εφαρμόσουμε τα μοντέλα μηχανικής και βαθιάς μάθησης.

Για το κομμάτι των αγγλικών χρησιμοποιήθηκε η συλλογή δεδομένων PHEME, που αποτελείται από tweets χωρισμένα σε rumours και non-rumours. Πιο συγκεκριμένα όπως αναφέρει και το [61] η συλλογή δεδομένων PHEME περιέχει 9 γεγονότα εκ των οποίων τα 5 (Charlie Hebdo, Sydney siege, Ferguson, Ottawa shooting, Germanwings-crash) είναι μεγάλα γεγονότα και τα υπόλοιπα 4 μικρά (Putin missing, Prince Toronto, Gurlitt, Ebola Essien). Τα πέντε μεγαλύτερα συμβάντα δημιουργούν μια πιο ισορροπημένη συλλογή δεδομένων καθώς όπως αναφέρει και η παραπάνω πηγή αυτά είναι σημαντικά γεγονότα κρίσης κατά τα οποία κοινοποιήθηκαν και συζητήθηκαν αρκετά, με αποτέλεσμα να υπάρξουν πολλές διαφορετικές πτυχές συζητήσεων για τα συμβάντα, που με την σειρά τους έφεραν και αρκετές ψευδείς πληροφορίες.

Σε αυτό το σημείο είναι καλό να διευκρινιστεί ότι σε αυτή την διπλωματική εργασία θα χρησιμοποιηθούν μόνο τα 4.661 από τα 6.425 threads tweets που έχουν αξιολογηθεί από τους ερευνητές για την εγκυρότητα του περιεχομένου τους και όχι όλα τα tweets που είναι στο σύνολό τους 105.353 και είναι σχόλια των παραπάνω threads. Επιπλέον επιλέγονται μόνο 4.661 tweets και όχι 6.425 που είναι τα συνολικά threads των Rumoured και Non-Rumoured, διότι σύμφωνα με το [61] τα Rumoured tweets της βάσης αξιολογήθηκαν σε true, false και unverified από επαγγελματία δημοσιογράφο με αποτέλεσμα να ορίζουμε ως ψευδείς πληροφορίες τα tweets που βρίσκονται μόνο στην κατηγορία των Rumours και συγκεκριμένα αυτά που έχουν αξιολογηθεί ως False.

Στον Πίνακας 1 βλέπουμε αναλυτικά τη δομή της βάσης PHEME όπως αυτή δίνεται από το [61] και στον Πίνακας 2 βλέπουμε πώς χωρίζονται τα Rumours των γεγονότων σε True, False και Unverified.

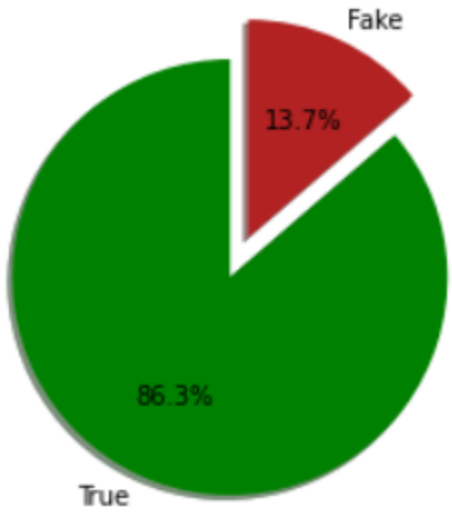
Πίνακας 1: Αναλυτική περιγραφή της PHEME βάσης δεδομένων για τα αγγλικά tweets

Events	Threads	Tweets	Rumours	Non- Rumours
Charlie Hebdo	2,079	38,268	458	1,621
Sydney siege	1,221	23,996	522	699
Ferguson	1,143	24,175	284	859
Ottawa shooting	890	12,284	470	420
Germanwings-crash	469	4,489	238	231
Putin missing	238	835	126	112
Prince Toronto	233	902	229	4
Gurlitt	138	179	61	77
Ebola Essien	14	226	14	0
Total	6,425	105,354	2,402	4,023

Πίνακας 2: Ανάλυση των Rumours σε True, False και Unverified

Events	Rumours	True	False	Unverified
Charlie Hebdo	458	193	116	149
Sydney siege	522	382	86	54
Ferguson	284	10	8	266
Ottawa shooting	470	329	72	69
Germanwings-crash	238	94	111	33
Putin missing	126	0	9	117
Prince Toronto	229	0	222	7
Gurlitt	61	59	0	2
Ebola Essien	14	0	14	0
Total	2,402	1067	638	697

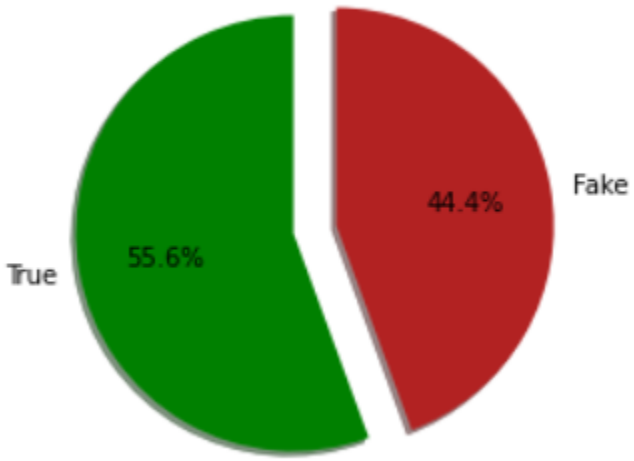
Έτσι βάσει των παραπάνω προκύπτει η παρακάτω μορφολογία των δεδομένων των αγγλικών tweets που χρησιμοποιήσαν τα μοντέλα της εργασίας μας ώστε να μπορέσουν να εκπαιδευτούν.



Εικόνα 5.1: Απεικόνιση των ποσοστών των δεδομένων True και Fake της αγγλικής συλλογής δεδομένων.

Όπως παρατηρούμε από την παραπάνω εικόνα, το σύνολο δεδομένων των αγγλικών tweets δεν είναι ομοιόμορφη δηλαδή αποτελείται από 86.3% από tweets που έχουν κατηγοριοποιηθεί ως True και 13.7% από tweets που έχουν κατηγοριοποιηθεί ως False. Συνήθως προτιμάται να υπάρχουν συλλογές δεδομένων που να είναι πιο ομοιόμορφα κατανεμημένα τα δεδομένα τους. Για αυτόν το λόγο, το πειραματικό κομμάτι των αγγλικών tweets πραγματοποιήθηκε με την παραπάνω

αναλογία δεδομένων και με μια παραλλαγή της ίδιας συλλογής με καλύτερη ομοιομορφία στα δεδομένα της όπως αυτή φαίνεται στην παρακάτω εικόνα. Το καινούργιο σύνολο δεδομένων που δημιουργήθηκε αποτελείται από 1438 threads, δηλαδή αποτελείται από το 30.8% του προηγούμενου συνόλου δεδομένων ενώ όπως φαίνεται και από την παρακάτω εικόνα αποτελείται από 55.6% tweets που κατηγοριοποιήθηκαν ως True και 44.4% από tweets που κατηγοριοποιήθηκαν ως False.

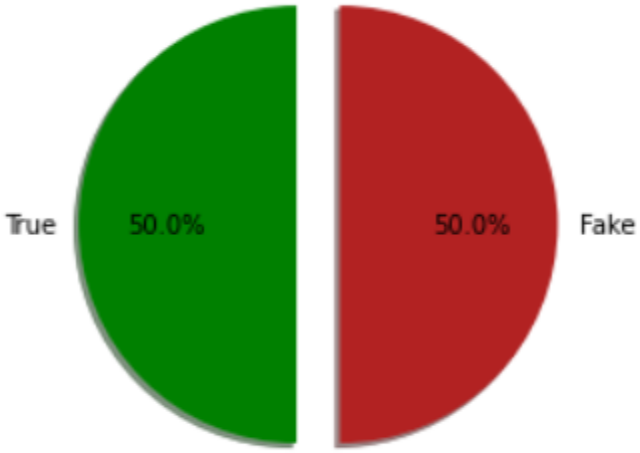


Εικόνα 5.2: Υποσύνολο της προηγούμενης συλλογής για να έχουμε ομοιόμορφες τις δύο κατηγορίες μας.

Ωστόσο όπως θα δούμε και στη συνέχεια το σύνολο δεδομένων των ελληνικών κειμένων είναι αρκετά μικρότερο από αυτό των αγγλικών. Για το λόγο αυτό θα δημιουργήσουμε ένα ακόμη υποσύνολο της βάσης PHEME όπως κάναμε και προηγούμενος, που θα έχει το διπλάσιο μήκος και την ίδια ομοιομορφία με αυτή της ελληνικής για να μπορέσουμε να βγάλουμε καλύτερα συμπεράσματα για τα αποτελέσματα των δεδομένων που αφορούν τα ελληνικά κείμενα. Επιλέγουμε να πάρουμε διπλάσιο μέγεθος από αυτής της ελληνικής και όχι αντίστοιχο αφού στην περίπτωση των αγγλικών tweets τα κείμενα είναι αρκετά πιο μικρά από αυτών των ελληνικών κειμένων και οι αλγόριθμοι δεν θα μπορούν να βρουν μοτίβα λέξεων εύκολα.

Για το κομμάτι των ελληνικών χαρακτήρων επιλέχθηκε η [62] όπου σύμφωνα με το αρθρογράφο του συγκεκριμένου άρθρου [63], αυτό το σύνολο δεδομένων αποτελείται από διάφορες ειδησεογραφικές θεματολογίες όπως πολιτική, οικονομία, πανδημία COVID-19 και τα παγκόσμια νέα. Η αξιολόγηση των κειμένων για την εγκυρότητά τους έγινε με την συγκέντρωση δεδομένων από αξιόπιστες ελληνικές εφημερίδες και ιστοσελίδες, ενώ επιπρόσθετα η εντόπιση των ψευδών ειδήσεων έγινε με την βοήθεια της ιστοσελίδας Ellinika Hoaxes που έχει πιστοποιηθεί από το διεθνές δίκτυο ελέγχου γεγονότων (IFCN) για την εγκυρότητα των δεδομένων της. Τέλος σε αυτό το σημείο είναι καλό να αναφερθεί ότι το συγκεκριμένο σύνολο δεδομένων δεν περιέχει ελληνικά tweets αλλά ελληνικά ειδησεογραφικά άρθρα που στο σύνολό τους είναι 100.

Όπως βλέπουμε από την παρακάτω εικόνα, το σύνολο δεδομένων των ελληνικών κειμένων είναι ομοιόμορφη και αποτελείται από 50 αληθινές ειδήσεις (True) και 50 ψευδείς ειδήσεις (False).



Εικόνα 5.3: Απεικόνιση των ποσοστών των κατηγοριών True και Fake στο ελληνική σύνολο δεδομένων.

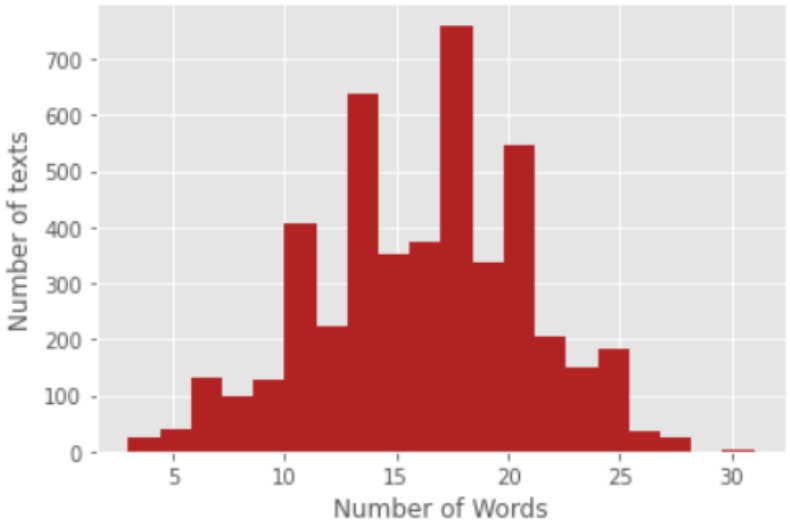
5.2. Experiments

5.2.1. Preprocessing of Datasets

Τα μοντέλα τόσο μηχανικής μάθησης όσο και βαθιάς μάθησης που πραγματοποιούνται στο πειραματικό κομμάτι αυτής της εργασίας αναλύθηκαν στο θεωρητικό κομμάτι της εργασίας. Για την κατασκευή αυτών των μοντέλων επιλέχθηκε η γλώσσα προγραμματισμού Python έκδοσης 3.8 και οι βιβλιοθήκες **tensorflow**, **keras**, **sklearn**, **torch**, **pycaret** και **pytorch**. Ωστόσο θα αναλυθούν μόνο τα μοντέλα βαθιάς μάθησης καθώς τα μοντέλα μηχανικής μάθησης δεν έχουν την περιπλοκότητα των μοντέλων της βαθιάς μάθησης. Έτσι θα παρουσιαστούν μόνο τα αποτελέσματά τους με τα αποτελέσματα και των υπολοίπων αλγορίθμων σε επόμενο κεφάλαιο.

