ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Αυτόματη αναγνώριση ρητορικής μίσους σε κείμενο με τεχνικές μηχανικής μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΑΝΑΣΤΑΣΙΑΣ ΣΚΛΑΒΟΥ

Επιβλέπων : Κωνσταντίνος Ι. Διαμαντάρας
Καθηγητής

Θεσσαλονίκη, Ιούνιος 2021

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Αυτόματη αναγνώριση ρητορικής μίσους σε κείμενο με τεχνικές μηχανικής μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΑΝΑΣΤΑΣΙΑΣ ΣΚΛΑΒΟΥ

Επιβλέπων : Κωνσταντίνος Ι. Διαμαντάρας
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΘΕΣΣΑΛΟΝΙΚΗΣ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Ιουλίου 2021.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Όνομα Επώνυμο	Όνομα Επώνυμο	'Ονομα Επώνυμο
Καθηγητής	Καθηγητής	Καθηγητής

Θεσσαλονίκη, Ιούνιος 2021

(Υπογραφή)

.....

ΑΝΑΣΤΑΣΙΑ ΣΚΛΑΒΟΥ

Αυτόματη αναγνώριση ρητορικής μίσους σε κείμενο με τεχνικές μηχανικής μάθησης

Περίληψη

Η ρητορική μίσους είναι λόγος που εκφράζει μίσος απέναντι σε συγκεκριμένες ομάδες, βάσει χαρακτηριστικών όπως η φυλή, η θρησκεία, το φύλο, ο σεξουαλικός προσανατολισμός ή η αναπηρία. Διάφορες πλατφόρμες, όπως Google, Twitter, Facebook, Microsoft, YouTube αναγνωρίζοντας ότι η διάδοση ρητορικής μίσους στο διαδίκτυο έχει αρνητικές συνέπειες, έχουν δεσμευτεί να εξετάζουν την πλειονότητα των έγκυρων ειδοποιήσεων για την απόσυρση παράνομης ρητορικής μίσους σε λιγότερο από 24 ώρες και, εφόσον απαιτείται, να αποσύρουν το εν λόγω περιεχόμενο ή να απενεργοποιούν την πρόσβαση σ' αυτό. Ο τεράστιος, όμως, όγκος των δεδομένων που παράγονται στις παραπάνω πλατφόρμες καθιστά αδύνατο τον εντοπισμό αναρτήσεων από διαχειριστές περιεχομένου ή από αναφορές χρηστών, κάνοντας αναγκαία τη χρήση αυτόματων εργαλείων εντοπισμού ρητορικής μίσους.

Λαμβάνοντας υπόψη τα παραπάνω, δημιουργήσαμε μοντέλα με σκοπό την επιτυχή κατάταξη των δεδομένων εισόδου σε τρεις κατηγορίες "Hate/Hateful","Offensive/Abusive" και "Neither/Normal", ανάλογα με το αν περιέχουν ρητορική μίσους, προσβλητική γλώσσα ή τίποτα από τα δύο, αναδεικνύοντας, παράλληλα την μέθοδο με την καλύτερη απόδοση. Επίσης, εξετάσαμε κατά πόσο τα μοντέλα που εκπαιδεύσαμε γενικεύουν σε διαφορετικά σύνολα δεδομένων.

Για τον σκοπό αυτό, μελετήσαμε τρεις διαφορετικές ενσωματώσεις λέξεων (GloVe, fastText, Bert) σε συνδυασμό με πληθώρα αλγορίθμων ταξινόμησης (GRU,LSTM,CNN). Τα πειράματά μας έδειξαν ότι η πιο αποδοτική μέθοδος είναι αυτή που χρησιμοποιεί ενσωματώσεις λέξεων Bert, ενώ ακολουθούν οι ενσωματώσεις λέξεων fastText και GloVe.

Λέξεις Κλειδιά: Ρητορική μίσους, Επεξεργασία φυσικής γλώσσας, GloVe, fastText, Bert, CNN, LSTM,GRU, ενσωματώσεις λέξεων.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

Hate speech is a speech that expresses hatred towards certain groups, based on characteristics such as race, religion, gender, sexual orientation or disability. Various platforms, such as Google, Twitter, Facebook, Microsoft, YouTube, recognizing that the spread of hate speech on the Internet has negative repercussions, have pledged to consider the majority of valid notices for the withdrawal of illegal hate speech in less than 24 hours and, if required, withdraw such content or disable access to it. However, the sheer volume of data generated on these platforms makes it impossible for content managers or user reports to detect such posts, necessitating the use of automated hate speech detection tools.

In view of the above, we created models to successfully classify the input data into three categories "Hate/Hateful", "Offensive/Abusive" and "Neither/Normal", depending on whether they contain hate speech, offensive language or neither, while highlighting the method with the better performance. We also investigated whether the models we trained generalize between different datasets.

For this purpose, we studied three different word embeddings (GloVe, fastText, Bert) in combination with a variety of classification algorithms (GRU, LSTM, CNN).. Our experiments have shown that the most efficient method is the one that uses Bert word embeddings, followed by fastText and GloVe word embeddings.

Keywords: Hate speech, Natural language processing, CNN, LSTM,GRU, GloVe, fastText, Bert, word embeddings

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1 *Εισαγωγή*

Στις πλατφόρμες Κοινωνικής Δικτύωσης όπως Twitter, Facebook κα., χρήστες από κάθε υπόβαθρο, φυλή, θρησκεία και εθνικότητα αλληλεπιδρούν, επικοινωνούν, και μοιράζονται ελεύθερα τις ιδέες, τις απόψεις και τις πεποιθήσεις τους. Από τη μία πλευρά, το παραπάνω έχει ως αποτέλεσμα την ανταλλαγή ιδεών και απόψεων ανθρώπων διαφορετικών ομάδων, ενώ από την άλλη ευνοείται η εξάπλωση και η υποκίνηση του μίσους, της βίας και των διακρίσεων κατά χρηστών ή ομάδων χρηστών με βάση το φύλο, τη θρησκεία, την φυλή, τον σεξουαλικό προσανατολισμό και την αναπηρία. Οι εταιρείες κοινωνικής δικτύωσης παρακινούν τους χρήστες τους να αναφέρουν ανάρμοστες δημοσιεύσεις, ώστε είτε αυτές να διαγράφονται είτε να γίνεται αναστολή λογαριασμού του χρήστη που δημοσίευσε την συγκεκριμένη δημοσίευση. Παρ' όλη την προσπάθεια των εταιρειών να περιορίσουν αυτά τα φαινόμενα η διαδικασία ελέγχου τέτοιων αναφορών όντας χειροκίνητη, μπορεί να οδηγήσει σε κατάχρηση αυτής της διαδικασίας σωπαίνοντας μειονότητες, θρησκευτικές πεποιθήσεις, ή καταστέλλοντας κριτικές σε επίσημες πολιτικές. Οι ανάρμοστες δημοσιεύσεις, που ονομάζονται και Ρητορική μίσους, μπορούν να οδηγήσουν ακόμη και σε σοβαρή σωματική ή ψυχική κακοποίηση και είναι επιτακτική ανάγκη να εντοπίζονται σωστά και να αποτρέπονται άμεσα.

1.1 Αντικείμενο διπλωματικής

Σε αυτή την εργασία, στοχεύουμε να μελετήσουμε μεθόδους ταξινόμησης για την ανίχνευση ρητορικής μίσους στο Twitter χρησιμοποιώντας διάφορα μοντέλα μηχανικής μάθησης όπως Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU) Convolutional Neural Networks (CNN), σε συνδυασμό με διαφορετικούς τύπους προ-εκπαιδευμένων ενσωματώσεων λέξεων όπως GloVe, fastText και Bert. Στοχεύουμε στο να κατηγοριοποιήσουμε, σωστά, τα δεδομένα εισόδου μας σε τρεις κατηγορίες. Αν τα δεδομένα περιέχουν ρητορική μίσους, θα πρέπει να κατηγοριοποιούνται ως "Hate/Hateful", αν περιέχουν προσβλητικό κείμενο, θα πρέπει να κατηγοριοποιούνται ως "Offensive/Abusive", ενώ αν δεν περιέχουν τίποτα από τα δύο, θα πρέπει να κατηγοριοποιούνται ως "Neither/Normal".

Για τον σκοπό αυτό, δημιουργήσαμε έναν αλγόριθμο προ-επεξεργασίας των δεδομένων εισόδου, ώστε να διευκολύνουμε όσο το δυνατόν περισσότερο την ταξινόμησή τους. Ενώ στη

συνέχεια πειραματιστήκαμε με πληθώρα αλγορίθμων ταξινόμησης όπως, Bidirectional Long Short-Term Memory (BLSTM) και Long Short-Term Memory (LSTM), Bidirectional Gated Recurrent Neural Network (BGRU) και Gated Recurrent Neural Network (GRU), καθώς και 2-D Convolutional Neural Networks σε συνδυασμό με προ-επεξεργασμένες ενσωματώσεις λέξεων (Embeddings) GloVe, fastText και Bert.

Τέλος, στοχεύουμε να εξετάσουμε κατά πόσο τα μοντέλα που εκπαιδεύσαμε να αναγνωρίζουν την ρητορική μίσους, γενικεύουν μεταξύ διαφορετικών συνόλων δεδομένων. Για τον σκοπό αυτό χρησιμοποιήσαμε δύο διαφορετικά σύνολα δεδομένων, τα οποία περιγράφονται αναλυτικά στην Ενότητα 4.1. Χρησιμοποιήσαμε, εναλλάξ, το ένα σύνολο δεδομένων για την εκπαίδευση του μοντέλου και το άλλο για την πρόβλεψη των δεδομένων και συγκρίναμε τα αποτελέσματα.

1.1.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήσαμε διάφορα συστήματα αυτόματου εντοπισμού ρητορικής μίσους.
2. Υλοποιήσαμε δύο αλγορίθμους για την προ-επεξεργασία του συνόλου δεδομένων μας. Έναν που χρησιμοποιήθηκε στις προ-εκπαιδευμένες ενσωματώσεις GloVe και fastText και έναν για τις ενσωματώσεις Bert.
3. Χρησιμοποιήσαμε προ-εκπαιδευμένες ενσωματώσεις λέξεων GloVe, fastText και Bert σε συνδυασμό με διάφορους αλγορίθμους CNN, LSTM και GRU. Αξιολογώντας την επίδοση των παραπάνω αλγορίθμων διαπιστώσαμε ότι οι ενσωματώσεις Bert έχουν την καλύτερη απόδοση, με τις ενσωματώσεις fastText να ακολουθούν και τις ενσωματώσεις GloVe να έχουν τη χειρότερη απόδοση. Επίσης, καταλήξαμε ότι οι αλγόριθμοι αποδίδουν διαφορετικά σε κάθε σύνολο δεδομένων, όσον αφορά τις ενσωματώσεις fastText και GloVe σε αντίθεση με το Bert όπου το αποδοτικότερο μοντέλο είναι κοινό.
4. Τέλος, χρησιμοποιώντας το επικρατέστερο σε απόδοση μοντέλο, εξετάσαμε κατά πόσο ένα μοντέλο που εκπαιδεύεται σε ένα σύνολο δεδομένων, μπορεί να παρουσιάσει παρόμοια απόδοση όταν προσπαθεί να προβλέψει τα δεδομένα ενός άλλου συνόλου δεδομένων. Καταλήξαμε στο συμπέρασμα ότι τα μοντέλα μας, σε αυτές τις περιπτώσεις, αποδίδουν λιγότερο πάνω 50%.

1.2 Οργάνωση κειμένου

Η εργασία μας είναι διαρθρωμένη ως εξής:

Εργασίες σχετικές με το αντικείμενο της διπλωματικής παρουσιάζονται στο Κεφάλαιο 2 . Η ανάλυση του θεωρητικού υπόβαθρου της εργασίας μας, όπως ο ορισμός της ρητορικής μίσους καθώς και η περιγραφή των αλγορίθμων ταξινόμησης δίνονται στο Κεφάλαιο 3. Στο Κεφάλαιο 4 αναλύουμε τα σύνολα δεδομένων που χρησιμοποιήσαμε ως είσοδο στα πειράματά μας και στο Κεφάλαιο 5 περιγράφουμε εν συντομία τα καλύτερα σε απόδοση πειράματα που εκτελέσαμε. Η απόδοση των πειραμάτων μας παρουσιάζεται στο Κεφάλαιο 6 ενώ στο Κεφάλαιο 7 αναλύονται οι αποδοτικότεροι αλγόριθμοι, και περιγράφονται οι πλατφόρμες και τα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν. Τέλος, η σύνοψη και τα συμπεράσματα της εργασίας μας παρουσιάζονται στο Κεφάλαιο 8.

2 *Σχετικές εργασίες*

Υπάρχει πληθώρα μελετών πάνω στον αυτόματο εντοπισμό ρητορικής μίσους στο διαδίκτυο και συγκεκριμένα στο Twitter. Σκεφτήκαμε να κατηγοριοποιήσουμε τις μελέτες αυτές σύμφωνα με το είδος της προσέγγισης που χρησιμοποιούν.

Έτσι, χωρίσαμε τις έρευνες σε δύο κατηγορίες. Αυτές που χρησιμοποίησαν προσεγγίσεις κλασσικής μηχανικής μάθησης (machine learning) και αυτές που χρησιμοποίησαν προσεγγίσεις βαθέως νευρωνικού δικτύου (deep neural network)

2.1 Προσέγγιση μηχανικής μάθησης

Υπάρχουν πολλές μελέτες που αξιοποιούν γλωσσικές αναπαραστάσεις, όπως ο σάκος λέξεων (Bag Of Words, BOW), συμπλέγματα λέξεων (n-grams) και συχνότητα όρου - αντίστροφη συχνότητα εγγράφου (Term Frequency–Inverse Document Frequency,TFIDF).

Οι (Davidson et al.,2017) [1] δημιούργησαν ένα σύνολο δεδομένων με 24K Tweets τα οποία σχολίασαν ως 'Hate', 'Offensive', 'Neither'. Ανέπτυξαν μια ποικιλία ταξινομητών πολλαπλών κατηγοριών (multi-class classifiers) όπως Logistic Regression, Naive Bayes, Decision Trees, Random Forests, και Linear SVMs. Χρησιμοποίησαν ένα σύνολο χαρακτηριστικών όπως Term Frequency – Inverse Document Frequency (TF-IDF) σταθμισμένα n-gram, καθώς και ορισμένα μεταδεδομένα σε επίπεδο tweet, όπως ο αριθμός των hashtag, retweets, URL κ.λπ. Η ενσωμάτωση των παραπάνω γλωσσικών χαρακτηριστικών βοηθά στον εντοπισμό της ρητορικής μίσους, αποδίδοντας στο μοντέλο πολύ καλή απόδοση, αλλά πάσχει από λεπτές διακρίσεις, όπως όταν χρησιμοποιούνται επιθετικοί όροι με θετική έννοια. Σε επόμενη μελέτη οι (Alshalan και Al-Khalifa, 2018)[2] χρησιμοποίησαν n-gram σταθμισμένα με τιμές TFIDF . Ο στόχος της χρήσης του TFIDF είναι η μείωση της επίδρασης των σημείων που μας δίνουν τις λιγότερες πληροφορίες λόγω της συχνής εμφάνισής τους στη συλλογή. Γίνεται χρήση των μοντέλων Logistic Regression, Naive Bayes και Support Vector Machines. Τα αποτελέσματα έδειξαν ότι η Logistic Regression δίνει καλύτερα αποτελέσματα. Τέλος οι (MacAvaney et al., 2019)[3] προτείνουν μια προσέγγιση βασισμένη σε ένα μοντέλο multi-view Support Vector Machine (mSVM) κάνοντας χρήση n-grams λέξεων και χαρακτήρων έως 5-grams ως διανύσματα χαρακτηριστικών. Συγκρίνουν το μοντέλο τους με διάφορα μοντέλα όπως BERT, Naive Bayes, SVM, Logistic Regression, fastText, Convolutional Gated Recurrent Unit (C-

GRU) σε διάφορα σύνολα δεδομένων όπως Stormfront, HatEval, TRAC(Facebook), HatebaseTwitter. Τα αποτελέσματα έδειξαν ότι τα επικρατέστερα μοντέλα είναι το Multi-view SVM και το BERT.

2.2 Προσέγγιση βαθέως νευρωνικού δικτύου

Οι περισσότερες μελέτες που ανήκουν σε αυτή την κατηγορία, χρησιμοποίησαν προ-εκπαιδευμένες ενσωματώσεις λέξεων Word2Vec[4], fastText[5] και GloVe[6]. Οι (Badjatiya et al., 2018)[7] πειραματίστηκαν με το σύνολο δεδομένων που παρέχουν οι Waseem και Hovy διερεύνησαν τρείς αρχιτεκτονικές βαθιάς μάθησης όπως fastText, CNN και LSTM. Συμπέραναν ότι οι ενσωματώσεις λέξεων που αρχικοποιήθηκαν σε τυχαία διανύσματα αποδίδουν ελαφρώς καλύτερα από τις ενσωματώσεις GloVe όταν αυτές συνδυάζονται με Gradient boosted decision tree (GBDT). Οι (Abuzayed και Elsayed, 2020)[8] πραγματοποίησαν μια προκαταρκτική μελέτη με την οποία διερεύνησαν την απόδοση 15 διαφορετικών μοντέλων κλασσικών και βαθέων νευρωνικών δικτύων για τον εντοπισμό ρητορικής μίσους σε αραβικά tweets. Στη συνέχεια έδειξαν μια απλή και γρήγορη προσέγγιση που εφαρμόζεται σε λιγότερο από τρεις ημέρες, η οποία όμως πέτυχε λογικές επιδόσεις. Η (Biere, 2018)[9] υλοποίησε ένα απλό μοντέλο CNN το οποίο παρά την πολύ καλή του ακρίβεια (accuracy), ταξινόμησε εσφαλμένα ρητορική χωρίς μίσος ως ρητορική μίσους ενώ ταξινόμησε σωστά την προσβλητική γλώσσα. Αυτό φαίνεται να οφείλεται στο ότι το σύνολο δεδομένων περιέχει στην πλειοψηφία του προσβλητική γλώσσα, οδηγώντας το μοντέλο να είναι προκατειλημμένο (biased), στην ταξινόμηση των tweets, ως προσβλητικών. Λόγω της δυσκολίας των προ-εκπαιδευμένων ενσωματώσεων Word2Vec, GloVe και fastText να συλλάβουν τη σημασιολογία του κειμένου, οι (Mozafari και Farahbakhsh, 2020)[10] προτείνουν την χρήση προ-εκπαιδευμένων γλωσσικών μοντέλων Bert μια τεχνική που ξεπερνά σε απόδοση τα άλλα μοντέλα. Επίσης οι (Mutanga et al., 2020)[12] προτείνουν ένα πιο ελαφρύ μοντέλο Bert από το οποίο έχουν αφαιρέσει ενσωματώσεις τύπου token και pooler, και ταυτόχρονα μείωσαν τον αριθμό των επιπέδων στο μισό μειώνοντας, έτσι, σημαντικά το αποτύπωμα του μοντέλου. Τέλος, οι (Swamy et al., 2019)[13] στην προσπάθειά τους να μελετήσουν αν, τα μοντέλα που παρουσιάζουν λογικά αποτελέσματα εκπαιδευόμενα σε ένα σύνολο δεδομένων μπορούν να παρουσιάζουν συγκριτικά καλά αποτελέσματα και σε άλλα σύνολα δεδομένων. Για τον λόγο αυτό υλοποίησαν μοντέλα LSVM, LSTM, ELMo και Bert, με το Bert να έχει τα καλύτερα αποτελέσματα.

