Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας
Ακαδημαϊκό Έτος 2022 - 23

Ανάλυση Συναισθήματος με Νευρωνικά Δίκτυα και Αποδοτική και Κατανεμημένη Εκπαίδευση Νευρωνικών Δικτύων

Πτυχιακή Εργασία

Σπυρίδων Δράντζιος

Επιβλέπουσα Καθηγήτρια: Γεωργία Κολωνιάρη

Ιούνιος 2023, Θεσσαλονίκη

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την καθηγήτρια Γεωργία Κολωνιάρη και τον καθηγητή Γεώργιο Ευαγγελίδη για τις γνώσεις που μου προσέφεραν και που μου προσέφεραν την έμπνευση για αυτήν την εργασία, μέσα από τα μαθήματά τους. Επίσης, τον καθηγητή Κωνσταντίνο Μαργαρίτη που μου μετέδωσε το ερευνητικό ενδιαφέρον του Κατανεμημένου Υπολογισμού και του Υπολογισμού Υψηλών Επιδόσεων, τα οποία και αποτέλεσαν σημαντικό κομμάτι αυτής της εργασίας.

Περίληψη

Η τεχνητή νοημοσύνη αναπτύσσεται ολοένα και περισσότερο, προσφέροντας πολλαπλά οφέλη στις ζωές των ανθρώπων, τα οποία είναι αδιαμφισβήτητα. Η χρήση της τεχνολογίας δύναται να βελτιώσει σημαντικά την καθημερινή ζωή, την παραγωγικότητα, την επιχειρηματικότητα και οποιονδήποτε τομέα την αξιοποιεί. Ένα σημαντικό κομμάτι της τεχνητής νοημοσύνης, το οποίο έχει γνωρίσει ιδιαίτερη ανάπτυξη είναι αυτό των νευρωνικών δικτύων, το οποίο μπορεί να χρησιμοποιηθεί για την αποτελεσματική ανάλυση μεγάλου όγκου δεδομένων, καθώς και για την εξαγωγή χρήσιμων συμπερασμάτων από πλευράς των επιχειρήσεων. Η χρήση των νευρωνικών δικτύων δύναται να δώσει μια ολοκληρωμένη εικόνα για το συναίσθημα που έχουν οι πελάτες για μια επιχείρηση ή οι άνθρωποι για ένα πρόσωπο. Κάθε επιχείρηση οφείλει να παρακολουθεί τι γνώμη έχουν οι πελάτες της για αυτήν, ώστε να μπορέσει να βελτιώνεται διαρκώς σε μια εποχή που τα δεδομένα θεωρούνται ο μοντέρνος χρυσός. Σκοπός της παρούσας εργασίας είναι η εξερεύνηση της τεχνολογίας των νευρωνικών δικτύων, όσον αφορά στην ανάλυση του συναισθήματος, το οποίο εξάγεται από κείμενο που έχουν γράψει οι πελάτες μιας επιχείρησης ή οι χρήστες μιας πλατφόρμας, καθώς και η αποτελεσματική και αποδοτική εκπαίδευση ενός νευρωνικού δικτύου σε ρεαλιστικό χρόνο, αξιοποιώντας υλισμικό και μεθόδους τελευταίας τεχνολογίας.

Πίνακας περιεχομένων

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1ο

Εισαγωγή

1.1 Επισκόπηση

Στην εποχή που η τεχνολογία πλέον αναπτύσσεται γρηγορότερα από ποτέ, η ανάγκη για άμεση και γρήγορη πληροφόρηση γίνεται όλο και πιο επιτακτική. Πάντοτε υπήρχε η τάση οι μηχανές να αναλαμβάνουν όποιες εργασίες μπορούν να αυτοματοποιηθούν, ώστε οι άνθρωποι να εστιάζουν σε αυτά που έχουν πραγματική σημασία. Αυτή η τάση έχει κορυφωθεί ακόμα περισσότερο στην εποχή που η τεχνητή νοημοσύνη γνωρίζει ιδιαίτερη άνθηση.

Ένα τέτοιο πρόβλημα, το οποίο αναλαμβάνει η τεχνητή νοημοσύνη είναι και η Ανάλυση Συναισθήματος (Sentiment Analysis), κατά το οποίο αναπτύσσονται μοντέλα, τα οποία πραγματοποιούν Επεξεργασία Φυσικής Γλώσσας και παρέχουν τη δυνατότητα εξαγωγής χρήσιμων συμπερασμάτων σχετικά με το γενικότερο συναίσθημα που χαρακτηρίζει το κείμενο προς επεξεργασία, το οποίο είτε μπορεί να αφορά σε κριτικές ενός προϊόντος στο διαδίκτυο, είτε σε σχόλια σε κάποια πλατφόρμα κοινωνικής δικτύωσης και πολλά ακόμα. Τα δεδομένα πλέον παράγονται με εξαιρετικά υψηλές ταχύτητες, οπότε και είναι ανθρωπίνως αδύνατο να διαβαστούν και να αξιολογηθούν από εξειδικευμένο προσωπικό, οπότε και αξιοποιείται η δύναμη της τεχνητής νοημοσύνης.

Όμως παρά τη χρησιμότητα αυτών των μοντέλων, δε μπορεί να παραληφθεί το γεγονός ότι αφήνουν πίσω τους σημαντικό ενεργειακό αποτύπωμα, λόγω των αυξημένων αναγκών σε υπολογιστικό φόρτο. Για αυτό το λόγο επιστρατεύονται ειδικά κατασκευασμένο υλισμικό και λογισμικό, που αποβαίνουν αποδοτικότερα, τόσο από άποψη ενέργειας, όσο και από άποψη χρόνου. Είναι εξαιρετικά ενδιαφέρον να διερευνηθούν και οι τρόποι, με τους οποίους μπορούν να κλιμακωθούν τα αναπτυσσόμενα μοντέλα τεχνητής νοημοσύνης, παραμένοντας όμως, όσο το δυνατόν περισσότερο φιλικά προς το περιβάλλον.

1.2 Αντικείμενο της εργασίας

Η παρούσα εργασία πραγματεύεται αντικείμενα από πολλαπλούς κλάδους της πληροφορικής, με σκοπό τη συνδυαστική αξιοποίηση γνώσεων και τεχνολογιών. Πιο συγκεκριμένα, η εργασία άπτεται της Μηχανικής Μάθησης, της Επεξεργασίας Φυσικής Γλώσσας, του Υπολογισμού Υψηλών Επιδόσεων, του Κατανεμημένου Υπολογισμού και των Τεχνολογιών Ιστού. Στόχος είναι η δημιουργία της βάσης για ένα εργαλείο τεχνητής νοημοσύνης, το οποίο θα είναι σε θέση να “διαβάζει” κείμενο και να εκτιμά τη συναισθηματική διάθεση του συγγραφέα ως αρνητική, θετική ή ουδέτερη.

Ωστόσο, η εκπαίδευση ενός τέτοιου εργαλείου απαιτεί τη διάθεση δεδομένων μεγάλου όγκου, ώστε να έχει τα επιθυμητά αποτελέσματα σε πραγματικές συνθήκες. Στο πλαίσιο της εργασίας χρησιμοποιήθηκε “μικρός” όγκος δεδομένων σε σχέση με τα δεδομένα που θα χρησιμοποιούσε κάποια εταιρεία. Παρόλα αυτά, αυτό το γεγονός δεν εκμηδενίζει την ανάγκη μελέτης των τρόπων, με τους οποίους μπορεί να επιταχυνθεί η εκπαίδευση του μοντέλου, τόσο σε επίπεδο hardware, όσο και σε software.

Όμως ένα τέτοιο εργαλείο δε θα ήταν το ίδιο χρήσιμο σε περίπτωση που δε θα μπορούσε να χρησιμοποιηθεί από χρήστες, οι οποίοι δεν έχουν εξειδικευμένες γνώσεις πληροφορικής. Για αυτό το σκοπό, πρόκειται να παρουσιαστεί ένα σενάριο χρήσης του μοντέλου, χρησιμοποιώντας τεχνολογίες ιστού. Οι χρήστες θα πρέπει να είναι σε θέση να ανοίγουν έναν περιηγητή και να μπορούν άμεσα να αξιοποιούν τη δύναμη της τεχνητής νοημοσύνης, γεγονός που βοηθά στον εκδημοκρατισμό της.

1.3 Δομή της εργασίας

Η εργασία αποτελείται από 7 κεφάλαια. Στο πρώτο κεφάλαιο τίθεται το θεωρητικό υπόβαθρο και στο δεύτερο πραγματοποιείται ανάλυση της προετοιμασίας των δεδομένων, τα οποία πρόκειται να χρησιμοποιηθούν για την εκπαίδευση των νευρωνικών δικτύων. Στο τρίτο κεφάλαιο αναπτύσσονται 2 μοντέλα, εξετάζεται η αποδοτικότητά τους και στο τέταρτο κεφάλαιο προτείνονται τρόποι αξιολόγησης των αποτελεσμάτων τους. Στο πέμπτο κεφάλαιο εξηγείται ο τρόπος με τον οποίο εξειδικευμένο υλισμικό επιταχύνει την εκπαίδευση των μοντέλων, ενώ στο έκτο κεφάλαιο παρουσιάζονται τρόποι εκπαίδευσης που εκτείνονται πέραν από τη χρήση ενός μόνο μηχανήματος. Στο έβδομο κεφάλαιο θίγεται το θέμα του εκδημοκρατισμού του εργαλείου μέσα από τη δημιουργία μιας διεπαφής, η οποία έχει τη μορφή API, η οποία δυνητικά μπορεί να γίνει προσβάσιμη μέσω διαδικτύου.

1.4 Εργαλεία που χρησιμοποιήθηκαν

Η εργασία εκπονήθηκε με Python 3.9.13 και χρησιμοποιήθηκαν τα παρακάτω εργαλεία:

Εργαλείο	**Σκοπός Χρήσης**
Pandas 2.0.1	Φόρτωση του dataset και πραγματοποίηση μαζικής επεξεργασίας επί των δεδομένων
Numpy 1.23.5	Πραγματοποίηση πράξεων επί πινάκων
Nltk 3.8.1	Επεξεργασία φυσικής γλώσσας
Scikit-learn 1.2.2	Διαχωρισμός dataset σε training και test set
Regular expression operations (re) 3.11	Επεξεργασία αλφαριθμητικών
Tensorflow 2.12.0	Εκπαίδευση νευρωνικών δικτύων
Keras 2.12.0	High level API για τη σύνθεση και την εκπαίδευση των νευρωνικών δικτύων
Flask 2.3.2	Δημιουργία API
Google API Client Library for Python 2.87.0	Αλληλεπίδραση με το YouTube Data API v3
Urllib3 1.26.15	Ανάκτηση παραμέτρων από συνδέσμους

Πίνακας 1.1 Εργαλεία που χρησιμοποιήθηκαν

Κεφάλαιο 2ο

Αποσαφηνίζοντας τους βασικούς όρους

Σε αυτό το κεφάλαιο τίθεται το θεωρητικό υπόβαθρο ως βάση για όσα ακολουθούν. Ωστόσο, και σε επόμενα κεφάλαια εξηγούνται οι απαραίτητοι όροι που χρησιμοποιούνται.

2.1 Τι είναι Ανάλυση Συναισθήματος (Sentiment Analysis);

Ως Sentiment Analysis ή στα ελληνικά Ανάλυση Συναισθήματος ορίζεται η επεξεργασία φυσικής γλώσσας με μεθόδους τεχνητής νοημοσύνης και μη, με σκοπό τη διερεύνηση του γενικότερου συναισθήματος που επικρατεί σε μια ομάδα ανθρώπων. Η συγκεκριμένη πρακτική παρέχει τη δυνατότητα εξαγωγής ενός συμπεράσματος σχετικά με το αν η μελετώμενη πληθυσμιακή ομάδα νιώθει θετικά, αρνητικά ή ουδέτερα ως προς ένα θέμα.

2.2 Τι είναι Μηχανική Μάθηση;

Η μηχανική μάθηση είναι ένας κλάδος της τεχνητής νοημοσύνης, το οποίο αξιοποιεί αλγόριθμους και δεδομένα με σκοπό τη μίμηση της διαδικασίας, με την οποία οι άνθρωποι μαθαίνουν. Η ακρίβεια των αλγορίθμων βελτιώνεται όλο και περισσότερο με την εκπαίδευση των μοντέλων και την παροχή μεγαλύτερου και ποιοτικού όγκου δεδομένων.

2.3 Τι είναι Εποπτευόμενη Μηχανική Μάθηση;

Ως εποπτευόμενη μηχανική μάθηση ορίζεται ο τομέας της μηχανικής μάθησης, κατά τον οποίο οι επιλεγόμενοι αλγόριθμοι εκπαιδεύονται σε δεδομένα, στα οποία έχει αποδοθεί μία ετικέτα, με σκοπό μελλοντικά να είναι σε θέση να κατηγοριοποιήσουν τα δεδομένα τα οποία δέχονται ως είσοδο. Γενικότερα, η μηχανική μάθηση χωρίζεται σε τέσσερα κλαδιά: εποπτευόμενη, μη εποπτευόμενη, ημιεποπτευόμενη και ενισχυμένη. Το μοντέλο που πρόκειται να αναπτυχθεί στην παρούσα εργασία εντάσσεται στην κατηγορία της εποπτευόμενης μηχανικής μάθησης.

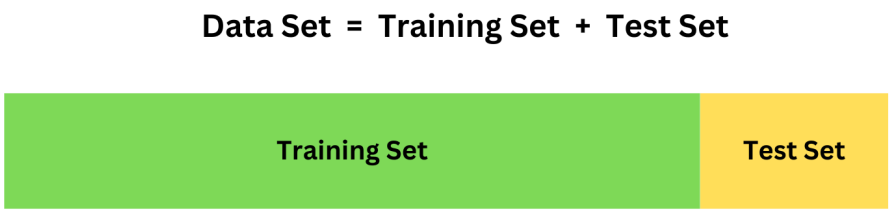
2.4 Τι είναι Εκπαίδευση ενός μοντέλου;

Ως εκπαίδευση ορίζεται η διαδικασία κατά την οποία το μοντέλο καλείται να λάβει αποφάσεις μέσα από τα δεδομένα που του έχουν δοθεί και να μάθει από αυτά. Σκοπός της εκπαίδευσης είναι το μοντέλο να μπορεί να αναγνωρίζει μοτίβα και να μπορεί να πραγματοποιεί προβλέψεις και εκτιμήσεις με μεγάλη ακρίβεια (accuracy). Σε κάθε στάδιο της εκπαίδευσης, το μοντέλο “διαβάζει” τα δεδομένα που του έχουν δοθεί και στη συνέχεια πραγματοποιεί ελέγχους σε δεδομένα που δεν έχει χρησιμοποιήσει για την εκπαίδευσή του, ώστε να διαπιστωθεί το πόσο ακριβές είναι τη δεδομένη χρονική στιγμή. Η ακρίβεια του μοντέλου τείνει να έχει ανοδική πορεία.

2.5 Τι είναι Σύνολο Εκπαίδευσης (Training Set) και Σύνολο Ελέγχου (Test Set);

Για την εκπαίδευση των μοντέλων που πρόκειται να αναπτυχθούν, είναι αναγκαίο να δοθεί ως είσοδος ένα σύνολο δεδομένων (dataset). Ουσιαστικά πρόκειται για έναν πίνακα με πολλαπλές στήλες, στον οποίο έχουν οργανωθεί τα δεδομένα με μεθοδικό τρόπο.

Πέραν της εκπαίδευσης, είναι χρήσιμο να πραγματοποιηθούν και δοκιμές προτού χρησιμοποιηθεί ένα μοντέλο. Τα δεδομένα της εκπαίδευσης και τα δεδομένα για τις δοκιμές πρέπει να έχουν την ίδια μορφή, ώστε να υπάρχει ένα μέτρο σύγκρισης. Για αυτό το λόγο, το dataset χωρίζεται σε 2 κομμάτια: το training set, δηλαδή τα δεδομένα της εκπαίδευσης και το test set, δηλαδή τα δεδομένα που πρόκειται να χρησιμοποιηθούν για την αξιολόγηση της απόδοσης του μοντέλου. Συνιστάται το dataset να χωρίζεται σε 80% training set και 20% test set ή 90% και 10% αντίστοιχα.