Πιο αναλυτικά όπως αναφερθήκαμε και σε προηγούμενο κεφάλαιο, πριν χρησιμοποιήσουμε τα δεδομένα μας στα μοντέλα ταξινόμησης, πρέπει πρώτα να τα προ επεξεργαστούμε ώστε να μπορούν τα μοντέλα να επιτύχουν το καλύτερο δυνατό αποτέλεσμα. Αυτό ισχύει και για τα σύνολα δεδομένων των αγγλικών tweets και για το σύνολο των ελληνικών κειμένων. Ωστόσο στην περίπτωση των αγγλικών tweets δεν επιλέχθηκε η μεθοδολογία της αφαίρεσης των stopwords καθώς μετά από ανάλυση των δεδομένων του συνόλου δεδομένων και με το γεγονός ότι έχουμε να κάνουμε με μικρού μήκους tweets, τα stopwords δεν αφαιρέθηκαν.

Πρώτο βήμα στην προ επεξεργασία κειμένων είναι να μετατρέψουμε τα κείμενά μας σε tokens όπως ακριβώς είδαμε και στο κεφάλαιο Tokenization and Lowercasing. Στη συνέχεια με την βοήθεια της εντολής **hist()** της βιβλιοθήκης **pandas** δημιουργήσαμε ένα ιστόγραμμα για κάθε ένα από τα σύνολα δεδομένων της εργασίας για να μπορέσουμε να έχουμε μια καλύτερη απεικόνιση των δεδομένων ώστε να διαλέξουμε στην συνέχεια το καταλληλότερο μέγεθος για να μετατρέψουμε όλες τις ακολουθίες των βάσεων μας στο ίδιο μέγεθος. Στην Εικόνα 5.4 βλέπουμε το ιστόγραμμα από το σύνολο δεδομένων των αγγλικών tweets.



Εικόνα 5.4: Ιστόγραμμα την αγγλικού συνόλου δεδομένων με bins = 20. Αριθμός λέξεων ανά tweet

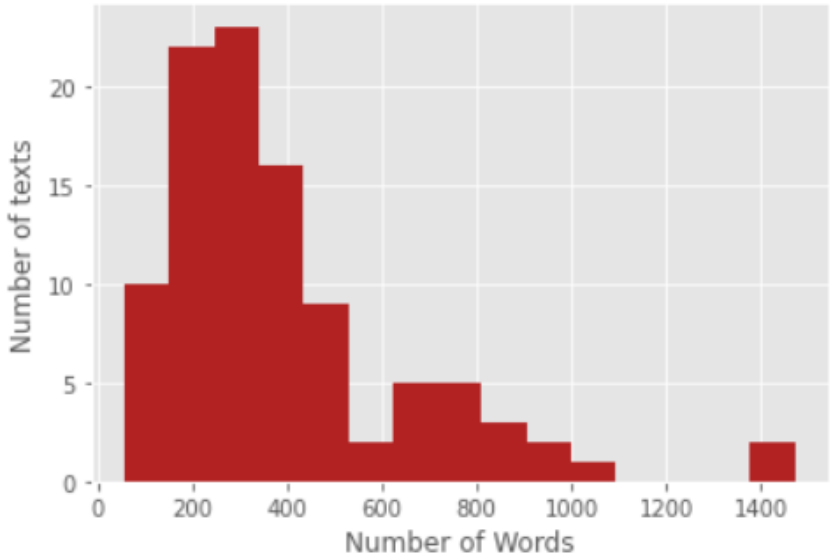
Στην περίπτωση του ελληνικού συνόλου δεδομένων δημιουργήθηκαν δύο ιστογράμματα, πριν και μετά από την αφαίρεση των stopwords από τα κείμενα καθώς σε αυτή την περίπτωση μετά από ανάλυση των δεδομένων διαπιστώθηκε ότι περιείχε αρκετό θόρυβο ο οποίος δεν θα βοηθούσε τα μοντέλα για την κατηγοριοποίηση των κειμένων στις σωστές κατηγορίες τους. Στην Εικόνα 5.5 βλέπουμε το ιστόγραμμα πριν την αφαίρεση των stopwords και στη Εικόνα 5.6 απεικονίζεται το ιστόγραμμα των κειμένων μετά την αφαίρεσή τους.

Για την αφαίρεση των ελληνικών stopwords έγινε χρήση της βιβλιοθήκης **spacy** και πιο συγκεκριμένα η **spacy.lang.el.stop_words**. Επιλέχθηκε αυτή η βιβλιοθήκη καθώς σε σχέση με την αντίστοιχη ελληνική βιβλιοθήκη της **nltk**, αποδείχθηκε ότι η συγκεκριμένη περιέχει **663** ελληνικά stopwords, έναντι της **nltk** που περιέχει **256** stopwords που στην πλειοψηφία τους είναι αρχαιοελληνικά stopwords. Επιπλέον μετά από ανάλυση των ελληνικών δεδομένων διαπιστώθηκε ότι τα κείμενα περιείχαν επιπλέον σύμβολα που δεν μπόρεσαν να απομακρυνθούν με την παραπάνω βιβλιοθήκη. Έτσι δημιουργήθηκε μια επιπρόσθετη συνάρτηση η **cleargreektext()** όπως αυτή φαίνεται και παρακάτω για να τα απομακρύνει.

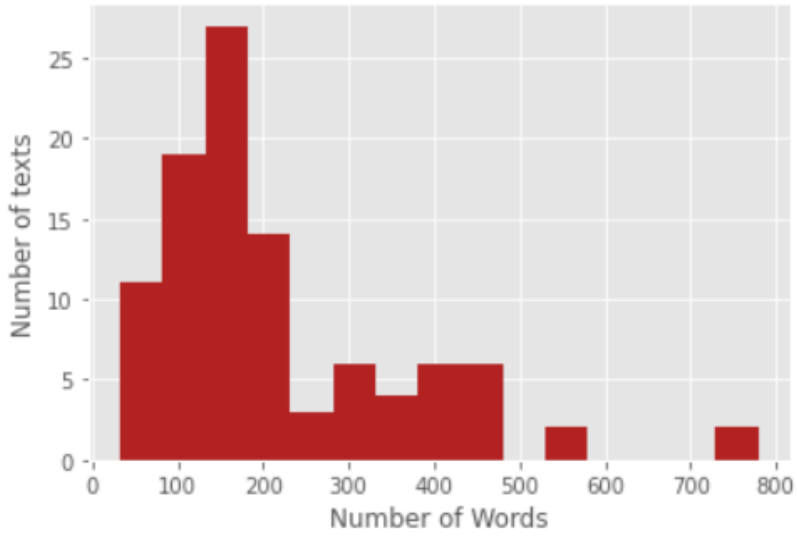
```
def cleargreektext(data):
    normalized = []
    for i in data:
        i = re.sub(' \n\n', ' ', i)
        i = re.sub('\n', '', i)
        i = re.sub(' - ', '', i)
        i = re.sub('- ', '', i)
        i = re.sub(', ', '', i)
        i = re.sub(' "', ' ', i)
        i = re.sub('" ', '', i)
        i = re.sub(', ', '', i)
        i = re.sub(' $', '', i)
```

```
        i = re.sub(' @', '', i)
        i = re.sub(': ', '', i)
        i = re.sub('«', '', i)
        i = re.sub('»', '', i)
        i = i.lower()
        normalized.append(i)
    return normalized
```

Κώδικας 1: Κώδικας από την συνάρτηση καθαρισμού συμβόλων



Εικόνα 5.5: Αριθμός λέξεων ανά κείμενο, πριν την αφαίρεση των stopwords από τα ελληνικά κείμενα.



Εικόνα 5.6: Αριθμός λέξεων ανά κείμενο, μετά την αφαίρεση των stopwords από τα ελληνικά κείμενα

Όπως φαίνεται από τα ιστογράμματα των ελληνικών κειμένων μετά την αφαίρεση των stopwords, τα κείμενα μίκρυναν αισθητά με αποτέλεσμα να οδηγούμαστε στο συμπέρασμα ότι τα μοντέλα μας θα βοηθηθούν αρκετά από αυτή την προ επεξεργασία κειμένου.

Τα δεδομένα των βάσεων τόσο των αγγλικών όσο και το ελληνικών κειμένων, χωρίστηκαν σε 30% δεδομένα που οι αλγόριθμοι ταξινόμησης θα προσπαθήσουν να τα κατηγοριοποιήσουν στην σωστή κατηγορία (testing data), δηλαδή οι ετικέτες αυτών των δεδομένων είναι άγνωστες για τα μοντέλα και σε 70% δεδομένα που τα μοντέλα θα εκπαιδευτούν πάνω σε αυτά όπου οι ετικέτες αυτών των δεδομένων είναι γνωστές στα μοντέλα (training data), ώστε να μπορέσουν να μάθουν τα μοτίβα των λέξεων μέσα στα κείμενα και να μπορέσουν στην συνέχεια να τρέξουν στα testing δεδομένα που προαναφέρθηκαν για να υπολογιστεί το ποσοστό επιτυχίας τους. Αυτή η διαδικασία χωρισμού των βάσεων επιτυγχάνεται με την εντολή **train_test_split()** της βιβλιοθήκης **sklearn**. Σε αυτό το σημείο είναι καλό να τονιστεί ότι σαν features για τα μοντέλα παίρνεται μόνο το κείμενο των βάσεων.

Με την βοήθεια των παραπάνω ιστογραμμάτων και την συνάρτηση της βιβλιοθήκης της tensorflow, **pad_sequences()** στην περίπτωση των μοντέλων βαθιάς μάθησης, δημιουργούμε ακολουθίες ίδιου μήκους για όλα τα κείμενά μας. Συγκεκριμένα τα αγγλικά κείμενα μετατρέπονται σε ακολουθίες μεγέθους **30** λέξεων ενώ τα ελληνικά κείμενα μετατρέπονται σε ακολουθίες μεγέθους **216** λέξεων. Με αυτή την τεχνική θέλουμε να βρούμε το καλύτερο μέγεθος που θα μετατραπούν οι ακολουθίες χωρίς όμως να χάνεται χρήσιμη πληροφορία των κειμένων ώστε τα μοντέλα βαθιάς μάθηση να μπορέσουν να λάβουν ίδιου μήκους ακολουθίες για να μπορέσουν να επεξεργαστούν τα δεδομένα, με όσο το δυνατό καλύτερα χαρακτηριστικά από αυτές γίνεται.

Σε αυτό το σημείο είναι καλό να αναφερθεί ότι για να χρησιμοποιήσουμε την εντολή **pad_sequences()**, θα πρέπει πρώτα να έχουμε μετατρέψει όλα τα κείμενά μας σε κωδικοποιημένες ακολουθίες, δηλαδή σε κάθε λέξη από τα λεξικά που δημιουργούνται κατά την προ επεξεργασία των κειμένων ανατίθεται ένας αριθμός. Αυτό επιτυγχάνεται με την εντολή **texts_to_sequences()** που είναι επίσης συνάρτηση της βιβλιοθήκης **tensorflow**.

Τέλος στα μοντέλα βαθιάς μάθησης που πραγματοποιήθηκαν, για τα βάρη των αγγλικών και ελληνικών λέξεων χρησιμοποιήθηκαν προ εκπαιδευμένα δεδομένα / λεξικά, που αναπαριστούν τις λέξεις των κειμένων σε διανύσματα 300 διαστάσεων τα οποία εκπαιδεύτηκαν με την χρήση της μεθόδου fastText πάνω σε δεδομένα της Common Crawl και της Wikipedia και πάρθηκαν από αυτή την ιστοσελίδα [64].

5.2.2. Deep Learning Models

Τα μοντέλα βαθιάς μάθησης που δημιουργήθηκαν στην γλώσσα προγραμματισμού python της έκδοσης 3.8 και είναι ίδια και για τις δύο γλώσσες (αγγλικά και ελληνικά), εκτός από την περίπτωση των μοντέλων BERT και Greek BERT που έχει φορτωθεί το ανάλογο προ εκπαιδευμένο μοντέλο για τις περιπτώσεις των αγγλικών και ελληνικών κειμένων αντίστοιχα. Αυτό που αλλάζει μέσα στα μοντέλα είναι το Embedding layer που τροποποιούμε τις παραμέτρους του, ώστε να ανταπεξέρχεται στις ανάγκες της εξεταζόμενης βάσης δεδομένων που θέλουμε. Για το λόγο αυτό σε αυτή την εργασία θα παρουσιαστούν μια φορά, ενώ στην συνέχεια θα είναι γνωστά και θα παρουσιάζονται μόνο τα δεδομένα από την διαδικασία εκπαίδευσης των μοντέλων.

Στο Embedding layer του κώδικα έχουν οριστεί ορισμένοι παράμετροι όπως **vocab_size_all, weights = [engfasttext_embedding_matrix_all], input_length = max_length** στην περίπτωση των αγγλικών κειμένων και αντίστοιχα **vocab_size_all_greek,**

weights=[greek_embedding_matrix_all], input_length = greek_max_length στην περίπτωση των ελληνικών κειμένων. Η πρώτη παράμετρος είναι το μέγεθος του λεξιλογίου που δημιουργείται κατά την διάρκεια της προετοιμασίας των δεδομένων που θα περαστούν στα μοντέλα, συν 1. Στην μεταβλητή **weights** τοποθετούμε ένα λεξικό με τα βάρη των λέξεων που πάρθηκαν από τα προ εκπαιδευμένα δεδομένα που αναφέραμε πιο πάνω, δηλαδή τοποθετούμε ένα **dict()** και στην παράμετρο **input_legth** τοποθετούμε το μήκος των ακολουθιών που ορίσαμε κατά την διάρκεια της προ επεξεργασίας των κειμένων, που όπως αναφέρθηκε στην περίπτωση των αγγλικών κειμένων είναι **30** και στην περίπτωση των ελληνικών κειμένων είναι **216**. Τέλος ο αριθμός **300** που έχει περαστεί ως όρισμα μέσα στο Embedding layer είναι οι διαστάσεις των διανυσμάτων που μετατρέπονται οι λέξεις με την βοήθεια των παραπάνω προ εκπαιδευμένων λεξικών.