3 Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο, παρουσιάζουμε τις έννοιες της Ελευθερίας λόγου, της Ελευθερίας λόγου στο διαδίκτυο ενώ στη συνέχεια ορίζουμε την ρητορική μίσους και παραθέτουμε δυσκολίες εντοπισμού της στο διαδίκτυο. Παρακάτω παραθέτουμε πληροφορίες για τα διάφορα μοντέλα στα οποία βασιστήκαμε στην υλοποίηση των δικών μας πειραμάτων.

3.1 Ελευθερία λόγου και ρητορική μίσους

3.1.1 Ελευθερία Λόγου

Η ελευθερία της έκφρασης αποτελεί ακρογωνιαίο λίθο της δημοκρατίας. Κάθε πρόσωπο ή κοινότητα έχει το δικαίωμα της ελευθερίας της γνώμης χωρίς φόβο αντίρρησης, λογοκρισίας ή νομικής κύρωσης, ανεξαρτήτως συνόρων.

Το δικαίωμα στην ελευθερία έκφρασης προστατεύεται από το άρθρο 19 της Οικουμενικής Διακήρυξης των Ανθρωπίνων Δικαιωμάτων[14], ενώ σε Ευρωπαϊκό επίπεδο καθορίζεται από το άρθρο 10 της Ευρωπαϊκής Σύμβασης των Δικαιωμάτων του Ανθρώπου.[15]

Ωστόσο η άσκηση του δικαιώματος της ελευθερίας της έκφρασης «εμπεριέχει καθήκοντα και υποχρεώσεις» και γι΄ αυτό μπορεί, όταν χρειάζεται, να υπόκειται σε ορισμένους περιορισμούς ώστε να προασπίζεται τα δικαιώματα ή τη φήμη των άλλων ή να προστατεύει την εθνική ασφάλεια, τη δημόσια τάξη, τη δημόσια υγεία ή την ηθική.

3.1.2 Ελευθερία Λόγου στο Διαδίκτυο

Η ανάπτυξη του διαδικτύου είχε ριζικές επιπτώσεις στην ανθρώπινη επικοινωνία, παρέχοντας πρόσβαση σε ειδήσεις, πληροφορίες και ιδέες χωρίς σύνορα. Με βάση το άρθρο 19 της Οικουμενικής Διακήρυξης των Ανθρωπίνων Δικαιωμάτων και το άρθρο 10 της Ευρωπαϊκής Σύμβασης των Δικαιωμάτων του Ανθρώπου, η διαδικτυακή έκφραση, απολαμβάνει την ίδια προστασία με τον προφορικό ή γραπτό λόγο.

Η πλειοψηφία των κρατών μελών της Ευρωπαϊκής Ένωσης τείνει να εφαρμόζει γενικούς νομικούς κανόνες και στην ηλεκτρονική έκφραση, ενώ ορισμένες χώρες έχουν θεσπίσει νομοθεσία για το διαδίκτυο που ρυθμίζει πτυχές, όπως η αποτροπή διάδοσης ρητορικής μίσους.

3.1.3 Ρητορική Μίσους στο Διαδίκτυο

Η ρητορική μίσους δεν έχει επίσημο νομικό ορισμό, αλλά υπάρχει συναίνεση ότι η ρητορική του μίσους είναι λόγος που εκφράζει μίσος σε συγκεκριμένες ομάδες με βάση χαρακτηριστικά όπως φυλή, θρησκεία, φύλο, σεξουαλικό προσανατολισμό ή αναπηρία.

Ένας περιεκτικός ορισμός, όπως ορίζεται από το Συμβούλιο της Ευρώπης, είναι ο παρακάτω: «Η ρητορική μίσους περιλαμβάνει κάθε μορφή έκφρασης που διαδίδει, υποκινεί, προωθεί ή δικαιολογεί το φυλετικό μίσος, την ξενοφοβία, τον αντισημιτισμό ή άλλες μορφές βίας και μίσους που βασίζονται στη μισαλλοδοξία, περιλαμβανομένου της μισαλλοδοξίας που εκφράζεται μέσα από τον επιθετικό εθνικισμό και τον εθνοκεντρισμό, τη διάκριση και το μίσος κατά μειονοτήτων και μεταναστών»[16].

Οι επιχειρήσεις τεχνολογίας πληροφοριών, όπως Google, Twitter, Facebook, Microsoft, YouTube αναγνωρίζουν ότι η διάδοση παράνομης ρητορικής μίσους στο διαδίκτυο έχει αρνητικές συνέπειες. Οι συνέπειες αυτές βαραίνουν όχι μόνο τις ομάδες ή τα πρόσωπα κατά των οποίων στρέφεται, αλλά και όσους υπερασπίζονται ανοικτά την ελευθερία, την ανοχή και την αποφυγή των διακρίσεων στις ανοικτές κοινωνίες, ενώ παράλληλα δημιουργούν κλίμα που δυσχεραίνει τον δημοκρατικό διάλογο στις διαδικτυακές πλατφόρμες. Οι παραπάνω επιχειρήσεις έχουν δεσμευτεί να εξετάζουν την πλειονότητα των έγκυρων ειδοποιήσεων για την απόσυρση παράνομης ρητορικής μίσους σε λιγότερο από 24 ώρες και, εφόσον απαιτείται, να αποσύρουν το εν λόγω περιεχόμενο ή να απενεργοποιούν την πρόσβαση σ' αυτό.

3.1.4 Twitter

Μεταξύ των πολλών υπαρχόντων κοινωνικών δικτύων, το Twitter κατατάσσεται σήμερα ως μία από τις κορυφαίες πλατφόρμες και είναι μια από τις πιο σημαντικές πηγές δεδομένων για ερευνητές. Το Twitter είναι ένα γνωστό δημόσιο δίκτυο μικρο-ιστολογίων σε πραγματικό χρόνο όπου, συχνά, εμφανίζονται νέα ακόμα και πριν από τα επίσημα μέσα ενημέρωσης. Χαρακτηρίζεται από το όριο σύντομων μηνυμάτων (280 χαρακτήρες) και τη μη φιλτραρισμένη ροή. Η χρήση του κλιμακώνεται ταχύτατα, ειδικά εν μέσω γεγονότων, με μέσο όρο δημοσίευσης 500 εκατομμυρίων tweets ανά ημέρα.

3.1.5 Δυσκολία ανίχνευσης ρητορικής μίσους στο διαδίκτυο

Η ρητορική μίσους στην πραγματικότητα είναι ένα γλωσσικό και κοινωνικό φαινόμενο με διάφορους τόνους και μορφές. Πολλές φορές το καταχρηστικό περιεχόμενο ορίζεται σε σχέση με τις προθέσεις των χρηστών, ενώ άλλες φορές οι προθέσεις τους είναι συγκαλυμμένες και δεν διακρίνονται άμεσα. Ένα ακόμα δύσκολο σημείο στον εντοπισμό είναι η πολυσημία

κάποιων λέξεων, όταν δηλαδή, λέξεις με την ίδια ορθογραφία έχουν διαφορετική έννοια ανάλογα με το πλαίσιο της πρότασης στην οποία βρίσκονται [17]. Τέλος, τα σύνολα δεδομένων (datasets) που χρησιμοποιούνται για την εκπαίδευση των συστημάτων εντοπισμού ρητορικής μίσους βασίζονται στις προσωπικές απόψεις ή ακόμα και το φύλο των χρηστών που «χαρακτήρισαν» τα δεδομένα. Επομένως είναι δύσκολο να δημιουργηθεί ένας καθολικός αλγόριθμος μηχανικής μάθησης που θα εντόπιζε τέτοια φαινόμενα.

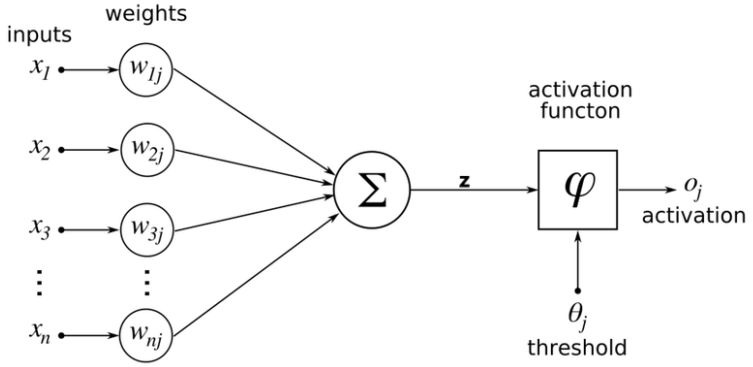
3.2 Ορισμός ταξινόμησης (Classification)

Η ταξινόμηση είναι η διαδικασία πρόβλεψης της κατηγορίας δεδομένων βάσει των δεδομένων εισόδου. Τα δεδομένα εισόδου χωρίζονται σε δύο ή περισσότερες κλάσεις, και η μηχανή πρέπει να κατασκευάσει ένα μοντέλο, το οποίο θα αντιστοιχίζει τα δεδομένα σε μία ή περισσότερες (multi-label ταξινόμηση) κλάσεις. Το παραπάνω εμπίπτει στην επιτηρούμενη μάθηση.

3.3 Τεχνητό Νευρωνικό δίκτυο (Artificial Neural Network)

Τα νευρωνικά δίκτυα είναι μαθηματικά μοντέλα που χρησιμοποιούν αλγόριθμους μάθησης, που μιμούνται τον ανθρώπινο εγκέφαλο για την αποθήκευση πληροφοριών. Όταν τα νευρωνικά δίκτυα χρησιμοποιούνται σε μηχανήματα, ονομάζονται «Τεχνητά Νευρωνικά Δίκτυα» (Artificial Neural Networks, ANNs). Οι νευρώνες ή αλλιώς κόμβοι, είναι η βασική μονάδα επεξεργασίας πληροφοριών και οι διασυνδέσεις μεταξύ τους ονομάζονται συνάψεις ή βάρη. Ο νευρώνας, λαμβάνει είσοδο από άλλους κόμβους ή από μια εξωτερική πηγή και

υπολογίζει μια έξοδο. Η έξοδος είναι, συνήθως, ένας αριθμός σύνοψης των εισόδων αφού περάσουν μέσω μιας γραμμικής συνάρτησης. Ένα βάρος σχετίζεται με κάθε είσοδο και αντιπροσωπεύει την σημασία αυτής της εισόδου στον νευρώνα. Μια μη-γραμμική συνάρτηση **f** περιέχεται στον νευρώνα και εφαρμόζεται στο άθροισμα των εισόδων του.[18]

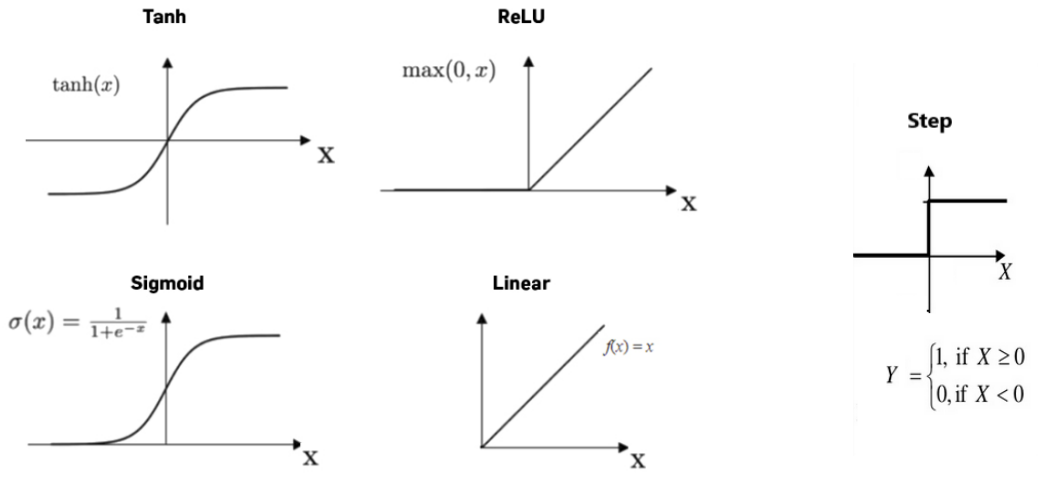


***Εικόνα 1**: Η δομή ενός τεχνητού νευρώνα ενός επιπέδου (Single layer perceptron)*

Στην **Εικόνα 1**, ο νευρώνας λαμβάνει εισόδους $\mathbf{x_1}$, $\mathbf{x_2}$ έως $\mathbf{x_n}$ στις οποίες σχετίζονται, αντίστοιχα, βάρη $\mathbf{w_{1j}}$, $\mathbf{w_{2j}}$ έως $\mathbf{w_{nj}}$. Στη συνέχεια, πολλαπλασιάζουμε τις εισόδους με τα βάρη και στη συνέχεια τα αθροίζουμε βάσει μιας μαθηματικής συνάρτησης και το δηλώνουμε ως **z**

$$z = \sum_{i=1}^{n} W_i x_i = W^T x$$

Η έξοδος του τεχνητού νευρώνα $\mathbf{o_j}$ υπολογίζεται βάσει μιας συνάρτησης ενεργοποίησης $\boldsymbol{\varphi}$ ενώ μερικές φορές εξαρτάται από ένα ορισμένο κατώφλι $\boldsymbol{\theta}$ (threshold). Οι συνήθεις συναρτήσεις ενεργοποίησης είναι οι παρακάτω : γραμμική (linear), βήματος (step), σιγμοειδής (sigmoid), υπερβολική εφαπτομένη (tanh) και διορθωμένης γραμμικής μονάδας (rectified linear unit, ReLu). **Εικόνα 2**

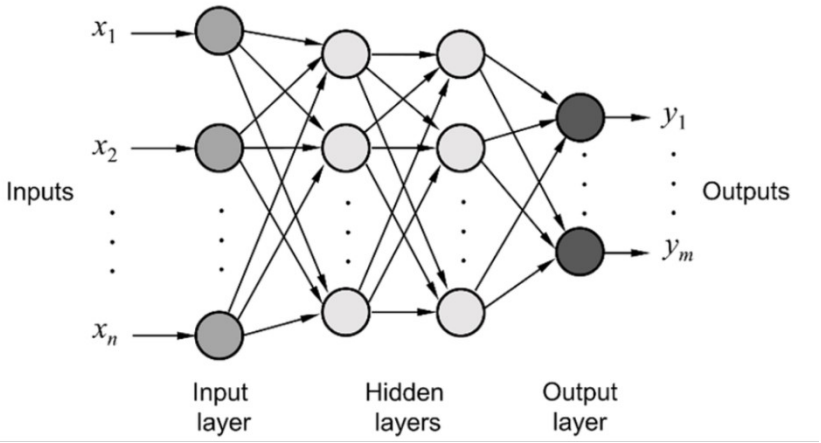


Εικόνα 2: *Συναρτήσεις Ενεργοποίησης νευρωνικού δικτύου*

3.3.1 Νευρωνικό δίκτυο πολλαπλών στρωμάτων (Multilayer neural network)

Το νευρωνικό δίκτυο πολλαπλών στρωμάτων, αποτελείται από τρία τουλάχιστον στρώματα κόμβων. Ένα στρώμα εισόδου, ένα στρώμα εξόδου και ένα ή περισσότερα κρυφά στρώματα . Εκτός από τους κόμβους εισόδου, κάθε κόμβος είναι ένας νευρώνας που χρησιμοποιεί μια συνάρτηση γραμμικής ενεργοποίησης. Κάθε κόμβος ενός στρώματος, συνδέεται με κάθε κόμβο του προηγούμενού του στρώματος ενώ η είσοδος από το στρώμα εισόδου πολλαπλασιάζεται με τα σχετικά βάρη κάθε συνδέσμου, διαδοχικά, μέχρι το στρώμα εξόδου. Το νευρωνικό δίκτυο μαθαίνει ενημερώνοντας επαναληπτικά τα βάρη μεταξύ των νευρώνων των κρυφών στρωμάτων προσπαθώντας να ελαχιστοποιήσει τη συνάρτηση απώλειας του

στρώματος εξόδου ή το σφάλμα εκπαίδευσης. Η παραπάνω επαναληπτική διαδικασία ονομάζεται αντίστροφη διάδοση (back propagation). Το δίκτυο, μπορεί να γίνει πιο ισχυρό προσθέτοντας κρυφά στρώματα. Τα Τεχνητά δίκτυα με πολλαπλά κρυμμένα στρώματα, ονομάζονται Βαθιά νευρωνικά δίκτυα.



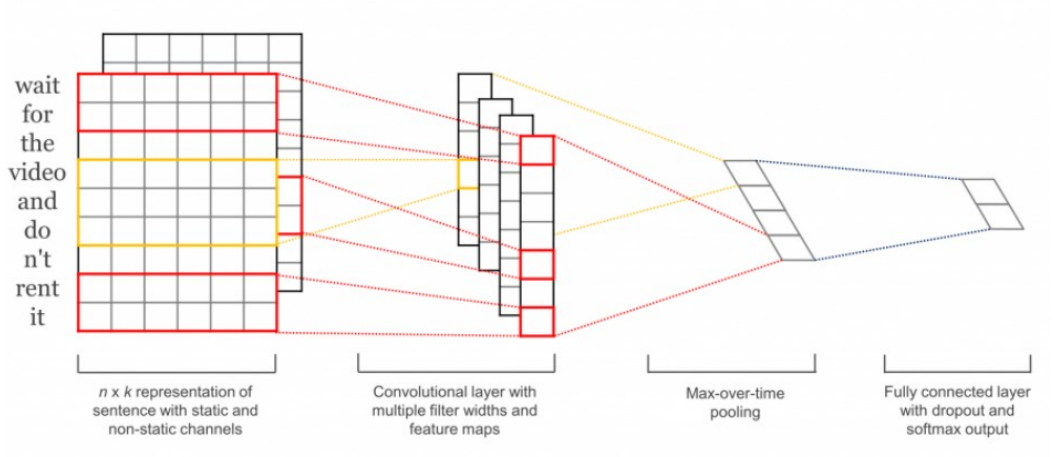
Εικόνα 3: *Νευρωνικό δίκτυο πολλαπλών στρωμάτων*

3.4 Βαθιά Νευρωνικά Δίκτυα (Deep Neural Networks,DNNs)

Το δίκτυο, μπορεί να γίνει πιο ισχυρό προσθέτοντας κρυφά στρώματα. Τα Τεχνητά δίκτυα με πολλαπλά κρυμμένα στρώματα, ονομάζονται Βαθιά νευρωνικά δίκτυα.