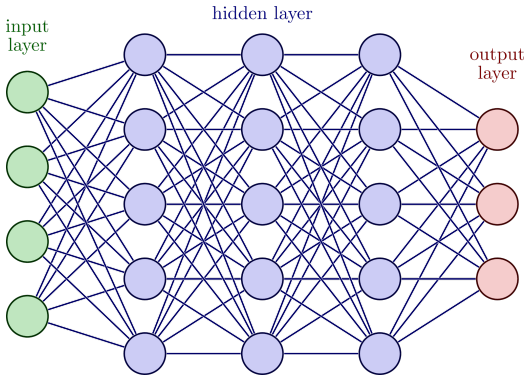
2.1 Διαχωρισμός του dataset σε training set και test set

2.6 Τι είναι Νευρωνικά Δίκτυα;

Τα νευρωνικά δίκτυα υπάγονται στην μηχανική μάθηση και αποτελούν την καρδιά των αλγορίθμων της βαθιάς μάθησης. Αποτελούνται από πολλαπλά επίπεδα κόμβων, οι οποίοι λαμβάνουν αριθμητικά σήματα. Οι κόμβοι επεξεργάζονται τα αριθμητικά σήματα και στέλνουν τα κατάλληλα μηνύματα στους κόμβους του επόμενου επιπέδου, οι οποίοι με τη σειρά τους λειτουργούν με τον ίδιο τρόπο. Η όλη διαδικασία επαναλαμβάνεται μέχρι το τελευταίο επίπεδο κόμβων, το οποίο και δημιουργεί την έξοδο. Τα νευρωνικά δίκτυα εκπαιδεύονται σε δεδομένα και μπορούν να αποδώσουν υψηλά επίπεδα ακρίβειας, όσον αφορά στις προβλέψεις που κάνουν. Γενικότερα, αποτελούνται από ένα επίπεδο, το οποίο δέχεται δεδομένα ως είσοδο, από κρυμμένα επίπεδα, τα οποία ενδεχομένως να είναι πολλαπλά και ένα επίπεδο, το οποίο επιστρέφει την έξοδο.

2.7 Τι είναι Βαθιά Μάθηση (Deep Learning);

Η βαθιά μάθηση είναι υποκατηγορία της μηχανικής μάθησης και αφορά σε νευρωνικά δίκτυα, τα οποία αποτελούνται από 3 ή περισσότερα επίπεδα, με σκοπό τη μίμηση της διαδικασίας με την οποία μαθαίνει ο άνθρωπος. Η αύξηση των επιπέδων αυξάνει το πόσο σύνθετο και πολύπλοκο είναι ένα μοντέλο. Ο συσχετισμός δεν είναι γραμμικός, αλλά γενικότερα όταν ακολουθούνται σωστές πρακτικές, όσο βαθύτερο είναι ένα ένα νευρωνικό δίκτυο, τόσο πιο ακριβές μπορεί να γίνει.



Εικόνα 2.2 Γενική δομή ενός νευρωνικού δικτύου
Πηγή: https://tikz.net/neural_networks/

2.8 Τι είναι Overfitting;

Το overfitting είναι ένα φαινόμενο κατά το οποίο το μοντέλο που εκπαιδεύτηκε είναι σε θέση να δώσει ακριβείς προβλέψεις όταν λαμβάνει δεδομένα από το training set, αλλά δεν έχει υψηλές αποδόσεις όταν λαμβάνει νέα δεδομένα. Δηλαδή το μοντέλο έχει “μάθει” πολύ καλά τα δεδομένα με τα οποία έχει εκπαιδευτεί και δεν είναι σε θέση να πραγματοποιήσει προβλέψεις με δεδομένα που δεν έχει “μάθει”.

Κεφάλαιο 3ο

Προετοιμασία των δεδομένων

Για το σκοπό της εργασίας, πρόκειται να χρησιμοποιηθούν κριτικές, οι οποίες έχουν δημοσιευτεί στην πλατφόρμα IMDb και που έχουν κατηγοριοποιηθεί ως θετικές ή αρνητικές. Το dataset[1] περιέχει 25.000 θετικές και 25.000 αρνητικές κριτικές στην αγγλική γλώσσα. Η δομή των δεδομένων περιγράφεται στον Πίνακα 3.1.

review	sentiment
A wonderful little production. The filming technique is very unassuming- very old-time-B...	positive
Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his par...	negative

Πίνακας 3.1 Η δομή του dataset

Η στήλη review περιέχει το κείμενο της κριτικής και η στήλη sentiment κατηγοριοποιεί την εκάστοτε κριτική ως αρνητική ή θετική.

Παρατηρείται ότι παρά την απλοϊκή δομή του πίνακα, τα δεδομένα χρήζουν περαιτέρω επεξεργασίας, ώστε να μετατραπούν σε μορφή που να γίνεται κατανοητή από ένα μοντέλο νευρωνικών δικτύων.

Τα δεδομένα πρέπει να περάσουν από διαδικασίες:

- Αφαίρεσης σημείων στίξης
- Αφαίρεσης ετικετών HTML
- Αφαίρεσης συνδέσμων
- Αφαίρεσης white space χαρακτήρων

Στη συνέχεια, όλοι οι χαρακτήρες πρέπει να μετατραπούν σε πεζούς και να αφαιρεθούν τα stopwords της αγγλικής γλώσσας, δηλαδή οι λέξεις που δεν προσδίδουν επιπλέον νόημα στο λόγο, αλλά λειτουργούν επικουρικά (για παράδειγμα οι λέξεις the, be κλπ). Το τελευταίο στάδιο της προετοιμασίας των δεδομένων συνίσταται στην λημματοποίηση όλων των λέξεων, ώστε να έρθουν σε μια κανονικοποιημένη αρχική μορφή, με σκοπό το μοντέλο να είναι σε θέση να εντοπίζει ευκολότερα το συναίσθημα.

[1] Πηγή δεδομένων: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Για την εφαρμογή όλων των παραπάνω χρησιμοποιήθηκε η βιβλιοθήκη Pandas για την εισαγωγή και την αποθήκευση του dataset και η βιβλιοθήκη nltk (Natural Language Toolkit) για τον εντοπισμό των stopwords. Στο παρακάτω κομμάτι κώδικα σε Python υλοποιούνται όλα τα ανωτέρω:

```
# Importing the libraries
import pandas as pd
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.corpus import stopwords

# Reading the dataset and validating it
df = pd.read_csv('IMDB Dataset.csv', sep=',')
df.head()

# Removing break tags
df['review'] = df['review'].str.replace('<br />', '')
# Removing URLs
df['review'] = df['review'].str.replace(r'https?://\S+|www\.\S+', '')
# Removing punctuation
df['review'] = df['review'].str.replace(r'[^\w\s]+', '')
# Removing new line characters
df['review'] = df['review'].str.replace(r'\n', '')
# Converting the reviews to lowercase
df['review'] = df['review'].apply(str.lower)

# Tokenization and Lemmatization of the reviews, Removing stop words
def lemmatize_text(text):
  stop_words = set(stopwords.words('english'))
  w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
  lemmatizer = nltk.stem.WordNetLemmatizer()
  lemmatized = [lemmatizer.lemmatize(w) for w in
  w_tokenizer.tokenize(text)]
  return [w for w in lemmatized if w not in stop_words]

df['review'] = df['review'].apply(lemmatize_text)
```

Στη συνέχεια, πρέπει τα δεδομένα να μετατραπούν σε πραγματικούς αριθμούς, ώστε να έρθουν στην επιθυμητή μορφή. Αρχικά, ορίζεται ως σύμβαση ότι η τιμή “positive” της στήλης sentiment αντιστοιχίζεται στον αριθμό 1.0 και η τιμή “negative” αντιστοιχίζεται στην τιμή 0.0:

```
# Binarizing the sentiment
# 0 --> Negative
# 1 --> Positive
df['sentiment'] = df['sentiment'].replace('negative', 0.0)
df['sentiment'] = df['sentiment'].replace('positive', 1.0)
```

Έπειτα, πρέπει όλες οι λέξεις που έχουν έχουν περάσει τη διαδικασία της λημματοποίησης να μετατραπούν σε ακολουθίες αριθμών, δηλαδή σε διανύσματα. Για αυτόν το σκοπό, πρόκειται να χρησιμοποιηθεί το πακέτο keras, το οποίο αποτελεί ένα API για ευκολότερο προγραμματισμό με Tensorflow σε Python. Το keras παρέχει ενσωματωμένο Tokenizer, ο οποίος λαμβάνει τις ακολουθίες λέξεων και αντιστοιχίζει κάθε μοναδική λέξη (token) σε έναν αριθμό. Στον κάτωθι κώδικα, ο Tokenizer λαμβάνει την ακολουθία χαρακτήρων κάθε κριτικής και επιστρέφει ένα διάνυσμα αριθμών. Αφού γίνει η μετατροπή, χρησιμοποιείται η συνάρτηση pad_sequences, ώστε όλα τα διανύσματα να έχουν το ίδιο μέγεθος:

```
# Tokenizing the reviews to float vectors
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
max_words = 5000
tokenizer = Tokenizer(filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n', lower=True,
num_words=max_words)
tokenizer.fit_on_texts(df['review'])
sequences = tokenizer.texts_to_sequences(df['review'])
reviews_temp = pad_sequences(sequences)
reviews = reviews_temp.astype(np.float32)
```

Έπειτα, πρέπει τα δεδομένα να διαχωριστούν σε test set και training set. Για να γίνει αυτό, θα χρησιμοποιηθεί η βιβλιοθήκη sklearn, η οποία παρέχει αυτή τη δυνατότητα. Τα δεδομένα πρόκειται να διαχωριστούν σε 80% training set και 20% test set. Για τον διαχωρισμό είναι αναγκαίο να δημιουργηθεί και ένας πίνακας 2 διαστάσεων, ο οποίος για κάθε κριτική θα κρατάει την ετικέτα ή αλλιώς το συναίσθημα που την χαρακτηρίζει. Τα παραπάνω υλοποιούνται στον εξής κώδικα:

```
# Splitting to train and test data
from sklearn.model_selection import train_test_split

# Categorizing the labels "Positive" and "Negative"
labels = np.array(df['sentiment'])
labels = tf.keras.utils.to_categorical(labels, 2, dtype="float32")

X_train, X_test, y_train, y_test = train_test_split(reviews, labels, random_state=0,
train_size=0.8, test_size=0.2)
```

Κατά σύμβαση ορίζεται ότι:

- X_train → είναι το 80% των κριτικών, το οποίο πρόκειται να χρησιμοποιηθεί για την εκπαίδευση
- X_test → είναι το 20% των κριτικών, το οποίο πρόκειται να χρησιμοποιηθεί για την επαλήθευση
- y_train → είναι οι ετικέτες που χαρακτηρίζουν τα δεδομένα του X_train
- y_test → είναι οι ετικέτες των κριτικών που χρησιμοποιούνται στην επαλήθευση

Τα δεδομένα είναι πλέον στην επιθυμητή μορφή, ώστε να χρησιμοποιηθούν στην εκπαίδευση. Ουσιαστικά, οι προτάσεις έχουν πλέον μετατραπεί σε διανύσματα ίδιου μήκους. Τα δεδομένα πλέον έχουν την εξής μορφή:

[[0.000e+00 0.000e+00 0.000e+00 ... 7.840e+02 3.673e+03 3.510e+02]
[0.000e+00 0.000e+00 0.000e+00 ... 1.753e+03 1.900e+01 1.340e+02]
[0.000e+00 0.000e+00 0.000e+00 ... 3.300e+01 1.500e+01 1.150e+02]
...
[0.000e+00 0.000e+00 0.000e+00 ... 2.650e+02 3.900e+02 3.588e+03]
[0.000e+00 0.000e+00 0.000e+00 ... 1.844e+03 2.346e+03 6.030e+02]
[0.000e+00 0.000e+00 0.000e+00 ... 8.980e+02 7.270e+02 1.000e+00]]

Κεφάλαιο 4ο

Δημιουργία 2 μοντέλων και εκπαίδευση

Για να γίνει εφικτή η αποτελεσματική επεξεργασία φυσικής γλώσσας, πρέπει να ληφθούν υπόψη όλες οι λέξεις που χρησιμοποιούνται εντός του κειμένου, καθώς και το γενικότερο περιβάλλον στο οποίο χρησιμοποιούνται. Ο τύπος νευρωνικών δικτύων που μπορεί να το κάνει αυτό ονομάζεται Recurrent Neural Network και χρησιμοποιείται συχνά σε τέτοιου είδους επεξεργασία, για παράδειγμα μετάφραση, αναγνώριση φωνής και φωνητική αναζήτηση.

Τα Recurrent Neural Networks διατηρούν ένα είδος μνήμης, η οποία χρησιμοποιείται για την αξιολόγηση των επόμενων δεδομένων προς είσοδο. Με αυτόν τον τρόπο, τα μοντέλα που πρόκειται να αναπτυχθούν θα είναι σε θέση να αντιληφθούν ολόκληρο το πλαίσιο στο οποίο χρησιμοποιούνται οι λέξεις και δε θα τις αξιολογούν ως αυτόνομες μονάδες. Οι λέξεις που πλέον έχουν μετατραπεί σε διανύσματα θα περνούν σειριακά από ένα Embedding Layer και το αποτέλεσμά τους θα εξάγεται από ένα Dense Layer.

Για τη δημιουργία και τη σύνθεση των μοντέλων πρόκειται να χρησιμοποιηθεί και πάλι το keras API.

4.1 Τι είναι Embedding Layer;

Τα δεδομένα έχουν περάσει από το στάδιο της επεξεργασίας και έτσι οι προτάσεις έχουν μετατραπεί σε διανύσματα. Όμως τα διανύσματα που έχουν δημιουργηθεί είναι πολύ αραιά και μεγάλα σε μέγεθος. Αυτό έχει ως αποτέλεσμα η επεξεργασία τους να καθίσταται δυσκολότερη. Το Embedding Layer παραλαμβάνει τα διανύσματα ως είσοδο και τα μετατρέπει σε μορφή που να επιτρέπει την ευκολότερη επεξεργασία τους. Ουσιαστικά πρόκειται για το επίπεδο κόμβων, το οποίο λαμβάνει τις λέξεις μια προς μια και τις προωθεί στο επόμενο επίπεδο.

4.2 Τι είναι Dense Layer;

Το Dense Layer είναι ένα από τα πιο δημοφιλή layers. Η αρχιτεκτονική τους έγκειται στο ότι κάθε κόμβος είναι απευθείας συνδεδεμένος με κάθε κόμβο του προηγούμενου layer και πρόκειται να βοηθήσει στην κατηγοριοποίηση των λέξεων ως θετικές ή αρνητικές. Το Dense Layer που θα χρησιμοποιηθεί στην ανάπτυξη των μοντέλων θα αποτελείται από 2 κόμβους, εφόσον το πρόβλημα αφορά τον διαχωρισμό των λέξεων σε θετικές και αρνητικές. Κάθε ένας από τους 2 κόμβους θα έχει ως έξοδο μια εκτίμηση αναφορικά με την κατηγορία της κάθε λέξης. Για παράδειγμα, μια λέξη που έχει περάσει από το μοντέλο είναι πολύ πιθανό να είναι 60% θετική και 40% αρνητική, άρα και αυτές θα είναι οι αντίστοιχες έξοδοι των κόμβων.

Έχοντας περιγράψει τα layers που πρόκειται να δέχονται τα δεδομένα και να παραδίδουν αποτελέσματα, είναι χρήσιμο να αναλυθεί και το ενδιάμεσο κομμάτι μεταξύ input και output.

