ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΓΙΑ ΠΡΟΒΛΕΨΗ ΕΝΤΟΛΩΝ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Διπλωματική Εργασία

του

Γερογιάννη Μιχαήλ

Θεσσαλονίκη,Φεβρουάριος 2020

ΤΕΧΝΙΚΕΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΓΙΑ ΠΡΟΒΛΕΨΗ ΕΝΤΟΛΩΝ ΣΕ ΠΕΡΙΒΑΛΛΟΝΤΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Γερογιάννης Μιχαήλ

Πτυχίο Επιχειρηματικού Σχεδιασμού & Πληροφοριακών Συστημάτων, Πάτρα, 2014

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Χατζηγεωργίου Αλέξανδρος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28/02/2020

Χατζηγεωργίου Αλέξανδρος	Σατρατζέμη Μαρία	Σακελλαρίου Ηλίας
.....	.....	.....

Γερογιάννης Μιχαήλ

.....

Περίληψη

Τα εργαλεία αυτόματης συμπλήρωσης κώδικα, που παρέχουν τα περιβάλλοντα προγραμματισμού, είναι πλέον αρκετά χρήσιμα στη σύγχρονη ανάπτυξη λογισμικού. Στην εργασία αυτή γίνεται χρήση νευρωνικών δικτύων, τα οποία αποτελούν μια υποκατηγορία της μηχανικής μάθησης. Συγκεκριμένα, με ένα αναδρομικό νευρωνικό δίκτυο-ανατροφοδοτούμενης μονάδας με πύλη (Recurrent Neural Network-Gated Recurrent Unit) γίνεται μια προσπάθεια πρόβλεψης των επόμενων tokens (λέξεις ή σύμβολα) που ακολουθούν το token που εισάγει αρχικά ένας προγραμματιστής κατα τη συγγραφή κώδικα. Για να γίνει η πρόβλεψη φιλικότερη προς το χρήστη, δημιουργήθηκε ένα εργαλείο που έχει τη μορφή ενός αναδυόμενου παραθύρου. Στο παράθυρο αυτό, ο χρήστης θα είναι σε θέση να εισάγει ένα token και το εργαλείο αυτόματα θα του παρέχει τις προβλέψεις του σχετικά με τα tokens που ακολουθούν το αρχικό που εισήγαγε ο χρήστης. Για να μάθει το νευρωνικό δίκτυο να παράγει ακριβείς προβλέψεις χρησιμοποιήθηκε ως είσοδος ένα σύνολο δεδομένων. Το σύνολο αυτό αποτελείται από ένα αρχείο κειμένου το οποίο συγκεντρώνει projects επιλεγμένα από τα αποθετήρια Github και Kaggle τα οποία περιέχουν κώδικα σε Java προσαρμοσμένο κατάλληλα ώστε να αφαιρεθεί ο στατιστικός θόρυβος που εμπεριείχε. Το δίκτυο που παρουσιάζεται παρακάτω δέχεται tokens και παρέχει προβλέψεις με βάση το συντακτικό της γλώσσας προγραμματισμού Java. Τέλος γίνεται μια σύγκριση των αποτελεσμάτων του εργαλείου αυτού με τα αποτελέσματα που παρέχουν άλλες τεχνικές νευρωνικών δικτύων πέραν των δικτύων ανατροφοδοτούμενης μονάδας με πύλη.

Λέξεις Κλειδιά:

Μηχανική μάθηση, νευρωνικά δίκτυα, naturalness, εργαλεία λογισμικού, συμπλήρωση κώδικα.

Abstract

The code autocomplete tools provided by the integrated development environments are now quite useful in modern software development. This research uses neural networks, which are a subcategory of machine learning. To be more specific, a Recurrent Neural Network-Gated Recurrent Unit attempts to predict the next tokens (words or symbols) that follow the initial token entered by a programmer when writing code. To make the prediction more user-friendly, a tool was created in the form of a popup window. In this window, the user will be able to enter a token and the tool will automatically provide him with predictions about the tokens following the first entered by the user. In order the neural network to be trained and produce accurate predictions, it uses a dataset as input. This dataset consists of a text file that gathers projects selected from the Github and Kaggle repositories that contain Java code adapted in such a way, so the statistical noise be removed from it. The network shown below accepts tokens and provides predictions based on the syntax of the Java programming language. Finally, a comparison of the results of this tool with the results provided by neural network techniques other than gated recurrent unit networks is made.