3.4.1 Συνελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network,CNN)

Τα Συνελικτικά Δίκτυα, χρησιμοποιούνται σε ποικιλία εφαρμογών. Αρχικά σχεδιάστηκαν για ανάλυση εικόνας, όμως τα τελευταία χρόνια χρησιμοποιούνται και σε εφαρμογές Επεξεργασίας Φυσικής Γλώσσας (Natural Language Process, NLP). Η αρχιτεκτονική ενός CNN για ταξινόμηση προτάσεων φαίνεται στην **Εικόνα 4**.



Εικόνα 4: *Αρχιτεκτονική CNN για ταξινόμηση προτάσεων* [19]. *Οι πυρήνες διαφορετικού μεγέθους σημειώνονται με διαφορετικό χρώμα*

Ως είσοδος στο παραπάνω μοντέλο, δίνεται ένας πίνακας n x k ενσωματώσεων λέξεων (Περιγράφονται στην Ενότητα 3.5), όπου n είναι ο μέγιστος αριθμός λέξεων της πρότασης και k είναι το μήκος της ενσωμάτωσης. Η έξοδος του μοντέλου είναι μια κλάση $y \in 0,1$ που αντιπροσωπεύει δύο διαφορετικές κατηγορίες για την είσοδο. Παρακάτω περιγράφονται οι τέσσερις τύποι στρωμάτων που σχηματίζουν ένα CNN.

Στρώμα Ενσωμάτωσης

Στο στρώμα ενσωμάτωσης, οι λέξεις της πρότασης, μετατρέπονται σε διανύσματα λέξεων.
Το στρώμα ενσωμάτωσης περιγράφεται αναλυτικά στην Ενότητα 3.5.

Συνελικτικό Στρώμα (CNN Layer)

Σε αυτό το στρώμα εισάγονται τα διανύσματα λέξεων από το στρώμα ενσωμάτωσης ώστε να εξαχθούν τα πιο σημαντικά n-gram χαρακτηριστικά και να δημιουργηθούν οι σημασιολογικές αναπαραστάσεις της πρότασης. Η συνέλιξη περιλαμβάνει ένα φίλτρο $W \in \mathbb{R}^{hk}$, το οποίο εφαρμόζεται σε ένα παράθυρο λέξεων h ώστε να παραχθεί ένα χαρακτηριστικό. Ένα φίλτρο c_i δημιουργείται από ένα παράθυρο λέξεων $x_{i:i+h-1}$ με βάση τον τύπο:

$$c_i = f(W \cdot x_{i:i+h-1} + b)$$

Το φίλτρο εφαρμόζεται σε κάθε πιθανό παράθυρο λέξεων της πρότασης $\{x_{1:h}, x_{2:h+1}, \dots, x_{n-h+1:n}\}$ ώστε να παραχθεί ο χάρτης χαρακτηριστικών:

$$c = [c_1, c_2, \dots, c_{n-h+1}]$$

Στρώμα Υπο-δειγματοληψίας (Pooling)

Το στρώμα υπο-δειγματοληψίας, μειώνει το χωρικό μέγεθος της αναπαράστασης εφαρμόζοντας μια λειτουργία, όπως, μέγιστο (max), άθροισμα (sum), μέσος όρος (average) ή ομαλοποίηση-L2 (L2-norm). Η υπο-δειγματοληψία εφαρμόζεται για δύο λόγους. Ο πρώτος είναι, να μειωθεί η διάσταση του χάρτη χαρακτηριστικών, κρατώντας παράλληλα τις σημαντικότερες πληροφορίες, και ο δεύτερος είναι για την μετατροπή μιας μεταβλητού μεγέθους εισόδου σε έναν πίνακα εξόδου σταθερού μεγέθους.

Πλήρως συνδεδεμένο στρώμα (Fully-Connected Layer)

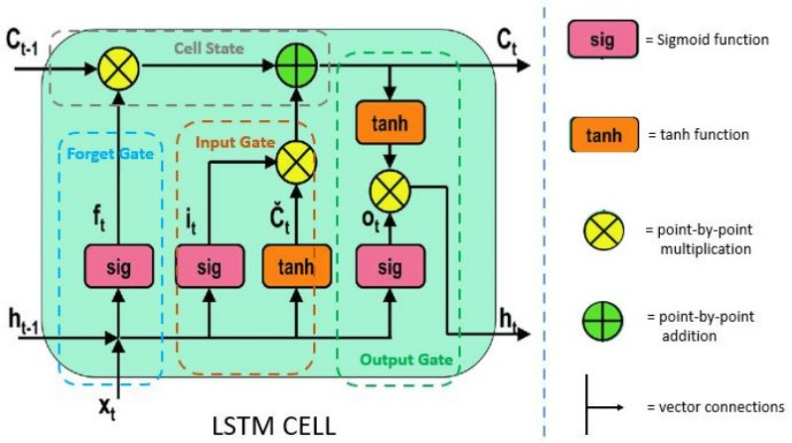
Στο πλήρες συνδεδεμένο στρώμα, όλοι οι νευρώνες έχουν συνδέσεις με όλες τις συνδέσεις του προηγούμενου στρώματος, σε αντιδιαστολή με το συνελικτικό στρώμα (convolutional layer) το οποίο συνδέεται μόνο με μια τοπική περιοχή στην είσοδο. Οι τιμές εξόδου υπολογίζονται από τον πολλαπλασιασμό ενός πίνακα με μια τιμή εξισορρόπησης συστηματικού σφάλματος. (bias offset)

3.4.2 Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Network, RNN)

Το RNN εκτελεί την ίδια λειτουργία για κάθε είσοδο δεδομένων, με την έξοδο να εξαρτάται από προηγούμενους υπολογισμούς. Απομνημονεύει κάθε προηγούμενη έξοδο δίνοντας κάθε έξοδο ως είσοδο στο επόμενο στρώμα. Κατά τη αντίστροφη διάδοση (back propagation) τα RNN υποφέρουν από το πρόβλημα των εξαφανιζόμενων κλίσεων (vanishing gradient). Οι κλίσεις είναι τιμές που χρησιμοποιούνται για την ενημέρωση των βαρών σε ένα νευρωνικό δίκτυο. Όταν η κλίση συρρικνώνεται και πλησιάζει το μηδέν , καθίσταται δύσκολη η εκπαίδευση του μοντέλου . Τα μοντέλα GRU[20] και LSTM[21] λύνουν το πρόβλημα αυτό.

Μακροπρόθεσμα Δίκτυα Μνήμης (Long Short-Term Memory Networks, LSTM)

Τα δίκτυα RNN δεν έχουν βραχυπρόθεσμη μνήμη. Όταν μια ακολουθία είναι αρκετά μεγάλη, δυσκολεύονται να μεταφέρουν πληροφορίες από τα προηγούμενα επίπεδα στα επόμενα, αφήνοντας έτσι πληροφορίες ζωτικής σημασίας. Το μοντέλο LSTM μοιάζει με ένα τυπικό μοντέλο RNN, αλλά κάθε κόμβος αντικαθίσταται από ένα κελί μνήμης. Το κελί μνήμης αποτελείται από μια πύλη εισόδου (input gate), μια πύλη εξόδου (output gate), μια πύλη λήθης (forget gate) και έναν νευρώνα που συνδέεται ξανά με τον εαυτό του. **Εικόνα 5**



Εικόνα 5:*LSTM* [22]

Forget Gate

Η forget gate αποφασίζει ποιες πληροφορίες πρέπει να διατηρηθούν εκτελώντας μια σιγμοειδή συνάρτηση με W_f (βάρος) και b_f (bias). Η πληροφορία από την τρέχουσα είσοδο X_t και την κρυφή κατάσταση h_{t-1} περνούν από μια σιγμοειδή συνάρτηση.

$$f_t = \sigma(W_f\,[h_{t-1}, x_t] + b_f$$

Input Gate

Η input gate προκειμένου να αποφασίσει ποιες πληροφορίες να αποθηκεύσει στην κατάσταση του κελιού, εκτελεί δύο βήματα. Στο πρώτο βήμα, η τρέχουσα κατάσταση X_t η κρυφή κατάσταση h_{t-1} περνούν από μια δεύτερη σιγμοειδή συνάρτηση και οι τιμές μετατρέπονται ανάμεσα σε 0 και 1, ώστε να αποφασιστεί ποιες τιμές θα ενημερωθούν i_t. Με 0 χαρακτηρίζονται οι σημαντικές και με 1 οι μη σημαντικές. Στη συνέχεια, οι ίδιες πληροφορίες περνούν μέσα από μια συνάρτηση tanh (υπερβολική εφαπτομένη), ώστε να δημιουργηθεί διάνυσμα $\tilde{C}_t$ με τιμές μεταξύ -1 και 1

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i$$
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Στο δεύτερο βήμα, αποφασίζεται και αποθηκεύεται η πληροφορία από τη νέα κατάσταση στην κατάσταση κελιού. Για να επιτευχθεί αυτό, πολλαπλασιάζεται η προηγούμενη κατάσταση C_{t-1} με το διάνυσμα λήθης f_t και στη συνέχεια προστίθενται οι νέες τιμές $i_t \times \tilde{C}_t$.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate

Η πύλη εξόδου καθορίζει την τιμή της επόμενης κρυφής κατάστασης h_t, εκτελώντας μια τρίτη σιγμοειδής συνάρτηση πάνω στην προηγούμενη κρυφή κατάσταση h_{t-1} και στην τρέχουσα

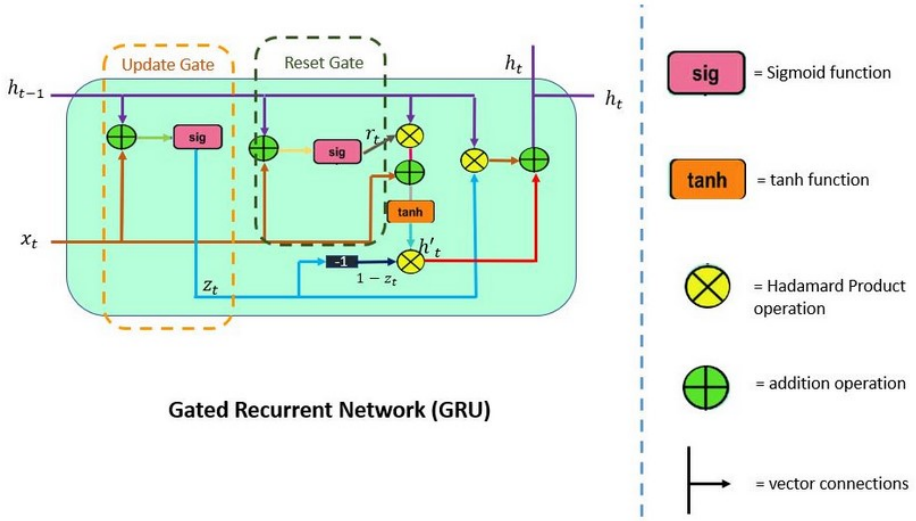
κατάσταση X_t. Στο τέλος, η έξοδος της σιγμοειδούς o_t πολλαπλασιάζεται με την συνάρτηση υπερβολικής εφαπτομένης (tanh) της νέας κατάστασης κελιού C_t.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o$$

$$h_t = o_t * \tanh(C_t)$$

Επαναλαμβανόμενες Μονάδες(Gated Recurrent Units,GRU)

Το μοντέλο GRU χρησιμοποιεί δύο πύλες, την πύλη ενημέρωσης (update gate) και την πύλη επαναφοράς (reset gate). Αυτές οι πύλες αποφασίζουν ποιες πληροφορίες θα πρέπει να διαβιβάζονται στην έξοδο και μπορούν να εκπαιδευτούν ώστε να κρατούν πληροφορίες από πολύ καιρό πριν χωρίς να χάνουν σημαντικά χαρακτηριστικά ενώ μπορούν να διαγράφουν χαρακτηριστικά που δεν σχετίζονται με την πρόβλεψη.



Εικόνα 6: *GRU* [23]

Update gate

Η Update gate z_t είναι υπεύθυνη για τον προσδιορισμό του όγκου των πληροφοριών, από τα προηγούμενα βήματα, που πρέπει να περάσουν στην επόμενη κατάσταση. Το διάνυσμα εισόδου x_t πολλαπλασιάζεται με τους πίνακες του βάρους των παραμέτρων W_z. Η προηγούμενη κρυφή κατάσταση h_{t-1} πολλαπλασιάζεται με το βάρος της. Στη συνέχεια, το άθροισμα των παραπάνω περνάει από μια σιγμοειδή συνάρτηση. Οι τιμές που προκύπτουν είναι ανάμεσα σε 0 και 1

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z$$

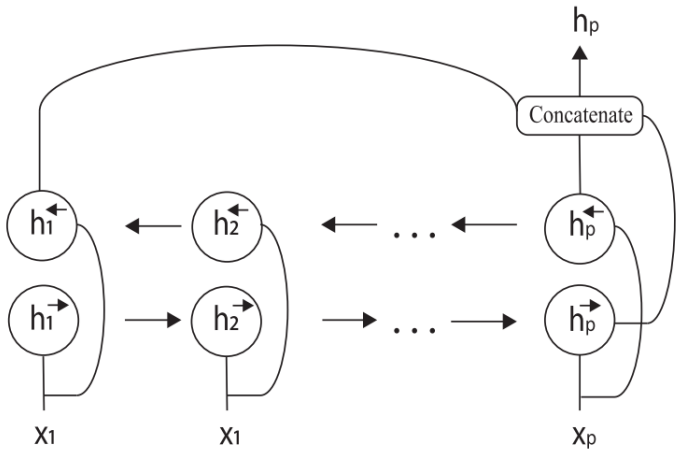
Reset gate

Η reset gate r_t χρησιμοποιείται για να αποφασιστεί πόσες από τις προηγούμενες πληροφορίες χρειάζεται να μην ληφθούν υπόψη. Τύπος είναι ίδιος με την πύλη update

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r$$

3.4.3 Αμφίδρομα RNN (Bidirectional RNN)

Τα αμφίδρομα επαναλαμβανόμενα δίκτυα, συνδυάζουν δύο δίκτυα RNN. Αυτή η δομή επιτρέπει την αντιστροφή της κατεύθυνσης της ακολουθίας, τροφοδοσία αυτού σε ένα ανεξάρτητο δίκτυο και συνένωση των κρυφών καταστάσεων που θα προκύψουν. Με αυτή την τεχνική, μπορούμε να προσθέσουμε πληροφορίες, σε ένα μοντέλο, με βάση τα συμφραζόμενα. Στην Εικόνα 7 φαίνεται η αρχιτεκτονική ενός τέτοιου μοντέλου.



Εικόνα 7: *Bidirectional RNN* [24]

Οι καταστάσεις h_p του σχήματος μπορούν να προέρχονται από LSTM ή GRU. Το δίκτυο με τις αντίστροφες καταστάσεις είναι ένα ανεξάρτητο αντίγραφο του αρχικού δικτύου. Οι κρυφές καταστάσεις που προκύπτουν συνενώνονται προκειμένου να σχηματιστούν

$$h_p = (h_{\overrightarrow{p}}, h_{\overleftarrow{1}})$$

Το κρυφό διάνυσμα τελικής κατάστασης μπορεί να χρησιμοποιηθεί στη μοντελοποίηση με τον ίδιο τρόπο όπως ένα απλό RNN.

3.5 Ενσωματώσεις Λέξεων (Word Embeddings)

Οι ενσωματώσεις λέξεων είναι μια από τις πιο δημοφιλείς αναπαραστάσεις λέξεων που γεφυρώνει την ανθρώπινη κατανόηση της γλώσσας με αυτήν μιας μηχανής. Είναι ικανές, σε ένα έγγραφο, να συλλάβουν το περιεχόμενο μιας λέξης, την σημασιολογική και συντακτική ομοιότητα καθώς και τη σχέση της λέξης με άλλες. Προκειμένου να υλοποιηθούν, κάθε μεμονωμένη λέξη αντιπροσωπεύεται ως διάνυσμα πραγματικής αξίας σε έναν προκαθορισμένο διανυσματικό χώρο. Κάθε λέξη χαρτογραφείται σε ένα διάνυσμα και οι τιμές του διανύσματος μαθαίνονται με βάση τη χρήση των λέξεων. Επομένως λέξεις που χρησιμοποιούνται με παρόμοιο τρόπο έχουν παρόμοιες αναπαραστάσεις, αποτυπώνοντας έτσι το νόημά τους.

3.6 Στρώμα Ενσωμάτωσης (Embedding Layer)

Το στρώμα ενσωμάτωσης είναι μια ενσωμάτωση λέξεων που μαθαίνεται από κοινού με ένα μοντέλο νευρωνικού δικτύου. Απαιτείται η προετοιμασία του κειμένου έτσι ώστε κάθε λέξη να είναι κωδικοποιημένη. Το μέγεθος του διανύσματος καθορίζεται ως μέρος του μοντέλου, με διαστάσεις 50, 100, 200 κ.α. και τα διανύσματα αρχικοποιούνται με μικρούς τυχαίους αριθμούς. Το στρώμα ενσωμάτωσης χρησιμοποιείται στο μπροστινό μέρος ενός νευρωνικού δικτύου και ταιριάζει σε αυτό, με έναν επιτηρούμενο τρόπο, χρησιμοποιώντας τον αλγόριθμο οπισθοδιάδοσης (backpropagation)

3.7 Καθολικά Διανύσματα για αναπαράσταση λέξεων(Global Vectors for Word Representation ,GloVe)

Το μοντέλο GloVe αναπτύχθηκε από τους Pennington et al. στο Πανεπιστήμιο του Στάνφορντ το 2014. Είναι ένα μοντέλο που μαθαίνει διανύσματα ή λέξεις κατασκευάζοντας ένα πίνακα περιεχομένου ή συνύπαρξης λέξεων χρησιμοποιώντας στατιστικά στοιχεία από ολόκληρο το σώμα κειμένου. Λαμβάνοντας υπόψη ένα σώμα κειμένου που έχει λέξεις **V**, ο πίνακας συνύπαρξης **X** θα είναι ένας πίνακας **VxV**, όπου η **i** σειρά και η **j** στήλη του **X, X_ij** υποδηλώνουν πόσες φορές η λέξη **i** έχει συνυπάρξει με την λέξη **j**. Ένα παράδειγμα, είναι ο **Πίνακας 1**.

Έστω η πρόταση : ***I play drums, I love drums and I love guitar***

	I	play	drums	love	guitar
I	0.0	1.0	0.0	2.0	0.0
play	1.0	0.0	1.0	0.0	0.0
drums	0.0	1.0	0.0	1.0	0.0
love	2.0	0.0	1.0	0.0	1.0
guitar	0.0	0.0	0.0	1.0	0.0

Πίνακας 1: *Πίνακας συνύπαρξης της πρότασης "I play drums, I love drums and I love guitar"*

Αν υπολογίσουμε τις πιθανότητες μερικών ζευγαριών λέξεων σύμφωνα με το θεώρημα του Bayes προκύπτει:

$$\mathbf{P(drums|play) = 1}$$

$$\mathbf{P(drums|love) = 0.5}$$

Επομένως, ο λόγος πιθανότητας που προκύπτει είναι:

$$\frac{\mathbf{P(drums|play)}}{\mathbf{P(drums|love)}} = \mathbf{2}$$

Δεδομένης μιας λέξης διερεύνησης, η αναλογία μπορεί να είναι μικρή, μεγάλη ή ίση με το 1. Επειδή αναλογία > 1 , μπορούμε να συμπεράνουμε ότι η πιο σχετική λέξη για τα "drums" είναι "play" σε σύγκριση με την λέξη "love". Ομοίως αν ο λόγος είναι κοντά στο 1, τότε και οι δύο λέξεις σχετίζονται με τα "drums".