4.3 Τι είναι LSTM Layer;

Το LSTM Layer κατατάσσεται στην κατηγορία των Recurrent Neural Networks και είναι ένας ειδικός τύπος νευρωνικών δικτύων, ο οποίος ειδικεύεται στην εκμάθηση μοτίβων και εξαρτήσεων και χρησιμοποιείται συχνά για την επεξεργασία φυσικής γλώσσας. Το LSTM Layer είναι φτιαγμένο έτσι ώστε να έχει τη δυνατότητα να επεξεργάζεται ακολουθίες πληροφοριών και όχι μεμονωμένες πληροφορίες. Η συγκεκριμένη δυνατότητα παρέχεται λόγω του ότι έχει ένα είδος μακροχρόνιας "μνήμης" ή αλλιώς βάρος, το οποίο επιλέγει να χρησιμοποιήσει όταν κρίνει ότι είναι απαραίτητο.

Με απλά λόγια, είναι σε θέση να λάβει υπόψη ολόκληρο το πλαίσιο της πρότασης και όχι μια λέξη τη φορά. Δύναται να αντιληφθεί ακόμα και το περιβάλλον στο οποίο χρησιμοποιείται η κάθε λέξη, καθώς και το πως σχετίζεται με τις υπόλοιπες, κάνοντας τους κατάλληλους υπολογισμούς.

Το LSTM Layer λαμβάνει υπόψη μόνο το πλαίσιο των δεδομένων του παρελθόντος. Δηλαδή, η τρέχουσα λέξη που υφίσταται επεξεργασία, συσχετίζεται μόνο με αυτές που προηγήθηκαν αυτής. Οι λέξεις που ακολουθούν πρόκειται να λάβουν υπόψη τις προηγούμενες λέξεις που έχουν υποστεί επεξεργασία, με σκοπό τη δημιουργία συσχετίσεων.

4.4 Τι είναι Bidirectional LSTM Layer;

Το Bidirectional LSTM Layer είναι όμοιο με το LSTM Layer. Η διαφορά του έγκειται στο ότι διαβάζει τα δεδομένα και προς τα εμπρός και προς τα πίσω. Κατ΄ αναλογία, αν έχουμε την πρόταση "I really liked this movie", το Bidirectional LSTM Layer πρόκειται κατά την εκπαίδευση να διαβάσει τις λέξεις και από αριστερά προς τα δεξιά, αλλά και από τα δεξιά προς τα αριστερά, που σημαίνει ότι θα λάβει την παραπάνω πρόταση και με τον εξής τρόπο ως είσοδο: "movie this liked really I".

Αυτή η πρακτική παρέχει τη δυνατότητα στο μοντέλο να δημιουργήσει συσχετίσεις και με τις λέξεις που έπονται μιας λέξης και όχι απλά να δημιουργήσει συσχετίσεις με τις λέξεις που προηγούνταν. Για παράδειγμα, σε ένα απλό LSTM Layer η λέξη "liked" θα συσχετιστεί με τις λέξεις "I really", ενώ με το Bidirectional LSTM Layer θα συσχετιστεί με όλες τις λέξεις της πρότασης, οδηγώντας σε δυνητικά καλύτερη απόδοση.

4.5 Τι είναι Sequential Model;

Το Sequential Model ή αλλιώς Ακολουθιακό μοντέλο είναι ο τρόπος δόμησης μοντέλου κατά τον οποίο τα layers προστίθενται ακολουθιακά. Ουσιαστικά, πρόκειται απλά για μια απλή διάταξη των layers το ένα μετά το άλλο.

4.6 Δόμηση πρώτου μοντέλου

Στον παρακάτω κώδικα ορίζεται ότι πρόκειται να δομηθεί ένα ακολουθιακό μοντέλο, το οποίο:

1. Δέχεται ως είσοδο δεδομένα και τα μετατρέπει σε διανύσματα διαστάσεων max_words επί 20.

2. Το δεύτερο επίπεδο είναι ενα LSTM Layer, το οποίο αναλύει τις λέξεις και βρίσκει τις συσχετίσεις μεταξύ τους.
3. Και το τρίτο επίπεδο είναι ένα Dense Layer, το οποίο παραδίδει 2 αριθμούς, οι οποίοι αποτελούν τις πιθανότητες κάθε λέξης να είναι θετική ή αρνητική.

```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop, Adam
from keras.callbacks import ModelCheckpoint

# Single LSTM Layer Model
model1 = Sequential()
model1.add(layers.Embedding(max_words, 20))
model1.add(layers.LSTM(20,dropout=0.3))
model1.add(layers.Dense(2,activation='softmax'))
model1.compile(optimizer='rmsprop',loss='categorical_crossentropy',
metrics=['accuracy'])

checkpoint1 = ModelCheckpoint("LSTM.hdf5", monitor='val_accuracy',
verbose=1, save_best_only=True, mode='auto', period=1, save_weights_only=False)

history = model1.fit(X_train, y_train, epochs=2, validation_data=(X_test,
y_test), callbacks=[checkpoint1])
```

Όσον αφορά στην εκπαίδευση, ορίζεται ότι θα παρακολουθούνται οι μετρικές accuracy, val_accuracy, loss και val_loss. Ειδικότερα:

- accuracy → πρόκειται για την ακρίβεια του μοντέλου και εν προκειμένω αφορά το ποσοστό των κριτικών του training set που έχουν καταταχθεί στη σωστή κλάση
- val_accuracy → πρόκειται για το ποσοστό των κριτικών του test set που έχουν καταταχθεί στη σωστή κλάση
- loss → πρόκειται για τη διαφορά των πραγματικών τιμών του training set με τις τιμές που προβλέπει το μοντέλο, καθώς εκπαιδεύεται
- val_loss → πρόκειται για τη διαφορά των τιμών του test set με τις τιμές που έχει προβλέψει το μοντέλο κατά τη διαδικασία της επαλήθευσης

4.7 Αποτελέσματα εκπαίδευσης του πρώτου μοντέλου

Το μοντέλο επιτυγχάνει αρκετά υψηλές τιμές accuracy και val_accuracy μετά από 2 epochs. Οι τιμές loss και val_loss είναι αρκετά κοντά, γεγονός που σημαίνει πως το μοντέλο αποδίδει το ίδιο καλά, είτε δεχτεί ως είσοδο τα δεδομένα στα οποία έχει εκπαιδευτεί, είτε δεδομένα τα οποία δεν γνωρίζει:

Epoch	accuracy	val_accuracy	loss	val_loss
1	0.8320	0.8741	0.3650	0.3053
2	0.8940	0.8872	0.2603	0.2706

Πίνακας 4.1 Αποτελέσματα εκπαίδευσης πρώτου μοντέλου

4.8 Δόμηση δεύτερου μοντέλου

Η μόνη διαφορά με το πρώτο μοντέλο είναι ότι χρησιμοποιείται Bidirectional LSTM Layer έναντι του απλού LSTM Layer:

```
# Bidirectional LSTM Model
model2 = Sequential()
model2.add(layers.Embedding(max_words, 20))
model2.add(layers.Bidirectional(layers.LSTM(20,dropout=0.3)))
model2.add(layers.Dense(2,activation='softmax'))
model2.compile(optimizer='rmsprop',loss='categorical_crossentropy',
metrics=['accuracy'])

checkpoint2 = ModelCheckpoint("BiLSTM.hdf5", monitor='val_accuracy',
verbose=1,save_best_only=True, mode='auto', period=1,save_weights_only=False)

history = model2.fit(X_train, y_train, epochs=2,validation_data=(X_test,
y_test),callbacks=[checkpoint2])
```

4.9 Αποτελέσματα εκπαίδευσης του δεύτερου μοντέλου

Το μοντέλο επιτυγχάνει παρόμοιες τιμές accuracy και val_accuracy μετά από 2 epochs.

Epoch	accuracy	val_accuracy	loss	val_loss
1	0.8293	0.8838	0.3706	0.2981
2	0.8934	0.8842	0.2617	0.2983

Πίνακας 4.2 Αποτελέσματα εκπαίδευσης δεύτερου μοντέλου

Αποφαίνεται ότι τα 2 μοντέλα έχουν σχεδόν ίδιες επιδόσεις. Οπότε για τη δοκιμή για τη συμπεριφορά σε νέες κριτικές στο επόμενο κεφάλαιο, πρόκειται να χρησιμοποιηθεί το πρώτο μοντέλο.

Κεφάλαιο 5ο

Ερμηνεία των προβλέψεων

5.1 Τι μορφή έχει η έξοδος του μοντέλου;

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, τα μοντέλα που έχουν δημιουργηθεί, δίνουν ως έξοδο 2 αριθμούς: ο ένας αφορά την πιθανότητα η λέξη να είναι αρνητική και ο άλλος να είναι θετική. Και οι 2 αριθμοί αποτελούν εκτιμήσεις του εκάστοτε μοντέλου. Η διαδικασία επαναλαμβάνεται για όλες τις λέξεις μιας κριτικής.

Για παράδειγμα, έστω η εξής **θετική** κριτική:

*This cute animated short features two comic icons - Betty Boop and Henry.

Henry is the bald, slightly portly boy from the comics who never speaks.

Well here he does speak!

He wants to get a puppy from Betty Boops pet store, and when he is left to mind the store - some hilarious hijinks ensue.

Betty sings a song about pets, Henry gets in a battle with birds and a monkey, but everything works out in the end.*

Μετά την προεπεξεργασία έχουμε την εξής λίστα λέξεων, η οποία πρόκειται να μετατραπεί σε διάνυσμα:

['cute', 'animated', 'short', 'feature', 'two', 'comic', 'icon', 'betty', 'boop', 'henry', 'henry', 'bald', 'slightly', 'portly', 'boy', 'comic', 'never', 'speaks', 'well', 'doe', 'speak', 'want', 'get', 'puppy', 'betty', 'boops', 'pet', 'store', 'left', 'mind', 'store', 'hilarious', 'hijinks', 'ensue', 'betty', 'sings', 'song', 'pet', 'henry', 'get', 'battle', 'bird', 'monkey', 'everything', 'work', 'end']

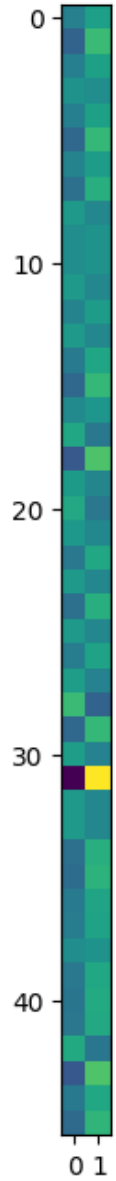
Μετά, έχουμε το διάνυσμα στο οποίο μετατράπηκε η κριτική:

[903.]
[1136.]
[223.]
...
...
...
[182.]
[57.]
[46.]

Το μοντέλο στη συνέχεια παραλαμβάνει το διάνυσμα και για κάθε μία από τις 46 λέξεις της κριτικής επιστρέφει την εκτίμηση:

[0.4591504 0.54084957]
[0.38504952 0.6149505]
...
...
...
[0.43896306 0.56103694]
[0.40143645 0.59856355]

Η εκτίμηση των λέξεων οπτικοποιημένη φαίνεται στην Εικόνα 5.1.



Εικόνα 5.1 Οπτικοποίηση της πρόβλεψης του μοντέλου

Στο παραπάνω γράφημα, η τιμή 0 αντιπροσωπεύει το αρνητικό και η τιμή 1 το θετικό. Πράγματι, παρατηρείται ότι η στήλη “1” έχει υψηλότερες τιμές από τη στήλη “0”, εφόσον τα χρώματα της τείνουν να είναι πιο ανοιχτόχρωμα, δηλαδή τείνουν περισσότερο προς το κίτρινο. Ωστόσο, είναι γνωστό εκ των προτέρων ότι η κριτική έχει αξιολογηθεί ως θετική. Μένει μόνο να βρεθεί ένας αποδοτικός τρόπος αξιολόγησης του αποτελέσματος, διότι σε ένα ρεαλιστικό σενάριο χρήσης του εργαλείου οι χρήστες πρέπει να λάβουν μια ξεκάθαρη απάντηση σχετικά με το αν η κριτική είναι αρνητική ή θετική.

5.2 Αξιολόγηση με το μέσο όρο

Η αξιολόγηση της κριτικής θα μπορούσε να πραγματοποιηθεί χρησιμοποιώντας το μέσο όρο κάθε στήλης του πίνακα, λαμβάνοντας ως δεδομένο ότι κάθε λέξη έχει την ίδια βαρύτητα. Στο συγκεκριμένο τρόπο αξιολόγησης λαμβάνεται υπόψη η μεγαλύτερη εκ των 2 πιθανοτήτων. Αν οι πιθανότητες είναι ίσες, τότε η κριτική χαρακτηρίζεται ως ουδέτερη:

```
avg = prediction.mean(axis=0)

print(avg[0] * 100, "% negative")
print(avg[1] * 100, "% positive")

if(avg[0]>avg[1]):
  print("The review is negative")
elif(avg[0]<avg[1]):
  print("The review is positive")
else:
  print("The review is neutral")
```

Η συγκεκριμένη κριτική δίνει το εξής αποτέλεσμα:

48.20 % negative
51.79 % positive
The review is positive

5.3 Αξιολόγηση κατά λέξη

Ένας εναλλακτικός τρόπος αξιολόγησης είναι ο εξής: λαμβάνεται υπόψη η υψηλότερη τιμή κάθε λέξης και κάθε λέξη λαμβάνει την ετικέτα positive ή negative. Στη συνέχεια, καταμετρώνται οι εμφανίσεις κάθε ετικέτας και το συμπέρασμα προκύπτει από την ετικέτα, η οποία έχει τις περισσότερες εμφανίσεις. Και πάλι θεωρείται ότι κάθε λέξη έχει την ίδια βαρύτητα στην κριτική.

Στον παρακάτω κώδικα δημιουργείται μια νέα λίστα classified, στην οποία προστίθεται κάθε φορά ένας αριθμός που χαρακτηρίζει τη λέξη που βρίσκεται στην ίδια θέση. Και πάλι οι θετικές λέξεις λαμβάνουν την τιμή 1 και οι αρνητικές την τιμή 0.

```
 # Interpreting the output by categorizing every word and then counting the positive
and the negative words
classified = []

for item in prediction:
  if item[0] > item[1]:
    classified.append(0)
  else:
    classified.append(1)

positive = sum(classified)
negative = len(classified) - positive

print("Positive words: ", positive)
print("Negative words: ", negative)

if(positive > negative):
  print("The review is positive")
elif (positive < negative):
  print("The review is negative")
else:
  print("The review is neutral")
```

Το περιεχόμενο της λίστας classified είναι το εξής:
[0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1]

Ωστόσο, παρατηρείται ότι με τον παραπάνω τρόπο αξιολόγησης η κριτική χαρακτηρίζεται ως ***ουδέτερη***, καθώς έχει 23 θετικές και 23 αρνητικές λέξεις. Υπενθυμίζεται ότι στο dataset η κριτική έχει κατηγοριοποιηθεί ως ***θετική***. Η εκτέλεση του παραπάνω κώδικα δίνει την εξής έξοδο:

Positive words: 23
Negative words: 23
The review is neutral

5.4 Αξιολόγηση θέτοντας κατώφλι

Ενδεχομένως οι 2 παραπάνω τρόποι αξιολόγησης να μη θεωρηθούν ιδιαίτερα κατάλληλοι για την ερμηνεία της εκτίμησης του μοντέλου. Μια εναλλακτική λύση θα ήταν να ληφθούν υπόψη μόνο τιμές, οι οποίες υπερβαίνουν μια τιμή - κατώφλι και στη συνέχεια να χρησιμοποιηθεί ένας από τους 2 παραπάνω τρόπους για την αξιολόγηση.