Πριν προχωρήσουμε στα μοντέλα μας, πρέπει να αναφερθούμε στον κώδικα που έχει γραφτεί για την εκπαίδευση τους. Αρχικά έχει οριστεί μεταβλητή **early_stop**, όπως αυτή φαίνεται στην **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** και τοποθετείται στην συνέχεια μέσα στην συνάρτηση εκπαίδευσης των μοντέλων, ώστε να παρακολουθεί την παράμετρο **val_loss** που είναι οι αστοχίες των μοντέλων πάνω στα **validation data** που χρησιμοποιούνται κατά την διάρκεια της εκπαίδευση των μοντέλων και να διακόπτει την διαδικασία εκπαίδευσης άμα η μεταβλητή **val_loss** που επιβλέπει ανεβεί σε 2 συνεχόμενα **epoch**. Στην περίπτωση των μοντέλων που παίρνουν ως είσοδο τις αγγλικές βάσεις δεδομένων, τα **validation data** αποτελούνται από το 10% των training data, ενώ στην περίπτωση των μοντέλων που διαπραγματεύονται τα ελληνικά κείμενα είναι 20% των training data, καθώς έχουμε να κάνουμε με πολύ μικρότερο σύνολο δεδομένων και θέλουμε να βοηθήσουμε όσο μπορούμε τα μοντέλα κατά την εκπαίδευσή τους χωρίς όμως να επηρεάζουμε σημαντικά τα δεδομένα εκπαίδευσης.

```
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
patience=2, restore_best_weights=True)
```

Κώδικας 2: Μεταβλητή early_stop για τη παρακολούθηση της μεταβλητής val_los.

Πίνακας 3: Παράμετροι της συνάρτησης EarlyStopping() της βιβλιοθήκης keras

Parameters	Description
monitor	τιμή που παρακολουθείται, δηλαδή: val_loss
petience	αριθμός epoch χωρίς βελτίωση μετά τις οποίες η εκπαίδευση θα διακοπεί
restore_best_weights	ορίζουμε αυτήν τη μεταβλητή σε True εάν θέλουμε να διατηρήσουμε τα καλύτερα βάρη αφού σταματήσει η διαδικασία εκπαίδευσης

Αφού έχουμε δημιουργήσει μια συνάρτηση που θα επιβλέπει την διαδικασία εκπαίδευσης των μοντέλων, πρέπει να ορίσουμε και την συνάρτηση **fit()** που είναι η συνάρτηση εκπαίδευσης για μοντέλα. Όπως βλέπουμε από την **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** η συνάρτηση **.fit()** που θα χρησιμοποιήσουμε παίρνει ως ορίσματα τις μεταβλητές **x_train**, **y_train**, **epochs**, **validation_split**, **banch_size** και **callbacks**.

```
.fit(X_train, y_train, epochs=10, validation_split=0.1, batch_size=50,
callbacks=[early_stop])
```

Κώδικας 3: Συνάρτηση για την εκπαίδευση των μοντέλων.

Πίνακας 4: Παράμετροι της συνάρτησης fit() της βιβλιοθήκης keras

Parameters	Description
x_train	τα δεδομένα που θα προσπαθήσουν να κατηγοριοποιήσουν οι αλγόριθμοι
y_train	τα δεδομένα προς εκπαίδευση
epochs	καθορίζει τον αριθμό των φορών που ο αλγόριθμος εκμάθησης θα λειτουργήσει σε ολόκληρο το σύνολο δεδομένων εκπαίδευσης
validation_split	καθορίζει το ποσοστό των δεδομένων που θα πάρει από τα training data ώστε να αξιολογούν το μοντέλο κατά την εκπαίδευσή του.
banch_size	καθορίζει τον αριθμό των δειγμάτων προς επεξεργασία πριν από την ενημέρωση των εσωτερικών παραμέτρων του μοντέλου
callbacks	ελέγχει την διαδικασία εκπαίδευση και μπορεί να επέμβει ανάλογα με τα ορίσματα που της δίνουμε

Επίσης στα μοντέλα CNN με χρήση ngrams, RNN και RNN-LSTM βλέπουμε ότι υπάρχει και ένα επίπεδο **Dropout(0.5)**. Αυτό το επίπεδο ορίζει τυχαία εισόδους στο 0 με συχνότητα ρυθμού 50% σε κάθε βήμα κατά τη διάρκεια του χρόνου εκπαίδευσης του μοντέλου. Αυτή η διαδικασία βοηθάει το μοντέλο να μαθαίνει καλύτερα τα δεδομένα και να εντοπίζει καλύτερα τα μοτίβα που θα το βοηθήσουν με το αποτρέπεται το overfeeding και overadjustment των δεδομένων.

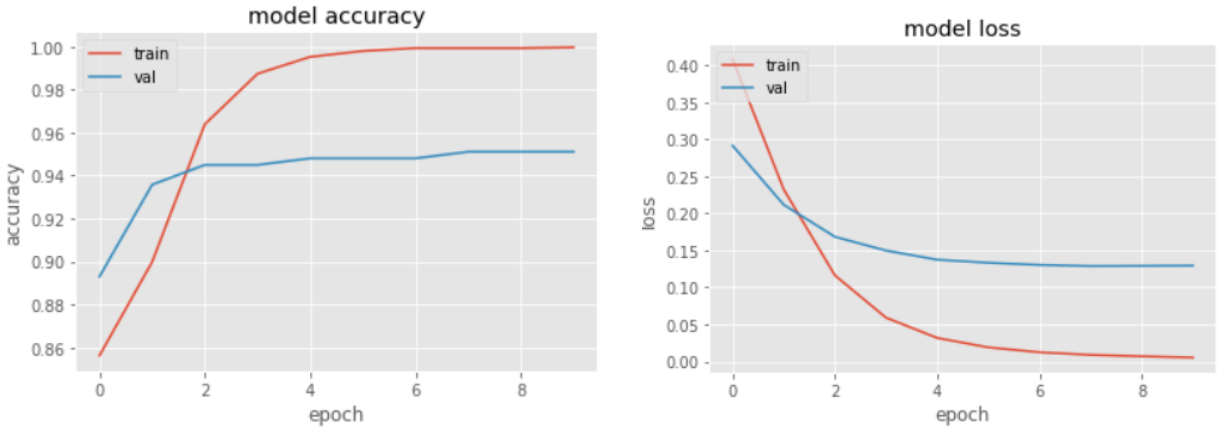
Τέλος σε αυτό το σημείο είναι καλό να αναφερθεί ότι ο κώδικας για την υλοποίηση των μοντέλων CNN με χρήση ngrams και BERT, χρησιμοποίησαν αρκετά χαρακτηριστικά των πηγών [65] και [66] αντίστοιχα.

5.2.2.1. Simple Artificial Neural Network

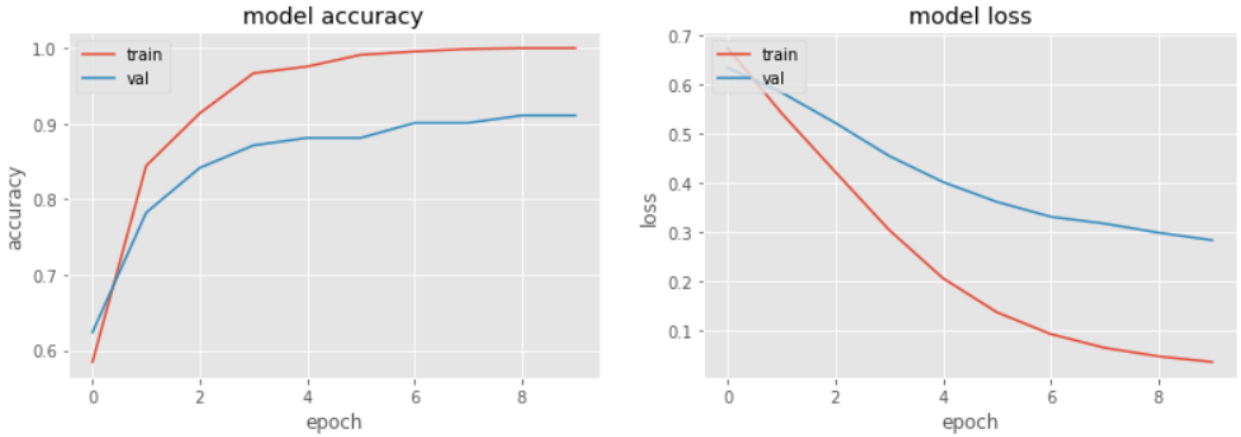
```
fasttext_simple_model = Sequential()
fasttext_simple_model.add(Embedding(vocab_size_all, 300,
weights=[engfasttext_embedding_matrix_all], input_length = max_length))
fasttext_simple_model.add(Flatten())
fasttext_simple_model.add(Dense(1, activation='sigmoid'))

fasttext_simple_model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
fasttext_simple_model.summary()
```

Κώδικας 4: Κώδικας για την υλοποίηση του απλού νευρωνικού δικτύου.

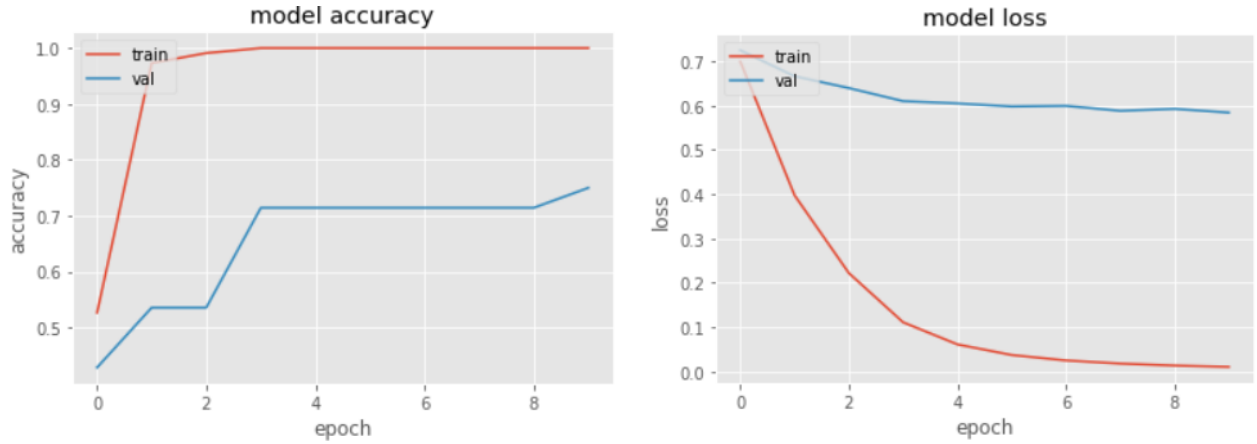
Training the model on PHEME dataset



Εικόνα 5.7: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 4661 tweets κατά την διάρκεια εκπαίδευσης του.

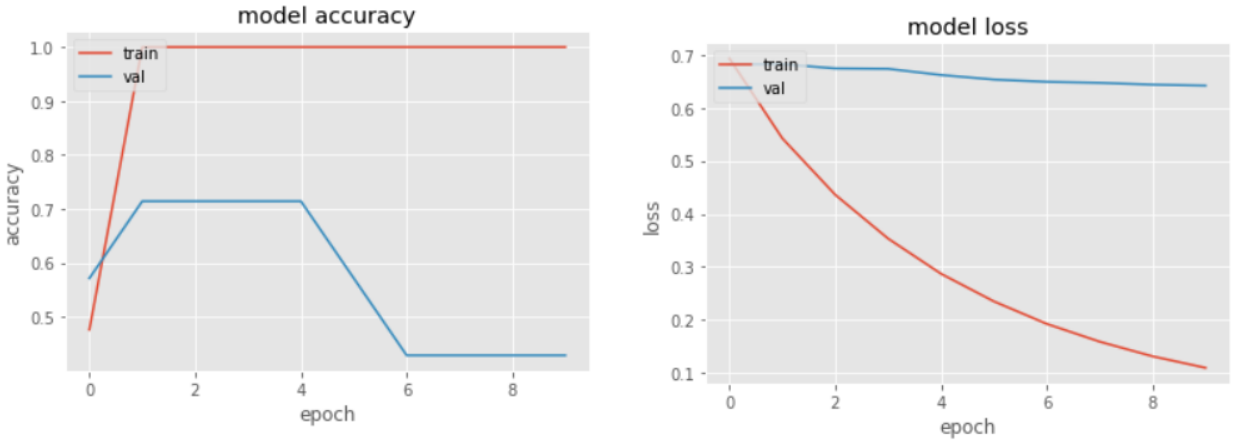


Εικόνα 5.8: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 1438 tweets κατά την διάρκεια εκπαίδευσης του.