Keywords:

Machine learning, neural networks, naturalness, software tools, code completion.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1. Εισαγωγή

1.1 Σκοπός – Στόχοι

Σκοπός της έρευνας είναι να αναδείξει τα πλεονεκτήματα που μπορεί να παρέχει η μηχανική μάθηση στα πλαίσια του κλάδου της επεξεργασίας φυσικής γλώσσας. Καθώς ο κώδικας αποτελεί μια γραπτή μορφή επικοινωνίας ανάμεσα στον άνθρωπο και τον υπολογιστή, είναι εύλογο να τον θεωρήσουμε ως προς τη μορφή του, παρόμοιο με τη γραπτή φυσική γλώσσα που χρησιμοποιείται ανάμεσα σε ανθρώπους.

Με βάση τη θεωρία αυτή, γίνεται μια σύνδεση των δυνατοτήτων της μηχανικής μάθησης και της επεξεργασίας φυσικής γλώσσας η οποία εμφανίζεται αρκετά συχνά στις εφαρμογές που χρησιμοποιούμε καθημερινά. Εφαρμογές μετάφρασης όπως η Google Translate, αναγνώρισης φωνής όπως η Google Assistant και η Siri, chatbots κ.α. είναι μερικά παραδείγματα των δυνατοτήτων της μηχανικής μάθησης σε συνδυασμό με την επεξεργασία φυσικής γλώσσας. Τα τελευταία χρόνια έχει γίνει σημαντική πρόοδος όσον αφορά τον τρόπο με τον οποίο «σκέφτονται» οι μηχανές και πλέον σε αρκετές περιπτώσεις αγγίζουν τον τρόπο με τον οποίο σκέφτεται ο ανθρώπινος νους.

Στα πλαίσια λοιπόν αυτών των δυνατοτήτων, γίνεται μια προσπάθεια δημιουργίας ενός εργαλείου, το οποίο θα βοηθάει τους προγραμματιστές να συντάξουν γρηγορότερα και αποτελεσματικότερα τον κώδικά τους σε ένα περιβάλλον προγραμματισμού.

1.2 Συνεισφορά

Μελετώντας διάφορες έρευνες σχετικές με το πεδίο της μηχανικής μάθησης και των νευρωνικών δικτύων, παρατίθεται μια νέα προσέγγιση η οποία αξιοποιεί τεχνικές που βασίζονται στα παραπάνω πεδία. Τα νευρωνικά δίκτυα παρέχουν μια ευρεία γκάμα τεχνικών, οι οποίες ποικίλουν ανάλογα με τη φύση του προβλήματος, όπου κανείς μπορεί να εφαρμόσει για να πάρει ως αποτέλεσμα αξιόπιστες και χρήσιμες προβλέψεις. Όσον αφορά τον κλάδο της επεξεργασίας φυσικής γλώσσας, είναι συνηθέστερη και αποτελεσματικότερη η εφαρμογή μιας υποκατηγορίας νευρωνικών δικτύων, τα Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Netwroks). Τα Αναδρομικά Νευρωνικά Δίκτυα χωρίζονται κι αυτά με τη σειρά τους σε υποκατηγορίες, μερικές εκ των οποίων χρησιμοποιήθηκαν από ερευνητές στη προσπάθειά τους να παρέχουν αποτελεσματικότερες προβλέψεις tokens κατά τη συγγραφή κώδικα. Πιο συγκεκριμένα οι ερευνητές, όπως θα φανεί και στη βιβλιογραφική επισκόπηση παρακάτω, χρησιμοποιούν τεχνικές όπως τα δίκτυα Long - Short Term Memory (LSTM), τα δίκτυα pointer, τα convolutional neural networks κ.α.