Από τα παραπάνω φαίνεται ότι είμαστε σε θέση να αντλήσουμε πληροφορίες για τη σχέση μεταξύ των λέξεων, χρησιμοποιώντας στατιστικά στοιχεία.

Το GloVe μαθαίνει να κωδικοποιεί τις πληροφορίες του λόγου πιθανότητας με τη μορφή διανυσμάτων λέξεων. Η πιο γενική μορφή του μοντέλου δίνεται από τη συνάρτηση:

λέξη διερεύνησης (probe word)

$w \in \mathbb{R}^d$ *είναι διανύσματα λέξεων*

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

συσχετίσεις μεταξύ των λέξεων w_i και w_j

πιθανότητες συνύπαρξης των λέξεων w_j και w_k

3.8 fastText

Αναπτύχθηκε από το τμήμα ερευνών Τεχνητής Νοημοσύνης του Facebook το 2015. Στο συγκεκριμένο μοντέλο τα διανύσματα δεν δημιουργούνται σε επίπεδο λέξεων αλλά σε επίπεδο συλλαβών (n-grams) και η αναπαράστασή τους είναι το άθροισμα αυτών των n-grams χαρακτήρων. Για παράδειγμα, αν έχουμε την λέξη “learning” με n=3, η αναπαράστασή της στο fastText με 3-gram θα είναι :

<le,lea,ear,arn,rni,nin,ing,ng>

Με αυτόν τον τρόπο επιτυγχάνεται ευκολότερη σύλληψη της σημασίας μικρότερων λέξεων καθώς και τον προθεμάτων(prefixes) και επιθεμάτων (suffixes).

Το fastText έχει καλύτερη απόδοση με τις άγνωστες λέξεις από ότι το GloVe. Στο fastText, ακόμα και αν μια λέξη δεν έχει χρησιμοποιηθεί κατά τη διάρκεια της εκπαίδευσης, μπορεί να χωριστεί στις συλλαβές (n-grams) ώστε να πάρει τις ενσωματώσεις της, ενώ το GloVe αποτυγχάνει να παρέχει οποιαδήποτε διανυσματική αναπαράσταση για λέξεις που δεν υπάρχουν στο λεξικό του μοντέλου.

3.9 Bert (Bidirectional Encoder Representations from Transformers)

Το Bert δημιουργήθηκε το 2018 από τον Jacob Devlin και τους συνεργάτες του στην Google.[25] Όπως αναφέραμε παραπάνω, οι προηγούμενες μέθοδοι δημιουργούν ένα διάνυσμα για κάθε λέξη στο λεξιλόγιο, έτσι για παράδειγμα η λέξη «έκτακτος» θα έχει την ίδια αναπαράσταση είτε πρόκειται για την έννοια «μη προκαθορισμένος» είτε πρόκειται για την έννοια «εξαιρετικός». Αντίθετα, στο Bert το διάνυσμα της λέξης δημιουργείται βάσει των υπόλοιπων λέξεων που υπάρχουν στην πρόταση ώστε πχ. η λέξη «έκτακτο» στις παρακάτω δύο προτάσεις να έχει διαφορετικό νόημα.

1. «Αυτή έχει έκτακτο συμβούλιο»
2. «Αυτή έχει έκτακτο χαρακτήρα»

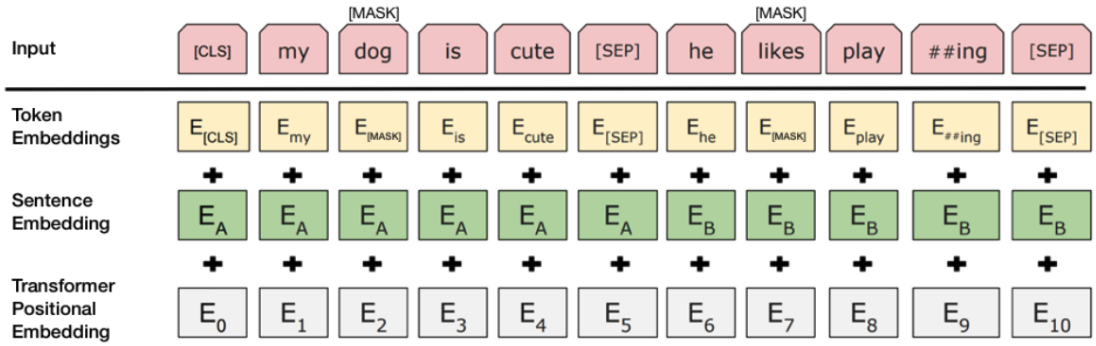
Το Bert είναι ένα αμφίδρομο μοντέλο το οποίο στηρίζεται στην αρχιτεκτονική του Μετασχηματιστή. Ο Μετασχηματιστής αποτελείται από δύο μηχανισμούς, τον κωδικοποιητή που διαβάζει το κείμενο εισόδου και τον αποκωδικοποιητή ο οποίος παράγει μια πρόβλεψη για το συγκεκριμένο πρόβλημα. Δεδομένου ότι ο σκοπός του Bert είναι να δημιουργήσει ένα γλωσσικό μοντέλο, μόνο ο μηχανισμός του κωδικοποιητή είναι απαραίτητος.

Το Bert αποτελείται από δύο βήματα. Στο πρώτο γίνεται η προ-εκπαίδευση του μοντέλου (pre-training) και στο δεύτερο γίνεται η προσαρμογή του (fine-tunning). Τα δύο αυτά βήματα περιγράφονται αναλυτικά στις Ενότητες 3.9.1 και 3.9.2

Προκειμένου το μοντέλο να χειρίζεται ποικιλία εργασιών, η αναπαράσταση εισόδου είναι σχεδιασμένη έτσι ώστε να αντιπροσωπεύει τόσο μια πρόταση όσο και ένα ζεύγος προτάσεων.

Το πρώτο σύμβολο σε κάθε ακολουθία εισόδου είναι το ειδικό σύμβολο ταξινόμησης [CLS] και η τελική κρυφή κατάστασή του χρησιμοποιείται ως η συνολική αναπαράσταση ακολουθίας σε προβλήματα ταξινόμησης. Τα ζεύγη προτάσεων, τοποθετούνται μαζί σε μία ακολουθία και διαχωρίζονται ως εξής: Πρώτα διαχωρίζονται με το ειδικό σύμβολο [SEP] και στη συνέχεια προστίθεται ένα εκπαιδευμένο διάνυσμα ενσωμάτωσης (learned embedding) σε κάθε ένα σύμβολο, που δείχνει σε ποια από τις δύο προτάσεις ανήκει (A ή B).

Για κάθε ένα σύμβολο, η αναπαράσταση εισόδου κατασκευάζεται αθροίζοντας τα διανύσματα ενσωμάτωσής του, τα διανύσματα ενσωμάτωσης του τμήματος/πρότασης (segment) στην οποία ανήκει και τα διανύσματα ενσωμάτωσης της θέσης στην οποία βρίσκεται μέσα στην ακολουθία. Όπως φαίνεται στην **Εικόνα 8**



Εικόνα 8: Αναπαράσταση εισόδου BERT.[25]

3.9.1 Προ-εκπαίδευση (pre-training)

Για την προ-εκπαίδευσή του BERT, χρησιμοποιούνται δύο στρατηγικές χωρίς επίβλεψη, το Γλωσσικό Μοντέλο Απόκρυψης και η Πρόβλεψη της επόμενης Πρότασης:

Το Γλωσσικό Μοντέλο Απόκρυψης (Masked Language Model)

Σε αυτό το μοντέλο, το 15% των λέξεων εισόδου κάθε ακολουθίας, αντικαθίσταται με το διακριτικό [MASK]. Ο σκοπός είναι, να προβλεφθεί η αρχική τιμή των λέξεων που έχουν αντικατασταθεί βάσει των υπόλοιπων λέξεων της ακολουθίας.

Για να επιτευχθεί αυτό απαιτείται:

- Προσθήκη ενός επιπέδου ταξινόμησης πάνω από την έξοδο του κωδικοποιητή.

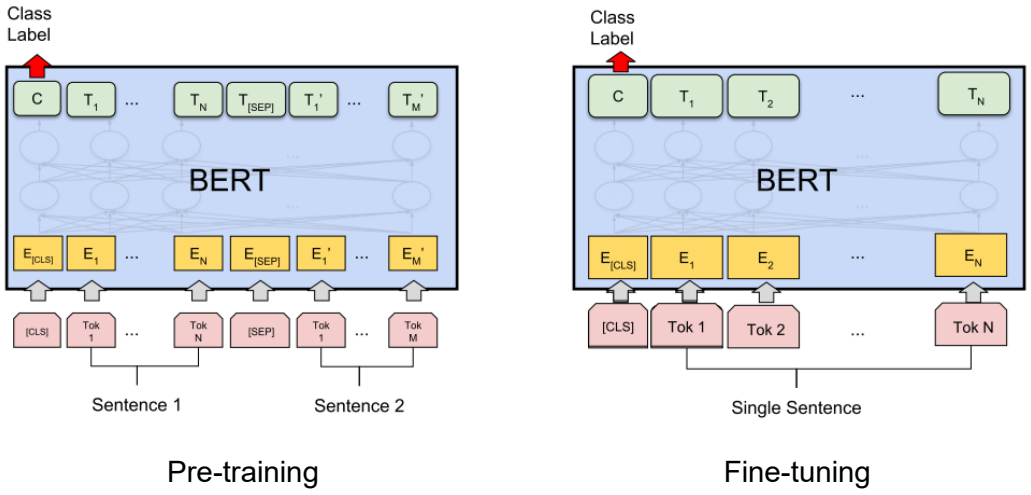
- Ο πολλαπλασιασμός των διανυσμάτων εξόδου με τον πίνακα ενσωματώσεων, μετατρέποντάς τα στη διάσταση του λεξιλογίου.
- Ο υπολογισμός της πιθανότητας κάθε λέξης του λεξιλογίου ,χρησιμοποιώντας Softmax

Πρόβλεψη της επόμενης Πρότασης (Next Sentence Prediction)

Προκειμένου να εκπαιδευτεί ένα μοντέλο που κατανοεί τις σχέσεις μεταξύ δύο προτάσεων χρησιμοποιείται ένα σύνολο δεδομένων που αποτελείται από ζεύγη προτάσεων. Στο 50% των περιπτώσεων η Β πρόταση είναι η πραγματική πρόταση που ακολουθεί την Α και χαρακτηρίζεται με την ετικέτα 'Είναι η Επόμενη' (IsNext) ενώ στις υπόλοιπες περιπτώσεις η πρόταση Β είναι μια τυχαία πρόταση και χαρακτηρίζεται με την ετικέτα 'Δεν είναι η επόμενη'(Not-Next)

3.9.2 Προσαρμογή (Fine-tuning)

Κατά την Προσαρμογή, το μοντέλο αρχικοποιείται με τις παραμέτρους του προηγούμενου βήματος και ξανά εκπαιδεύεται χρησιμοποιώντας δεδομένα που είναι επισημειωμένα από τις προηγούμενες εργασίες. Παρ' όλο που τα δεδομένα αρχικοποιούνται με τις ίδιες προ-εκπαιδευμένες παραμέτρους, κάθε πρόβλημα έχει ξεχωριστά μοντέλα προσαρμογής.



***Εικόνα 9**: Οι διαδικασία της προ-εκπαίδευσης και της προσαρμογής για το Bert σε πρόβλημα κατηγοριοποίησης*[25]

3.10 Προ-εκπαιδευμένες Ενσωματώσεις Λέξεων(Pretrained Word Embeddings)

Οι προ-εκπαιδευμένες ενσωματώσεις λέξεων εκπαιδεύονται σε μεγάλα σύνολα δεδομένων, αποθηκεύονται και στη συνέχεια χρησιμοποιούνται για την εκπαίδευση άλλων μοντέλων. Η εκμάθηση ενσωματώσεων λέξεων από το μηδέν είναι ένα δύσκολο πρόβλημα λόγω της σπανιότητας των δεδομένων εκπαίδευσης αλλά και του μεγάλου όγκου εκπαιδευόμενων παραμέτρων. Τα προ-εκπαιδευμένα μοντέλα ενσωματώσεων είναι ικανά να ενισχύσουν την απόδοση ενός μοντέλου Επεξεργασίας της Φυσικής Γλώσσας (Natural Language Processing, NLP

3.11 Μετρικές απόδοσης

Παρακάτω παρουσιάζουμε μερικές από τις συνηθέστερες μετρικές απόδοσης αλγορίθμων μηχανικής μάθησης.

Confusion Matrix (Πίνακας σύγχυσης)

Μας δίνει έναν πίνακα ως έξοδο που περιγράφει την πλήρη απόδοση του μοντέλου. Πρόκειται για έναν πίνακα NxN, όπου το (i,j) στοιχείο του ισούται με το πλήθος των σημείων που, ενώ προέρχονται από την κλάση i, καταχωρούνται στην κλάση j. Δίνει πληροφορίες σχετικά με το αν κάποιες κλάσεις έχουν την τάση να συγχέονται με άλλες κλάσεις.

	πρόβλεψη PREDICTED CLASS		
πραγματική ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	f_{11} TP	f_{10} FN
	Class=No	f_{01} FP	f_{00} TN

Πίνακας 2: *Πίνακας Σύγχυσης*

Στον **Πίνακα 2** αναπαρίστανται οι παρακάτω τιμές

TP (true positive) $\mathbf{f_{11}}$**:** οι περιπτώσεις στις οποίες, προβλέψαμε Yes και η πραγματική έξοδος ήταν Yes

FN (false negative) $\mathbf{f_{10}}$**:** οι περιπτώσεις στις οποίες, προβλέψαμε No και η πραγματική έξοδος ήταν No

FP (false positive) $\mathbf{f_{01}}$**:** οι περιπτώσεις στις οποίες, προβλέψαμε Yes και η πραγματική έξοδος ήταν No

TN (true negative) $\mathbf{f_{00}}$**:** οι περιπτώσεις στις οποίες, προβλέψαμε No και η πραγματική έξοδος ήταν Yes

Accuracy

Η ακρίβεια είναι ο λόγος του αριθμού των σωστών προβλέψεων προς τον συνολικό αριθμό των προβλέψεων.

$$\text{Accuracy}=\frac{TP+TN}{\text{συνολικός αριθμός προβλέψεων}}$$

Η μετρική αυτή, λειτουργεί καλά μόνο όταν υπάρχει ίσος αριθμός δειγμάτων που ανήκουν σε κάθε κλάση.

Precision

Μετρά τι ποσοστό των θετικών ταυτοποιήσεων που προέρχονται από την κλάση i ταξινομήθηκαν σωστά στην κλάση αυτή

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

Recall

Μετρά τι ποσοστό των θετικών ταυτοποιήσεων που ταξινομούνται στην κλάση i και ανήκουν πράγματι στην κλάση αυτή

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

F_1score

Ο αρμονικός μέσος των Precision και Recall.

$$\text{F}_1=2 * \frac{P*R}{P+R}$$

Υπάρχουν τρεις παραλλαγές του f_1 score, όπως:

- η **average-macro** : Υπολογίζει την τιμή του F1, υπολογίζοντας τις μετρήσεις για κάθε κλάση. Δεν λαμβάνεται υπόψη η άνιση κατανομή στις κλάσεις
- η **weighted-macro**: Υπολογίζει την τιμή του F1, για κάθε κλάση ξεχωριστά, βάσει υποστήριξης (ο αριθμός των πραγματικών παρουσιών για κάθε κλάση). Ως εκ τούτου η κλάση με τα περισσότερα παραδείγματα, υπερτερεί σε σχέση με τις άλλες.
- η **micro :** Υπολογίζει την τιμή του F1 ,λαμβάνοντας υπόψη, τον ολικό αριθμό των αληθώς θετικών (TP), ψευδώς αρνητικών (FN) και ψευδώς θετικών (FP) όλων των κλάσεων, χωρίς να ευνοεί κάποια κλάση.

4 Δεδομένα Εισόδου

Σε αυτό το κεφάλαιο αναλύουμε τα δεδομένα που χρησιμοποιήσαμε ως είσοδο στα πειράματά μας. Για την εξαγωγή όσο το δυνατόν καλύτερων αποτελεσμάτων, είναι πολύ σημαντική η κατανόηση και η σωστή προ-επεξεργασία των δεδομένων εισόδου.

4.1 Σύνολο Δεδομένων (Dataset)

Στην εργασία μας, χρησιμοποιήσαμε δύο dataset που είναι δημόσια διαθέσιμα. Το πρώτο dataset δημιουργήθηκε από τους Davidson et al.[1] και διαθέτει 24.783 tweets στα Αγγλικά τα οποία έχουν χαρακτηριστεί, μη αυτόματα, από το Crowdflower[26] ως "Hate" (ρητορική μίσους), "Offensive" (προσβλητικό), "Neither" (κανένα από τα δύο). Μόνο το 5,77% του dataset είναι χαρακτηρισμένο ως "ρητορική μίσους", ενώ το 77,43% ως προσβλητικό και το 16,8% ως κανένα από τα δύο. Τα παραπάνω φαίνονται στον **Πίνακα 3**

Κλάση (Class)	Αριθμός Tweets
Hate	1.430 (5,77%)
Offensive	19.190 (77.43%)
Neither	4.163 (16.8%)
Σύνολο	24.783

Πίνακας 3: *Κατηγορίες dataset Davidson*

Όπως φαίνεται από τα παραπάνω ποσοστά των τριών κατηγοριών, το σύνολο δεδομένων που επιλέξαμε είναι ανομοιογενές ως προς τη κατανομή των κλάσεων. Δεδομένου ότι η ρητορική μίσους ,και σε πραγματικές συνθήκες, εμφανίζεται σπάνια δεν πραγματοποιήσαμε τεχνικές υπερ-δειγματοληψίας ή υπο-δειγματοληψίας για να προσαρμόσουμε την κατανομή τάξεων ώστε να παρέχουμε τα σύνολα δεδομένων όσο το δυνατόν πιο ρεαλιστικά.

Το δεύτερο dataset δημιουργήθηκε από τους Founta et al.[27] και διαθέτει 100.000 tweets στα Αγγλικά τα οποία έχουν χαρακτηριστεί, μη αυτόματα, από το Crowdflower[26] ως "Normal" (κανονικό, δεν ανήκει σε καμία από τις επόμενες κατηγορίες), "Abusive" (προσβλητικό), "Hateful" (ρητορική μίσους), "Spam". Ως "Spam", χαρακτηρίζεται η ανάρτηση που περιλαμβάνει διαφημίσεις, πώληση προϊόντων ενηλίκων, σύνδεση με κακόβουλους

ιστότοπους, απόπειρες ηλεκτρονικού ψαρέματος και άλλα είδη ανεπιθύμητων πληροφοριών που εκτελούνται, συνήθως, επανειλημμένα. Προκειμένου να μπορέσουμε να διαχειριστούμε, πιο εύκολα, το παραπάνω σύνολο δεδομένων, αφαιρέσαμε την κλάση "spam" και στη συνέχεια μειώσαμε τα δεδομένα στην κλάση "Normal" κατά 35.000 και "Abusive" κατά 10.000. Έτσι το τελικό σύνολο δεδομένων μας περιέχει 17.150 δεδομένα στην κλάση "Abusive" ποσοστό 41%, 4965 στην κλάση "Hateful" ποσοστό 12% και 19229 στην κλάση "Normal" ποσοστό 46%. Τα παραπάνω φαίνονται στον **Πίνακα 4**.