Εικόνα 5.9: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 200 tweets κατά την διάρκεια εκπαίδευσης του.

Training the model on Greek Dataset



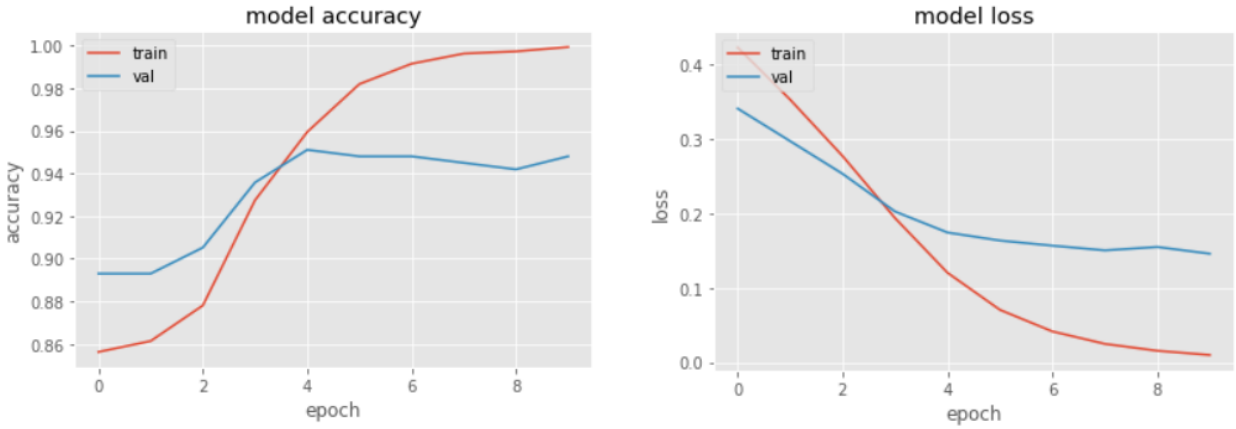
Εικόνα 5.10: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο ελληνικό σύνολο δεδομένων κατά την διάρκεια εκπαίδευσης του

5.2.2.2. Convolutional Neural Network

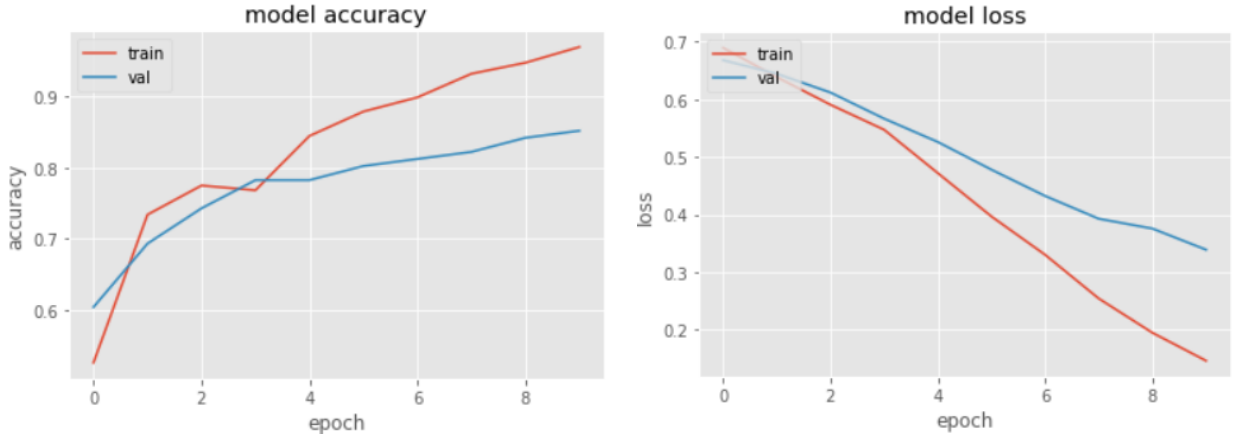
```
fasttext_simple_cnn = Sequential()
fasttext_simple_cnn.add(Embedding(vocab_size_all, 300,
weights=[engfasttext_embedding_matrix_all], input_length = max_length))
fasttext_simple_cnn.add(Conv1D(300, 5, activation='relu'))
fasttext_simple_cnn.add(MaxPooling1D())
fasttext_simple_cnn.add(Flatten())
fasttext_simple_cnn.add(Dense(250, activation='relu'))
fasttext_simple_cnn.add(Dense(1, activation='sigmoid'))

                        fasttext_simple_cnn.summary()
```

Κώδικας 5: Κώδικας για την υλοποίηση του συνελικτικού νευρωνικού δικτύου.

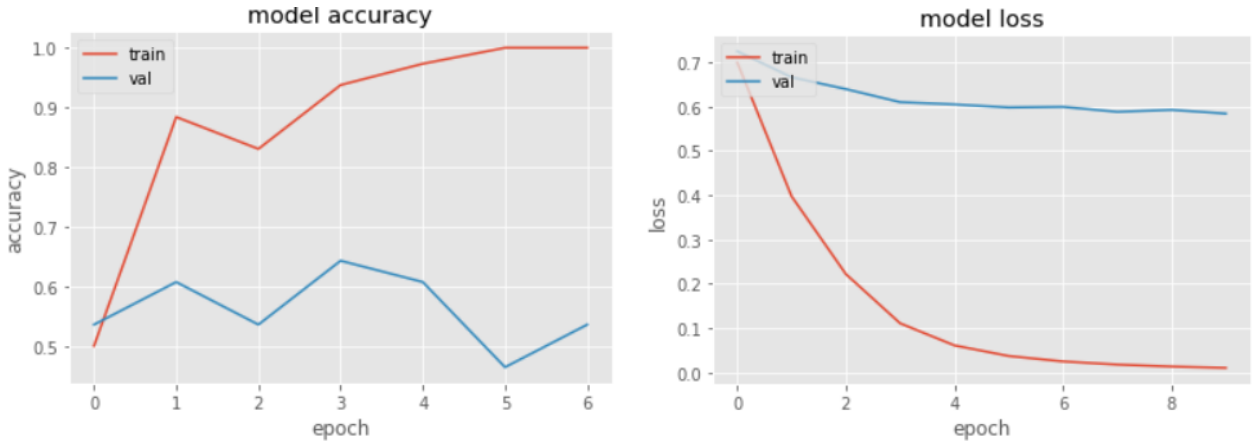
Training the model with PHEME dataset



Εικόνα 5.11: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 4661 tweets κατά την διάρκεια εκπαίδευσης του.

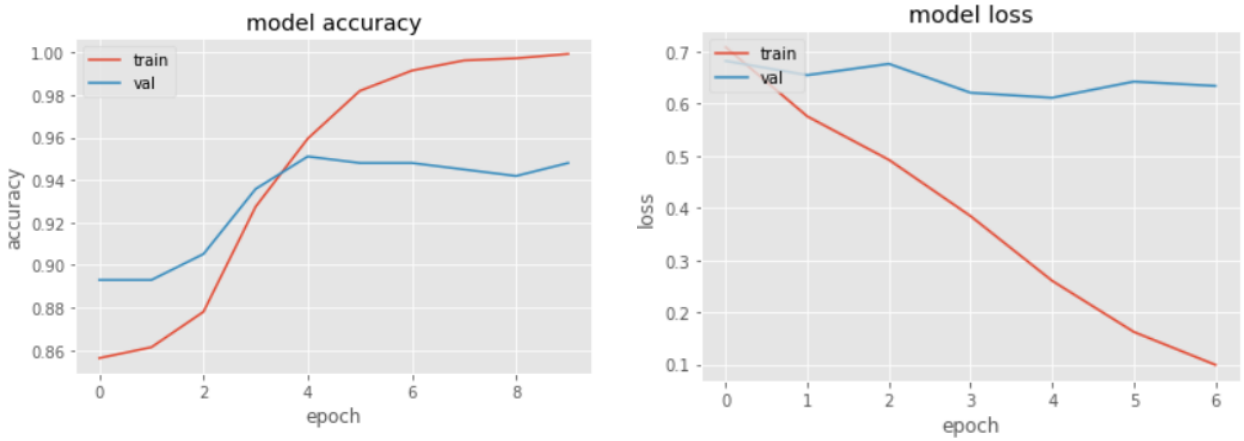


Εικόνα 5.12: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 1438 tweets κατά την διάρκεια εκπαίδευσης του.



Εικόνα 5.13: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 200 tweets κατά την διάρκεια εκπαίδευσης του.

Training the model on Greek Dataset



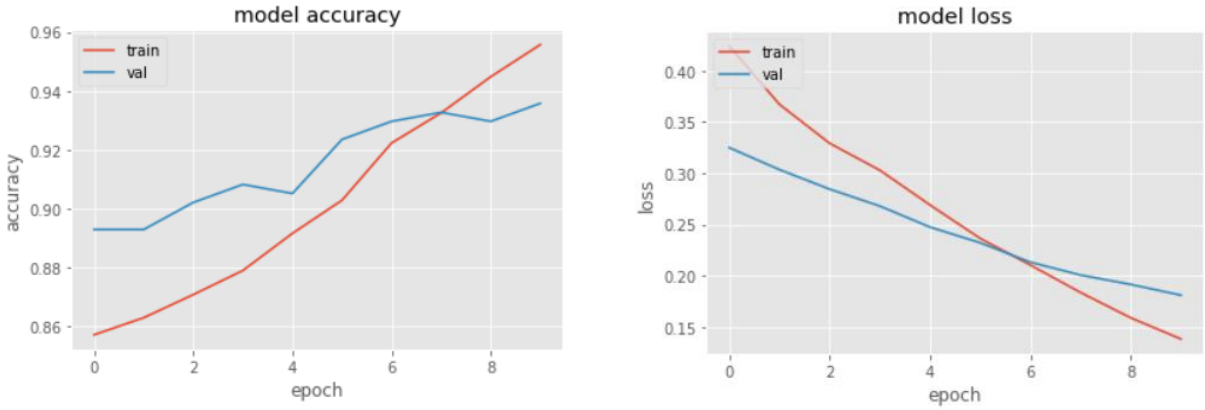
Εικόνα 5.14: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο ελληνικό σύνολο δεδομένων

5.2.2.3. Convolutional Neural Networks with use of ngrams

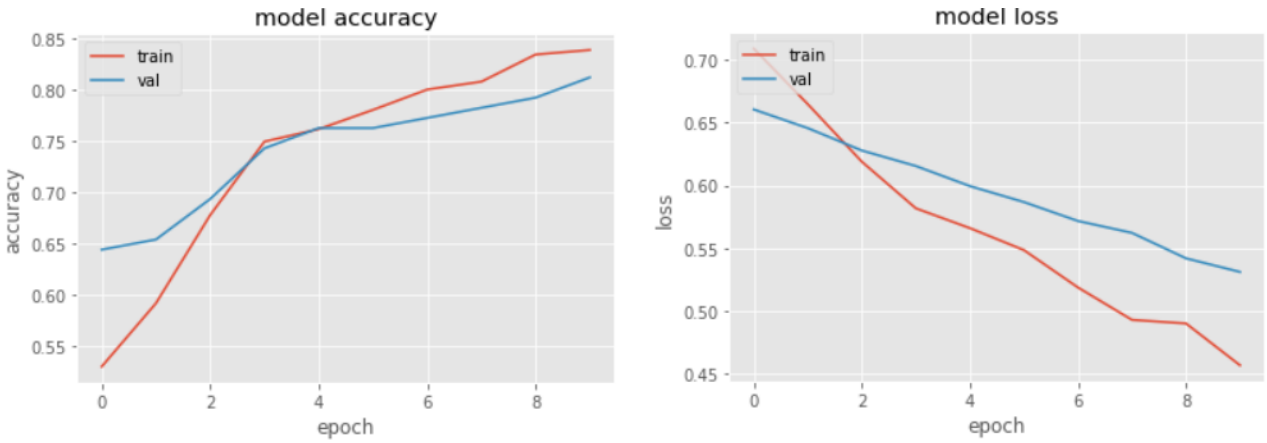
```
# n_grams = filter sizes
n_grams = [4, 5, 6]


text_input = tf.keras.layers.Input(shape=(max_length,), name='input')
text_embed = tf.keras.layers.Embedding(vocab_size_all, 300,
input_length = max_length,
weights=[engfasttext_embedding_matrix_all])(text_input)
branches = []

for n in n_grams:
    branch = Conv1D(filters= 200, kernel_size=n, activation= 'relu',
name='Conv_'+'_'+str(n))(text_embed)
    branch = MaxPooling1D(pool_size = max_length-n+1, strides=None,
padding='valid', name='MaxPooling_'+'_'+str(n))(branch)
    branch = Flatten(name='Flatten_'+'_'+str(n))(branch)
    branches.append(branch)

concate_layer = concatenate(branches, axis=-1)
drop_layer = Dropout(0.5)(concate_layer)
dense_layer = Dense(1, activation='sigmoid', name='output')(drop_layer)

fasttext_mulmodel = tf.keras.models.Model(inputs = text_input, outputs
= dense_layer)

fasttext_mulmodel.compile(loss={'output': 'binary_crossentropy'},
optimizer='adam')
fasttext_mulmodel.summary()
```

Κώδικας 6: Κώδικας για την υλοποίηση του συνελικτικού νευρωνικού δικτύου με χρήση ngrams.