Στην εργασία αυτή παρουσιάζεται μια νέα προσέγγιση για την πρόβλεψη του επόμενου token κατά τη συγγραφή κώδικα, η οποία βασίζεται στην τεχνική Αναδρομικού Νευρωνικού Δικτύου Ανατροφοδοτούμενης Μονάδας με Πύλη (Recurrent Neural Network - Gated Recurrent Unit). Το δίκτυο αυτό δέχεται ως είσοδο ένα σύνολο δεδομένων σε μορφή κειμένου το οποίο αποτελείται από συνδυασμό διαφόρων projects γραμμένα στη γλώσσα προγραμματισμού Java. Ύστερα από την επεξεργασία του συνόλου δεδομένων με τη χρήση της τεχνικής νευρωνικού δικτύου Gated Recurrent Unit (GRU) παρέχεται μια αρκετά ακριβής πρόβλεψη των επόμενων tokens που ακολουθούν το αρχικό που εισήγαγε ο χρήστης. Για να φανεί η λειτουργία του δικτύου στην πράξη κι όχι ως ποσοστό ακρίβειας και μόνο, δημιουργήθηκε ένα εργαλείο που έχει τη μορφή αναδυόμενου παραθύρου ώστε ο χρήστης να μπορεί να βλέπει τα tokens που παρήχθησαν από την πρόβλεψη και ενδεχομένως να τα χρησιμοποιήσει στη συνέχεια της συγγραφής του κώδικά του.

1.3 Διάρθρωση της μελέτης

- Στο κεφάλαιο 2 παρατίθεται η βιβλιογραφική επισκόπηση άρθρων που διαπραγματεύονται αντίστοιχο πρόβλημα, καθώς και οι ορισμοί εννοιών που χρησιμοποιούνται.

- Στο κεφάλαιο 3 παρουσιάζονται οι απαιτήσεις της εφαρμογής που αναπτύχθηκε. Παρουσιάζονται τα εργαλεία, οι βιβλιοθήκες και η γλώσσα προγραμματισμού που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

- Το κεφάλαιο 4 αναφέρεται στη σχεδίαση και υλοποίηση της εφαρμογής. Αρχικά γίνεται μια γνωριμία με την εφαρμογή ώστε να είναι σε θέση ο αναγνώστης να κατανοήσει τη λειτουργία της εφαρμογής. Έπειτα παρουσιάζονται οι μέθοδοι που χρησιμοποιήθηκαν για την ανάπτυξη του μοντέλου νευρωνικού δικτύου.

- Στο κεφάλαιο 5 γίνεται παρουσίαση της εφαρμογής. Παρουσιάζονται μερικοί πίνακες οι οποίοι προκύπτουν από την εφαρμογή καθώς και μερικά screenshots με παραδείγματα της εφαρμογής στη πράξη.

- Το κεφάλαιο 6 αφορά την αξιολόγηση της εφαρμογής, την επίδειξη αποτελεσμάτων, καθώς και τη σύγκριση της με άλλες παρόμοιες εφαρμογές με τη μορφή πινάκων.

- Τέλος στο κεφάλαιο 7 παρατίθενται τα συμπεράσματα, οι περιορισμοί της έρευνας, μερικές προτάσεις για μελλοντική επέκταση της εφαρμογής, η βιβλιογραφία καθώς και οι διαδικτυακές πηγές.

2. Βιβλιογραφική Επισκόπηση – Θεωρητικό Υπόβαθρο

2.1 Βιβλιογραφική επισκόπηση

Οι Li et al. [2017] στη προσπάθειά τους να προβλέψουν το επόμενο token που ακολουθεί κατά τη συγγραφή κώδικα χρησιμοποιούν ως είσοδο, ώστε να εκπαιδευτεί το νευρωνικό δίκτυο, δύο σύνολα δεδομένων. Το ένα σύνολο αφορά ένα μεγάλο αρχείο με κώδικα διαφόρων project γραμμένα σε Python και JavaScript και το δεύτερο αφορά ένα αρχείο κειμένου το οποίο δημιούργησαν οι ίδιοι που περιέχει διάφορα tokens βασισμένα στο συντακτικό των Python και JS.

Το μοντέλο Αναδρομικού Νευρωνικού Δικτύου που χρησιμοποιούν, βασίζεται στη μέθοδο Long Short Term Memory (LSTM). Όλα τα Αναδρομικά Νευρωνικά Δίκτυα (RNN) έχουν τη μορφή μίας αλυσίδας επαναλαμβανόμενων ενοτήτων του ίδιου του νευρωνικού δικτύου. Επομένως, το νευρωνικό δίκτυο, πέρα από το σύνολο δεδομένων που έχει ως είσοδο για να εκπαιδευτεί, χρησιμοποιεί με αναδρομικό τρόπο στοιχεία από την εκμάθησή του ώστε να βοηθηθεί στην πορεία της εκπαίδευσής του. Τα δίκτυα μακράς βραχείας - διάρκειας μνήμης ή αλλιώς Long Short Term Memory (LSTM), είναι ένα ειδικό είδος RNN ικανό να μαθαίνει από μεγάλες ακολουθίες δεδομένων.