Κλάση (Class)	Αριθμός Tweets
Normal	19.229 (46%)
Abusive	17.150 (41%)
Hateful	4.965 (12%)
Σύνολο	41.344

Πίνακας 4: *Κατηγορίες dataset Founta*

4.2 Καθαρισμός δεδομένων από γλώσσα αργκό

Από τις μεγαλύτερες προκλήσεις που αντιμετωπίσαμε, ήταν ο καθαρισμός των δεδομένων μας από γλώσσα αργκό και συντομογραφίες που χρησιμοποιούνται κατά κόρον σε εφαρμογές όπως το Tweeter. Ο τρόπος γραφής, σε τέτοιες εφαρμογές, τις περισσότερες φορές είναι χαλαρός και δεν ακολουθεί κανόνες ορθογραφίας. Επίσης δημιουργούνται νέες λέξεις που δεν υπάρχουν στο τυπικό λεξικό, όπως για παράδειγμα η λέξη "blaxican" που αποτελείται από τις λέξεις "black" και "Mexican" ή η λέξη "nigglet" που σημαίνει "baby nigger". Προκειμένου να επιλύσουμε αυτό το πρόβλημα, δημιουργήσαμε ένα λεξικό όρων το οποίο εμπλουτίσαμε με τις ερμηνείες τέτοιων λέξεων. Αξίζει να σημειωθεί ότι η λέξη "n*gger" εμφανίζεται με δώδεκα (12) διαφορετικούς τρόπους γραφής στα δεδομένα μας, η λέξη "n*ggers" με οκτώ (8) και η λέξη "f*cking" με οκτώ (8). Στον **Πίνακα 5** παρουσιάζονται οι διαφορετικές εκδοχές των παραπάνω λέξεων.

n*gger	**niggers**	**f*cking**
n*qqa	n*gs	f*ccing
n*gguh	n*ghas	fckn
n*ggah	n*gguhs	f*ckn
n*ggaa	n*ggaz	f*ccin

n*gga	n*ggas	f*kn
n*gg	n*ggahs	f*ccn
n*ga	n*gas	fawking
n*g	n*ccas	f*ckin
n*cca		
j*gga		
j*gaboo		
coon		

Πίνακας 5: Διαφορετικές εκδοχές γραφής των λέξεων n*gger, n*ggers,f*cking

4.3 Ανάλυση των δεδομένων εισόδου

Σε αυτήν την ενότητα, παραθέτουμε μια ανάλυση των συνόλων δεδομένων μας, ως προς τη συχνότητα των λέξεων τους.

Στις εικόνες (**Εικόνα 9.1 και Εικόνα 9.2)** φαίνεται η κατανομή των λέξεων που αποτελούν το σύνολο δεδομένων Davidson και Founta, αντίστοιχα, βάσει της συχνότητας εμφάνισής τους. Όσο μεγαλύτερη απεικονίζεται μια λέξη, τόσο πιο συχνά εμφανίζεται στο σύνολο δεδομένων. Αντίστοιχα στους πίνακες (**Πίνακας 6.1, Πίνακας 6.2)** εμφανίζονται οι 10 πιο συχνά εμφανιζόμενες λέξεις για κάθε σύνολο δεδομένων.

Εικόνα 9.1: *Κατανομή των λέξεων που αποτελούν το σύνολο δεδομένων Davidson*

Εικόνα 9.2: *Κατανομή των λέξεων που αποτελούν το σύνολο δεδομένων Founta*

Πιο συχνές λέξεις	
Λέξη	**Αρ. Εμφάνισης**
b*tch	11439 (6.4%)
not	5802 (3.3%)
wh*re	4648 (2.6%)
n*gger	3221 (1.8%)
like	2929 (1.6%)
f*ck	2865 (1.6%)

Πίνακας 6.1: *Κατανομή των 10 πιο συχνά εμφανιζόμενων λέξεων στο σύνολο δεδομένων Davidson*

Πιο συχνές λέξεις	
Λέξη	**Αρ. Εμφάνισης**
f*ck	11960 (3.5%)
not	9149 (2.7%)
!	7493 (2.2%)
like	3500 (1%)
love	2572 (0.8%)
get	2100 (0.6%)

Πίνακας 6.2: *Κατανομή των 10 πιο συχνά εμφανιζόμενων λέξεων στο σύνολο δεδομένων Founta*

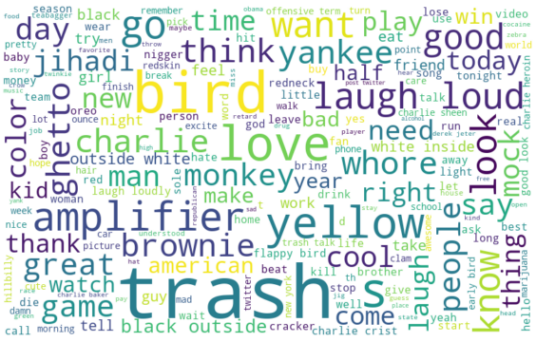
Παρακάτω, απεικονίζονται οι πιο συχνά εμφανιζόμενες λέξεις της κάθε κατηγορίας κλάσης. Για το σύνολο δεδομένων Davidson, η **Εικόνα 10.1** αφορά στην κλάση "Hate", η **Εικόνα 11.1** στην κλάση "Offensive" και τέλος η **Εικόνα 12.1** στην κλάση "Neither". Για το σύνολο δεδομένων Founta, η **Εικόνα 10.2** αφορά στην κλάση "Hateful", η **Εικόνα 11.2** στην κλάση "Abusive" και τέλος η **Εικόνα 12.2** στην κλάση "Normal".

Εικόνα 10.1:*Λέξεις όπως, n*gger, f*ggot και b*tch είναι οι πιο συχνά εμφανιζόμενες στην κατηγορία "Hate"*

Εικόνα 10.2:*Λέξεις όπως, n*gger, f*ck, h*te και id*ot είναι οι πιο συχνά εμφανιζόμενες στην κατηγορία "Hateful"*

Εικόνα 11.1:*Λέξεις όπως, b*tch, wh*re και n*gger είναι οι πιο συχνά εμφανιζόμενες στην κατηγορία "Offensive"*

Εικόνα 11.2:*Λέξεις όπως, f*ck, b*tch, και f*cking είναι οι πιο συχνά εμφανιζόμενες στην κατηγορία "Abusive"*



Εικόνα 12.1:*Λέξεις όπως, trash και bird είναι οι πιο συχνά εμφανιζόμενες στην κατηγορία "Neither"*

Εικόνα 12.2:*Λέξεις όπως, love και amplifier είναι οι πιο συχνά εμφανιζόμενες στην κατηγορία "Normal"*

Από τα παραπάνω "σύννεφα λέξεων", μπορούμε να διαπιστώσουμε, ότι δεν θα μπορούσαμε να κατατάξουμε ένα tweet μόνο με τον εντοπισμό μεμονωμένων λέξεων που εμφανίζονται σε μια κλάση. Υπάρχουν λέξεις όπως πχ. 'n*gger' και 'b*tch' που εμφανίζονται εξίσου στην κλάση 'Hate' και 'Offensive'. Επομένως, είναι απαραίτητο να χρησιμοποιηθούν τεχνικές που λαμβάνουν υπόψη και τα συμφραζόμενα, όπως, οι ενσωματώσεις λέξεων.

4.4 Σύγκριση των συνόλων δεδομένων.

Συγκρίναμε τα σύνολα δεδομένων μεταξύ τους, ως προς το περιεχόμενό τους, χρησιμοποιώντας τη βιβλιοθήκη DataComPy[28] η οποία εξάγει μια αναφορά σύγκρισης. Η αναφορά έδειξε ότι τα δύο σύνολα δεδομένων έχουν μόνο 47 κοινές εγγραφές και, το σημαντικότερο, ότι ίδιες προτάσεις είναι διαφορετικά χαρακτηρισμένες. Στον **Πίνακα 7**, φαίνονται μερικές από τις εν λόγω διαφορές.

Πρόταση	Davidson	Founta
H*te b*tch	abusive	hateful
F*ck retard	hateful	abusive
N*gger mad	abusive	hateful
H*te wh*re	abusive	hateful
N*gger watch f*ck h*te b*tch	abusive	hateful
N*gger f*ck	abusive	hateful

Πίνακας 7: *Διαφορές στον χαρακτηρισμό ίδιων προτάσεων στα δύο σύνολα δεδομένων.*

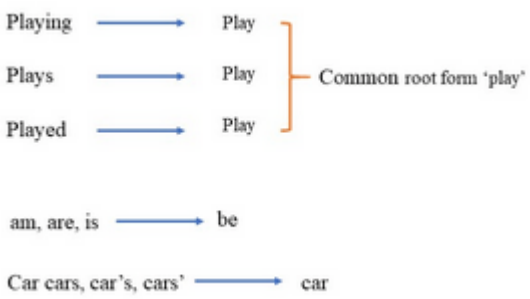
5 Πειράματα

Σε αυτήν την ενότητα περιγράφουμε, αναλυτικά, τα επικρατέστερα, σε απόδοση, πειράματα που εκτελέσαμε καθώς και την προ-επεξεργασία που εφαρμόσαμε πάνω στα δεδομένα εισόδου.

5.1 Προ-επεξεργασία των δεδομένων εισόδου

Χρησιμοποιήσαμε τα σύνολα δεδομένων που περιγράφονται στην Ενότητα 4.1. Οι δημοσιεύσεις των χρηστών στο Twitter παρ' όλο το μικρό μέγεθός τους περιέχουν πολλά περιττά στοιχεία. Προκειμένου να μετατρέψουμε τα tweets μας σε κατάλληλη μορφή ώστε να είναι κατανοητά από μηχανή, χρησιμοποιήσαμε την παρακάτω προ-επεξεργασία στα δεδομένα μας:

- Αφαιρέσαμε τις ετικέτες (#hastags), αναφορές (@) και διευθύνσεις διαδικτύου (http://)
- Επίσης, αφαιρέσαμε τους αριθμούς και τα σημεία στίξης εκτός από το '!', το οποίο αποδίδει νόημα στην πρόταση.
- Μετατρέψαμε όλους τους χαρακτήρες σε πεζούς ,
- Μετατρέψαμε επιμήκεις λέξεις στην κανονική μορφή τους. Πχ. 'yeeeees" σε 'yes'
- Μετατρέψαμε τις συντομογραφίες στην εκτενή μορφή τους ώστε να έχουμε μια ομοιομορφία στις χρησιμοποιούμενες λέξεις. πχ. "I'm" μετατροπή σε "I am"
- Μετατρέψαμε το κείμενο από ιδιωματική γλώσσα (αργκό) σε πιο λόγια μορφή. Λέξεις όπως 'lol' μετατράπηκαν σε "laugh out loud", "plz" σε "please" και "fmtkfytfo" σε "for me to know for you to find out"
- Εκτελέσαμε ορθογραφικό έλεγχο χρησιμοποιώντας τη βιβλιοθήκη Textblob[29]
- Αφαιρέσαμε από το κείμενό μας, λέξεις διακοπής (stopwords), με εξαίρεση τις λέξεις που υποδηλώνουν άρνηση (not, no, neither, no one, nobody, nothing, nowhere, least). Οι λέξεις διακοπής είναι λέξεις που εμφανίζονται πάρα πολλές φορές μέσα σε ένα κείμενο και δεν προσφέρουν στην κατηγοριοποίηση, ενώ η αφαίρεσή τους μειώνει το σύνολο των λέξεων του κειμένου.
  πχ = {'a', 'is', 'at', 'the', 'in'}
- Τέλος, εφαρμόσαμε Λεμματισμό (Lemmatizing), ώστε οι λέξεις μας να μετατραπούν στην αρχική μορφή τους με βάση τη μορφή της λέξης. πχ



Μερικά παραδείγματα του καθαρισμού των datasets φαίνονται στον **Πίνακα 8**

Αρχικό Tweet	Tweet μετά από την προ-επεξεργασία
@haylgroot .....bitch 😒 yeah it was me... Caught me red handed, sorry @jennholly	b*tch yeah catch red hand sorry
Dawg oriental flavored noodles go so f*ckin hard b RT @J_Dun54: @ItsNotHarold Lmfaoo! F*ckin coon. Smh	dude oriental flavor noodle f*ck hard laugh fat as ! f*cking n*gger shake head

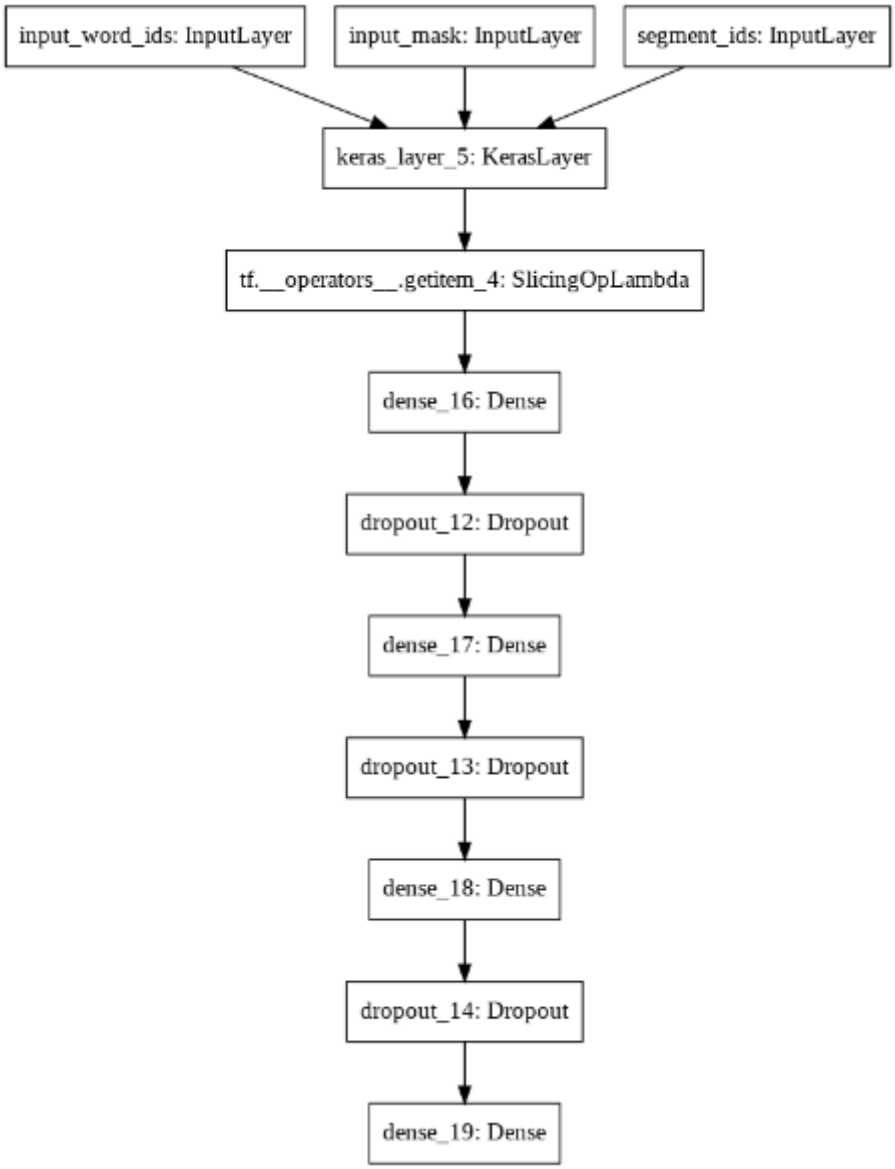
Πίνακας 8: *Παραδείγματα καθαρισμού των datasets*

5.2 Τα επικρατέστερα μοντέλα

Σε αυτήν την ενότητα περιγράφουμε το επικρατέστερο, σε απόδοση, μοντέλο που υλοποιήσαμε. Στα δύο σύνολα δεδομένων που εξετάσαμε, η καλύτερη απόδοση επιτεύχθηκε με το ίδιο μοντέλο Bert.

5.2.1 Υλοποίηση Bert στα σύνολα δεδομένων Davidson και Founta

Το επικρατέστερο σε απόδοση μοντέλο, είναι κοινό για τα σύνολα δεδομένων Davidson και Founta. Χρησιμοποιεί προ-εκπαιδευμένες ενσωματώσεις Bert $_{base}$, όπως αυτές περιγράφονται στην Ενότητα 6.3, και τρία πλήρως συνδεδεμένα στρώματα 128 νευρώνων, με συναρτήσεις ενεργοποίησης την ReLu. Ανάμεσα τους υπάρχουν στρώματα Dropout κατά τα οποία αγνοούνται τυχαίοι νευρώνες με ποσοστό 10% . Στο στρώμα εξόδου χρησιμοποιείται η συνάρτηση Softmax. Στην **Εικόνα 13** φαίνεται η σχηματική αναπαράσταση του μοντέλου.



***Εικόνα 13**: Σχηματική αναπαράσταση μοντέλου $BERT_{base}$*

6 *Αξιολόγηση*

Σε αυτήν την ενότητα παρουσιάζουμε τα πειράματα αξιολόγησης των τεχνικών που χρησιμοποιήσαμε.

6.1 Παράμετροι αξιολόγησης

Αξιολογούμε την απόδοση κάθε προσέγγισης χρησιμοποιώντας τη μετρική weighted-averaged F_1 score. Θεωρήσαμε ότι η παραπάνω μετρική προσφέρει μια καλή επισκόπηση της απόδοσης των προσεγγίσεών μας μιας και τα δύο σύνολα δεδομένων που χρησιμοποιούμε παρουσιάζουν άνιση κατανομή των κλάσεων τους.

6.2 Οργάνωση πειραμάτων

Για την υλοποίηση των πειραμάτων μας χρησιμοποιήσαμε Python η οποία είναι η πιο δημοφιλής Γλώσσα προγραμματισμού μηχανικής μάθησης. Η Python προσφέρει μια μεγάλη ποικιλία βιβλιοθηκών και έτοιμων προ-εκπαιδευμένων μοντέλων που διευκολύνουν την ανάπτυξη εφαρμογών μηχανικής μάθησης. Οι πιο δημοφιλείς από αυτές που χρησιμοποιήσαμε είναι η Scikit-learn[30] και το Keras[31] με σύστημα υποστήριξης Tensorflow (backend). Επιπλέον, η υλοποίηση του Bert βασίστηκε και στην βιβλιοθήκη Huggingface[29].

Παρακάτω περιγράφονται λεπτομερώς τα στοιχεία που χρησιμοποιήσαμε στα πειράματά μας.

α) Προ-εκπαιδευμένες ενσωματώσεις λέξεων

Στα πειράματά μας χρησιμοποιήσαμε προ-εκπαιδευμένες ενσωματώσεις τόσο για το GloVe όσο και για το fastText και Bert.