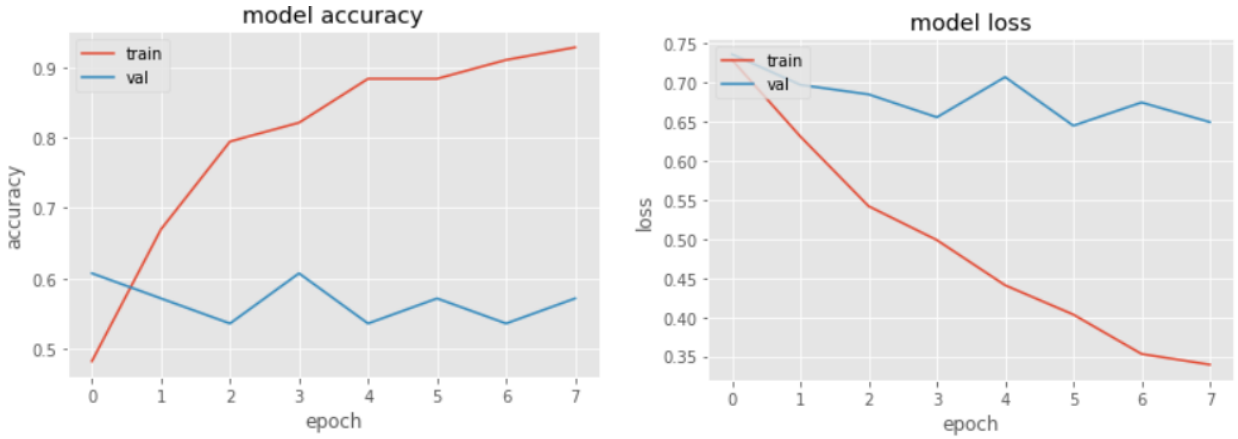
Training the model on PHEME dataset



Εικόνα 5.15: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch στο σύνολο δεδομένων PHEME με τα 4661 tweets κατά την διάρκεια εκπαίδευσης του.

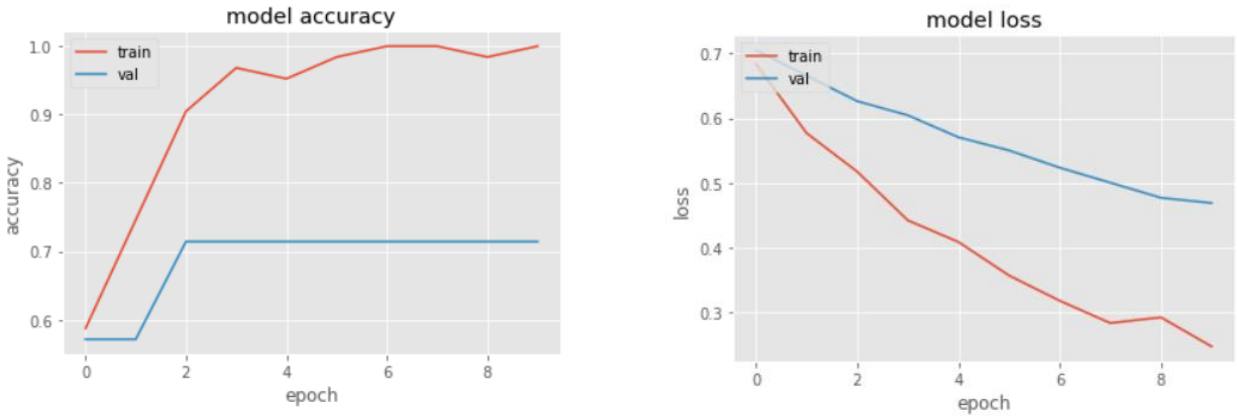


Εικόνα 5.16: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 1438 tweets κατά την διάρκεια εκπαίδευσης του.



Εικόνα 5.17: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 200 tweets κατά την διάρκεια εκπαίδευσης του.

Training the model on Greek Dataset



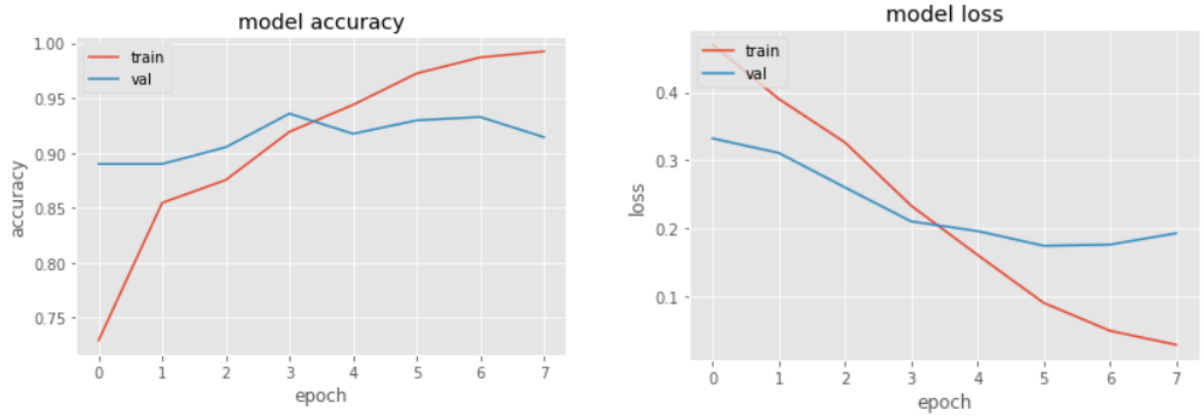
Εικόνα 5.18: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο ελληνικό σύνολο δεδομένων κατά την διάρκεια εκπαίδευσης του.

5.2.2.4. Recurrent Neural Network

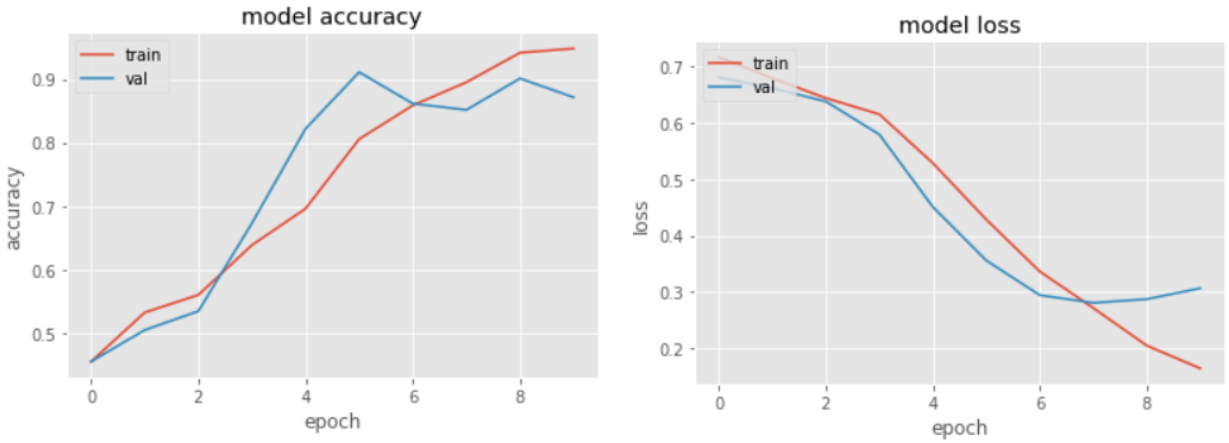
```
model_rnn_fasttext = Sequential()

model_rnn_fasttext.add(Embedding(vocab_size_all, 300,
weights=[engfasttext_embedding_matrix_all], input_length = max_length))
model_rnn_fasttext.add(Bidirectional(tf.keras.layers.SimpleRNN(64, )))
model_rnn_fasttext.add(Dropout(0.5))
model_rnn_fasttext.add(Dense(64))
model_rnn_fasttext.add(Dense(1))

model_rnn_fasttext.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
model_rnn_fasttext.summary()
```

Κώδικας 7: Κώδικας για την υλοποίηση του επαναλαμβανόμενου νευρωνικού δικτύο.

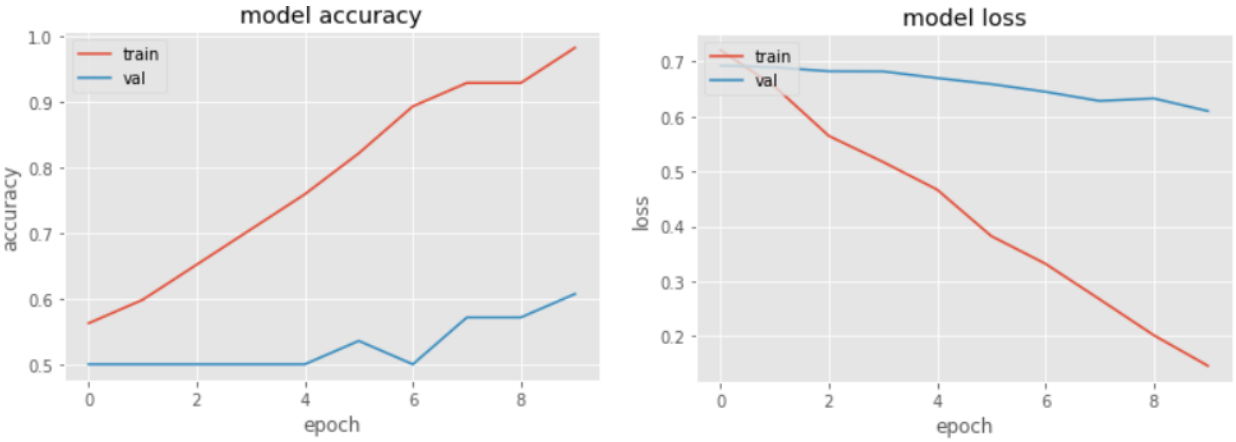
Training the model on PHEME dataset



Εικόνα 5.19: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 4661 tweets κατά την διάρκεια εκπαίδευσης του.

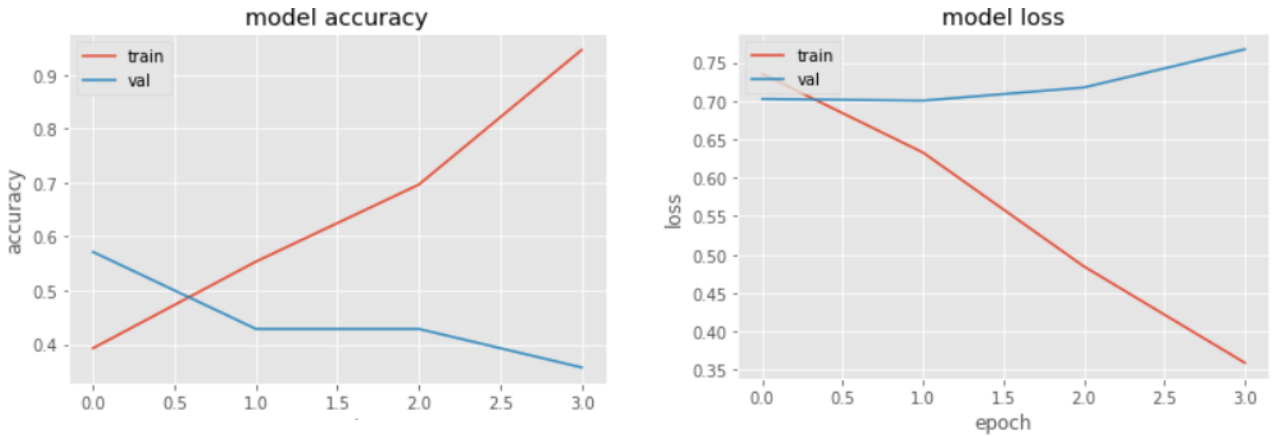


Εικόνα 5.20: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 1438 tweets κατά την διάρκεια εκπαίδευσης του.



Εικόνα 5.21: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 200 tweets κατά την διάρκεια εκπαίδευσης του.

Training the model on Greek Dataset



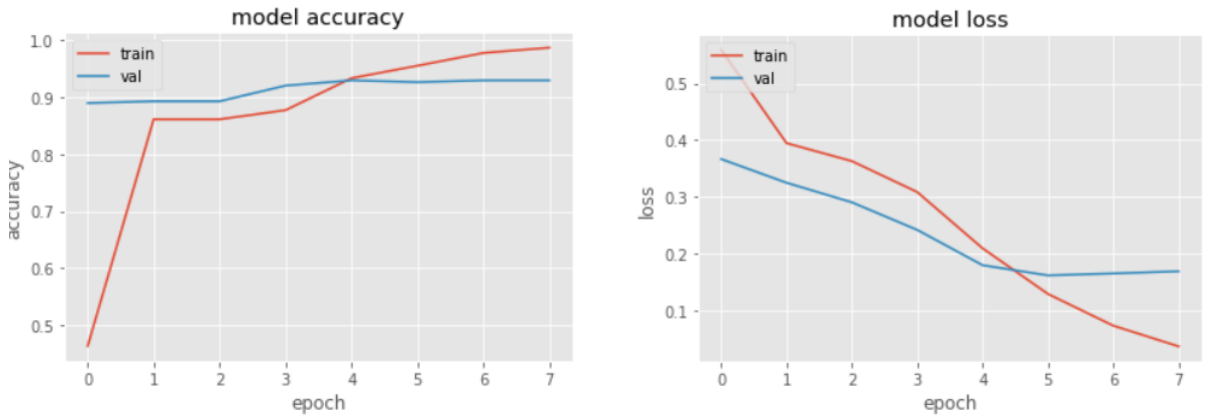
Εικόνα 5.22: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο ελληνικό σύνολο δεδομένων κατά την διάρκεια εκπαίδευσης του.