Η επιλογή που πραγματοποιεί το νευρωνικό δίκτυο, ανάμεσα στα δύο αυτά σύνολα ώστε να κάνει τη πρόβλεψή του, βασίζεται στο ποιο από τα δύο σύνολα δεδομένων θα του παρέχει κάθε φορά τη μεγαλύτερη ακρίβεια για την πρόβλεψη.

Ουσιαστικά, το μοντέλο εκπαιδεύεται και με τα δύο σύνολα δεδομένων και παράγει προβλέψεις με διαφορετική ακρίβεια για κάθε σύνολο ξεχωριστά. Επομένως, το token το οποίο θα προταθεί από το νευρωνικό δίκτυο στον χρήστη, θα προέρχεται από το σύνολο δεδομένων που παρείχε μεγαλύτερη ακρίβεια στο δίκτυο κατά την εκπαίδευσή του. Η εισαγωγή δύο συνόλων δεδομένων προς εκπαίδευση στο νευρωνικό δίκτυο, το βοηθά να είναι πιο ακριβές στη πρόβλεψή του καθώς έχει μεγαλύτερο εύρος επιλογών. Για παράδειγμα αν κάποια tokens λείπουν ή δεν εμφανίζονται αρκετές φορές σε ένα σύνολο δεδομένων, το νευρωνικό δίκτυο θα παράγει ένα αναξιόπιστο αποτέλεσμα. Με την εισαγωγή ενός δεύτερου συνόλου δεδομένων, οι ερευνητές προσπαθούν να εξαλείψουν το ενδεχόμενο που αναφέρεται στο παράδειγμα.

Οι Bhoopchand et al. [2016] αναπτύσσουν ένα πρόγραμμα το οποίο παρέχει προτάσεις για τα επόμενα tokens βασισμένες σε ένα αρκετά μεγάλο σύνολο δεδομένων που περιέχει κώδικα σε Python. Με την τεχνική των sparse pointer networks την οποία χρησιμοποιούν, ο αλγόριθμος είναι σε θέση να «στοχεύσει» και να προτείνει το token με τη μεγαλύτερη πιθανότητα να είναι το επόμενο σε σειρά.

Τα δίκτυα pointer networks είναι μια παραλλαγή του μοντέλου sequence to sequence with attention. Κάθε στρώμα νευρώνων, αντί να μεταφράζει ολόκληρες συστοιχίες δεδομένων, τους αποδίδει μια σειρά από δείκτες με ξεχωριστή βαρύτητα στον καθένα, ώστε να είναι ευκολότερο να «διαβαστούν» από το αμέσως επόμενο στρώμα κ.ο.κ.

Για παράδειγμα, αν θέλουμε να μεταφράσουμε τη φράση "Τι κάνεις σήμερα;" από τα ελληνικά στα κινέζικα θα έχουμε μια είσοδο 3 λέξεων και μια έξοδο 7 συμβόλων (今天你在做什麼?). Είναι σαφές ότι δεν μπορούμε να χρησιμοποιήσουμε ένα απλό νευρωνικό δίκτυο για να αποκωδικοποιήσουμε κάθε λέξη από την ελληνική πρόταση στην κινεζική πρόταση. Στο ίδιο παράδειγμα το μοντέλο pointer network θα προσέθετε ένα δείκτη βαρύτητας σε κάθε σύμβολο ώστε να γίνει πιο εύκολη η αποκωδικοποίησή του. Για την αντιμετώπιση προβλημάτων όπως αυτό χρησιμοποιούνται τα μοντέλα pointer networks και sequence to sequence networks.

Στην έρευνα των Cho et al. [2014] παρατηρούμε την τεχνική encode - decode όπου αρχικά κωδικοποιείται το σύνολο δεδομένων ώστε να γίνει πιο «εύπεπτο» για το μοντέλο που χρησιμοποιούν και αποκωδικοποιείται όταν παραχθεί η πρόβλεψη.

Η κωδικοποίηση ενός μηνύματος αφορά ένα σύστημα κωδικοποιημένων εννοιών και για να το δημιουργήσει ο αποστολέας πρέπει να καταλάβει πώς αντιλαμβάνεται ο δέκτης τον κόσμο γύρω του.