Για το GloVe, χρησιμοποιήσαμε διανύσματα λέξεων που εκπαιδεύτηκαν σε 2 δισεκατομμύρια tweets με κάθε λέξη να είναι ένα διάνυσμα μεγέθους 200.

Για το fastText, χρησιμοποιήσαμε 1 εκατομμύριο διανύσματα λέξεων εκπαιδευμένα σε διαδικτυακά κείμενα της Wikipedia 2017, του Πανεπιστημίου του Μέριλαντ (University of Maryland, Baltimore County, UMBC) και σε ένα σύνολο δεδομένων ειδήσεων από το Statistical Machine Translation, statmt.org[32]), με κάθε λέξη να είναι ένα διάνυσμα μεγέθους 300.

Τέλος, για το Bert, χρησιμοποιήσαμε ένα προ-εκπαιδευμένο μοντέλο που αποτελείται από 12 επίπεδα κωδικοποιητών (L), 768 κρυφούς κόμβους (H), 12 κεφαλές (A) και σύνολο 110 εκατομμύρια παραμέτρους. Τα δεδομένα προέρχονται από ένα μεγάλο σύνολο κειμένων της Wikipedia και του BookCorpus[33], ενός συνόλου δεδομένων με +10,000 βιβλία διάφορων θεμάτων .

β) Σύνολο Δεδομένων

Χρησιμοποιήσαμε τα σύνολα δεδομένων που περιγράφονται στην Ενότητα 4.1. Στη συνέχεια χρησιμοποιήσαμε την προ-επεξεργασία που περιγράφεται στην Ενότητα 5.1 κατά την υλοποίηση των μοντέλων GloVe, fastText και Bert. Ειδικότερα στην περίπτωση του Bert, επαναλάβαμε την παραπάνω προ-επεξεργασία χωρίς να αφαιρέσουμε τα stopwords και χωρίς να εφαρμόσουμε Λεμματισμό. Στην αρχή κάθε πειράματος χωρίσαμε το σύνολο δεδομένων στα δύο σε ποσοστό 70-30, εξασφαλίζοντας τον ίσο καταμερισμό των κλάσεων. Το 70% χρησιμοποιήθηκε για την εκπαίδευση (train) του μοντέλου μας και το 30% για τον έλεγχό του (test).

Στο 70% εφαρμόσαμε Stratified 10-Fold Cross Validation για GloVe, fastText και Stratified 5-Fold Cross Validation για Bert, ώστε να κρατήσουμε ένα σύνολο δεδομένων με ίσο καταμερισμό των κλάσεων, κρατώντας από το test set ένα 20% για validation κατά την εκπαίδευση και υπολογίζοντας weighted-average f_1 score, βρίσκοντας τον μέσο όρο του μετά το πέρας των Fold.

γ) Ρύθμιση υπερ-παραμέτρων

Σε όλα μας τα πειράματα εφαρμόσαμε, early-stopping παρακολουθώντας την μείωση της απώλειας κατά την επικύρωση του μοντέλου (validation loss), ώστε να αποφύγουμε το overfitting, με patience 5 για όλα τα μοντέλα. Επίσης χρησιμοποιήσαμε κατηγορηματική εγκάρσια εντροπία (categorical cross entropy), ως συνάρτηση απώλειας και τον βελτιστοποιητή (Optimizer) Adam, για το Glove, με ρυθμό μάθησης 2e -4 και 3e-5 για το Bert ενώ για fastText χρησιμοποιήσαμε τον βελτιστοποιητή RMSprop με ρυθμό μάθησης 3e -5. Ορίσαμε batch size 512 για GloVe και fastText και 32 για Bert. Το batch size είναι ο αριθμός των δειγμάτων (samples) που εξετάζονται σε μια εποχή (epoch) πριν ενημερωθούν τα βάρη. Στο Glove και fastText, ο πίνακας δεικτών, θα συμπεριλάβει τις 20,000 πιο συχνές λέξεις του συνόλου δεδομένων, ενώ το μέγιστο μήκος της αναπαράστασής τους (Max Length) θα είναι 180, σε κάθε περίπτωση, μεγαλύτερου ή μικρότερου μήκους, αυτό θα περικοπεί ή θα γεμίσει με μηδενικά, αντίστοιχα. Στην περίπτωση του Bert, το μήκος κάθε αναπαράστασης λέξης θα

είναι 90. Τα παραπάνω φαίνονται στον **Πίνακα 9**. Οι υπόλοιπες, επιμέρους, υπερ-παράμετροι που χρησιμοποιήσαμε αναλύονται στην Ενότητα 7.1

	Patience	**Categorical Cross entropy**	**Optimizer**	**Learning Rate**	**Batch size**	**Max Length**
GloVe	5	√	Adam	2e-4	512	180
fastText	5	√	RMSprop	3e-5	512	180
Bert	5	√	Adam	3e-5	32	90

Πίνακας 9*: Βασικοί υπερ-παράμετροι GloVe, fastText,Bert*

δ) Γενίκευση συνόλων δεδομένων

Μετά την ολοκλήρωση των πειραμάτων μας για την εύρεση του επικρατέστερου σε απόδοση μοντέλου, διενεργήσαμε μια νέα σειρά πειραμάτων με σκοπό να εξετάσουμε κατά πόσο ένα μοντέλο που έχει εκπαιδευτεί σε ένα σύνολο δεδομένων και παρουσιάζει λογικά αποτελέσματα, μπορεί να παρουσιάζει συγκριτικά καλά αποτελέσματα σε άλλα σύνολα δεδομένων, δηλαδή αν γενικεύει. Για τον σκοπό αυτό, διαλέξαμε τα δύο καλύτερα σε απόδοση μοντέλα μας, ένα για κάθε σύνολο δεδομένων. Στη συνέχεια, χρησιμοποιήσαμε το εκπαιδευμένο μοντέλο του ενός συνόλου δεδομένων για να προβλέψουμε τα δεδομένα του άλλου συνόλου δεδομένων και συγκρίναμε τα αποτελέσματα τους.

6.3 Αποτελέσματα

Στους πίνακες (**Πίνακας 10** και **Πίνακας 11)** παρουσιάζονται όλα τα αποτελέσματα των πειραμάτων μας ανά σύνολο δεδομένων.

Συγκεντρωτικός πίνακας πειραμάτων (Davidson dataset)	
GloVe	**f1- weighted**
BLSTM + LSTM +GloVe	91,63
GRU + GloVe	91,444
BGRU + GRU + GloVe	91,393
Dense + Glove	91,273
BGRU+BLSTM + GloVe	91,189
LSTM + GloVe	91,177
BLSTM + BGRU + GloVe	91,16
BGRU + GloVe	91,136

BLSTM + GloVe	91,074
BLSTM + BGRU + GloVe + MultiHead Attention	90,844
Conv2d+GloVe	90,548
fastText	**f1- weighted**
LSTM + fastText	91,79
BGRU + fastText	91,714
Dense + fastText	91,682
BGRU+BLSTM + fastText	91,63
BGRU + GRU + fastText	91,624
BLSTM + BGRU + fastText	91,541
GRU + fastText	91,541
BLSTM + LSTM + fastText	91,467
Conv2d + fastText	91,388
BLSTM + fastText	91,318
BLSTM + BGRU + fastText + MultiHead Attention	91,193
Bert	**f1- weighted**
Bert + 3*Dense	**92,622**
Bert + CNN	92,054
Bert + BGRU	91,967
Bert + Dense + BGRU+ Dense	91,33

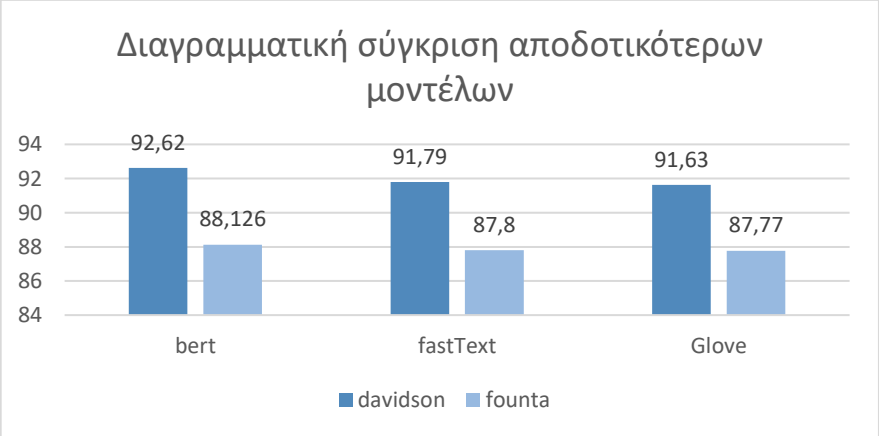
Πίνακας 10: *Συγκεντρωτικός πίνακας πειραμάτων στο dataset του Davidson*

Συγκεντρωτικός πίνακας πειραμάτων (Founta dataset)	
GloVe	**f1- weighted**
BGRU + GRU + GloVe	87,779
BGRU + GloVe	87,772
GRU + GloVe	87,668
BLSTM + BGRU + GloVe	87,534
BLSTM + BGRU + GloVe + MultiHead Attention	87,534
Dense + Glove	87,314
LSTM + GloVe	87,13
BLSTM + GloVe	87,089
BGRU + BLSTM + GloVe	86,883
BLSTM + LSTM + GloVe	86,093
Conv2d + GloVe	85,297
fastText	**f1- weighted**
BLSTM + BGRU + fastText + MultiHead Attention	87,80

GRU + fastText	87,692
BLSTM + BGRU + fastText	87,614
BGRU + fastText	87,582
BLSTM + LSTM + fastText	87,459
BGRU + GRU + fastText	87,449
LSTM + fastText	87,435
Conv2d + fastText	87,363
BGRU + BLSTM + fastText	87,239
Dense + fastText	87,238
BLSTM + fastText	87,007
Bert	**f1- weighted**
Bert + 3*Dense	**88.126**
Bert + Dense + GRU + Dense	87,86
Bert	87,555
Bert + BGRU	87,483
Bert +CNN	87,195

Πίνακας 11: *Συγκεντρωτικός πίνακας πειραμάτων στο dataset της Founta*

Όπως παρατηρούμε από τους παραπάνω συγκεντρωτικούς πίνακες πειραμάτων, και στα δύο σύνολα δεδομένων, τα μοντέλα που παρουσιάζουν την καλύτερη απόδοση είναι αυτά που χρησιμοποιούν τις προ-εκπαιδευμένες ενσωματώσεις Bert. Τα μοντέλα μας έχουν απόδοση **92,62%** για το σύνολο δεδομένων Davidson και **88.126%** για το σύνολο δεδομένων Founta. Ακολουθούν τα μοντέλα που χρησιμοποιούν προ-εκπαιδευμένες ενσωματώσεις fastText με **91,79%** και **87,8%** αντίστοιχα, ενώ τελευταία έρχονται τα μοντέλα που χρησιμοποιούν προ-εκπαιδευμένες ενσωματώσεις Glove, με **91,63%** και **87,77%** για κάθε σύνολο δεδομένων. Στο **Διάγραμμα 1** φαίνονται τα παραπάνω.



Διάγραμμα 1: Διαγραμματική σύγκριση αποδοτικότερων μοντέλων σε κάθε κατηγορία.

Τέλος, τα πειράματά μας, σε σχέση με το αν τα μοντέλα που εκπαιδεύσαμε έχουν ακρίβεια γενίκευσης σε διαφορετικά μοντέλα δεδομένων, φαίνονται στον **Πίνακα 12**.

	Davidson	Founta
Davidson	**0,92**	0,44
Founta	0,31	**0,88**

Πίνακας 12: *Οι σειρές αφορούν στα σύνολα δεδομένων εκπαίδευσης και οι στήλες στα σύνολα δεδομένων δοκιμής*

Στον παραπάνω πίνακα φαίνεται πως κανένα από τα δύο σύνολα δεδομένων δεν έχει καλή απόδοση όταν δοκιμάζεται σε διαφορετικό σύνολο δεδομένων από αυτό στο οποίο έχει εκπαιδευτεί. Η επίδοση, μετρούμενη σε f1-score, στο σύνολο δεδομένων Davidson έπεσε κατά **52%**, και αντίστοιχα, στο Founta κατά **64%**, υποδεικνύοντας ότι τα μοντέλα έχουν μικρή απόδοση γενίκευσης.

6.4 Σύνοψη συμπερασμάτων αξιολόγησης

Μπορούμε να συμπεράνουμε ότι τα μοντέλα LSTM και GRU υπερτερούν, σε απόδοση, σε σχέση με τα μοντέλα που χρησιμοποιούν CNN, με την αρχιτεκτονική GRU να υπερτερεί σε σχέση με την αρχιτεκτονική LSTM. Τέλος, από τα αποτελέσματα, φαίνεται ότι οι αλγόριθμοι αποδίδουν διαφορετικά σε κάθε σύνολο δεδομένων. Για παράδειγμα, το πιο αποδοτικό μοντέλο με χρήση GloVe στο σύνολο δεδομένων Davidson χρησιμοποιεί αρχιτεκτονική Bidirectional LSTM και LSTM ενώ η ίδια αρχιτεκτονική έρχεται προτελευταία στο σύνολο δεδομένων Founta, με την αρχιτεκτονική Bidirectional GRU και GRU να έχει την καλύτερη απόδοση. Εξαίρεση αποτελεί το επικρατέστερο σε απόδοση μοντέλο που χρησιμοποιεί προ-εκπαιδευμένες ενσωματώσεις Bert, όπου ο αλγόριθμος είναι κοινός και για τα δύο σύνολα δεδομένων. Στο σύνολο δεδομένων Davidson, το επικρατέστερο μοντέλο μας, ήταν κατώτερο σε απόδοση, από όλα τα μοντέλα που μελετήσαμε. Για παράδειγμα οι [13] υλοποιώντας Bert είχαν απόδοση F1-macro **77.27%** ενώ η δική μας απόδοση σε F1-macro ήταν **65.75%.** Όμως, οι [13] στο σύνολο δεδομένων Founta, είχαν κατώτερη απόδοση από τη δική μας, με F1-macro **69.60%** έναντι F1-macro **76.334%**.

Τέλος, σύμφωνα με τα αποτελέσματα της γενίκευσης των συνόλων δεδομένων, συμπεράναμε ότι τα σύνολα δεδομένων έχουν μικρή απόδοση γενίκευσης σε διαφορετικά σύνολα δεδομένων. Σύμφωνα, με την σύγκριση που διενεργήσαμε στην **Ενότητα 4.4,** η παραπάνω μείωση της απόδοσης, οφείλεται στην ανομοιογένεια των δεδομένων που αποτελούν τα δύο σύνολα δεδομένων, καθώς και στη διαφορετικό τρόπο χαρακτηρισμού κοινών προτάσεων από τους σχολιαστές (annotators).

7 Τεχνικές λεπτομέρειες

Στις επόμενες ενότητες παραθέτουμε λεπτομέρειες υλοποίησης καθώς και τεχνικά χαρακτηριστικά των πλατφόρμων και προγραμμάτων που χρησιμοποιήθηκαν στην υλοποίηση αυτής εργασίας.

7.1 Λεπτομέρειες υλοποίησης

Στο Παράρτημα Ι παραθέτουμε χαρακτηριστικά κομμάτια κώδικα από το στάδιο της προ-επεξεργασίας των δεδομένων μας ενώ στο Παράρτημα ΙΙ περιγράφεται ο κώδικας του καλύτερου σε απόδοση μοντέλου με χρήση Bert. Ειδικά στην περίπτωση του Bert, αντιμετωπίσαμε αρκετές δυσκολίες όσον αφορά την ρύθμιση των υπερ-παραμέτρων μας. Αναγκαστήκαμε να ορίσουμε τις χαμηλότερες δυνατές τιμές ώστε ο αλγόριθμός μας να μπορεί να εκτελεστεί από την πλατφόρμα Colab και να μην υπερβούμε το όριο των διαθέσιμων υπολογιστικών πόρων που παραχωρούνται, ανά χρήστη, από την πλατφόρμα.

7.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Τα πειράματά μας αναπτύχθηκαν και εκτελέστηκαν στις πλατφόρμες Google Colaboratory[24], Kaggle[24] οι οποίες προσφέρουν, δωρεάν, ένα Jupyter Notebook και στον υπολογιστή που μας παραχωρήθηκε από τον κ. Κωνσταντίνο Διαμαντάρα με τα ακόλουθα χαρακτηριστικά:

- Nvidia Titan Xp (12GB, 3800 cores)
- Μνήμη 16GB
- CPU Intel i5

Η προ-επεξεργασία του συνόλου δεδομένων μας αναπτύχθηκε και εκτελέστηκε τοπικά σε ένα Εικονικό Μηχάνημα (Virtual Machine,VM) με τα παρακάτω χαρακτηριστικά:

- Hypervisor: MS Hyper V

- RAM: 8GB
- HD: 4GB
- OS: UBUNTU SERVER 20.04.01 LTS

Το Εικονικό Μηχάνημα εγκαταστάθηκε στην προαναφερόμενη πλατφόρμα Hyper V της Microsoft σε έναν υπολογιστή με τα παρακάτω χαρακτηριστικά:

- CPU: IntelI5-6400T
- RAM:12GB
- HD1:256GB NVMe
- HD2:1TB SATA3
- OS: Windows 10 pro x64 v.20H2

Η πλατφόρμα Colab είναι ένα περιβάλλον ανάπτυξης εφαρμογών Python που «τρέχει» στο πρόγραμμα περιήγησης χρησιμοποιώντας το Google Cloud. Δεν απαιτεί καμία ρύθμιση για χρήση, ενώ παρέχει δωρεάν πρόσβαση σε υπολογιστικούς πόρους συμπεριλαμβανομένων των GPU και TPU. Η μέγιστη διάρκεια χρήσης ανά σύνοδο (session) είναι οι 12 ώρες.
Η πλατφόρμα Kaggle παρέχει ένα υπολογιστικό περιβάλλον συγγραφής κώδικα Python ή R, που τρέχει στο πρόγραμμα περιήγησης και μας παρέχει GPU, TPU. Κάθε σύνοδος έχει μέγιστη διάρκεια 9 ώρες. Η CPU που μας παρέχεται έχει 4 πυρήνες και 16GB RAM, αντίστοιχα η GPU έχει 2 πυρήνες και 13 GB RAM, ενώ η TPU έχει 4 πυρήνες και 16 GB RAM.
Και οι δύο πλατφόρμες που αναφέρθηκαν παραπάνω μας δίνουν την δυνατότητα συνεργατικής συγγραφής κώδικα. Επίσης έχουν προ-εγκατεστημένες πολλές απαραίτητες βιβλιοθήκες Μηχανικής Μάθησης (Machine Learning).

8 *Επίλογος*

Στο κεφάλαιο αυτό παρουσιάζονται συνοπτικά τα συμπεράσματα της μελέτης μας και δίνονται προτάσεις για μελλοντικές επεκτάσεις.