Με αυτόν τον τρόπο, λαμβάνονται υπόψη μόνο οι λέξεις που, σύμφωνα με το μοντέλο, προσδίδουν παραπάνω συναισθηματικό φόρτο στην κριτική. Στον παρακάτω κώδικα, διατηρούνται μόνο οι τιμές που είναι ίσες ή ξεπερνούν το 0.60:

```
# Keeping only the values which are equal or greater than the threshold and then
using the 2 above evaluation ways to interpret the output
threshold = 0.6
sorted_values = []

for item in prediction:
   if(item[0] >= threshold or item[1] >= threshold):
      sorted_values.append(item)
```

Ο πίνακας μετά την εφαρμογή του threshold είναι:

[0.6006154 0.39938462]
[0.36646938 0.6335306]
[0.6080683 0.3919317]
[0.37732375 0.6226762]
[0.19134864 0.8086514]
[0.6158053 0.38419467]
[0.3878549 0.61214507]

Χρησιμοποιώντας τους 2 προηγούμενους τρόπους για την αξιολόγηση των αποτελεσμάτων, η έξοδος που λαμβάνεται είναι η εξής:

Using the Mean Value:

44.96 % negative
55.03 % positive
The review is positive

By counting the positive and negative words:

Positive words: 4
Negative words: 3
The review is positive

Παρατηρείται ότι η χρήση του threshold δίνει καλύτερα αποτελέσματα, καθώς και οι 2 τρόποι δίνουν το ίδιο αποτέλεσμα, δηλαδή ότι η κριτική είναι **θετική**. Φυσικά, το threshold μπορεί να παραμετροποιηθεί και τα αποτελέσματα να αλλάξουν. Υπάρχουν διάφοροι τρόποι για να αξιολογηθεί η εκτίμηση του μοντέλου. Η επιλογή του ιδανικού τρόπου εξαρτάται και από το πλαίσιο στο οποίο θα χρησιμοποιηθεί το μοντέλο.

Μπορεί οι κριτικές να περάσουν από ακόμα ένα φιλτράρισμα πριν την εκπαίδευση, ώστε να χρησιμοποιηθούν μόνο οι λέξεις που θεωρείται ότι προσδίδουν μεγαλύτερη αξία στο νόημα και το αίσθημα που αποπνέει από την κριτική. Θα μπορούσαν να χρησιμοποιηθούν λεξικά, τα οποία προσδίδουν ακόμα περισσότερη πληροφορία στην κάθε λέξη, είτε η πληροφορία αφορά ένα εύρος συναισθημάτων σε μορφή λίστας, είτε αριθμητική αποτίμηση των συναισθημάτων. Βέβαια, αυτό αποτελεί μια εναλλακτική προσέγγιση με κάποιο έτοιμο λεξικό ή κάποια σχετική βιβλιοθήκη.

Υπάρχουν πολλές μέθοδοι για την αξιολόγηση των αποτελεσμάτων και την προεπεξεργασία των δεδομένων, οι οποίες δυνητικά θα μπορούσαν να αλλάξουν το αποτέλεσμα και τον τρόπο ερμηνείας της εκτίμησης του μοντέλου. Η ερμηνεία μπορεί να αλλάζει ανάλογα με την περίπτωση. Η σύνθεση του τρόπου αξιολόγησης υπόκειται στην κρίση του καθενός και την εφαρμογή στην οποία θα χρησιμοποιηθεί το εκάστοτε μοντέλο.

Κεφάλαιο 6ο

Αποδοτική εκπαίδευση

Η εκπαίδευση ενός μοντέλου τεχνητής νοημοσύνης μπορεί να αποτελέσει μια πραγματικά επίπονη διαδικασία για το εκάστοτε υπολογιστικό σύστημα. Εκτός αυτού, η εκπαίδευση μπορεί να αποδειχθεί ιδιαίτερα χρονοβόρα, γεγονός που δυσχεραίνει σημαντικά την όλη διαδικασία. Για αυτό το λόγο, έχουν αναπτυχθεί διάφορες μέθοδοι, οι οποίες αξιοποιούν συγκεκριμένες αρχιτεκτονικές και ειδικά κατασκευασμένους επιταχυντές.

Το Tensorflow, ως βιβλιοθήκη, υποστηρίζει πληθώρα στρατηγικών, οι οποίες χρησιμοποιούνται υπό διαφορετικές συνθήκες. Ως στρατηγική ορίζεται η μέθοδος εκπαίδευσης που πρόκειται να χρησιμοποιηθεί. Στο επόμενο κεφάλαιο πρόκειται να αναλυθούν οι διαθέσιμες στρατηγικές. Ωστόσο, είναι πολύ χρήσιμο αρχικά να αναλυθούν σε μεγαλύτερο βάθος οι υπολογισμοί που πραγματοποιούνται στα νευρωνικά δίκτυα, καθώς και οι 2 κύριοι τύποι επιταχυντών που χρησιμοποιούνται.

6.1 Γιατί υπάρχει η ανάγκη για αποδοτική εκπαίδευση;

Κάθε μοντέλο νευρωνικών δικτύων αποτελείται από κόμβους και επίπεδα. Σε πολλούς από τους κόμβους πραγματοποιούνται πολύπλοκες μαθηματικές πράξεις, οι οποίες προϋποθέτουν σημαντική υπολογιστική ισχύ, καθώς και υψηλά ποσοστά ενέργειας, τα οποία αφήνουν το αποτύπωμά τους στο περιβάλλον. Τέτοιου είδους πράξεις, για παράδειγμα, είναι οι πολλαπλασιασμοί πινάκων και ο υπολογισμός των συναρτήσεων ενεργοποίησης κάθε νευρώνα.

Αυτές οι πράξεις πολύ συχνά αφορούν πίνακες πολύ μεγάλων διαστάσεων, των οποίων ο υπολογισμός θα αργούσε σημαντικά στην περίπτωση που χρησιμοποιούταν μόνο η CPU. Όμως ακόμα και η χρήση πολυπύρηνων CPU δε βελτιώνει σημαντικά το χρόνο εκπαίδευσης, διότι εξ ορισμού η CPU έχει να εκτελέσει πολλές διαφορετικές εργασίες και διεργασίες. Για παράδειγμα, μια CPU δε δύναται να αφοσιωθεί πλήρως στον πολλαπλασιασμό δύο πινάκων, διότι έχει να φροντίσει για την ομαλή λειτουργία του λειτουργικού συστήματος και την επικοινωνία με τις περιφερειακές συσκευές.

Αυτό το γεγονός οδήγησε στη δημιουργία επιταχυντών, οι οποίοι αναλαμβάνουν τις υπολογιστικά απαιτητικές εργασίες, ώστε και να μειωθεί ο χρόνος εκπαίδευσης, αλλά και να χρησιμοποιηθούν ακόμα πιο αποδοτικά οι πόροι του υπολογιστικού συστήματος. Οι επιταχυντές όχι μόνο κάνουν πιο αποδοτική την εκπαίδευση, αλλά την καθιστούν κιόλας πιο φιλική προς το περιβάλλον, καταναλώνοντας λιγότερη ηλεκτρική ενέργεια. Οι 2 κύριοι επιταχυντές που χρησιμοποιούνται είναι οι GPUs και TPUs.

Ο σχεδιασμός των επιταχυντών τόσο σε επίπεδο hardware, όσο και σε επίπεδο software είναι αυτός που κάνει πραγματικά τη διαφορά στην αποδοτική εκπαίδευση.

6.2 Τι είναι η GPU και πως επιταχύνει τους υπολογισμούς;

Οι GPU ή αλλιώς Μονάδες Επεξεργασίας Γραφικών είναι συσκευές hardware, οι οποίες αρχικά σχεδιάστηκαν με σκοπό την ταχύτερη πραγματοποίηση πράξεων που απαιτούσαν τα γραφικά, αλλά πλέον χρησιμοποιούνται για υπολογισμούς γενικού σκοπού, δεδομένου ότι υπάρχει κάποιος φόρτος εργασίας.

Από την ταξινόμηση του Flynn είναι γνωστή η ιδέα του **Single Process Multiple Data** (SIMD), κατά την οποία υπάρχει ένας μεγάλος όγκος δεδομένων, στα οποία και πρέπει να εφαρμοστεί η ίδια πράξη. Για παράδειγμα, η αλλαγή της φωτεινότητας των χρωμάτων ενός παιχνιδιού απαιτεί την εφαρμογή της ίδιας πράξης στα 3 χρώματα από τα οποία αποτελείται ένα pixel.

Κατά την ίδια λογική, άρχισαν να κατασκευάζονται μονάδες επεξεργασίας γενικού σκοπού (GPGPU), οι οποίες πλέον χρησιμοποιούνται σε μια πληθώρα εφαρμογών, συμπεριλαμβανομένης της τεχνητής νοημοσύνης.

Όσον αφορά στα νευρωνικά δίκτυα, οι GPU επιστρατεύονται στο κομμάτι της πραγματοποίησης πράξεων με πίνακες. Για παράδειγμα, ένας πολλαπλασιασμός 2 πινάκων εντάσσεται στην κατηγορία SIMD, διότι επαναλαμβάνονται οι ίδιες εντολές για κάθε γραμμή και στήλη των 2 πινάκων.

Το πλεονέκτημα που προσφέρουν οι GPU σε σχέση με τις CPU είναι ότι διαθέτουν πολλαπλάσιο αριθμό αριθμό πυρήνων, οι οποίοι μπορούν να εκτελέσουν την ίδια πράξη πάνω σε διαφορετικά κομμάτια των πινάκων. Οι GPU γενικότερα διαθέτουν εκατοντάδες ή ακόμα και χιλιάδες πυρήνες, οι οποίοι εκτελούν παράλληλα την εκάστοτε επιθυμητή πράξη, εκτελώντας τις ίδιες εντολές σε υποπολλαπλάσιο χρόνο.

6.3 Τι είναι η TPU και πως επιταχύνει τους υπολογισμούς;

Η ανάγκη πολλαπλασιασμού πινάκων πολύ μεγάλων διαστάσεων οδήγησε τη Google στην ανάπτυξη hardware που εξειδικεύεται πάνω σε αυτό το κομμάτι. Το hardware αυτό ονομάζεται TPU (Tensor Processing Unit) ή Μονάδα Επεξεργασίας Τανυστών.

Οι TPU διαθέτουν μια Μονάδα Πολλαπλασιασμού Πινάκων (MXU ή Matrix Multiplication Unit). Η MXU παρέχει τη δυνατότητα πραγματοποίησης εκατοντάδων χιλιάδων υπολογισμών σε ένα χτύπο του ρολογιού. Αν κανείς παρομοιάσει μια CPU, μια GPU και μια TPU με εκτυπωτές, τότε η CPU θα εκτύπωνε τους χαρακτήρες έναν προς ένα, η GPU γραμμή προς γραμμή, ενώ η TPU θα εκτύπωνε ένα έγγραφο σελίδα προς σελίδα.

Οι TPU διαθέτουν ενσωματωμένα κυκλώματα, τα οποία ειδικεύονται στις πιο συνηθισμένες πράξεις που περιλαμβάνονται στα νευρωνικά δίκτυα. Οι εντολές μηχανής που λαμβάνουν πολλαπλασιάζουν απευθείας πίνακες μεγάλων διαστάσεων και υλοποιούν σε επίπεδο hardware τις πιο συνηθισμένες συναρτήσεις ενεργοποίησης νευρώνων, προσφέροντας ακόμα περισσότερη επιτάχυνση.

Πέραν από την επιτάχυνση των πράξεων, οι TPU είναι πολύ περισσότερο φιλικές προς το περιβάλλον και αποδοτικότερες ως προς την κατανάλωση ηλεκτρικής ενέργειας. Σε έρευνα που πραγματοποίησε η Google[2] σε συγκεκριμένα μοντέλα νευρωνικών δικτύων, κατέληξε στα εξής αποτελέσματα:

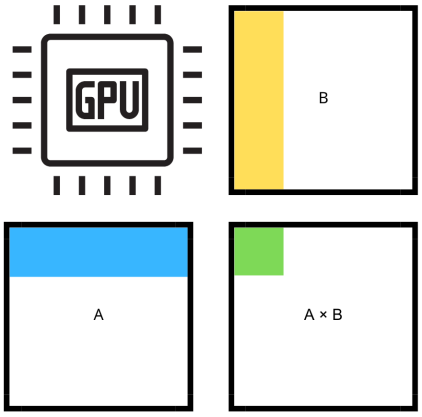
	Προβλέψεις / sec	**Απόδοση / Watt**
CPU	5.482	1
GPU	13.194	2.9
TPU	225.000	83

Πίνακας 6.1 Αποδοτικότητα CPU, GPU και TPU ως προς τις προβλέψεις

6.4 Πως υλοποιεί τον πολλαπλασιασμό πινάκων σε επίπεδο hardware η GPU;

Υπάρχουν πάρα πολλές μέθοδοι πολλαπλασιασμού πινάκων για τις GPU, οι οποίες έχουν ως στόχο να βελτιστοποιήσουν τη διαδικασία με σκοπό τη μέγιστη αποδοτικότητα. Όλες οι μέθοδοι προσπαθούν να ελαχιστοποιήσουν τη μεταφορά δεδομένων από τη μνήμη, να αξιοποιήσουν στο έπακρο τη μνήμη cache και να μεγιστοποιήσουν τα επίπεδα παραλληλισμού.

Ωστόσο, η πιο συνηθισμένη μέθοδος που ακολουθείται είναι ο διαχωρισμός του τελικού πίνακα σε μικρά πλακίδια και η δημιουργία νημάτων στη GPU. Το κάθε νήμα αναλαμβάνει να υπολογίσει το αναληφθέν πλακίδιο, εφόσον έχει παραλάβει τα κατάλληλα κομμάτια του πρώτου και του δεύτερου πίνακα.



Εικόνα 6.1 Πολλαπλασιασμός πινάκων σε GPU με διαχωρισμό του τελικού πίνακα σε πλακίδια

[2]https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

Για παράδειγμα, ένα νήμα της GPU θα λάμβανε το γαλάζιο κομμάτι του πίνακα A και το κίτρινο κομμάτι του πίνακα B, ώστε να υπολογίσει το πράσινο κομμάτι του πίνακα A × B. Το πόσο μπορεί να παραλληλιστεί ο πολλαπλασιασμός εξαρτάται από το πόσο ισχυρή είναι η εκάστοτε GPU και από το πόσα νήματα μπορεί να δημιουργήσει.

Παρόλο που ο συσχετισμός δεν είναι απόλυτα γραμμικός, διότι υπάρχουν και άλλες παράμετροι, όσα περισσότερα νήματα μπορεί να δημιουργήσει μια GPU, τόσα περισσότερα πλακίδια θα υπολογίσει παράλληλα και τόσο πιο σύντομα θα τελειώσει.

6.5 Πως υλοποιεί τον πολλαπλασιασμό πινάκων σε επίπεδο hardware η TPU;

Η TPU έχει υλοποιημένη την αρχιτεκτονική systolic array, σύμφωνα με την οποία το αποτέλεσμα των πολλαπλασιαζόμενων πινάκων αποθηκεύεται σε ένα πλέγμα κόμβων, του οποίου οι διαστάσεις είναι ίσες με τις διαστάσεις του πίνακα που προκύπτει μετά τον πολλαπλασιασμό. Κάθε κόμβος έχει αποθηκευμένο έναν αριθμό. Είναι χρήσιμο ο τρόπος λειτουργίας της αρχιτεκτονικής να παρουσιαστεί χρησιμοποιώντας ένα παράδειγμα. Έστω ο εξής πολλαπλασιασμός 2 πινάκων:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

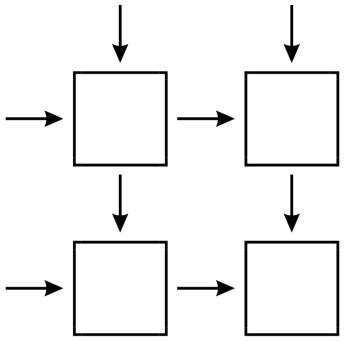
Εικόνα 6.2 Πολλαπλασιασμός δύο πινάκων διαστάσεων 2×2

Το αποτέλεσμα του πολλαπλασιασμού θα είναι αυτό που φαίνεται στην Εικόνα 6.3.

$$\begin{bmatrix} ae + bg & af + bh \\ ce + gd & cf + dh \end{bmatrix}$$

Εικόνα 6.3 Αποτέλεσμα του πολλαπλασιασμού 2 πινάκων διαστάσεων 2×2

Ο προκύπτων πίνακας είναι και αυτός διάστασης 2×2. Υπό πραγματικές συνθήκες, το πλέγμα που εκτελεί τις πράξεις εντός της TPU είναι μεγαλύτερο, αλλά για λόγους απλοποίησης θα θεωρηθεί ότι το κύκλωμα μιας TPU είναι και αυτό μεγέθους 2×2 όπως φαίνεται στην Εικόνα 6.4.