Η αποκωδικοποίηση ενός μηνύματος είναι το πώς ο δέκτης μπορεί να καταλάβει και να ερμηνεύσει το μήνυμα. Πρόκειται για μια διαδικασία ερμηνείας και μετάφρασης των κωδικοποιημένων πληροφοριών σε κατανοητή μορφή.

Ένα παράδειγμα κωδικοποίησης και αποκωδικοποίησης μηνύματος είναι η χρήση νοηματικής γλώσσας η οποία διαθέτει λεκτικές και συντακτικές δομές για να εκφράσει οποιαδήποτε αφηρημένη έννοια.

Οι Hinton et al. [2012] κάνουν λόγο για feed - forward convolutional neural networks (CNNs). Αυτού του είδους τα νευρωνικά δίκτυα είναι δίκτυα που αποτελούνται από πολλά στρώματα νευρώνων, όπου κάθε νευρώνας σε ένα συγκεκριμένο στρώμα φιλτράρει το αποτέλεσμα που δίνουν οι νευρώνες του προηγούμενου στρώματος. Ουσιαστικά αφαιρείται ο δείκτης bias και εφαρμόζεται μια μη γραμμική ενεργοποίηση λειτουργίας (activation function) στο αποτέλεσμα των νευρώνων του προηγούμενου στρώματος πριν αυτό προχωρήσει στο επόμενο στρώμα. Ο δείκτης bias αναφέρεται ουσιαστικά στον στατιστικό θόρυβο που ενδεχομένως περιέχει το αρχικό σύνολο δεδομένων του δικτύου και η ενεργοποίηση λειτουργίας καθορίζει την έξοδο του προηγούμενου στρώματος.

Οι Liu et al. [2016] εξηγούν την έννοια των αφηρημένων δέντρων σύνταξης ή αλλιώς Abstract Syntax Tree (AST). Ένα AST είναι ένας τρόπος παρουσίασης του συντακτικού μιας γλώσσας προγραμματισμού ως μια ιεραρχικού τύπου δομή. Στην εργασία τους, θεωρούν τα δεδομένα εισαγωγής τους, ως μια μορφή AST και έτσι το πρόβλημα αυτόματης πρόβλεψης/συμπλήρωσης κώδικα μετατρέπεται σε πρόβλημα πρόβλεψης του επόμενου κόμβου του AST.

Οι Allamanis et al. [2018] μιλούν για την υπόθεση της φυσικότητας. Εξηγούν ότι ο κώδικας θα μπορούσε να θεωρηθεί ως μια μορφή ανθρώπινης επικοινωνίας καθώς οι λέξεις και τα σύμβολα που χρησιμοποιούνται όταν κανείς γράφει κώδικα έχουν παρόμοιες ιδιότητες με αυτά της φυσικής γλώσσας. Οι ιδιότητες αυτές μπορούν να αξιοποιηθούν για την κατασκευή καλύτερων εργαλείων συγγραφής κώδικα.

Η εκμετάλλευση των παρόμοιων ιδιοτήτων μεταξύ κώδικα και φυσικής γλώσσας βρίσκει πολλές εφαρμογές. Μελέτες πάνω σε μεγάλα κείμενα φυσικής γλώσσας χρησιμοποιήθηκαν με μεγάλη επιτυχία σε εφαρμογές όπως η αναγνώριση ομιλίας, η μετάφραση, η διόρθωση σφαλμάτων κ.λπ.

Η υπόθεση της φυσικότητας θεωρεί τη συγγραφή κώδικα ως μια μορφή επικοινωνίας, όπου συναντά κανείς πλούσια μοτίβα παρόμοια με τη φυσική γλώσσα, επιτρέποντας έτσι τη δημιουργία εργαλείων μηχανικής μάθησης. Ιδανικά μέσω στατιστικών μεθόδων που επιτρέπουν σε ένα σύστημα να κάνει υποθέσεις, είναι δυνατό να προβλεφθεί με σχετική ακρίβεια η επόμενη ενέργεια που θέλει να κάνει ο προγραμματιστής κατά τη συγγραφή κώδικα.

Η φυσικότητα του κώδικα φαίνεται να έχει ισχυρή βάση. Οι προγραμματιστές προτιμούν να γράφουν και να διαβάζουν κώδικα που είναι συμβατικός και οικείος, καθώς αυτό βοηθά στην κατανόηση και τη διατήρηση των συστημάτων λογισμικού. Ο κώδικας που παίρνει γνωστές μορφές είναι πιο διαφανής αφού το νόημά του είναι πιο εύκολα κατανοητό σε έναν έμπειρο αναγνώστη. Έτσι, η υπόθεση της φυσικότητας οδηγεί απρόσκοπτα σε μια έννοια "πρόβλεψης κώδικα", δεδομένου ότι ο κώδικας ακολουθεί συγκεκριμένα μοτίβα συμβολοσειρών και δημιουργεί προβλέψιμα πρότυπα τα οποία μπορούν να αξιοποιηθούν.