8.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία, μελετήσαμε τρία διαφορετικά μοντέλα ενσωματώσεων λέξεων, σε συνδυασμό με πληθώρα αλγορίθμων ταξινόμησης, με σκοπό να αναδείξουμε το μοντέλο με την καλύτερη απόδοση. Η έρευνά μας, έδειξε ότι το καλύτερο, σε απόδοση, μοντέλο είναι αυτό που χρησιμοποιεί ενσωματώσεις Bert ακολουθούμενο από τις ενσωματώσεις fastText και GloVe.

Στο σύνολο δεδομένων Davidson, το επικρατέστερο μοντέλο μας, ήταν κατώτερο σε απόδοση, από όλα τα μοντέλα που μελετήσαμε. Για παράδειγμα οι [13] υλοποιώντας Bert είχαν απόδοση F1-macro **77.27%** ενώ η δική μας απόδοση σε F1-macro ήταν **65.75%**. Όμως, στο σύνολο δεδομένων Founta, η μελέτη [13] είχε κατώτερη απόδοση από την δική μας με F1-macro **69.60%** έναντι F1-macro **76.334%.**

Επίσης, καταλήξαμε ότι οι αλγόριθμοι αποδίδουν διαφορετικά σε κάθε σύνολο δεδομένων, όταν χρησιμοποιούνται ενσωματώσεις fastText και GloVe σε αντίθεση με τη χρήση ενσωματώσεων Bert όπου το αποδοτικότερο μοντέλο είναι κοινό. Συνοψίζοντας, παρατηρήσαμε ότι όταν ένα μοντέλο που εκπαιδεύεται σε ένα σύνολο δεδομένων προσπαθεί να προβλέψει τα δεδομένα ενός άλλου συνόλου δεδομένων, η απόδοσή του είναι αρκετά χαμηλότερη είτε λόγω της ανομοιογένειας των δεδομένων που αποτελούν τα δύο σύνολα δεδομένων είτε λόγω του διαφορετικού τρόπου χαρακτηρισμού κοινών προτάσεων τους από τους σχολιαστές (annotators).

8.2 Μελλοντικές επεκτάσεις

Σε μελλοντική έρευνα, θα ήταν σκόπιμο να μελετήσουμε, επιπλέον τεχνικές με σκοπό να εμπλουτίσουμε το αποδοτικότερο μοντέλο μας, ώστε να δημιουργήσουμε ένα νέο μοντέλο που θα υπερτερεί σε απόδοση από αυτά που ερευνήσαμε.

9 *Βιβλιογραφία*

[1] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," no. Icwsm, pp. 512–515, 2017.

[2] A. Gaydhani, V. Doma, S. Kendre, and L. Bhagwat, "Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach," 2018, [Online]. Available: http://arxiv.org/abs/1809.08651.

[3] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," *PLoS One*, vol. 14, no. 8, p. e0221152, 2019, doi: 10.1371/journal.pone.0221152.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space." Accessed: Jun. 24, 2021. [Online]. Available: http://ronan.collobert.com/senna/.

[5] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information." [Online]. Available: http://www.isthe.com/chongo/tech/comp/fnv.

[6] J. Pennington, R. Socher, and C. D. Manning, "GloVe : Global Vectors for Word Representation."

[7] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep Learning for Hate Speech Detection in Tweets," no. 2.

[8] A. Abuzayed and T. Elsayed, "Quick and Simple Approach for Detecting Hate Speech in Arabic Tweets," *Proc. 4th Work. Open-Source Arab. Corpora Process. Tools, with a Shar. Task Offensive Lang. Detect.*, no. May, pp. 109–114, 2020, [Online]. Available: https://www.aclweb.org/anthology/2020.osact-1.18.

[9] S. Biere and M. B. Analytics, "Hate Speech Detection Using Natural Language Processing Techniques," 2018.

[10] M. Mozafari and R. Farahbakhsh, "Hate Speech Detection and Racial Bias Mitigation in Social Media based on BERT model," pp. 1–30, 2020.

[11] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." [Online]. Available: https://github.com/tensorflow/tensor2tensor.

[12] R. T. Mutanga, N. Naicker, and O. O. Olugbara, "Hate speech detection in twitter using transformer methods," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 9, pp. 614–620, 2020, doi: 10.14569/IJACSA.2020.0110972.

[13] S. D. Swamy, A. Jamatia, and B. Gambäck, "Studying generalisability across abusive language detection datasets," *CoNLL 2019 - 23rd Conf. Comput. Nat. Lang. Learn. Proc. Conf.*, pp. 940–950, 2019, doi:

10.18653/v1/k19-1088.
[14] “Οικουμενικη διακηρυξη για τα ανθρωπινα δικαιωματα 10,” 1948.
[15] “Ευρωπαϊκή Σύμβαση Δικαιωμάτων του Ανθρώπου Ευρωπαϊκή Σύμβαση Δικαιωμάτων του Ανθρώπου.”
[16] Ευρωπαϊκή Επιτροπή, “ΚΩΔΙΚΑΣ ΣΥΜΠΕΡΙΦΟΡΑΣ ΓΙΑ ΤΗΝ ΚΑΤΑΠΟΛΕΜΗΣΗ ΤΗΣ ΠΑΡΑΝΟΜΗΣ ΡΗΤΟΡΙΚΗΣ ΤΟΥ ΜΙΣΟΥΣ ΣΤΟ ΔΙΑΔΙΚΤΥΟ,” pp. 1–4, 2016.
[17] B. Vidgen, R. Tromble, A. Harris, S. Hale, and H. Margetts, “Challenges and frontiers in abusive content detection,” no. Section 2, pp. 80–93, 2019.
[18] T. Harvard, “Neural Network Models for Hate Speech Classification in Tweets Neural Network Models for Hate.”
[19] Y. Kim, “Convolutional neural networks for sentence classification,” *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, pp. 1746–1751, 2014, doi: 10.3115/v1/d14-1181.
[20] K. Cho *et al.*, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, pp. 1724–1734, 2014, doi: 10.3115/v1/d14-1179.
[21] R. C. Staudemeyer and E. R. Morris, “Understanding LSTM – A tutorial into Long Short-Term Memory Recurrent Neural Networks,” *arXiv*, pp. 1–42, 2019.
[22] G. Singhal, “Introduction to LSTM Units in RNN | Pluralsight.” https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn (accessed Jan. 23, 2021).
[23] G. Singhal, “LSTM versus GRU Units in RNN | Pluralsight.” https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn (accessed Jan. 23, 2021).
[24] N. Joselson, “and Bi-Directional LSTM models for use with Twitter conversations) Emotion Classi cation with Natural Language Processing (Comparing BERT and Bi-Directional LSTM models for use with Twitter conversations),” no. June 2019, 2020, doi: 10.13140/RG.2.2.16482.27844.
[25] 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, no. Mlm, pp. 4171–4186, 2019.
[26] “Hate Speech Identification - dataset by crowdflower | data.world.” https://data.world/crowdflower/hate-speech-identification (accessed Jan. 23, 2021).
[27] A. M. Founta *et al.*, “Large scale crowdsourcing and characterization of twitter abusive behavior,” *12th Int. AAAI Conf. Web Soc. Media, ICWSM 2018*, pp. 491–500, 2018, doi: 10.5281/zenodo.1443348.
[28] “DataComPy — datacompy 0.7.2 documentation.” https://capitalone.github.io/datacompy/ (accessed May 31, 2021).
[29] “Philosophy — transformers 4.2.0 documentation.” https://huggingface.co/transformers/philosophy.html (accessed Jan. 24, 2021).

[30] “scikit-learn: machine learning in Python — scikit-learn 0.24.1 documentation.” https://scikit-learn.org/stable/ (accessed Jan. 24, 2021).
[31] “Keras: the Python deep learning API.” https://keras.io/ (accessed Jan. 24, 2021).
[32] “Statistical Machine Translation.” http://statmt.org/ (accessed Jan. 23, 2021).
[33] Y. Zhu *et al.*, “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 19–27, 2015, doi: 10.1109/ICCV.2015.11.

Παράρτημα Ι

Προ-επεξεργασία συνόλων δεδομένων

Στην **Ενότητα 5.1** περιγράψαμε τα βήματα που ακολουθήσαμε κατά την προ-επεξεργασία των συνόλων δεδομένων μας. Παρακάτω παραθέτουμε αναλυτικά την υλοποίηση σε κώδικα Python. Στο **Παράρτημα ΙΙΙ**, παραθέτουμε λεπτομερή παρουσίαση της οθόνης εξόδου των 10 πρώτων εγγραφών του συνόλου δεδομένων Davidson, σε κάθε βήμα εκτέλεσης του προγράμματος.

Αφού φορτώσουμε όλες τις απαραίτητες βιβλιοθήκες, εκτελούμε την παρακάτω εντολή η οποία έχει ως αποτέλεσμα την αύξηση του πλάτους στήλης ώστε να μπορούμε να βλέπουμε ολόκληρα τα tweets στην οθόνη εξόδου του προγράμματος.

```
pd.options.display.max_colwidth = 200
```

Στη συνάρτηση που περιγράφεται παρακάτω, δηλώνουμε όλα τα πιθανά emoticons, ώστε να μπορέσουμε να τα διαγράψουμε στη συνέχεια

```
def removeEmoticons(text):
     text = re.sub(':\)|;\)|:-\)|\(-:|:-D|=D|:P|xD|X-p|\^\^|:-
*|\^\.\^|\^\-\^|\^\_\^|\,-\)|\)-:|:\'\(|:\(|:-
\(|:\S|T\.T|\.\_\.|:<|:-\S|:-<|\*\-\*|:O|=O|=\-O|O\.o|XO|O\_O|:-
\@|=/|:/|X\-\(|>\.<|>=\(|D:', '', text)
    return text
```

Η παρακάτω συνάρτηση μετατρέπει στην κανονική μορφή τους, τις λέξεις που περιέχουν περισσότερους από 2 συνεχόμενους χαρακτήρες, εκτός και αν εντοπιστούν στο λεξικό `wordnet`.

```
def replaceElongated(word):
    repeat_regexp = re.compile(r'(\w)(\1{2,})')
    repl = r'\1'
    if wordnet.synsets(word):
        return word
    repl_word = repeat_regexp.sub(repl, word)
    if repl_word != word:
        return replaceElongated(repl_word)
```

```
        else:
            return repl_word
```

Διαβάζουμε το σύνολο δεδομένων που θα επεξεργαστούμε και στη συνέχεια δημιουργούμε μια νέα στήλη όπου θα αποθηκεύσουμε τα "καθαρά" tweets, την οποία ονομάσαμε "tidy_tweet"

```
#import dataset
filename = 'hatedataset3.csv'
names=['column_a','count','hate_speech','offensive_language','ne
ither','class','tweet']
data=pd.read_csv(filename,names=names,sep=";")
print(data.shape)

# create a new column
data['tidy_tweet'] = data['tweet']
```

Στον παρακάτω κώδικα, αφαιρούμε τους επόμενους χαρακτήρες "#,@,http,&,RT" μαζί με το κείμενο που τους ακολουθεί. Τέλος αφαιρούμε τους αριθμούς. Για να πετύχουμε τα παραπάνω, χρησιμοποιούμε την εντολή replace η οποία αντικαθιστά μια συμβολοσειρά (string) με μια άλλη.

```
print("before")
print (data['tidy_tweet'].head(10))
print("\n"*2)
print("removing @,# etc.....")
# remove @,#,http,&
data['tidy_tweet']=data['tidy_tweet'].str.replace(r'#\w+ ?'," ")
data['tidy_tweet']=data['tidy_tweet'].str.replace(r'@[^\s]+',"
")
data['tidy_tweet']=data['tidy_tweet'].str.replace(r'http:\S+',"
")
data['tidy_tweet']=data['tidy_tweet'].str.replace(r'&\w+ ;', "
")
data['tidy_tweet']=data['tidy_tweet'].str.replace(r'RT'," ")
#remove numbers
data['tidy_tweet']=data['tidy_tweet'].str.replace("[0-9]", " ")
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Χρησιμοποιούμε τη συνάρτηση apply η οποία καλεί τη συνάρτηση `lambda` και την εφαρμόζει σε κάθε σειρά του συνόλου δεδομένων επιστρέφοντας την στήλη `data['tidy_tweet']` χωρίς να περιέχει emoticons.

```
print("removing emoticons...")
#replace emoticons
data['tidy_tweet'] = data['tidy_tweet'].apply(lambda x:
    removeEmoticons(x))
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Χρησιμοποιώντας την ίδια τεχνική με παραπάνω, μετατρέπουμε το κείμενό μας σε πεζούς χαρακτήρες

```
print("lowercasing...")
#lower case
data['tidy_tweet'] = data['tidy_tweet'].apply(lambda x:
    " ".join(x.lower() for x in x.split()))
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Χρησιμοποιούμε την παραπάνω τεχνική και την συνάρτηση `replaceElongated`, ώστε να μετατρέψουμε τις λέξεις με παραπάνω από δύο συνεχόμενους χαρακτήρες, σε κανονικές.

```
print("replacing elongated words...")
#replace elongated
data['tidy_tweet'] = data['tidy_tweet'].apply(lambda x:
    replaceElongated(x))
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Το παρακάτω τμήμα κώδικα φέρνει μεγάλες αλλαγές στα δεδομένα μας, μιας και αντικαθιστά λέξεις που υπάρχουν μέσα στο αρχείο slang3.txt με λέξεις που υπάρχουν μέσα στο σύνολο δεδομένων, χρησιμοποιώντας regular expressions. Με τη χρήση τους μπορούμε να ελέγξουμε και να ταιριάξουμε οποιονδήποτε ακολουθία χαρακτήρων με μια συμβολοσειρά.

- `abbr_dict = dict(csvfile)`-> δημιουργεί ένα λεξικό λέξεων από το αρχείο slang3.txt.

- `abbr_dict= {r'(\b){}(\b)'.format(k):r'\1{}\2'.format(v)` -> μετατρέπει το λεξικό στην παρακάτω μορφή, ώστε να είναι δυνατή η εύρεση το 1ου όρου στο σύνολο δεδομένων και η αντικατάστασή του με τον 2ο όρο. Η εύρεση και αντικατάσταση γίνεται με την εντολή `replace`

```
πχ abbr_dict
    {'(\\b)nigguh(\\b)': '\\1nigger\\2',
     '(\\b)juss(\\b)': '\\1just\\2',
     '(\\b)afaik(\\b)': '\\1as far as i know\\2',
     '(\\b)lmo(\\b)': '\\1leave me alone\\2',
```

```
print('removing slang and correcting abbreviations.....')
#remove slang
filename = "slang3.txt"
csvfile =
csv.reader(open(filename),delimiter=',',quoting=csv.QUOTE_NONE)
abbr_dict = dict(csvfile)
abbr_dict = {r'(\b){}(\b)'.format(k):r'\1{}\2'.format(v) for k,v
in abbr_dict.items()}
data['tidy_tweet'] = data['tidy_tweet'].astype(str)
data['tidy_tweet'].replace(abbr_dict,regex=True,inplace=True)
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Αφαιρούμε όλα τα σημεία στίξης εκτός από το '!'', το οποίο θεωρούμε ότι προσθέτει νόημα σε μία πρόταση.

```
print('removing punctuations.....')
#remove punctuations
data['tidy_tweet'] =
data['tidy_tweet'].str.replace('[^\w\s\!]'," ")
#remove duplicate !
data['tidy_tweet'] = data['tidy_tweet'].str.replace(r'!\S+',"!")
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Χρησιμοποιούμε την βιβλιοθήκη `TextBlob` και κάνουμε ορθογραφικό έλεγχο στα δεδομένα μας.

```
print("doing spell correction...")
#spell correction
```

```
data['tidy_tweet'].apply(lambda x: str(TextBlob(x).correct()))
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Όταν ένα token δεν υπάρχει στο λεξικό των stopwords, επιστρέφει στο σύνολο δεδομένων μας. Η εντολή `join`, βοηθάει στο να παραληφθούν όλα τα κενά που δημιουργήθηκαν κατά την εκκαθάριση των δεδομένων. Επίσης, από τη λίστα των stopwords εξαιρούμε τις λέξεις `{"not", "no", "neither", "no one", "nobody", "nothing", "nowhere", "least"}`

```
print("tokenizing and removing stopwords")
tweets_processed = []
all_stopwords_gensim = STOPWORDS
sw_list = {"not","no","neither", "no one",
    "nobody","nothing","nowhere","least"}
all_stopwords_gensim = STOPWORDS.difference(sw_list)
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Παρακάτω εφαρμόζουμε Λεμματισμό χρησιμοποιώντας την βιβλιοθήκη `WordNet`

```
print("lemmatizing...")
lemmatizer=WordNetLemmatizer()
wordnet_map={"N":wordnet.NOUN,"V":wordnet.VERB,"J":wordnet.ADJ,"
R":wordnet.ADV}
def lemmatize_words(text):
    pos_tagged_text=nltk.pos_tag(text.split())
    return " ".join([lemmatizer.lemmatize
(word,wordnet_map.get(pos[0],wordnet.NOUN)) for word,pos in
pos_tagged_text])
data['tidy_tweet'] =data['tidy_tweet'].apply(lambda text:
lemmatize_words(text))
print (data['tidy_tweet'].head(10))
print("\n"*2)
```

Το νέο, καθαρισμένο σύνολο δεδομένων, γράφεται σε νέο αρχείο το οποίο τροφοδοτείται ως είσοδος στα μοντέλα GloVe και fastText. Στο αρχείο εισόδου του Bert, παραλείπουμε την αφαίρεση των stopwords και τον λεμματισμό.

```
print("Any moment now.Writing to file...")
#write to file
```

```
data.to_csv('preprocessed.csv', index=False, header=False,
    sep=';',encoding='utf-8')
```

Παράρτημα ΙΙ

Υλοποίηση επικρατέστερου μοντέλου Bert

Στην Ενότητα 5.2.1 περιγράψαμε το μοντέλο που χρησιμοποιήσαμε για τις ενσωματώσεις λέξεων Bert. Παρακάτω παρουσιάζουμε τον κώδικα του μοντέλου αυτού. Ο παρακάτω κώδικας αναφέρεται στο σύνολο δεδομένων Davidson.
Ο κώδικάς μας ξεκινά με τη φόρτωση των απαραίτητων βιβλιοθηκών καθώς και της δημιουργίας του επιπέδου (layer) Bert.