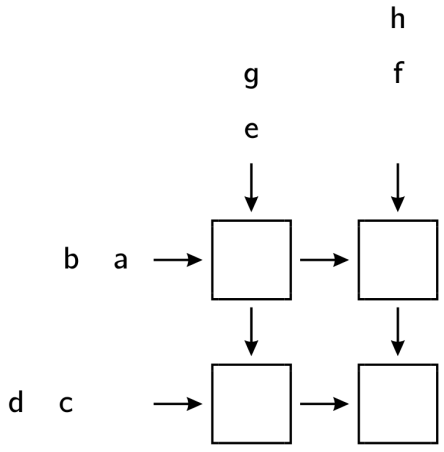
Εικόνα 6.4 Απλοποιημένη οπτικοποίηση της αρχιτεκτονικής systolic array

Κάθε κόμβος στο πλέγμα εκτελεί την πράξη D = A * B + C. Όπου A και B είναι οι αριθμοί που φτάνουν στον κόμβο και C είναι ο αριθμός που έχει αποθηκευμένο μέσα του ο κόμβος. Αναλυτικότερα κάθε κόμβος:

1. Λαμβάνει έναν αριθμό από πάνω του και έναν από αριστερά του (A και B)
2. Πολλαπλασιάζει τους 2 αριθμούς (A * B)
3. Αθροίζει το αποτέλεσμα με τον αριθμό που έχει από προηγούμενη πράξη και τον αποθηκεύει (A * B + C)
4. Στέλνει στον από κάτω κόμβο τον αριθμό που έλαβε άνωθεν
5. Στέλνει στον κόμβο προς τα δεξιά τον αριθμό που έλαβε από αριστερά

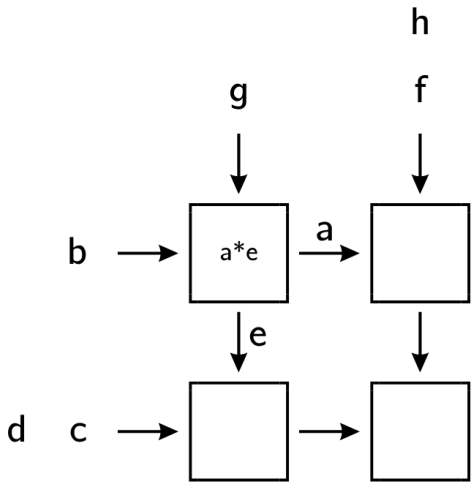
Οι κόμβοι λαμβάνουν σταδιακά τα στοιχεία των πινάκων που πρέπει να χρησιμοποιήσουν για τους υπολογισμούς. Η ταχύτητα αυτού του τρόπου έγκειται και στο γεγονός ότι τα στοιχεία των πινάκων διαβάζονται μόνο μια φορά από τη μνήμη, καθώς "ρέουν" από τον ένα κόμβο στον άλλο.

Η TPU τοποθετεί στο **αριστερό μέρος** του πλέγματος **τα στοιχεία των γραμμών του πρώτου** πίνακα και στο **πάνω μέρος** του πλέγματος **τα στοιχεία των στηλών του δεύτερου** πίνακα όπως φαίνεται στην Εικόνα 6.5



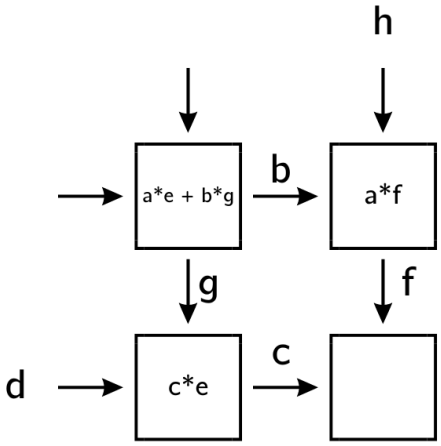
Εικόνα 6.5 Βήμα 0 του πολλαπλασιασμού πινάκων σε systolic array

Στη συνέχεια, κάθε κόμβος πρόκειται με τη σειρά του να εκτελέσει τα 5 παραπάνω βήματα, δημιουργώντας σταδιακά τον πίνακα του αποτελέσματος. Ουσιαστικά, ο πίνακας του αποτελέσματος ξεκινά υπολογίζοντας τα πιο πάνω αριστερά στοιχεία και συνεχίζει μέχρι να φτάσει στο κάτω δεξιά στοιχείο. Η διαδικασία αυτή απεικονίζεται παρακάτω (Εικόνα 6.6).



Εικόνα 6.6 Βήμα 1 του πολλαπλασιασμού πινάκων σε systolic array

Ο πάνω αριστερά κόμβος πραγματοποίησε τον πολλαπλασιασμό a*e και τον αποθήκευσε στον πάνω αριστερά κόμβο. Ύστερα, ο πάνω αριστερά κόμβος προωθεί τους αριθμούς που έλαβε στα κελιά που αντιστοιχούν προς την κατεύθυνση που κινείται κάθε αριθμός.

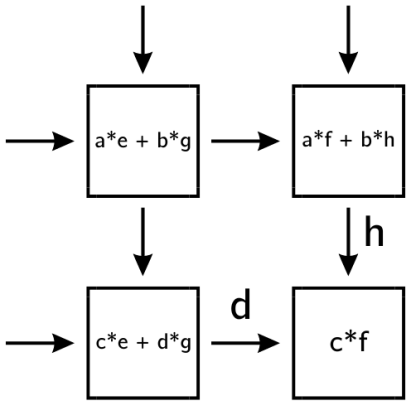


Εικόνα 6.7 Βήμα 2 του πολλαπλασιασμού πινάκων σε systolic array

Στο δεύτερο βήμα (Εικόνα 6.7), το πάνω αριστερά κελί έλαβε τους αριθμούς b και g, τους πολλαπλασίασε και άθροισε το αποτέλεσμα με το αποτέλεσμα της προηγούμενης πράξης (a*e). Οπότε το περιεχόμενο του πάνω αριστερά κελιού είναι πλέον το αποτέλεσμα του πολλαπλασιασμού της πρώτης γραμμής του πρώτου πίνακα με την πρώτη στήλη του δεύτερου πίνακα.

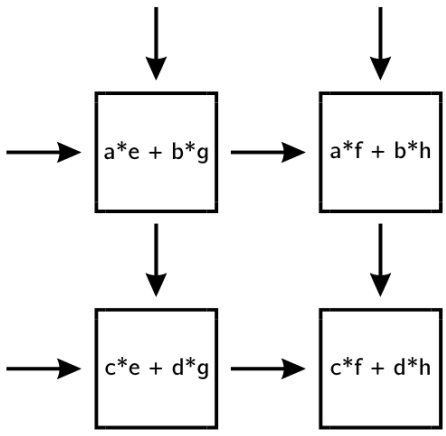
Ο πάνω αριστερά κόμβος, στη συνέχεια, προώθησε τους αριθμούς που έλαβε στους επόμενους κόμβους που αντιστοιχούν.

Ο πάνω δεξιά κόμβος έχει λάβει από το προηγούμενο βήμα του αριθμούς a και f, οπότε τους πολλαπλασιάζει και αποθηκεύει το αποτέλεσμα. Αντίστοιχα, ο κάτω αριστερά κόμβος έχει λάβει τους αριθμούς c και e, τους οποίους πολλαπλασιάζει, αποθηκεύοντας το αποτέλεσμα. Οι 2 αυτοί κόμβοι προωθούν τους αριθμούς στον αντίστοιχο επόμενο κόμβο. Αν δεν υπάρχει επόμενος κόμβος, τότε δεν πραγματοποιείται προώθηση.



Εικόνα 6.8 Βήμα 3 του πολλαπλασιασμού πινάκων σε systolic array

Στην Εικόνα 6.8, ο πάνω αριστερά κόμβος έχει ήδη τελειώσει τις πράξεις που έπρεπε να κάνει. Ο πάνω δεξιά κόμβος έλαβε τους αριθμούς b και h, τους πολλαπλασίασε και άθροισε το αποτέλεσμα με τον αριθμό που είχε ήδη αποθηκευμένο. Ο κάτω αριστερά έλαβε τους αριθμούς d και g, τους πολλαπλασίασε και άθροισε το αποτέλεσμα με τον αριθμό που είχε ήδη αποθηκευμένο. Ο κάτω δεξιά κόμβος είχε λάβει τους αριθμούς c και f, τους πολλαπλασίασε και τους αποθήκευσε. Τώρα μένει μόνο να λάβει τους αριθμούς που του στέλνουν οι άλλοι 2 κόμβοι, ώστε να τελειώσει και αυτός.



Εικόνα 6.9 Βήμα 4 του πολλαπλασιασμού πινάκων σε systolic array

Οπότε και μετά από 4 βήματα η TPU έχει ολοκληρώσει τον πολλαπλασιασμό δύο 2×2 πινάκων (Εικόνα 6.9).

5.6 Εκπαίδευση του μοντέλου με GPU και TPU

Σε αυτό το κομμάτι του κεφαλαίου, θα πραγματοποιηθεί σύγκριση του χρόνου εκπαίδευσης του πρώτου μοντέλου με GPU και TPU, αγνοώντας το overfitting. Σκοπός είναι να δημιουργηθεί μεγάλος φόρτος εργασίας, ώστε να εξεταστεί και σε πρακτικό επίπεδο η αποδοτικότητα του hardware. Το μοντέλο πρόκειται να εκπαιδευτεί σε 20 epochs.

Η GPU που χρησιμοποιήθηκε είναι η NVIDIA T4 Tensor Core και TPU v2.

Hardware	**Χρόνος σε sec**	**Χρόνος σε min**
GPU (T4)	835.27	13.92
TPU v2	826.88	13.78

Πίνακας 6.2 Αποτελέσματα εκπαίδευσης με GPU και TPU

Παρατηρείται (Πίνακας 6.2) ότι η χρήση της TPU για το συγκεκριμένο μοντέλο δεν επέφερε σημαντική επιτάχυνση της εκπαίδευσης, ακόμα και με πολλαπλάσιο φόρτο εργασίας. Πιο συγκεκριμένα, η μείωση του χρόνου εκπαίδευσης ανέρχεται σε μόλις περίπου 1%. Ωστόσο, αυτό δε σημαίνει ότι η TPU απέτυχε να εκπληρώσει το σκοπό για τον οποίο δημιουργήθηκε.

Πιο συγκεκριμένα, η GPU που χρησιμοποιήθηκε, υλοποιεί και αυτή ένα είδος της αρχιτεκτονικής systolic array, η οποία διαμερίζει τον τελικό πίνακα σε πλακίδια διαστάσεων 4×4, των οποίων ο πολλαπλασιασμός πραγματοποιείται εντός ενός tensor core. Η συγκεκριμένη GPU διαθέτει 320 tensor cores, οι οποίοι χρησιμοποιούνται αποκλειστικά για τον πολλαπλασιασμό πυκνών πινάκων και δουλεύουν παράλληλα σε διαφορετικά πλακίδια.

Εν τέλει, η διαφορά μεταξύ της συγκεκριμένης GPU και της TPU έγκειται στο ότι η GPU έχει αναπτυχθεί έχοντας κατά νου ότι πρόκειται να χρησιμοποιηθεί και για υπολογισμούς γενικού σκοπού, ενώ η TPU έχει δημιουργηθεί αποκλειστικά για την επιτάχυνση της εκπαίδευσης των νευρωνικών δικτύων και είναι ένα κύκλωμα που προορίζεται αποκλειστικά για χρήση σε συγκεκριμένες εφαρμογές (ASIC ή Application-Specific Integrated Circuit).

Αν το ίδιο μοντέλο εκπαιδευόταν σε GPU χωρίς tensor cores, τότε η διαφορά του χρόνου εκπαίδευσης θα ήταν πολλαπλάσια.

Κεφάλαιο 7ο

Κατανεμημένη εκπαίδευση

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, το Tensorflow παρέχει διάφορες μεθόδους εκπαίδευσης, ανάλογα με τον τύπο του υλισμικού και της αρχιτεκτονικής. Οι μέθοδοι εκπαίδευσης ή αλλιώς στρατηγικές, διαφέρουν ως προς την κατανομή του φόρτου εργασίας. Πιο συγκεκριμένα, η κατανομή μπορεί να γίνει:

- Σε πολλαπλές GPUs στο ίδιο μηχάνημα
- Σε πολλαπλές TPUs στο ίδιο μηχάνημα
- Σε πολλαπλά μηχανήματα με ομοιογενές ή μη hardware

Η εκπαίδευση μπορεί να γίνει με σύγχρονο ή ασύγχρονο τρόπο. Ο σύγχρονος τρόπος χωρίζει το φόρτο εργασίας σε μικρότερα κομμάτια και κάθε συσκευή hardware αναλαμβάνει ένα κομμάτι. Στη συνέχεια, πραγματοποιείται συγκέντρωση των αποτελεσμάτων και η διαδικασία επαναλαμβάνεται έως ότου ολοκληρωθούν οι υπολογισμοί. Αυτή η πρακτική ονομάζεται Data Parallelism, καθώς πραγματοποιείται παράλληλη εργασία σε διαφορετικά κομμάτια των δεδομένων. Η συγκεκριμένη πρακτική μπορεί να εφαρμοστεί σε κάθε τύπο μοντέλου.

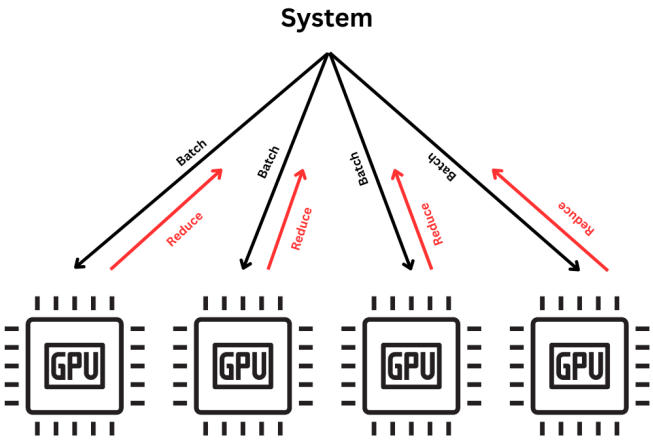
Στην περίπτωση που η εκπαίδευση γίνεται με ασύγχρονο τρόπο και πάλι ο φόρτος χωρίζεται σε μικρότερα κομμάτια, στέλνεται στις συσκευές και πραγματοποιείται και πάλι αναγωγή του αποτελέσματος. Η διαφορά έγκειται στο ότι τα βήματα της εκπαίδευσης πρέπει να είναι ανεξάρτητα μεταξύ τους, ώστε να μην πρέπει να περιμένουν οι συσκευές μέχρι να τελειώσουν όλα τα κομμάτια για να προχωρήσουν στο επόμενο βήμα. Αυτή η πρακτική ονομάζεται Model Parallelism, διότι το μοντέλο πραγματοποιεί διαφορετικούς ανεξάρτητους μεταξύ τους υπολογισμούς, των οποίων τα αποτελέσματα ενδεχομένως να χρησιμοποιηθούν συνδυαστικά σε κάποιο επόμενο βήμα. Το Model Parallelism χρησιμοποιείται σε πιο περίπλοκα μοντέλα. Στην πράξη, τις περισσότερες φορές χρησιμοποιείται η σύγχρονη εκπαίδευση.