5.2.2.5. Recurrent Neural Networks with LSTM layer

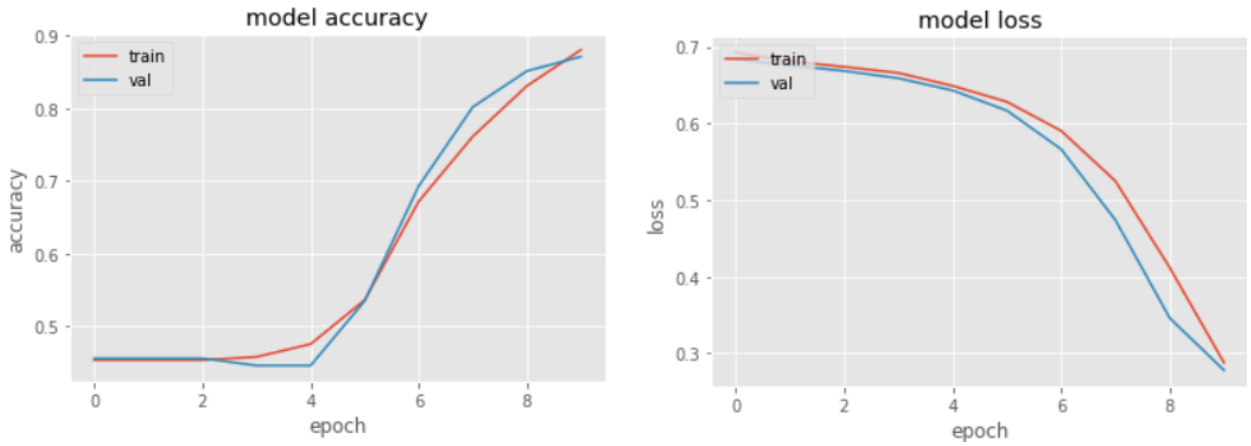
```
model_lstm_fasttext = Sequential()
model_lstm_fasttext.add(Embedding(vocab_size_all, 300,
weights=[engfasttext_embedding_matrix_all], input_length = max_length))
model_lstm_fasttext.add(Bidirectional(tf.keras.layers.LSTM(64, )))
model_lstm_fasttext.add(Dropout(0.5))
model_lstm_fasttext.add(Dense(64))
model_lstm_fasttext.add(Dense(1))

model_lstm_fasttext.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])
model_lstm_fasttext.summary()
```

Κώδικας 8: Κώδικας για την υλοποίηση του επαναλαμβανόμενου νευρωνικού δικτύου με LSTM επίπεδο.

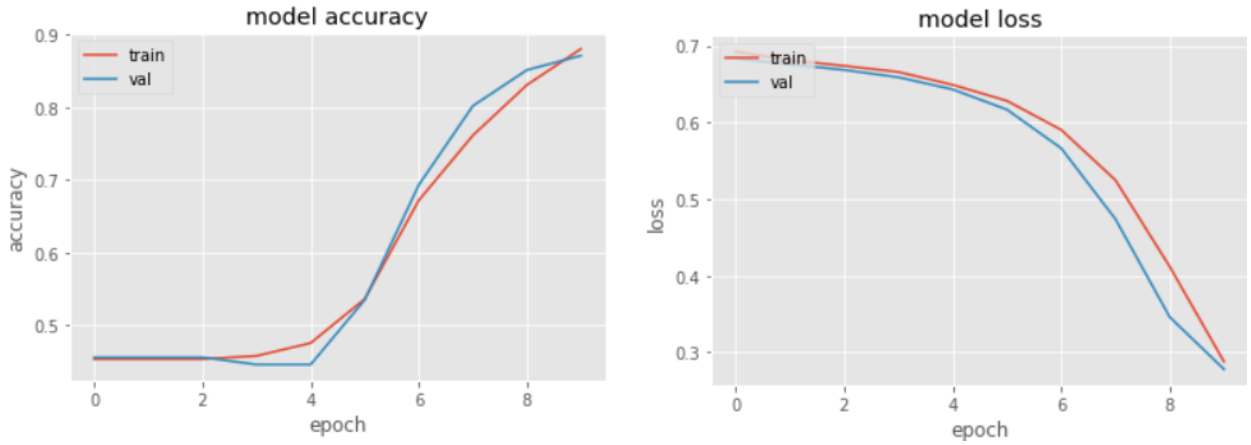
Training the model on PHEME dataset



Εικόνα 5.23: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 4661 tweets κατά την διάρκεια εκπαίδευσης του.

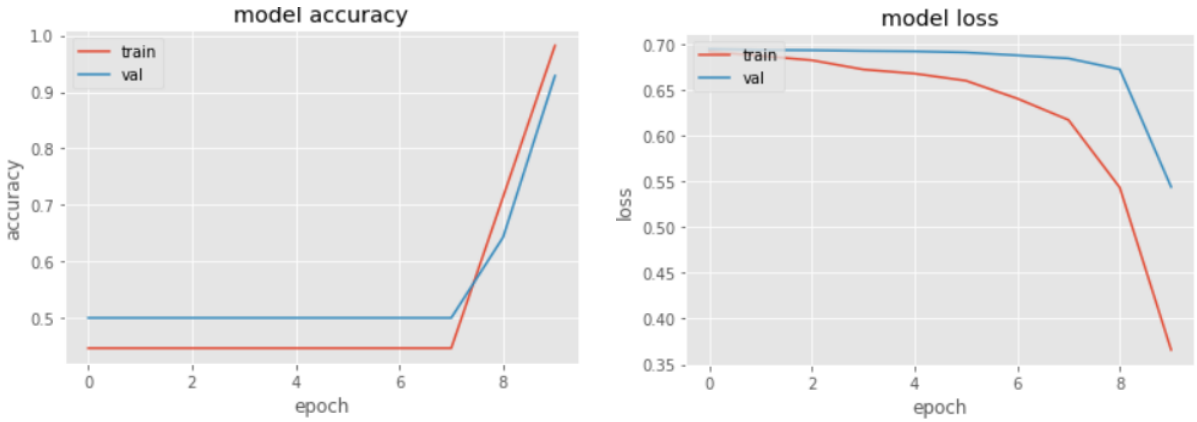


Εικόνα 5.24: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 1438 tweets κατά την διάρκεια εκπαίδευσης του.



Εικόνα 5.25: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο σύνολο δεδομένων PHEME με τα 200 tweets κατά την διάρκεια εκπαίδευσης του.

Training the model on Greek Dataset



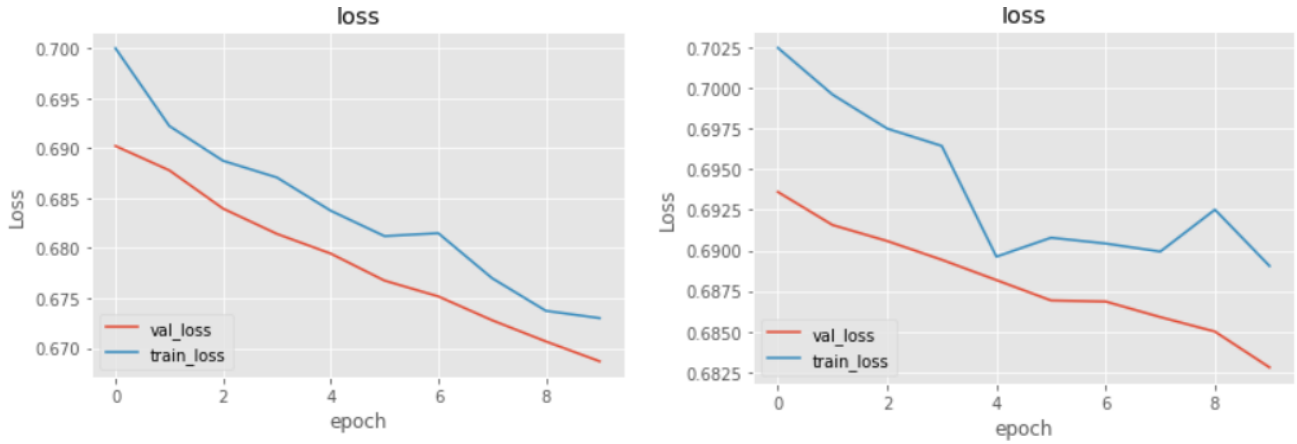
Εικόνα 5.26: Απεικόνιση της ακρίβειας και της απώλειας του μοντέλου σε συνάρτηση με το κάθε epoch πάνω στο ελληνικό σύνολο δεδομένων κατά την διάρκεια εκπαίδευσης του.

BERT

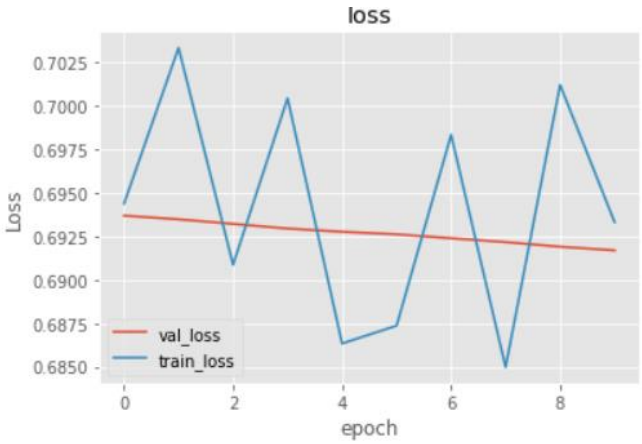
```
class BERT_Arch(nn.Module):

    def __init__(self, bert):

      super(BERT_Arch, self).__init__()
      self.bert = bert
      self.dropout = nn.Dropout(0.5)
      self.relu =  nn.ReLU()
      self.fc1 = nn.Linear(768,512)
      self.fc2 = nn.Linear(512,2)
      self.softmax = nn.LogSoftmax(dim=1)
    def forward(self, sent_id, mask):

      cls_hs = self.bert(sent_id, attention_mask=mask)['pooler_output']
      x = self.fc1(cls_hs)
      x = self.relu(x)
      x = self.dropout(x)
      x = self.fc2(x)
      x = self.softmax(x)

      return x

bert = AutoModel.from_pretrained('bert-base-uncased')
model = BERT_Arch(bert)
```

Κώδικας 9: Κώδικας για την υλοποίηση του επαναλαμβανόμενου BERT μοντέλου.

Training the model with PHEME dataset

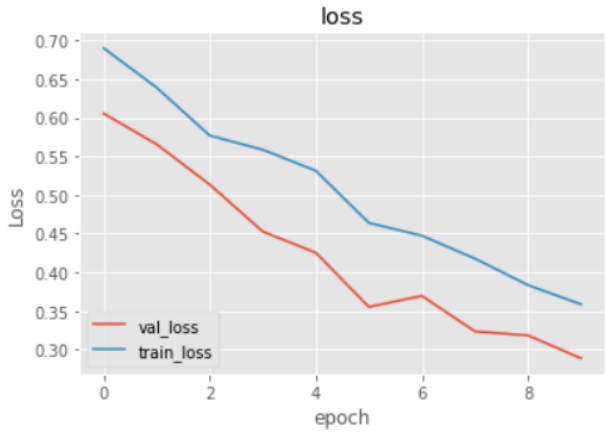


Εικόνα 5.27: Βλέπουμε τις αστοχίες του μοντέλου BERT πάνω στο σύνολο PHEME με τα 4.661 tweets (αριστερή εικόνα) και στο υποσύνολό της με τα 1.438 (δεξιά εικόνα)



Εικόνα 5.28: Απεικόνιση των αστοχιών του μοντέλου BERT πάνω στο υποσύνολο του συνόλου δεδομένων PHEME με τα 200 tweets.

Training the model with Greek Dataset



Εικόνα 5.29: Απεικόνιση των αστοχιών του μοντέλου BERT πάνω στο ελληνικό σύνολο δεδομένων.

5.2.3. Results

Πρώτα από όλα, παρατηρώντας τα δεδομένα εκπαίδευσης των αλγορίθμων βαθιάς μάθησης που εφαρμόστηκαν πάνω στο ελληνικό σύνολο δεδομένων, διαπιστώνουμε ότι τα μοντέλα δυσκολεύονται να βρουν χρήσιμα μοτίβα λέξεων από τα κείμενα καθώς το validation accuracy των μοντέλων είναι αρκετά χαμηλό ενώ ταυτόχρονα το validation loss είναι αρκετά υψηλό. Ωστόσο την ίδια παρατήρηση κάνουμε και για τα μοντέλα που εφαρμόστηκαν για το αγγλικό σύνολο δεδομένων των 200 tweets, που έφεραν παρόμοια αποτελέσματα με αυτά των ελληνικών κειμένων. Έτσι οδηγούμαστε στο συμπέρασμα ότι το πρόβλημα των μοντέλων έγκειται στο γεγονός ότι δεν υπάρχουν αρκετά δεδομένα εκπαίδευσης για να μάθουν τα απαραίτητα σημασιολογικά μοτίβα των λέξεων ώστε να αποδώσουν καλύτερα πάνω στα εξεταζόμενα δεδομένα. Σε συνάρτηση με τα αποτελέσματα των μοντέλων μηχανικής μάθησης των ελληνικών κειμένων, που φαίνεται να έχουν υψηλά ποσοστά επιτυχίας που είναι παρόμοια με αυτών των αγγλικών βάσεων, μπορούμε να υποθέσουμε ότι με ένα μεγαλύτερο ελληνικό σύνολο δεδομένων θα είχαμε παρόμοια ποσοστά επιτυχίας με αυτά της αγγλικής βάσης των 4.661 tweets στα μοντέλα βαθιάς μάθησης.