Οι γλώσσες προγραμματισμού γεφυρώνουν το χάσμα ανάμεσα στους υπολογιστές και τον ανθρώπινο νου. Ο πηγαίος κώδικας επικοινωνεί κατά μήκος δύο καναλιών, ένα που αφορά τους ανθρώπους και ένα που αφορά τους υπολογιστές. Οι άνθρωποι πρέπει να καταλάβουν τον κώδικα για να τον διαβάσουν και να τον γράψουν, και οι υπολογιστές πρέπει να είναι σε θέση να τον εκτελέσουν.

Η διόγκωση του κώδικα οδηγεί στις ομοιότητες και στις διαφορές μεταξύ του κώδικα και της φυσικής γλώσσας. Παρόλο που ο κώδικας και το κείμενο είναι παρόμοιες έννοιες, ο κώδικας είναι σχετικά ένα νέο πεδίο προβληματισμού για τις υπάρχουσες τεχνικές μηχανικής μάθησης και επεξεργασίας φυσικής γλώσσας. Στη συνέχεια της εργασίας τους στρέφουν τη προσοχή τους στα πιθανοτικά μοντέλα μηχανικής μάθησης.

Το πιθανοτικό μοντέλο του πηγαίου κώδικα είναι μια κατανομή πιθανότητας πάνω από τα τεχνητά στοιχεία κώδικα. Όπως κάνουν όλα τα μοντέλα μηχανικής μάθησης τα πιθανοτικά μοντέλα κάνουν κι αυτά με τη σειρά τους απλουστευμένες υποθέσεις σχετικά με τα υπάρχοντα δεδομένα. Αυτές οι υποθέσεις καθιστούν τα μοντέλα εύκολα στη κατανόηση και στη χρήση, όμως είναι πάντα πιθανή η περίπτωση σφάλματος. Στη προσπάθειά τους να παρουσιάσουν τα υπέρ και τα κατά κάθε μοντέλου μηχανικής μάθησης, οι ερευνητές ομαδοποιούν τα μοντέλα σε τρεις κατηγορίες:

- **Code - generating models:** Τα μοντέλα αυτά ορίζουν μια κατανομή πιθανότητας πάνω από τον κώδικα και τον διασπούν σε απλούστερα τμήματα κώδικα με τη βοήθεια τεχνικών όπως η abstract syntax tree.
- **Representational models:** Τα μοντέλα αυτά λαμβάνουν μια αφηρημένη αναπαράσταση του κώδικα ως είσοδο. Το αποτέλεσμα του μοντέλου αποδίδει μια κατανεμημένη πιθανότητα πάνω από τα στοιχεία του κώδικα. Με αυτόν τον τρόπο γίνεται απλούστερη η πρόβλεψή τους.

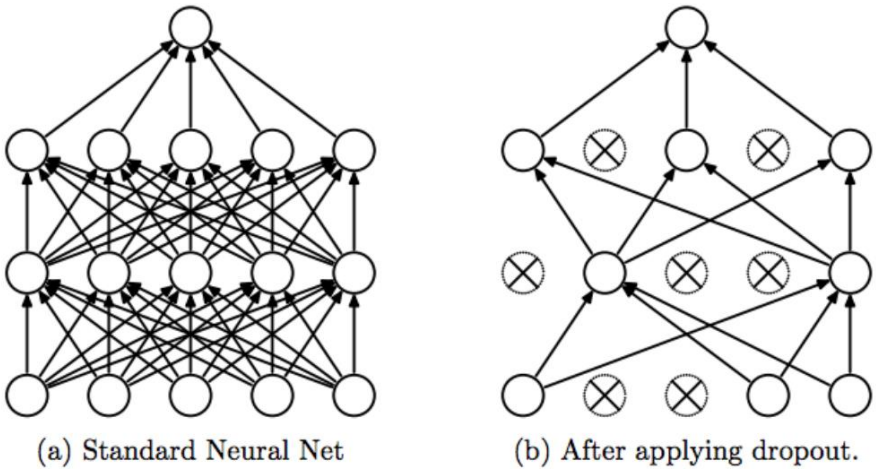
- **Pattern mining models:** Τα μοντέλα αυτά προσπαθούν να βρουν μοτίβα εντός του κώδικα ώστε να γίνει πιο εύκολη η πρόβλεψή τους.