Φορτώνουμε τις απαραίτητες βιβλιοθήκες - πακέτα

```
from tensorflow.keras import layers
import tensorflow as tf
from sklearn.metrics import f1_score
import seaborn as sns
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import gc
import csv
import numpy as np
import pandas as pd
import keras
from keras import models
from keras.models import Model, load_model, Sequential
from keras import layers
from keras.layers import *
from keras import regularizers
from keras.engine import InputSpec, Layer
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, Callback,
EarlyStopping
from keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import  Input
```

```
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report
from tqdm import tqdm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.utils.np_utils import to_categorical
from keras.utils.vis_utils import plot_model
from sklearn.model_selection import KFold,StratifiedKFold
from sklearn import model_selection
use_gpu=True
```

θα χρειαστούμε έναν `Tokenizer`

```
!wget --quiet
https://raw.githubusercontent.com/tensorflow/models/master
/official/nlp/bert/tokenization.py
```

Δημιουργούμε το στρώμα (layer) Bert

```
!pip install bert-tensorflow
!pip install sentencepiece
import tensorflow_hub as hub
import bert
from keras.preprocessing.text import Tokenizer
module_url =
'https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-
768_A-12/2'
bert_layer = hub.KerasLayer(module_url, trainable=True)
```

Συνδέουμε το google `colab` με το φάκελο που περιέχει τα αρχεία μας στο `google drive`.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Φορτώνουμε στον κώδικά μας το σύνολο δεδομένων που θα χρησιμοποιηθεί, καθορίζουμε τα πεδία από τα οποία αποτελείται και διώχνουμε τυχόν άδειες εγγραφές που προέκυψαν κατά την προ-επεξεργασία. Τέλος, κρατάμε μόνο τα πεδία με τα οποία θα ασχοληθούμε στη συνέχεια,

δηλαδή, το "tidy_tweet" που περιέχει τα απαλλαγμένα από θόρυβο και περιττά στοιχεία, tweets, και το πεδίο κλάσης "h_class".

```
path = "/content/gdrive/My Drive/dataset/preproc_bert.csv"
names=['column_a','count','hate_speech','offensive_languag
e','neither','hclass','tweet','tidy_tweet']
data = pd.read_csv(path,names=names,sep=";")
data = data[['tidy_tweet', 'hclass']]
#drop empty lines
data=data.dropna()
y=data.hclass
X=data.tidy_tweet
```

Εμφανίζουμε σε plot την κατανομή των τριών κλάσεων

```
data['hclass'].value_counts()
fig1 = sns.countplot(x= 'hclass',data = data)
plt.title('Class Distribution')
plot = fig1.get_figure()
plot.savefig('Count Plot.png')
```

Αρχικοποιούμε τους πίνακες που θα χρησιμοποιήσουμε για να κρατήσουμε διάφορες μετρικές σε κάθε Fold

```
acc_per_fold = []
loss_per_fold = []
f1_per_fold = []
f1w_per_fold = []
mdl=[]
```

Δηλώνουμε διάφορες υπερ-παραμέτρους που θα χρησιμοποιηθούν όπως ο αριθμός των Folds (num_folds), ο μέγιστος αριθμός των εποχών (EPOCHS), το μέγιστο μήκος ακολουθίας (MAX_LEN) και το μέγεθος του batch (BATCH_SIZE)

```
num_folds = 10
EPOCHS =200
BATCH_SIZE = 32
MAX_LEN=90
```

Χωρίζουμε τα δεδομένα εισόδου σε `X_train, y_train και X_test, y_test,` με ποσοστό 70-30 αντίστοιχα. Είναι σημαντικό, στο `train` και `test` να είναι ισάριθμα κατανεμημένες οι τρεις κλάσεις, ώστε να εκπαιδευτεί σωστά το μοντέλο μας. Το παραπάνω επιτυγχάνεται μέσα από την παράμετρο `stratify`.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30,stratify=y, random_state=42)
```

Στην συνάρτηση `bert_encode` ορίζονται τα εξής:

Αρχικά, το κείμενο χωρίζεται σε tokens. Στην αρχή και στο τέλος κάθε ακολουθίας token προστίθενται τα διακριτικά [CLS]και [SEP] αντίστοιχα. Αν η ακολουθία έχει διαφορετικό μέγεθος από το μέγιστο μήκος ακολουθίας, τότε, εφαρμόζεται η τεχνική pad, κατά την οποία είτε περικόπτεται μια μεγαλύτερη ακολουθία είτε γεμίζει με μηδενικά το τέλος μιας μικρότερης ακολουθίας.

```
def bert_encode(texts, tokenizer, max_len=MAX_LEN):
    all_tokens = []
    all_masks = []
    all_segments = []

    for text in texts:
        text = tokenizer.tokenize(text)
        text = text[:max_len-2]
        input_sequence = ["[CLS]"] + text + ["[SEP]"]
        pad_len = max_len - len(input_sequence)
        tokens =
tokenizer.convert_tokens_to_ids(input_sequence)
        tokens += [0] * pad_len
        pad_masks = [1] * len(input_sequence) + [0] *
pad_len
        segment_ids = [0] * max_len
        all_tokens.append(tokens)
        all_masks.append(pad_masks)
        all_segments.append(segment_ids)
```

```
    return np.array(all_tokens), np.array(all_masks),
np.array(all_segments)
```

Δημιουργία του μοντέλου

```
def build_model(bert_layer, max_len=MAX_LEN):
  input_word_ids = tf.keras.Input(shape=(max_len,),
dtype=tf.int32, name="input_word_ids")
  input_mask = tf.keras.Input(shape=(max_len,),
dtype=tf.int32, name="input_mask")
  segment_ids = tf.keras.Input(shape=(max_len,),
dtype=tf.int32, name="segment_ids")
  pooled_output, sequence_output =
bert_layer([input_word_ids, input_mask, segment_ids])

clf_output= sequence_output[:,0,:]
model=models.Sequential()
net=tf.keras.layers.Dense(128,activation='relu')(clf_outpu
t)
net=tf.keras.layers.Dropout(0.1)(net)
net=tf.keras.layers.Dense(128,activation='relu')(net)
net=tf.keras.layers.Dropout(0.1)(net)
net=tf.keras.layers.Dense(128,activation='relu')(net)
net=tf.keras.layers.Dropout(0.1)(net)
out=tf.keras.layers.Dense(3,activation='softmax')(net)


model=tf.keras.models.Model(inputs=[input_word_ids,input_m
ask,segment_ids],outputs=out)
  optimizer = Adam(lr=3e-5)
  model.compile(optimizer=optimizer,
loss='categorical_crossentropy', metrics=['accuracy'])
  return model
vocabulary_file =
bert_layer.resolved_object.vocab_file.asset_path.numpy()
```

```
lower_case =
bert_layer.resolved_object.do_lower_case.numpy()
tokenizer =
tokenization.FullTokenizer(vocabulary_file,lower_case)

test_input = bert_encode(X_test, tokenizer,
max_len=MAX_LEN)
test_labels = keras.utils.to_categorical(y_test,
num_classes=3)
```

Χρήση της `StratifiedKFold`, για χωρισμό των δεδομένων του `Train set` σε `train` και `valid`, με ίση κατανομή των κλάσεων. Σε κάθε fold θα κρατούνται τα 9 για εκπαίδευση ενώ το 10ο θα χρησιμοποιείται για έλεγχο

```
skfold = StratifiedKFold(n_splits=num_folds, shuffle=True)
```

Για κάθε `fold` τα `train` και `valid` μετασχηματίζονται μέσα από τη συνάρτηση `bert_encode`, σε κατάλληλη μορφή, για επεξεργασία από το `bert`

```
fold_no = 1
for train,valid in skfold.split(X_train,y_train):
    print('\n' * 3)
    print("Training on Fold: ",fold_no)


    Xtrain = X_train.iloc[train]
    ytrain = y_train.iloc[train]
    Xvalid = X_train.iloc[valid]
    yvalid = y_train.iloc[valid]


    train_input = bert_encode(Xtrain, tokenizer,
max_len=MAX_LEN)
    train_labels = keras.utils.to_categorical(ytrain,
num_classes=3)
    valid_input = bert_encode(Xvalid, tokenizer,
max_len=MAX_LEN)
```

```
    valid_labels = keras.utils.to_categorical(yvalid,
num_classes=3)
```

Αρχικοποιείται, εκ νέου, το στρώμα Bert ώστε να αρχικοποιούνται και τα αντίστοιχα βάρη σε κάθε `fold`

```
    try:
        del bert_layer
    except:
        pass

    module_url =
"https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-
768_A-12/2"
    bert_layer = hub.KerasLayer(module_url,
trainable=True)
```

Δημιουργείται το μοντέλο μέσω της εντολής `compile` και στη συνέχεια εκπαιδεύονται τα δεδομένα εισόδου μας σε αυτό μέσω της εντολής `fit`.Κατά την εκπαίδευση των δεδομένων χρησιμοποιείται η παράμετρος `callbacks` της εντολής `fit`, με την οποία:

- Παρακολουθείται την τιμή της μεταβλητής val_loss, ώστε όταν αυτή σταματήσει να μειώνεται και με ανοχή πέντε συνεχόμενων μη μειώσεων, να διακόπτεται η εκπαίδευση.
- Σε κάθε `epoch` που υπάρχει βελτίωση της τιμής της val_loss, αποθηκεύονται τα βάρη με την εντολή `model_checkpoint` ώστε να τα φορτωθούν, ξανά, μετά το τέλος της εκπαίδευσης του συγκεκριμένου `fold`, στη φάση ελέγχου του μοντέλου

```
    model = build_model(bert_layer, max_len=MAX_LEN)
    file_path = "fold " + str(fold_no) + "
best_model.hdf5"

    model_checkpoint = ModelCheckpoint(file_path,
verbose=1, save_best_only=True)
    monitor = EarlyStopping(monitor='val_loss',mode='min',
patience=2, verbose=1,restore_best_weights=True)
    bert_history = model.fit(train_input
                             , train_labels
```

```
                                    , epochs= EPOCHS
                                    , batch_size=BATCH_SIZE
                                    ,
validation_data=(valid_input, valid_labels)
                                    , verbose=1
                                    , callbacks=[monitor,
model_checkpoint])
```

Αξιολόγηση μοντέλου στο συγκεκριμένο `fold`

```
    print("loading weights...")
     model.load_weights(file_path)
     scores =model.evaluate(test_input, test_labels)
```

Πρόβλεψη κλάσεων

```
    print("predicting...")
     pred = model.predict(test_input)
     predict_classes = np.argmax(pred,axis=1)
     expected_classes = np.argmax(test_labels,axis=1)
     label = np.argmax(test_labels,axis = 1)
     target_names = ['hate 0', 'offensive 1', 'neither 2']
     print(classification_report(expected_classes,
predict_classes,target_names=target_names))
     f1 =
f1_score(predict_classes,expected_classes,average='macro')
     f1w =
f1_score(predict_classes,expected_classes,average='weighte
d')
     f1w_per_fold.append(f1w* 100)
     f1_per_fold.append(f1* 100)
     print(f'Score for fold {fold_no}:
{model.metrics_names[0]} of {scores[0]:.5};
{model.metrics_names[1]} of {scores[1]:.5}%')
     acc_per_fold.append(scores[1])
     loss_per_fold.append(scores[0])
     mdl.append(file_path)
```

```
    fold_no = fold_no + 1
```

Εμφανίζονται οι τιμές των μετρικών Accuracy, Loss, F1-macro και F1- weighted σε κάθε fold και στη συνέχεια οι μέσοι όροι των παραπάνω μετρικών από όλα τα folds

```
print('------------------------------------------------')
print('Score per fold')
for i in range(0, len(acc_per_fold)):
    print(f'> Fold {i+1} - Loss: {loss_per_fold[i]} -
Accuracy: {acc_per_fold[i]:.5}%- F-macro:
{f1_per_fold[i]:.5}- F-weigted: {f1w_per_fold[i]:.5}')
    print('------------------------------------------------')
print('Average scores for all folds:')
print(f'> Accuracy: {np.mean(acc_per_fold):.5} (+-
{np.std(acc_per_fold):.5})')
print(f'> Loss: {np.mean(loss_per_fold):.5}')
print(f'> F1-macro: {np.mean(f1_per_fold):.5}')
print(f'> F1-weigthed: {np.mean(f1w_per_fold):.5}')
print('------------------------------------------------')
```

Παράρτημα ΙΙΙ

Αποτελέσματα κατά τη διάρκεια της προ-επεξεργασίας στο σύνολο δεδομένων Davidson

BEFORE
1.!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
2.!!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!!
3.!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever f*ck a b*tch and she start to cry? You be confused as sh*t
4.!!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny
5.!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya 
6.!!!!!!!!!!!!!!!!!!"@T_Madison_x: The sh*t just blows me..claim you so faithful and down for somebody but still f*cking with hoes! 😂😂😂"
7.!!!!!!"@__BrighterDays: I cannot just sit up and HATE on another bitch .. I got too much sh*t going on!"
8.!!!!“@selfiequeenbri: cause I'm tired of you big b*tches coming for us skinny girls!!”
9." & you might not get ya b*tch back & thats that "
10." @rhythmixx_ :hobbies include: fighting Mariam"b*tch
removing @,# etc.....
1.!!! As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
2. !!!!! boy dats cold...tyga dwn bad for cuffin dat hoe in the st place!!
3.!!!!!!! Dawg!!!! You ever f*ck a b*tch and she start to cry? You be confused as sh*t

4.!!!!!!!!! she look like a tranny

5. !!!!!!!!!!!!! The sh*t you hear about me might be true or it might be faker than the b*tch who told it to ya & ;

6. !!!!!!!!!!!!!!!!!!" The shit just blows me..claim you so faithful and down for somebody but still f*cking with hoes! & ;& ;& ;"

7. !!!!!!" I cannot just sit up and HATE on another b*tch .. I got too much sh*t going on!"

8. !!!!& ; cause I'm tired of you big b*tches coming for us skinny girls!!& ;

9. " & you might not get ya b*tch back & thats that "

10. " :hobbies include: fighting Mariam"b*tch

removing emoticons...

1.!!! As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...

2. !!!!! boy dats cold...tyga dwn bad for cuffin dat hoe in the st place!!

3.!!!!!!! Dawg!!!! You ever f*ck a b*tch and she start to cry? You be confused as sh*t

4.!!!!!!!!! she look like a tranny

5. !!!!!!!!!!!!! The sh*t you hear about me might be true or it might be faker than the b*tch who told it to ya & ;

6.!!!!!!!!!!!!!!!!!!" The sh*t just blows me..claim you so faithful and down for somebody but still f*cking with hoes! & ;& ;& ;"

7.!!!!!!" I cannot just sit up and HATE on another b*tch .. I got too much sh*t going on!"

8.!!!!& ; cause I'm tired of you big b*tches coming for us skinny girls!!& ;

9. " & you might not get ya b*tch back & thats that "

10. " hobbies include fighting Mariam"b*tch

lowercasing...

1.!!! as a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...

2.!!!!! boy dats cold...tyga dwn bad for cuffin dat hoe in the st place!!

3.!!!!!!! dawg!!!! you ever f*ck a b*tch and she start to cry? you be confused as sh*t

4.!!!!!!!!! she look like a tranny

5.!!!!!!!!!!!!! the sh*t you hear about me might be true or it might be faker than the b*tch who told it to ya & ;

6. !!!!!!!!!!!!!!!!!!" the sh*t just blows me..claim you so faithful and down for somebody but still f*cking with hoes! & ;& ;& ;"

7.!!!!!!" i cannot just sit up and hate on another b*tch .. i got too much sh*t going on!"

8. !!!!& ; cause i'm tired of you big b*tches coming for us skinny girls!!& ; **9.** " & you might not get ya b*tch back & thats that " **10.** " hobbies include fighting mariam"b*tch
replacing elongated words... **1.**!!! as a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out... **2.**!!!!! boy dats cold...tyga dwn bad for cuffin dat hoe in the st place!! **3.**!!!!!!! dawg!!!! you ever f*ck a b*tch and she start to cry? you be confused as sh*t **4.**!!!!!!!!! she look like a tranny **5.**!!!!!!!!!!!!! the sh*t you hear about me might be true or it might be faker than the b*tch who told it to ya & ; **6.**!!!!!!!!!!!!!!!!!!" the sh*t just blows me..claim you so faithful and down for somebody but still f*cking with hoes! & ;& ;& ;" **7.**!!!!!!" i cannot just sit up and hate on another b*tch .. i got too much sh*t going on!" **8.** !!!!& ; cause i'm tired of you big b*tches coming for us skinny girls!!& ; **9.** " & you might not get ya b*tch back & thats that " **10.** " hobbies include fighting mariam"b*tch
removing slang and correcting abbreviations..... **1.** !!! as a woman you should not complain about cleaning up your house. &amplifier; as a man you should always take the trash out... **2.** !!!!! boy that is cold...thank you god always dwn bad for cuffin that wh*re in the something place!! **3.** !!!!!!! dude!!!! you ever f*ck a bitch and she start to cry? you be confused as sh*t **4.** !!!!!!!!! she look like a tranny **5.** !!!!!!!!!!!!! the sh*t you hear about me might be true or it might be faker than the b*tch who told it to your & ; **6.** !!!!!!!!!!!!!!!!!!" the sh*t just blows me..claim you so faithful and down for somebody but still f*cking with wh*res! & ;& ;& ;" **7.** !!!!!!" i cannot just sit up and hate on another b*tch .. i got too much sh*t going on!" **8**. !!!!& ; because i am tired of you great b*tches coming for us skinny girls!!& ; **9.** " &amplifier; you might not get your b*tch back &amplifier; that is that " **10.** " hobbies include fighting mariam"b*tch
removing punctuations.....

1. ! as a woman you should not complain about cleaning up your house amplifier as a man you should always take the trash out

2. ! boy that is cold thank you god always dwn bad for cuffin that wh*re in the something place!

3. ! dude! you ever f*ck a b*tch and she start to cry? you be confused as sh*t

4. ! she look like a tranny

5. ! the sh*t you hear about me might be true or it might be faker than the b*tch who told it to your

6.! the sh*t just blows me claim you so faithful and down for somebody but still f*cking with wh*res!

7. ! i cannot just sit up and hate on another b*tch i got too much sh*t going on!

8. ! because i am tired of you great b*tches coming for us skinny girls!

9. amplifier you might not get your b*tch back amplifier thats that

10. hobbies include fighting mariam b*tch

doing spell correction...

1.! as a woman you should not complain about cleaning up your house amplifier as a man you should always take the trash out

2.! boy that is cold thank you god always down bad for cuffin that wh*re in the something place!

3.! dude! you ever f*ck a b*tch and she start to cry you be confused as sh*t

4.! she look like a tranny

5.! the sh*t you hear about me might be true or it might be faker than the b*tch who told it to your

6.! the sh*t just blows me claim you so faithful and down for somebody but still f*cking with wh*res!

7.! i can not just sit up and hate on another b*tch i got too much sh*t going on!

8.!because i am tired of you great b*tches coming for us skinny girls!

9. amplifier you might not get your b*tch back amplifier that is that

10.hobbies include fighting mariam b*tch

tokenizing and removing stopwords

1. ! woman not complain cleaning house amplifier man trash

2. ! boy cold thank god down bad cuffin wh*re place !

3. ! dude ! f*ck bitch start confused sh*t

4. ! look like tranny

5. ! sh*t hear true faker b*tch told

6. ! sh*t blows claim faithful somebody f*cking wh*res !

7. ! not sit hate b*tch got sh*t going !

8. ! tired great b*tches coming skinny girls !

9. amplifier not b*tch amplifier

10. hobbies include fighting mariam b*tch
lemmatizing... **1.** ! woman not complain clean house amplifier man trash **2.** ! boy cold thank god down bad cuffin wh*re place ! **3.** ! dude ! f*ck b*tch start confuse sh*t **4.** ! look like tranny **5.** ! sh*t hear true faker b*tch tell **6.** ! sh*t blow claim faithful somebody f*ck wh*re ! **7.** ! not sit hate b*tch get sh*t go ! **8.** ! tire great b*tch come skinny girl ! **9.** amplifier not b*tch amplifier **10.** hobby include fight mariam b*tch