Επιπλέον βλέποντας τους παρακάτω πίνακες παρατηρούμε ότι τα καλύτερα αποτελέσματα εντοπίζονται από τα μοντέλα που εκπαιδεύτηκαν πάνω στο αγγλικό σύνολο δεδομένων με τα 4.661 tweets καθώς τα μοντέλα είχαν μεγάλο εύρος εκπαιδευόμενων δεδομένων που τα βοήθησαν να βρουν περισσότερα ιδιαίτερα μοτίβα στα κείμενα. Ενώ με μια πρώτη ματιά φαίνεται ότι όλα τα μοντέλα έχουν φέρει καλύτερα αποτελέσματα στο προαναφερθέν σύνολο δεδομένων, το μοντέλο BERT και συγκεκριμένα η παραλλαγή του, το Greek BERT που εφαρμόστηκε πάνω στο ελληνικό σύνολο δεδομένων, έφερε τα καλύτερα αποτελέσματα με ποσοστά 100% σε όλες τις κατηγορίες αξιολόγησης του μοντέλου. Αυτό που κάνει ακόμα πιο ενδιαφέρον το παραπάνω ποσοστό, είναι ότι το επόμενο καλύτερο F1-score που λαμβάνουμε από μοντέλο για το ελληνικό σύνολο δεδομένων, είναι 88,89% και είναι από το μοντέλο μηχανικής μάθησης Naïve Bayes, ενώ το αμέσως επόμενο καλύτερο F1-score από μοντέλο βαθιάς μάθησης για το ελληνικό σύνολο είναι μόλις 73,33% του RNN-LSTM μοντέλου.

Άλλο ένα ενδιαφέρον συμπέρασμα που βγάζουμε από τους παρακάτω πίνακες αποτελεσμάτων, είναι ότι σε όλες τις βάσεις δεδομένων, το μοντέλο RNN-LSTM έφερε καλύτερα αποτελέσματα τόσο στο accuracy όσο και στο F1-score από το μοντέλο RNN, επιβεβαιώνοντας την θεωρία που μελετήσαμε ότι τα επαναλαμβανόμενα νευρωνικά μοντέλα RNN που κάνουν χρήση LSTM κελιών, αντιμετωπίζουν καλύτερα το πρόβλημα vanishing gradient και επιτρέπουν

στο επαναλαμβανόμενο νευρωνικό μοντέλο να εκπαιδεύεται καλύτερα πάνω στα δεδομένα, χωρίς να χάνονται χρήσιμες πληροφορίες κατά την διαδικασία εκπαίδευσής του. Επίσης το ίδιο συμπέρασμα μπορούμε να βγάλουμε και για το μοντέλο μηχανικής μάθησης SVM στις αγγλικές βάσεις δεδομένων . Ωστόσο στην περίπτωση των ελληνικών κειμένων φαίνεται ότι ο αλγόριθμος Naïve Bayes φέρνει καλύτερα αποτελέσματα, ενώ το μοντέλο SVM έρχεται τρίτο σε απόδοση.

Πίνακας 5: Επιδόσεις των μοντέλων πάνω στα testing data, που εκπαιδεύτηκαν στο σύνολο δεδομένων PHEME με τα 4.661 threads.

Models	Accuracy	Precision	Recall	F1 score
Simple ANN	94,21%	99,50%	94,10%	96,72%
CNN	94,28%	98,33%	95,17%	96,72%
CNN with ngrams	92,20%	98,83%	92,60%	95,61%
RNN	91,27%	96,09%	93,90%	94,98%
RNN-LSTM	93,78%	97,17%	95,66%	96,41%
BERT	82,38%	93,39%	85,67%	89,36%
SVM	96,71%	97,07%	99,17%	98,11%
NB	90,28%	89,84%	99.80%	94,65%
RF	95,57%	95,31%	99,75%	97,48%
KNN	94,71%	97,48%	96,34%	96,91

Πίνακας 6: Επιδόσεις των μοντέλων πάνω στα testing data, που εκπαιδεύτηκαν στο σύνολο δεδομένων PHEME με τα 1438 threads.

Models	Accuracy	Precision	Recall	F1 score
Simple ANN	87,50 %	92,80 %	86,56 %	89,57 %
CNN	84,49 %	90,00 %	84,26 %	87,04 %
CNN with ngrams	79,62 %	88,00 %	79,13 %	83,33 %
RNN	84,25 %	86,00 %	86,69 %	86,34 %
RNN-LSTM	84,95 %	84,80 %	88,70 %	86,70 %
BERT	69,23%	72,85%	70,83%	71,83%
SVM	93,29%	93,68%	94,80%	94,23%
NB	91,90%	93,17%	92,80%	92,99%
RF	89,12%	89,80%	91,60%	90,69%
KNN	88,89%	95,91%	84,40%	89,79%

Πίνακας 7: Επιδόσεις των μοντέλων πάνω στα testing data, που εκπαιδεύτηκαν στο σύνολο δεδομένων PHEME με τα 200 threads.

Models	Accuracy	Precision	Recall	F1 score
Simple ANN	68,33%	63,63%	75,00%	68,85%
CNN	63,33%	63,63%	67,74%	65,62%
CNN with ngrams	56,66%	57,57%	61,29%	59,37%
RNN	65,00%	45,45%	83,33%	58,82%
RNN-LSTM	70,00%	51,51%	89,47%	65,38%
BERT	66,66%	66,66	66,66%	66,6%
SVM	85,00%	98,51%	72,73%	84,21%
NB	80,00%	97,54%	63,64%	77,78%
RF	78,33%	85,71%	72,73%	78,69%
KNN	73,33%	90,48%	57,58%	70,37%

Πίνακας 8: Επιδόσεις των μοντέλων πάνω στα testing data, που εκπαιδεύτηκαν στο ελληνικό σύνολο δεδομένων

Models	Accuracy	Precision	Recall	F1 score
Simple ANN	63,33 %	83,33 %	52,63 %	64,51 %
CNN	60,00 %	75,00 %	50,00 %	60,00 %
CNN with ngrams	60,00 %	83,33 %	50,00 %	62,50 %
RNN	66,66 %	33,33 %	66,66 %	44,44 %
RNN-LSTM	73,33 %	91,66 %	61,11 %	73,33 %
BERT	100%	100%	100%	100%
SVM	86,67%	75,00%	99,07%	85,71%
NB	90,00%	80,00%	99,97%	88,89%
RF	60,00%	50,00%	99,54%	66,67%
KNN	90,00%	84,62%	91,67%	88,00%

6. Conclusion and Future work

Παρατηρώντας τα αποτελέσματα από τις αξιολογήσεις των μοντέλων βλέπουμε ότι στην περίπτωση των αγγλικών tweets, παρόλο που έχουμε να κάνουμε με κείμενα μήκους 20 με 30 λέξεων, τα μοντέλα τόσο μηχανικής μάθησης όσο και βαθιάς μάθησης κατάφεραν να τα κατηγοριοποιήσουν στις σωστές κατηγορίες τους, οδηγώντας μας στο συμπέρασμα ότι τα ψευδή και αληθή tweets έχουν μοτίβα λέξεων που μπορούν να βρεθούν και να επεξεργαστούν από τα μοντέλα της εργασίας. Επίσης, παρόλο που στην περίπτωση της βάσης δεδομένων των 4.661 tweets έχουμε ανομοιόμορφες τις δυο κατηγορίες Fake και True, οι αλγόριθμοι κατάφεραν να ταξινομήσουν με επιτυχία τα κείμενα. Ωστόσο και στην περίπτωση του συνόλου δεδομένων των 1.438 tweets, τα μοντέλα έφεραν πάρα πολύ υψηλά ποσοστά F1-score ενισχύοντας την προηγούμενη εικασία μας ότι τα κείμενα των δύο κατηγοριών έχουν σημαντικές διαφορές στον τρόπο γραφής τους με αποτέλεσμα να δημιουργούνται μοτίβα που μπορούν να εντοπιστούν από τα μοντέλα.

Στην περίπτωση του ελληνικού συνόλου δεδομένων με τα εκατό ειδησεογραφικά άρθρα, βλέπουμε ότι παρόλο του μικρού μήκους συνόλου, τα μοντέλα κατάφεραν να βρουν και σε αυτή την περίπτωση χρήσιμα μοτίβα λέξεων παρόλο των σχετικά χαμηλών ποσοστών τους επιτυχίας. Ωστόσο όπως αναφέραμε και σε προηγούμενη παράγραφο εικάζουμε ότι το πρόβλημα έγκειται στο γεγονός ότι δεν υπάρχουν αρκετά δεδομένα εκπαίδευσης και όχι στα μοντέλα ή στα προ εκπαιδευμένα λεξικά που εφαρμόστηκαν σε αυτή την εργασία. Αυτή η εικασία ενισχύεται με τα αποτελέσματα του αγγλικού συνόλου δεδομένων με τα 200 tweets που φαίνεται οι αλγόριθμοι βαθιάς μάθησης να μην φέρνουν καλά αποτελέσματα λόγω της έκτασής της.

Ακόμη, παρατηρώντας τους πίνακες των αποτελεσμάτων, βλέπουμε ότι οι αλγόριθμοι μηχανικής μάθησης (SVM, NB, RF και KNN) φαίνεται να φέρνουν καλύτερα αποτελέσματα τις περισσότερες φορές από τους αλγόριθμους βαθιάς μάθησης. Έτσι οδηγούμαστε στο συμπέρασμα ότι, όταν έχουμε μικρά σύνολα δεδομένων, τα μοντέλα βαθιάς μάθησης δεν μπορούν να μάθουν εύκολά τα σημασιολογικά μοτίβα των κειμένων, καθώς όπως βλέπουμε και από τους πίνακες εκπαίδευσης των μοντέλων βαθιάς μάθησης του κεφαλαίου 5.2.2 λόγο της μικρής έκτασης των συνόλων δεδομένων, φαίνεται να γίνεται ένα μικρό overffiting των δεδομένων. Αυτό παρατηρείται κυρίως στην περίπτωση του συνόλου, με τα 200 tweets, όπου τα χαμηλά ποσοστά σφάλματος σε συνάρτηση των υψηλών ποσοστών ακρίβειας στα δεμένα εκπαίδευσης, υποδηλώνουν overffiting των δεδομένων. Ωστόσο για τα μοντέλα μηχανικής μάθησης και ιδιαίτερα για το μοντέλο SVM, συμπεραίνουμε ότι σε τέτοιες περιπτώσεις καταφέρνουν να φέρουν καλύτερα αποτελέσματα.

Τέλος το μοντέλο BERT που εκπαιδεύτηκε, πάνω στα δεδομένα των συνόλων των αγγλικών tweets φαίνεται να μην φέρνει υψηλά ποσοστά ακρίβειας, σε αντίθεση των αποτελεσμάτων από το ελληνικό σύνολο δεδομένων παρόλο της μικρής σε έκταση σύνολο. Η ειδοποιός διαφορά των αγγλικών και ελληνικών κειμένων είναι ότι στην περίπτωση του ελληνικού συνόλου δεδομένων, έχουμε κείμενα μήκους 800 λέξεων ενώ στην περίπτωση των αγγλικών κειμένων έχουμε κείμενα 20-30 λέξεων. Έτσι οδηγούμαστε στο συμπέρασμα ότι το μοντέλο BERT δεν χρειάζεται απαραίτητα μεγάλα σύνολα δεδομένων αλλά κείμενα που να έχουν μεγαλύτερη έκταση άνω των 30 λέξεων.

Μελλοντικά προτείνεται η δημιουργία ενός ελληνικού συνόλου δεδομένων ανάλογης έκτασης με της PHEME (1438) και με μια ανάλογη ομοιομορφία δεδομένων ώστε να μπορέσουμε να συγκρίνουμε τα μοντέλα που αναπτύχθηκαν σε αυτή την εργασία, σε επαρκή ελληνικά δεδομένα. Επίσης για μελλοντική έρευνα προτείνεται η σύγκριση του προεκπαιδευμένου λεξικού

μεθοδολογίας GloVe με αυτό της εργασίας, ώστε να μελετηθεί αν υπάρχουν αλλαγές στα αποτελέσματα των μοντέλων λόγο των προεκπαιδευμένων λεξικών.

7. Bibliography

[1] X. Zhou and R. Zafarani, "A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities," 2020. Accessed: Dec. 09, 2020. [Online]. Available: https://www.researchgate.net/profile/Xinyi_Zhou26/publication/342762209_A_Survey_of_Fake_News_Fundamental_Theories_Detection_Methods_and_Opportunities/links/5f1644f9a6fdcc3ed71b23cd/A-Survey-of-Fake-News-Fundamental-Theories-Detection-Methods-and-Opportun.