Οι διαθέσιμες στρατηγικές που πρόκειται να αναλυθούν παρακάτω είναι οι εξής:

- MirroredStrategy
- TPUStrategy
- MultiWorkerMirroredStrategy
- ParameterServerStrategy
- CentralStorageStrategy

7.1 MirroredStrategy

Η εν λόγω στρατηγική χρησιμοποιείται για σύγχρονη εκπαίδευση σε ένα μηχάνημα με πολλαπλές GPU. Κάθε GPU κρατά ένα ξεχωριστό αντίγραφο των μεταβλητών που χρειάζεται. Η ενημέρωση όλων των μεταβλητών σε κάθε επιταχυντή πραγματοποιείται με τη μορφή ενημερώσεων της ειδικής μεταβλητής MirroredVariable. Για τη συγκέντρωση των αποτελεσμάτων, το σύστημα πραγματοποιεί αναγωγή, για την οποία και χρησιμοποιούνται πολύπλοκοι αλγόριθμοι, οι οποίοι μειώνουν σημαντικά το φόρτο της ενημέρωσης και του συγχρονισμού.



Εικόνα 7.1 Οπτικοποίηση της MirroredStrategy

Η χρήση αυτής της στρατηγικής είναι απλή. Απλώς, πρέπει να δοθούν οι GPU που πρόκειται να χρησιμοποιηθούν. Αν δεν δοθεί η παράμετρος devices, τότε χρησιμοποιούνται όσες GPU μπορούν να εντοπιστούν από το σύστημα εκτέλεσης. Για παράδειγμα, έστω το μοντέλο που αναπτύχθηκε σε προηγούμενο κεφάλαιο:

```
 # Mirrored Strategy
 strategy = tf.distribute.MirroredStrategy()

with strategy.scope():
  # Single LSTM Layer Model
  model1 = Sequential()
  model1.add(layers.Embedding(max_words, 20))
  model1.add(layers.LSTM(20,dropout=0.3))
  model1.add(layers.Dense(2,activation='softmax'))
  model1.compile(optimizer='rmsprop',loss='categorical_crossentropy',
  metrics=['accuracy'])

 checkpoint1 = ModelCheckpoint("best_model1.hdf5", monitor='val_accuracy',
```

```
verbose=1,save_best_only=True, mode='auto', period=1,save_weights_only=False)

history = model1.fit(X_train, y_train, epochs=20,validation_data=(X_test,
y_test),callbacks=[checkpoint1])
```

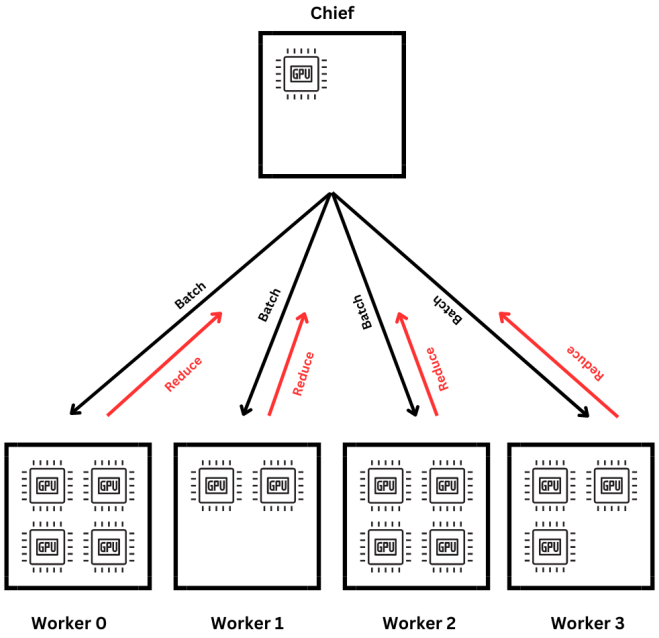
7.2 TPUStrategy

Η TPUStrategy πραγματοποιεί την ίδια ακριβώς διαδικασία με τη MirroredStrategy, με τη διαφορά ότι χρησιμοποιούνται TPU συσκευές και όχι GPU:

```
# TPU Strategy
strategy = tf.distribute.MirroredStrategy()

with strategy.scope():
  # Single LSTM Layer Model
  model1 = Sequential()
  model1.add(layers.Embedding(max_words, 20))
  model1.add(layers.LSTM(20,dropout=0.3))
  model1.add(layers.Dense(2,activation='softmax'))
  model1.compile(optimizer='rmsprop',loss='categorical_crossentropy',
  metrics=['accuracy'])

checkpoint1 = ModelCheckpoint("best_model1.hdf5", monitor='val_accuracy',
verbose=1,save_best_only=True, mode='auto', period=1,save_weights_only=False)

history = model1.fit(X_train, y_train, epochs=20,validation_data=(X_test,
y_test),callbacks=[checkpoint1])
```

7.3 MultiWorkerMirroredStrategy

Με τη συγκεκριμένη στρατηγική, μπορεί να πραγματοποιηθεί σύγχρονη κατανεμημένη εκπαίδευση σε πολλαπλά μηχανήματα, τα οποία ενδεχομένως να έχουν πολλαπλές GPU. Ουσιαστικά, πρόκειται για ομάδα μηχανημάτων, της οποίας τα μέλη, όπως και στη MirroredStrategy, κρατούν δικά τους αντίγραφα των μεταβλητών και εργάζονται σε διαφορετικά κομμάτια των υπολογισμών. Πραγματοποιείται και σε αυτήν την περίπτωση αναγωγή των υπολογισμών σε κάθε βήμα της εκπαίδευσης όπως φαίνεται στην Εικόνα 7.2.



Εικόνα 7.2 Οπτικοποίηση της MultiWorkerMirroredStrategy

Να σημειωθεί ότι στην παραπάνω εικόνα και ο chief θεωρείται και αυτός worker, αναλαμβάνοντας τις επιπλέον εργασίες, όπως τη δημιουργία των checkpoint και άλλα. Ωστόσο, θα πρέπει πρώτα να δημιουργηθεί μια μεταβλητή, η οποία θα περιέχει όλες τις απαραίτητες πληροφορίες για τους workers:

```
os.environ["TF_CONFIG"] = json.dumps(
{
       "cluster":{
           "worker": ["host1:port", "host2:port", "host3:port", "host4:port",
                      "host5:port"]
       },
       "task":{
           "type": "worker",
           "index": 0
       }
   }
)
```

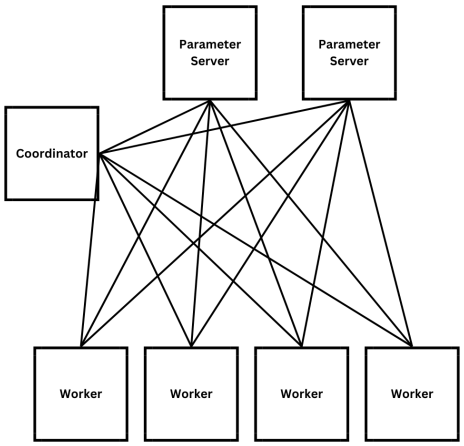
Πρακτικά, κάθε worker θα είχε διαφορετική διεύθυνση IP, οπότε θα έπρεπε κάθε worker να έχει δική του μεταβλητή με τις παραπάνω λεπτομέρειες, καθώς και στο task να λάβει τη δική του εργασία. Για παράδειγμα, ο πρώτος worker θα είχε "index": 1. Ακόμα, παρέχεται η δυνατότητα η επικοινωνία των μηχανημάτων να πραγματοποιηθεί με gRPC ή NCCL (NVIDIA Collective Communication Library).

Ο κώδικας για την εκπαίδευση παραμένει ο ίδιος. Το μόνο που αλλάζει είναι το είδος της στρατηγικής:

```
# MultiWorker Mirrored Strategy
strategy = tf.distribute.MultiWorkerMirroredStrategy()

with strategy.scope():
 # Single LSTM Layer Model
 model1 = Sequential()
 model1.add(layers.Embedding(max_words, 20))
 model1.add(layers.LSTM(20,dropout=0.3))
 model1.add(layers.Dense(2,activation='softmax'))
 model1.compile(optimizer='rmsprop',loss='categorical_crossentropy',
 metrics=['accuracy'])

checkpoint1 = ModelCheckpoint("best_model1.hdf5", monitor='val_accuracy',
verbose=1,save_best_only=True, mode='auto', period=1,save_weights_only=False)

history = model1.fit(X_train, y_train, epochs=20,validation_data=(X_test,
y_test),callbacks=[checkpoint1])
```

7.4 ParameterServerStrategy

Η συγκεκριμένη στρατηγική αναπτύχθηκε για μοντέλα μεγάλου μεγέθους και στηρίζεται στη δικτύωση. Εφαρμόζεται το Data Parallelism, όπως στη MirroredStrategy. Η εργασία και πάλι χωρίζεται σε μικρότερα κομμάτια, τα οποία αναλαμβάνουν οι Workers. Η διαφορά έγκειται στο ότι υπάρχει ένας συντονιστής (Coordinator), ο οποίος αναλαμβάνει να χωρίσει την εργασία σε κομμάτια και να πραγματοποιήσει τις αναθέσεις στους Workers. Οι Workers λαμβάνουν όλες τις μεταβλητές και τις τιμές που χρειάζονται από τους Parameter Servers (Εικόνα 7.3).

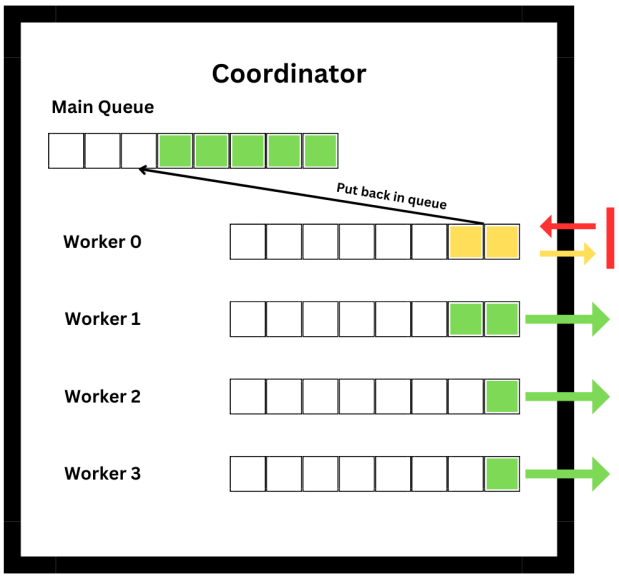


Εικόνα 7.3 Οπτικοποίηση της ParameterServerStrategy

Λόγω του ότι οι μεταβλητές και οι πληροφορίες που χρειάζεται ένα μοντέλο μεγάλου μεγέθους, ενδεχομένως να μη μπορούν να αποθηκευτούν σε ένα μηχάνημα, οι Parameter Servers αναλαμβάνουν την αποθήκευση και την αποστολή των δεδομένων στους κατάλληλους Workers. Οι Workers πραγματοποιούν τους απαραίτητους υπολογισμούς και επιστρέφουν τα αποτελέσματα στους Parameter Servers, όπου και πραγματοποιείται αναγωγή.

Για τον καταμερισμό της εργασίας, ο Coordinator δημιουργεί στο εσωτερικό του μια ουρά από εργασίες, τις οποίες και αναθέτει στους Workers. Οι Workers λειτουργούν ως αυτόνομες μονάδες, οι οποίες δεν επικοινωνούν με άλλους Workers. Οι μόνες επικοινωνίες που πραγματοποιούνται είναι με τον Coordinator για να λάβουν τις εργασίες που τους ανατέθηκαν και η ανάκτηση και αποθήκευση των δεδομένων στους Parameter Servers. Με αυτόν τον τρόπο, επιτυγχάνεται υψηλή ανεκτικότητα όσον αφορά στα σφάλματα των Workers, καθώς ο Coordinator είναι σε θέση να αντιληφθεί εύκολα και γρήγορα αν ένας Worker λειτουργεί με μη αναμενόμενο τρόπο. Ωστόσο, πρέπει να διασφαλίζεται από τους διαχειριστές ότι ο Coordinator λειτουργεί κανονικά, καθώς το σύστημα ενδεχομένως να καταστεί μη λειτουργικό. Για αυτό το λόγο, πρέπει να δημιουργούνται συχνά checkpoints, ώστε αφού επανέλθει και πάλι ο Coordinator και συνδεθεί με τα υπόλοιπα μέλη, οι εργασίες να συνεχίσουν από εκεί που σταμάτησαν.

Για παράδειγμα στην Εικόνα 7.4, ο Coordinator έχει πάρει από την ουρά εργασιών 2 εργασίες για τον Worker 0 και τις έχει προσθέσει στη δική του ουρά. Κατά την προσπάθειά του να αποστείλει τις εργασίες στον Worker 0 προκύπτει κάποιο σφάλμα. Σε αυτήν την περίπτωση, προσθέτει πίσω στην κύρια ουρά τις εργασίες που είχε αναθέσει στον Worker 0, με σκοπό να τις αναθέσει σε κάποιον άλλο.



Εικόνα 7.4 Οπτικοποίηση της λογικής του Coordinator

Για να εφαρμοστεί αυτή η στρατηγική:

1. Αρχικά, πρέπει να οριστεί το είδος της στρατηγικής

2. Να οριστεί το cluster των μηχανημάτων, καθώς και ο τρόπος τεμαχισμού του dataset
3. Να πραγματοποιηθεί compilation του μοντέλου
4. Να γίνει fitting του μοντέλου, να οριστούν τα epoch, καθώς και τα βήματα ανά epoch

```
# Prepare a strategy to use with the cluster and variable partitioning info.
strategy = tf.distribute.experimental.ParameterServerStrategy(
    cluster_resolver=...,
    variable_partitioner=...)

# A dataset function takes a `input_context` and returns a `Dataset`
def dataset_fn(input_context):
  dataset = tf.data.Dataset.from_tensors(...)
  return dataset.repeat().shard(...).batch(...).prefetch(...)

# With `Model.fit`, a `DatasetCreator` needs to be used.
input = tf.keras.utils.experimental.DatasetCreator(dataset_fn=...)

with strategy.scope():
  model = ...  # Make sure the `Model` is created within scope.
model.compile(optimizer=..., loss=..., steps_per_execution=..., ...)

# Optional callbacks to checkpoint the model, back up the progress, etc.
callbacks = [tf.keras.callbacks.ModelCheckpoint(...), ...]

# `steps_per_epoch` is required with `ParameterServerStrategy`.
model.fit(input, epochs=..., steps_per_epoch=..., callbacks=callbacks)
```

7.5 CentralStorageStrategy

Πρόκειται για μια απλούστερη στρατηγική, η οποία μοιάζει πολύ με τη MirroredStrategy. Σε αυτήν την περίπτωση, υπάρχει μόνο ένα μηχάνημα, το οποίο ενδεχομένως να έχει πολλαπλές GPU. Όλες οι μεταβλητές και οι τιμές τους δημιουργούνται, αποθηκεύονται και ενημερώνονται από τη CPU του μηχανήματος. Ουσιαστικά, η ενημέρωση όλων των μεταβλητών λαμβάνει χώρα σε ένα μόνο μέρος, εξαλείφοντας την αναγκαιότητα χρήσης allreduce αλγορίθμων.

Σε σύγκριση με τη MirroredStrategy, είναι απλούστερη, διότι δε χρειάζεται να γίνει αναγωγή. Η MirroredStrategy δημιουργεί για κάθε GPU ένα ξεχωριστό αντίγραφο μεταβλητών. Σε κάθε βήμα πρέπει να γίνει ενημέρωση όλων των μεταβλητών σε όλες τις συσκευές, ενώ η CentralStorageStrategy διατηρεί τις μεταβλητές σε ένα κεντρικό σημείο, καθιστώντας έτσι την ενημέρωση των μεταβλητών απλή.