[2] R. Fletcher and R. K. Nielsen, "Are people incidentally exposed to news on social media? A comparative analysis," *New Media Soc.*, vol. 20, no. 7, pp. 2450–2468, Jul. 2018, doi: 10.1177/1461444817724170.

[3] A. Bondielli and F. Marcelloni, "A survey on fake news and rumour detection techniques," *Inf. Sci. (Ny).*, vol. 497, pp. 38–55, Sep. 2019, doi: 10.1016/j.ins.2019.05.035.

[4] B. Thornton, "The Moon Hoax: Debates About Ethics in 1835 New York Newspapers," *J. Mass Media Ethics*, vol. 15, no. 2, pp. 89–100, 2000, doi: 10.1207/s15327728jmme1502_3.

[5] X. Zhou, R. Zafarani, K. Shu, and H. Liu, "Fake News: Fundamental theories, detection strategies and challenges," in *WSDM 2019 - Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, 2019, pp. 836–837, doi: 10.1145/3289600.3291382.

[6] N. Todorova and A. M. Mills, "WHY DO PEOPLE SHARE FAKE NEWS? A SOCIOTECHNICAL MODEL OF MEDIA EFFECTS," *Int. J. Knowl. Manag.*, vol. 14, no. 3, pp. 1–20, 2018, doi: 10.4018/ijkm.2018070101.

[7] E. C. Tandoc, Z. W. Lim, and R. Ling, "Defining 'Fake News': A typology of scholarly definitions," *Digit. Journal.*, vol. 6, no. 2, pp. 137–153, 2018, doi: 10.1080/21670811.2017.1360143.

[8] R. Oshikawa, J. Qian, and W. Y. Wang, "A Survey on Natural Language Processing for Fake News Detection," 2020. Accessed: Dec. 13, 2020. [Online]. Available: https://arxiv.org/abs/1811.00770.

[9] S. Memon, G. A. Mallah, K. N. Memon, A. Shaikh, S. K. Aasoori, and F. U. H. Dehraj, "Comparative study of truncating and statistical stemming algorithms," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 2, pp. 563–568, 2020, doi: 10.14569/ijacsa.2020.0110272.

[10] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Inf.*, vol. 10, no. 4, pp. 1–68, 2019, doi: 10.3390/info10040150.

[11] Stanford University, "Stemming and lemmatization," *Cambridge University Press*, 2009. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (accessed May 11, 2021).

[12] J. Pollock, E. Waller, and R. Politt, "Speech and language processing," *Day-to-Day Dyslexia Classr.*, pp. 16–28, 2010, doi: 10.4324/9780203461891_chapter_3.

[13] V. Mohan, "Preprocessing Techniques for Text Mining-An Overview Privacy Preserving

Data Mining View project," 2015. Accessed: May 11, 2021. [Online]. Available: https://www.researchgate.net/publication/339529230.

[14] "Introduction to Word Embedding and Word2Vec | by Dhruvil Karani | Towards Data Science." https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa (accessed May 17, 2021).

[15] M. Naili, A. H. Chaibi, and H. H. Ben Ghezala, "Comparative study of word embedding methods in topic segmentation," in *Procedia Computer Science*, Jan. 2017, vol. 112, pp. 340–349, doi: 10.1016/j.procs.2017.08.009.

[16] "Topic Modeling using Latent Semantic Analysis." https://iq.opengenus.org/topic-modeling-lsa/ (accessed Dec. 09, 2021).

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space." Accessed: May 18, 2021. [Online]. Available: http://ronan.collobert.com/senna/.

[18] J. Lilleberg, Y. Zhu, and Y. Zhang, "Support vector machines and Word2vec for text classification with semantic features," in *Proceedings of 2015 IEEE 14th International Conference on Cognitive Informatics and Cognitive Computing, ICCI*CC 2015*, Sep. 2015, pp. 136–140, doi: 10.1109/ICCI-CC.2015.7259377.

[19] Z. Li, "A Beginner's Guide to Word Embedding with Gensim Word2Vec Model," 2019. Accessed: Dec. 09, 2021. [Online]. Available: https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92.

[20] D. Kimothi, P. Biyani, J. M. Hogan, A. Soni, and W. Kelly, "Learning supervised embeddings for large scale sequence comparisons," *PLoS One*, vol. 15, no. 3, p. e0216636, Mar. 2020, doi: 10.1371/journal.pone.0216636.

[21] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation." Accessed: May 18, 2021. [Online]. Available: http://nlp.

[22] bin wang, angela wang, fenxiao chen, yuncheng wang, c-c jay kuo, and B. Wang, "Evaluating word embedding models: methods and experimental results," *SIP*, vol. 8, pp. 1–14, 2019, doi: 10.1017/ATSIP.2019.12.

[23] C. Cortes, V. Vapnik, and L. Saitta, "Support-Vector Networks Editor," Kluwer Academic Publishers, 1995.

[24] M. J. Er, R. Venkatesan, and W. Ning, "An online universal classifier for binary, multi-class and multi-label classification," in *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, 2017, pp. 3701–3706, doi: 10.1109/SMC.2016.7844809.

[25] L. Schultebraucks, "Introduction to Support Vector Machines," *Medium*. https://medium.com/@LSchultebraucks/introduction-to-support-vector-machines-9f8161ae2fcb (accessed Dec. 09, 2021).

[26] V. Malhotra, "ML06: Intro to Multi-class Classification," *Medium*. https://medium.com/analytics-vidhya/ml06-intro-to-multi-class-classification-e61eb7492ffd (accessed Dec. 09, 2021).

[27] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends,"

Neurocomputing, vol. 408, pp. 189–215, Sep. 2020, doi: 10.1016/j.neucom.2019.10.118.

[28] S. Fatima, "Text Document categorization using support vector machine," *Int. Res. J. Eng. Technol.*, vol. 4, no. 2, pp. 141–147, 2017, Accessed: Oct. 14, 2021. [Online]. Available: www.irjet.net.

[29] J. Xu, X. Liu, Z. Huo, C. Deng, F. Nie, and H. Huang, "Multi-Class Support Vector Machine via Maximizing Multi-Class Margins," 2017.

[30] C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multiclass Support Vector Machines," 2002. Accessed: Dec. 28, 2020. [Online]. Available: https://mycourses.aalto.fi/pluginfile.php/508959/mod_resource/content/1/multiclass-svm.pdf.

[31] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, "An up-to-date comparison of state-of-the-art classification algorithms," *Expert Syst. Appl.*, vol. 82, pp. 128–150, Oct. 2017, doi: 10.1016/j.eswa.2017.04.003.

[32] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[33] Y. Al Amrani, M. Lazaar, and K. E. El Kadirp, "Random forest and support vector machine based hybrid approach to sentiment analysis," in *Procedia Computer Science*, Jan. 2018, vol. 127, pp. 511–520, doi: 10.1016/j.procs.2018.01.150.

[34] I-king-of-ml, "Random Forest and how it works," *Medium*, 2019. https://medium.com/@rdhawan201455/random-forest-and-how-it-works-67f408e43a43 (accessed Dec. 10, 2021).

[35] V. Bijalwan, V. Kumar, P. Kumari, and J. Pascual, "KNN based machine learning approach for text and document mining," *Int. J. Database Theory Appl.*, vol. 7, no. 1, pp. 61–70, 2014, doi: 10.14257/ijdta.2014.7.1.06.

[36] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text Classification Algorithms: A Survey," doi: 10.3390/info10040150.

[37] S. Jiang, G. Pang, M. Wu, and L. Kuang, "An improved K-nearest-neighbor algorithm for text categorization," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 1503–1509, Jan. 2012, doi: 10.1016/j.eswa.2011.08.040.

[38] "KNN Classification using Sklearn Python - DataCamp." https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn (accessed Dec. 10, 2021).

[39] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing," *J. Artif. Intell. Res.*, vol. 57, pp. 345–420, Nov. 2016, doi: 10.1613/jair.4992.

[40] "Understanding Feedforward Neural Networks | LearnOpenCV." https://learnopencv.com/understanding-feedforward-neural-networks/ (accessed Dec. 10, 2021).

[41] Q. Nguyen, M. C. Mukkamala, and M. Hein, "Neural Networks Should Be Wide Enough to Learn Disconnected Decision Regions," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 9, pp. 6001–6011, Feb. 2018, Accessed: Nov. 14, 2021. [Online]. Available: http://arxiv.org/abs/1803.00094.

[42] "Single-layer Neural Networks in Machine Learning (Perceptrons) - DEV Community." https://dev.to/codeperfectplus/single-layer-neural-networks-in-machine-learning-perceptrons-18n8 (accessed Dec. 10, 2021).

[43] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, and G. Petrova, "Nonlinear Approximation and (Deep) ReLU Networks," *Constr. Approx.*, Apr. 2021, doi: 10.1007/s00365-021-09548-z.

[44] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," Mar. 2018, Accessed: Nov. 11, 2021. [Online]. Available: https://arxiv.org/abs/1803.01271v2.

[45] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," Nov. 2015, Accessed: Dec. 10, 2021. [Online]. Available: https://arxiv.org/abs/1511.08458v2.

[46] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020, doi: 10.1007/s10462-020-09825-6.

[47] A. Jacovi, O. Sar Shalom, and Y. Goldberg, "Understanding Convolutional Neural Networks for Text Classification," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Sep. 2018, pp. 56–65, doi: 10.18653/v1/W18-5408.

[48] J. Browniee, "Best Practices for Text Classification with Deep Learning," 2020. https://machinelearningmastery.com/best-practices-document-classification-deep-learning/ (accessed Dec. 10, 2021).

[49] Y. Zhang and B. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," pp. 253–263, Oct. 2015, Accessed: Nov. 17, 2021. [Online]. Available: http://arxiv.org/abs/1510.03820.

[50] L. Qing, W. Linhong, and D. Xuehai, "A Novel Neural Network-Based Method for Medical Text Classification," *Futur. Internet*, vol. 11, no. 12, p. 255, Dec. 2019, doi: 10.3390/fi11120255.

[51] "CS 230 - Recurrent Neural Networks Cheatsheet." https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks (accessed Nov. 21, 2021).

[52] B. Vivek Singh, "BASIC ARCHITECTURE OF RNN AND LSTM," 2017, Accessed: Oct. 10, 2021. [Online]. Available: https://pydeeplearning.weebly.com/blog/basic-architecture-of-rnn-and-lstm.

[53] Saurabh Rathor, "Simple RNN vs GRU vs LSTM :- Difference lies in More Flexile control ," *Medium*. https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57 (accessed Dec. 10, 2021).

[54] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks," Sep. 2019, Accessed: Nov. 22, 2021. [Online]. Available: http://arxiv.org/abs/1909.09586.

[55] E. López, C. Valle, H. Allende, E. Gil, and H. Madsen, "Wind Power Forecasting Based on Echo State Networks and Long Short-Term Memory," *Energies*, vol. 11, no. 3, p. 526, Feb. 2018, doi: 10.3390/en11030526.

[56] “Constant Error Carousel Definition,” *DeepAI*. https://deepai.org/machine-learning-glossary-and-terms/constant error carousel (accessed Dec. 10, 2021).

[57] C. Tallec and Y. Ollivier, “Unbiasing Truncated Backpropagation Through Time,” 2017.

[58] S. M. Lakew, M. Cettolo, and M. Federico, “A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation,” *COLING 2018 - 27th Int. Conf. Comput. Linguist. Proc.*, pp. 641–652, Jun. 2018, Accessed: Feb. 06, 2022. [Online]. Available: https://arxiv.org/abs/1806.06957v2.

[59] A. Vaswani *et al.*, “Attention is All you Need,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, Accessed: Feb. 06, 2022. [Online]. Available: https://arxiv.org/abs/1806.06957v2.

[60] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, pp. 4171–4186, Oct. 2018, Accessed: Nov. 11, 2021. [Online]. Available: https://arxiv.org/abs/1810.04805v2.

[61] K. Elena, L. Maria, and Z. Arkaitz, “PHEME dataset for Rumour Detection and Veracity Classification,” 2018. https://figshare.com/articles/dataset/PHEME_dataset_for_Rumour_Detection_and_Veracity_Classification/6392078 (accessed Jan. 04, 2022).

[62] derevirn, “Greek Fake News Detector,” *GitHub*. https://github.com/derevirn/gfn-detector (accessed Jan. 05, 2022).

[63] Giannis Tolios, “How I Created a Fake News Detector with Python ,” *Towards Data Science*, Apr. 14, 2021. https://towardsdatascience.com/how-i-created-a-fake-news-detector-with-python-65b1234123c4 (accessed Feb. 06, 2022).

[64] “fastText.” https://fasttext.cc/ (accessed Jan. 05, 2022).

[65] “What is an N-gram Multichannel Convolutional Neural Network for Text Classification.” https://gdcoder.com/what-is-an-n-gram-multichannel-convolutional-neural-network-for-text-classification/ (accessed Jan. 06, 2022).

[66] “Fake News Detection with Machine Learning, using Python | by Piero Paialunga | Towards Data Science.” https://towardsdatascience.com/fake-news-detection-with-machine-learning-using-python-3347d9899ad1 (accessed Jan. 06, 2022).