Κεφάλαιο 8ο

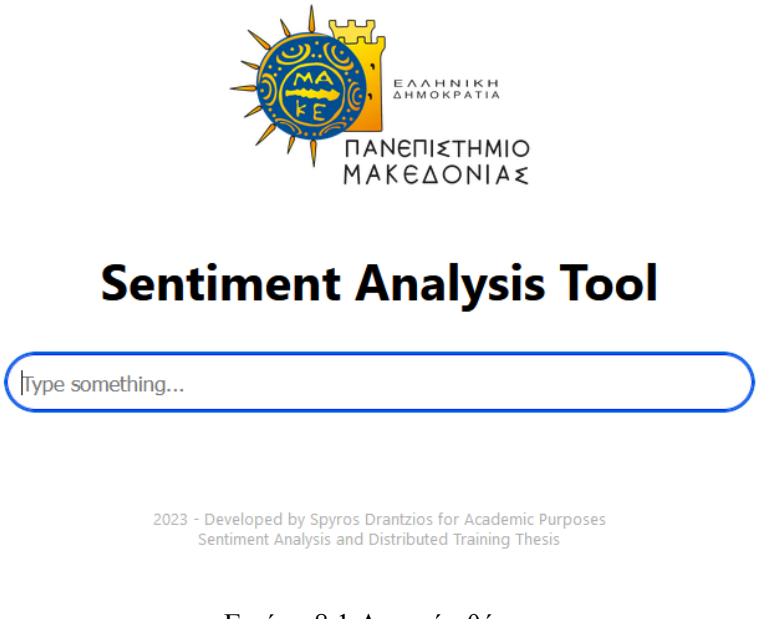
Σενάριο Χρήσης

Κάθε μοντέλο τεχνητής νοημοσύνης αναπτύσσεται έχοντας κατά νου κάποιο σκοπό. Σκοπός της συγκεκριμένης εργασίας είναι δοκιμαστεί το μοντέλο που αναπτύχθηκε και σε πρακτικό επίπεδο. Να χρησιμοποιηθεί και από απλούς χρήστες και να πειραματιστούν με το εργαλείο. Για αυτόν το λόγο, δημιουργήθηκε ένας server, ο οποίος υλοποιεί την αξιολόγηση κειμένου και επιστρέφει την απάντηση (response) στον τελικό χρήστη. Πιο συγκεκριμένα:

- Ο server υλοποιήθηκε σε Python με το Flask, το οποίο είναι framework για την κατασκευή API
- Ο server παραλαμβάνει το κείμενο που έχει πληκτρολογήσει ο χρήστης
- Εκτελεί την κατάλληλη επεξεργασία και προετοιμασία του κειμένου
- Το ήδη εκπαιδευμένο μοντέλο εκτελεί τις προβλέψεις
- Ο server χρησιμοποιεί 2 από τους τρόπους που αναφέρθηκαν στο κεφάλαιο 5 (αξιολόγηση με το μέσο όρο και αξιολόγηση κατά λέξη) για να παράσχει ερμηνεία των προβλέψεων στο χρήστη
- Δημιουργεί και επιστρέφει στον τελικό χρήστη ένα response, το οποίο περιέχει την ερμηνεία του αποτελέσματος και πληροφορίες σχετικά με αυτήν

8.1 Τι βλέπει ο χρήστης

Ανοίγοντας τη σελίδα, ο χρήστης βλέπει απευθείας μια φόρμα, στην οποία μπορεί να πληκτρολογήσει το κείμενο που επιθυμεί (Εικόνα 8.1).



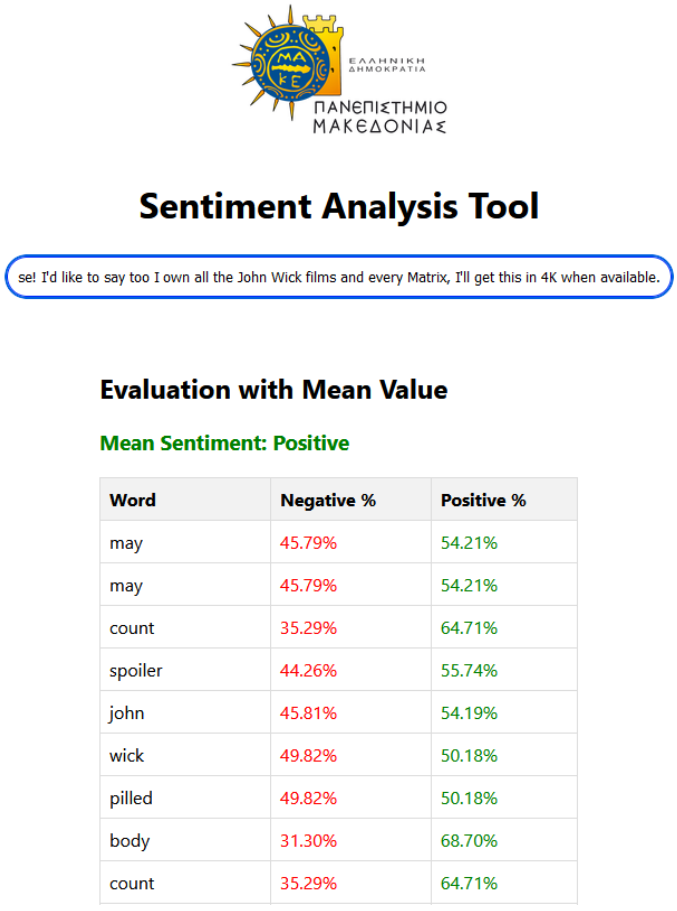
Εικόνα 8.1 Αρχική οθόνη

Έστω ότι ο χρήστης εισάγει το εξής κείμενο:

«May or may not count as a spoiler but John Wick pilled up more of a body count than Jason Vorhees sure different genre yet both great killing machines! Chad Staleski created such a great action extravaganza quad of films and one of my favorite trivia's he was the stunt doable for Keanu Reeves in the The Matrix back in the day; fast forward to 2014 they work together on one of the greatest action films of our generation. John Wick 4 is ultimate experience that is a must see I can't praise enough for how much fun it is.

I might get vibes for my thoughts I tried not to spoil it's obvious the combination of kill count has added up the rest left to the viewers, the action and writing is incredible for Chapter 4 that everyone involved deserves praise! I'd like to say too I own all the John Wick films and every Matrix, I'll get this in 4K when available.»

Αφού πατήσει Enter, τότε αρχικά βλέπει την αξιολόγηση του κειμένου με βάση το μέσο όρο. Για κάθε λέξη εμφανίζεται η εκτίμηση όπως φαίνεται στην Εικόνα 8.2.



Εικόνα 8.2 Αποτελέσματα μέσου όρου

Στην Εικόνα 8.3, στο κάτω μέρος του πίνακα, εμφανίζεται ο μέσος όρος για κάθε στήλη του πίνακα. Για το συμπέρασμα λαμβάνεται η μεγαλύτερη εκ των δύο τιμών, εκτός και αν είναι ίσες, που τότε αξιολογείται ως ουδέτερο.

film	37.29%	62.71%
every	66.27%	33.73%
matrix	51.08%	48.92%
get	49.12%	50.88%
4k	49.82%	50.18%
available	48.35%	51.65%
Mean	0.49	0.51

Εικόνα 8.3 Μέσοι όροι

Αντίστοιχα στην Εικόνα 8.4 εμφανίζεται και η αξιολόγηση του κειμένου λαμβάνοντας υπόψη την κατηγοριοποίηση της κάθε λέξης.

Evaluation with Counting Occurrences

Count Sentiment: Positive

Word	Negative	Positive
may		positive
may		positive
count		positive
spoiler		positive
john		positive
wick		positive
pilled		positive
body		positive
count		positive
jason		positive
vorhees		positive
sure		positive

Εικόνα 8.4 Αποτελέσματα καταμέτρησης ετικετών

Και σε αυτήν την περίπτωση, στο κάτω μέρος του πίνακα εμφανίζεται το πλήθος των θετικών και των αρνητικών λέξεων. Το συμπέρασμα προκύπτει και πάλι από το μεγαλύτερο εκ των δύο αριθμών. Αν οι

αριθμοί είναι ίσοι, τότε το κείμενο αξιολογείται ως ουδέτερο. Σε αυτό το παράδειγμα, το κείμενο έχει αξιολογηθεί ως θετικό (Εικόνα 8.5).

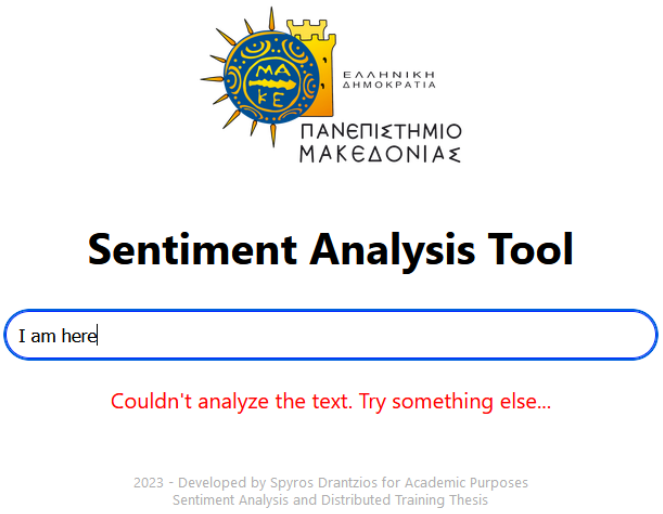
every	negative	
matrix	negative	
get		positive
4k		positive
available		positive
Count	30	62

Εικόνα 8.5 Άθροισμα ετικετών

Σε περίπτωση που το μοντέλο δεν είναι σε θέση να εξάγει ένα συμπέρασμα από το κείμενο που έχει δοθεί, τότε εμφανίζεται μήνυμα σφάλματος. Το μοντέλο μπορεί να μην είναι σε θέση να πραγματοποιήσει πρόβλεψη για διάφορους λόγους. Για παράδειγμα:

- Ο χρήστης έχει πληκτρολογήσει λίγες λέξεις
- Οι λέξεις που έχουν πληκτρολογηθεί κατατάσσονται όλες στην κατηγορία των stop words
- Υπάρχει κάποιο σφάλμα από πλευράς του server

Μια τέτοια περίπτωση είναι η αποστολή ενός request, το οποίο αφορά το κείμενο “I am here”, το οποίο αποτελείται μόνο από stop words, τα οποίο αφαιρούνται κατά την προεπεξεργασία του κειμένου. Ως εκ τούτου, προκαλείται σφάλμα, διότι το μοντέλο δε δύναται να αξιολογήσει, εφόσον δεν έχει στη διάθεσή του κείμενο (Εικόνα 8.6)



Εικόνα 8.6 Οθόνη σφάλματος

8.2 Από την πλευρά του server

Ο server αποτελεί ουσιαστικά ένα απλό API, στο οποίο ο client στέλνει αίτημα στο “.../evaluate” μαζί με την παράμετρο “text” και ο server επιστρέφει ένα αρχείο json με την αξιολόγηση. Για την αξιολόγηση του κειμένου έχει δημιουργηθεί ένα ξεχωριστό αντικείμενο evaluator της κλάσης Evaluator, το οποίο περιλαμβάνει μεθόδους που υλοποιούν την προεπεξεργασία και την προετοιμασία των δεδομένων, όπως έγινε στο πρώτο κεφάλαιο. Ο κώδικας του server:

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from evaluator import Evaluator

# Creating the app, editing the CORS policy and initializing the evaluator
app = Flask("YouTube Sentiment Analysis")
CORS(app)
evaluator = Evaluator()

@app.route("/evaluate", methods=["GET"])
def evaluate_text():
  text = request.args.get("text")

  if text is None:
      return jsonify({"error": "Please provide a 'text' parameter in the query."},
      400)

  response = evaluator.evaluate(text)
  return jsonify(response)

# Starting the server
app.run()
```

8.3 Περαιτέρω εξέλιξη της ιδέας

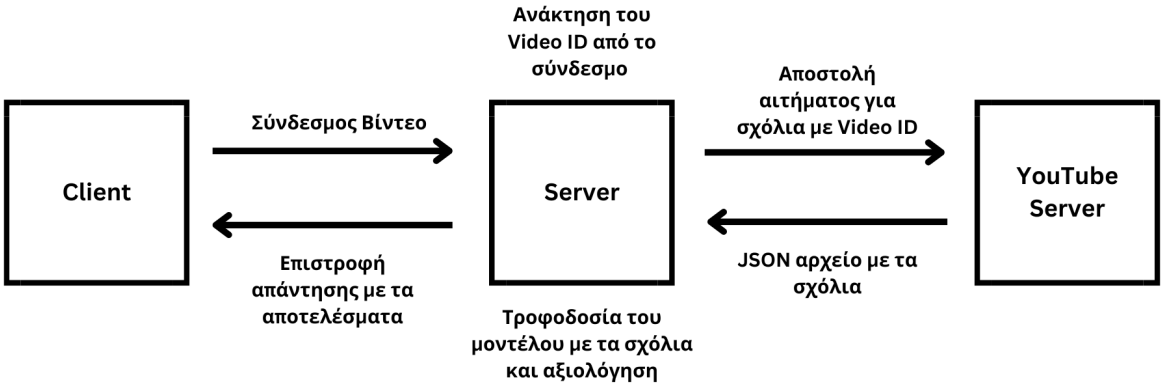
Η λειτουργικότητα της εφαρμογής θα μπορούσε να επεκταθεί περισσότερο. Για παράδειγμα, θα μπορούσαν να αξιοποιηθούν APIs από πλατφόρμες και social media, όπως το YouTube και το TikTok, ώστε να πραγματοποιείται ανάκτηση και αξιολόγηση των σχολίων από κάθε βίντεο και κανάλι. Σκοπός αυτού θα ήταν η παροχή περισσότερων πληροφοριών σχετικά με το πως νιώθουν οι χρήστες για κάποιο δημιουργό περιεχομένου και ούτω καθεξής.

Αυτό θα προϋπέθετε να γίνουν περαιτέρω βελτιστοποιήσεις σε διάφορους τομείς. Πιο συγκεκριμένα:

- Το χρησιμοποιούμενο μοντέλο θα έπρεπε να εκπαιδευτεί με δεδομένα που προέρχονται από την κατάλληλη πλατφόρμα, ώστε να αποδίδει καλύτερα σε τέτοιου είδους περιεχόμενο.
- Να ενταχθεί στην εκπαίδευση ο συνεχής αυξανόμενος αριθμός από emoji και περιπτώσεις στις οποίες χρησιμοποιούνται, ώστε να δημιουργηθούν οι κατάλληλες συσχετίσεις.
- Το training set να περιέχει δεδομένα από περισσότερες γλώσσες, διότι πλέον το ψηφιακό περιεχόμενο κυκλοφορεί σε παγκόσμια κλίμακα, γεγονός που έχει ως αποτέλεσμα αρκετά σχόλια σε ένα βίντεο να είναι και σε άλλες γλώσσες εκτός της Αγγλικής (π.χ. Ισπανικά).
- Να χρησιμοποιούνται ξεχωριστά λεξικά, μέθοδοι λημματοποίησης και tokenizers και για τις υπόλοιπες γλώσσες, ώστε να μπορεί να γίνει η κατάλληλη προετοιμασία του κειμένου
- Να δημιουργηθεί μια πιο στιβαρή μέθοδος αξιολόγησης του κειμένου (π.χ. συνδυασμός των ανωτέρω μεθόδων), η οποία ενδεχομένως να χρησιμοποιεί και βάρη για τις λέξεις μέσα στις προτάσεις

8.4 Ανάκτηση σχολίων μέσω του YouTube Data API v3

Ένα πρώτο βήμα για την επέκταση της λειτουργικότητας είναι η ανάκτηση των σχολίων από βίντεο στο YouTube. Ειδικότερα, ο τελικός χρήστης θα πρέπει να είναι σε θέση να παρέχει το σύνδεσμο προς κάποιο βίντεο και η εφαρμογή να μπορεί να πραγματοποιεί αξιολόγηση όλων ή ενός πλήθους των σχολίων. Η διαδικασία απεικονίζεται παρακάτω:



Εικόνα 8.7 Ανάκτηση σχολίων από το API του YouTube

Για την ανάκτηση των σχολίων, δημιουργήθηκε για το server η κλάση YouTubeCommentScraper, της οποίας ένα αντικείμενο λαμβάνει ένα σύνδεσμο και πραγματοποιεί τη διαδικασία. Πιο συγκεκριμένα:

- Καλείται η μέθοδος scrape_comments, στην οποία παράμετρος είναι ο σύνδεσμος του βίντεο.
- Καλείται η μέθοδος get_video_id, η οποία πραγματοποιεί parsing του αναγνωριστικού του βίντεο από το σύνδεσμο.
- Στη συνέχεια, έχοντας το Video ID, η scrape_comments στέλνει κατάλληλο αίτημα στο API του YouTube και επιστρέφει τη λίστα με τα 100 πρώτα σχόλια. Ο αριθμός των σχολίων είναι ενδεικτικός και μπορεί να προσαρμοστεί ανάλογα.

Ο κώδικας της κλάσης:

```
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from urllib.parse import urlparse, parse_qs

class YouTubeCommentScraper:

   def __init__(self):
       self.api_key = "API_KEY"
       self.youtube = build("youtube", "v3", developerKey=self.api_key)

   def get_video_id(self, value):
       query = urlparse(value)
       # Parsing the video ID
       if query.hostname == 'youtu.be':
           return query.path[1:]
       if query.hostname in ('www.youtube.com', 'youtube.com'):
           if query.path == '/watch':
               p = parse_qs(query.query)
               return p['v'][0]
           if query.path[:7] == '/embed/':
               return query.path.split('/')[2]
           if query.path[:3] == '/v/':
               return query.path.split('/')[2]
       # Failed to get the video ID
       return None

   def scrape_comments(self, url):
       video_id = self.get_video_id(url)
       try:
           # Make the API request to retrieve comments
           response = self.youtube.commentThreads().list(
               part="snippet",
               videoId=video_id,
               textFormat="plainText",
               maxResults=100).execute()

           comments = []
           # Saving the comments into the comments array
           while response:
               for item in response["items"]:
```

```
                comment =
                item["snippet"]["topLevelComment"]["snippet"]["textDisplay"]
                comments.append(comment)

            if "nextPageToken" in response:
                response = self.youtube.commentThreads().list(
                    part="snippet",
                    videoId=video_id,
                    textFormat="plainText",
                    maxResults=100,
                    pageToken=response["nextPageToken"]
                ).execute()
            else:
                break

        return comments

    except HttpError as e:
        return "Something went wrong"
```

Η λειτουργικότητα και η χρήση του μοντέλου μπορεί να επεκταθεί με πολλούς τρόπους. Οι παράμετροι που πρέπει να ληφθούν υπόψη είναι αρκετές, διότι πάντοτε υπάρχει και ο απρόβλεπτος παράγοντας. Φυσικά, όσο περισσότερα τα δεδομένα της εκπαίδευσης, τόσο καλύτερη θα είναι και η απόδοση του μοντέλου.

8.5 Φορητότητα του server

Η μεταφορά του κώδικα από ένα μηχάνημα σε ένα άλλο συχνά αποτελεί πρόβλημα, διότι πρέπει να γίνουν πολλές ρυθμίσεις. Για αυτό το λόγο, πολλές φορές ακόμα και σε εμπορικές εφαρμογές, προτιμάται η μεταφορά του κώδικα σε έναν περιέκτη (container), ώστε να "τρέχει" σε οποιοδήποτε περιβάλλον εκτέλεσης. Ένα container αποτελεί ουσιαστικά ένα πακέτο που περιλαμβάνει τον κώδικα και όλες τις εξαρτήσεις του. Το container είναι μικρότερο σε μέγεθος από μια εικονική μηχανή και όταν εκτελείται αποτελεί αυτόνομη διεργασία.

Για τους ανωτέρω λόγους, καθώς και για να είναι εύκολη η δοκιμή και η αλληλεπίδραση με το εκπαιδευμένο μοντέλο, δημιουργήθηκε ένα container, το οποίο περιλαμβάνει όλα τα περιεχόμενα του server. Όλα τα απαραίτητα αρχεία είναι διαθέσιμα σε αποθετήριο στο Github.[3] Το Dockerfile, που περιέχει τις οδηγίες σύνθεσης έχει τον εξής κώδικα:

[3] https://github.com/spyrosdran/Thesis

```
FROM python:3.9
COPY ./requirements.txt requirements.txt

RUN pip install tensorflow
RUN pip install --upgrade -r ./requirements.txt

COPY app.py .
COPY evaluator.py .
COPY IMDB_Dataset.csv .
COPY LSTM.hdf5 .
COPY BiLSTM.hdf5 .

CMD ["flask", "run", "--host=0.0.0.0", "--port", "5000"]
```

Στον παραπάνω κώδικα, ορίζεται ότι πρόκειται να χρησιμοποιηθεί ως βάση μια εικόνα (image) ενός container, η οποία έχει ήδη την Python 3.9. Στο αρχείο requirements.txt περιέχονται όλες οι εξαρτήσεις του κώδικα. Το αρχείο αντιγράφεται εντός του container και έπειτα με την εντολή RUN, στις 2 επόμενες γραμμές εγκαθίστανται όλα τα απαιτούμενα πακέτα. Το Tensorflow εγκαθίσταται ξεχωριστά, διότι ενδεχομένως το container που χτίζεται να χρειάζεται κάποια έκδοση, η οποία να υποστηρίζει την εκάστοτε αρχιτεκτονική του μηχανήματος.

Στις επόμενες 5 σειρές, αντιγράφονται όλα τα απαραίτητα αρχεία που πρέπει να έχει ο server για να ξεκινήσει να λειτουργεί. Αντιγράφονται και τα 2 μοντέλα που έχουν εκπαιδευτεί, ώστε να επιλεχθεί το επιθυμητό, όμως από προεπιλογή στον κώδικα χρησιμοποιείται το εκπαιδευμένο μοντέλο που επιστρατεύει το BiLSTM layer.

Στην τελευταία γραμμή, ορίζεται ότι κατά την εκκίνηση, το container θα εκτελεί στη γραμμή εντολών την εξής εντολή η οποία είναι απαραίτητη για την εκκίνηση του server:

```
flask run –host=0.0.0.0 –port 5000
```

Το docker image χτίζεται με την παρακάτω εντολή, έχοντας ανοίξει τη γραμμή εντολών στο φάκελο του server:

```
docker build -t sentiment-analysis .
```

Για να ξεκινήσει ένα container από το image που δημιουργήθηκε, χρησιμοποιείται η εξής εντολή:

```
docker run -p 5000:5000 sentiment-analysis
```

Στην παραπάνω εντολή πραγματοποιείται αντιστοίχηση της θύρας 5000 του container, με τη θύρα 5000 του μηχανήματος. Οπότε αφού ξεκινήσει η εκτέλεση του container, μπορεί κανείς να αλληλεπιδράσει με την εφαρμογή, ανοίγοντας το αρχείο index.html, το οποίο βρίσκεται στο φάκελο Client.

Κεφάλαιο 9

Συμπεράσματα

Σε μια εποχή που πραγματοποιούνται εξαιρετικές προσπάθειες οι υπολογιστές να έρθουν πιο κοντά στον άνθρωπο, η επεξεργασία φυσικής γλώσσας έχει γίνει πιο επιτακτική από ποτέ. Η δύναμη της τεχνητής νοημοσύνης γεφυρώνει το χάσμα και προσφέρει περισσότερες δυνατότητες.

Ωστόσο, η επεξεργασία φυσικής γλώσσας αποδεικνύεται ένα αρκετά σύνθετο πρόβλημα, καθώς είναι πολλοί οι παράγοντες που πρέπει να ληφθούν υπόψη. Η αγγλική γλώσσα είναι ένα πάρα πολύ καλό σημείο εκκίνησης για την ανάλυση συναισθήματος, αλλά αποδεικνύεται ανεπαρκής, όταν πρόκειται το εκάστοτε μοντέλο να χρησιμοποιηθεί σε πραγματικές συνθήκες, όπου πλέον το περιεχόμενο που παράγεται χαρακτηρίζεται από μεγάλη ποικιλία γλωσσών, εικονιδίων, emoji και όλους τους νέους τρόπους, τους οποίους οι χρήστες του διαδικτύου εφευρίσκουν, ώστε το κείμενο που γράφουν να γίνεται όλο και πιο εκφραστικό.

Η ανάπτυξη μοντέλων υψηλής ακρίβειας είναι μια διαδικασία, η οποία χρήζει μελέτης εις βάθος, καθώς πρέπει η εκπαίδευση να γίνει σε ρεαλιστικό χρόνο και το μοντέλο να μπορεί να λειτουργεί αποδοτικά, τόσο από άποψη χρόνου, όσο και από άποψη ενέργειας. Η εξέλιξη του hardware έχει επιφέρει δραστικές αλλαγές στους 2 ανωτέρω παράγοντες και ειδικότερα η χρήση της αρχιτεκτονικής systolic array, μαζί με τις βελτιστοποιήσεις που επιδέχεται. Οι TPU και οι Tensor Cores το έχουν καταφέρει αυτό σε υψηλό επίπεδο, καθώς τα αποτελέσματα των μετρήσεων της αποδοτικότητάς τους είναι εντυπωσιακά.

Θα ήταν εξαιρετικά ενδιαφέρον να πραγματοποιηθούν πειράματα, τα οποία θα αξιοποιούσαν όλες τις στρατηγικές εκπαίδευσης που προσφέρει το Tensorflow, όμως αυτό προϋποθέτει την ύπαρξη κατάλληλης υποδομής. Μπορεί να χρησιμοποιηθεί το Google Cloud για την κλιμάκωση των πειραμάτων, αλλά η παραμετροποίηση που πρέπει να γίνει, ξεπερνά τα όρια αυτής της εργασίας. Όμως η κατανεμημένη εκπαίδευση ενός μοντέλου είναι ένα άκρως ενδιαφέρον θέμα, το οποίο θα μπορούσε να αποτελεί ολόκληρο βιβλίο, το οποίο θα αναλύει σε μεγαλύτερο βάθος όλες τις πιθανές επιλογές, τη συνδεσιμότητα των κόμβων, καθώς και όλους τους διαφορετικούς αλγορίθμους αναγωγής των αποτελεσμάτων, οι οποίοι επηρεάζουν σημαντικά τους χρόνους εκπαίδευσης.

Εν τέλει, οι χρήστες ενός τέτοιου μοντέλου θα επιζητούσαν χρήσιμα συμπεράσματα. Αυτό θα προϋπέθετε την επέκταση της εφαρμογής, ώστε να διασυνδέεται με διαφορετικές πλατφόρμες, με σκοπό την ανάκτηση δεδομένων σε πραγματικό χρόνο. Επιπλέον, το μοντέλο θα μπορούσε να επεκταθεί και σε διαφορετικά σενάρια χρήσης. Ένα παράδειγμα, θα μπορούσε να είναι η πρόβλεψη του συναισθήματος που πρόκειται να έχουν οι άνθρωποι, αφότου διαβάσουν ένα κείμενο, μια ιστοσελίδα, ένα βιβλίο. Η τεχνητή νοημοσύνη εξελίσσεται συνεχώς και με ταχύτατους ρυθμούς. Οι άνθρωποι θα μπορούν πλέον να εξάγουν ακόμα περισσότερα συμπεράσματα από τα δεδομένα που έχουν στη διάθεσή τους.

Το μέλλον φαίνεται πολλά υποσχόμενο.

Αναφορές

Γεωργία Κολωνιάρη, Γεώργιος Ευαγγελίδης. Ανάκτηση Πληροφορίας και Μηχανές Αναζήτησης (Information Retrieval and Search Engines), https://openeclass.uom.gr/courses/DAI148/.

Google. “Distributed training with TensorFlow.” *TensorFlow*, https://www.tensorflow.org/guide/distributed_training.

Google. “A friendly introduction to distributed training (ML Tech Talks).” *YouTube*, 30 December 2021, https://www.youtube.com/watch?v=S1tN9a4Proc.

Google. “An in-depth look at Google's first Tensor Processing Unit (TPU).” *Google Cloud*, 12 May 2017, https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu.

Google. “MirroredStrategy demo for distributed training.” *YouTube*, 30 December 2021, https://www.youtube.com/watch?v=xzSCvXDcX68.

Google. “Train and run machine learning models faster | Cloud TPU.” *Google Cloud*, https://cloud.google.com/tpu.

Google. “Training and Test Sets: Splitting Data | Machine Learning.” *Google for Developers*, 18 July 2022, https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

Grubnyak, Alina. “An easy tutorial about Sentiment Analysis with Deep Learning and Keras.” *Towards Data Science*, 8 October 2020, https://towardsdatascience.com/an-easy-tutorial-about-sentiment-analysis-with-deep-learning-and-keras-2bf52b9cba91.

Hagoort, Niels. “Exploring the GPU Architecture | VMware.” *VMware | The Cloud Platform Tech Zone*, 29 October 2020, https://core.vmware.com/resource/exploring-gpu-architecture#.

Heo, Minsuk. “[TensorFlow 2 Deep Learning] Dense Layer.” *YouTube*, 10 February 2020, https://www.youtube.com/watch?v=lor2LnEVn8M.

“How Do You Train Artificial Intelligence (AI)?” *TELUS International*, 19 May 2021, https://www.telusinternational.com/insights/ai-data/article/how-to-train-ai.

Huang, Zhibin, et al. “GPU computing performance analysis on matrix multiplication.” *Ietresearch*, 16 December 2019, https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/joe.2018.9178.

IBM. “What is Machine Learning?” *IBM*, https://www.ibm.com/topics/machine-learning.

IBM. “What is Supervised Learning?” *IBM*, https://www.ibm.com/topics/supervised-learning.

Ibrahim, Mostafa. “What is a Tensor Processing Unit (TPU)? And how does it work?” *Towards Data Science*, 1 March 2021, https://towardsdatascience.com/what-is-a-tensor-processing-unit-tpu-and-how-does-it-work-dbbe6ecbd8ad.

IDRIS. “TensorFlow: Multi-GPU and multi-node data parallelism.” *IDRIS*, 17 April 2023, http://www.idris.fr/eng/jean-zay/gpu/jean-zay-gpu-tf-multi-eng.html.

Karagiannakos, Sergios. “Distributed Deep Learning training: Model and Data Parallelism in Tensorflow.” *AI Summer*, 22 October 2020, https://theaisummer.com/distributed-training/.

Levinas, Mantas. “Everything You Need to Know About GPU Architecture and How It Has Evolved.” *Cherry Servers*, 23 March 2021, https://www.cherryservers.com/blog/everything-you-need-to-know-about-gpu-architecture.

N, Lakshmipathi. “IMDB Dataset of 50K Movie Reviews.” *Kaggle*, 2019, https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews.

Naderan, Mahmood. “Systolic array, how matrices are multiplied?” *YouTube*, 9 March 2019, https://www.youtube.com/watch?v=cmy7LBaWuZ8.

NVIDIA. “Matrix Multiplication Background User's Guide.” *NVIDIA Docs Hub*, 1 February 2023, https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html.

NVIDIA. “Matrix Multiplication Background User's Guide.” *NVIDIA Docs Hub*, 1 February 2023, https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html.

NVIDIA. “NVIDIA T4 Tensor Core GPU for AI Inference.” *NVIDIA*, https://www.nvidia.com/en-us/data-center/tesla-t4/.

Rosenblatt, Frank. “An Introduction To Mathematics Behind Neural Networks.” *Towards Data Science*, 7 October 2020, https://towardsdatascience.com/introduction-to-math-behind-neural-networks-e8b60dbbdeba.

Run:ai. “TensorFlow Multiple GPU: 5 Strategies and 2 Quick Tutorials.” *Run:ai*, https://www.run.ai/guides/multi-gpu/tensorflow-multi-gpu-strategies-and-tutorials.

SAP. “Τι είναι η μηχανική μάθηση;.” *SAP*, 5 January 2023, https://www.sap.com/greece/products/artificial-intelligence/what-is-machine-learning.html.

Smola, Alex. “4.3 Deep and Bidirectional LSTMs.” *YouTube*, 18 January 2021, https://www.youtube.com/watch?v=XfWBk3KjUyM.

“What is LSTM - Introduction to Long Short Term Memory.” *Intellipaat*, 4 March 2023, https://intellipaat.com/blog/what-is-lstm/?US.

Zvornicanin, Enes. “What Are Embedding Layers in Neural Networks?” *Baeldung*, 24 May 2023, https://www.baeldung.com/cs/neural-nets-embedding-layers.