Οι Zaremba et al. [2014] χρησιμοποιούν την τεχνική dropout για να βελτιστοποιήσουν το μοντέλο τους. Είναι αρκετά πιθανό να πέσει κανείς στην παγίδα της «υπερφόρτωσης» δεδομένων στο μοντέλο του.

Τα νευρωνικά δίκτυα που εκπαιδεύονται σε πολύ μεγάλα σύνολα δεδομένων μπορεί να υπερκεράσουν τα δεδομένα εκπαίδευσης. Αυτό εμπεριέχει τον κίνδυνο στατιστικού θορύβου στα δεδομένα εισόδου, πράγμα που έχει ως αποτέλεσμα την κακή απόδοση του μοντέλου. Επιπρόσθετα, αυξάνεται το σφάλμα της γενίκευσης εξαιτίας της υπερφόρτωσης δεδομένων εισόδου.

Στο σημείο αυτό λοιπόν έρχεται να βοηθήσει η τεχνική dropout. Ουσιαστικά είναι μια μέθοδος τακτοποίησης που προσεγγίζει την κατάρτιση ενός μεγάλου αριθμού νευρωνικών δικτύων με διαφορετικές αρχιτεκτονικές.

Κατά τη διάρκεια της εκπαίδευσης, κάποιες τυχαίες έξοδοι στρωμάτων του νευρωνικού δικτύου αγνοούνται ή "αποχωρούν". Αυτό έχει ως αποτέλεσμα να κάνει το στρώμα πιο εύκολο στη κατανόηση από το επόμενο στρώμα. Με τη τεχνική dropout το μοντέλο μαθαίνει να χειρίζεται τη διαδικασία τροφοδότησής του ώστε να γίνει πιο ακριβές στις προβλέψεις του. Στην **Εικόνα 1** φαίνονται οι συνδέσεις μεταξύ των νευρώνων πριν και μετά την εφαρμογή της τεχνικής dropout.



Εικόνα 1: Εφαρμογή τεχνικής dropout

https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

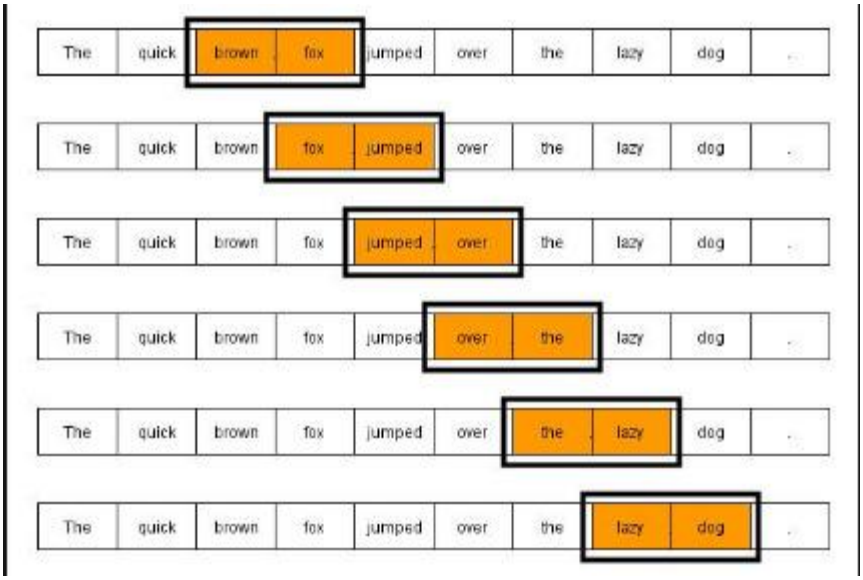
Οι Hellendoorn & Devanbu [2017] στην έρευνά τους χρησιμοποιούν τον όρο n-gram model. Ένα n-gram μοντέλο είναι μια ακολουθία λέξεων N. Για παράδειγμα στην πρόταση "Αυτό είναι ένα ωραίο τραγούδι" ένα 2-gram μοντέλο θα διαμορφώσει τη πρόταση ως εξής:

«Αυτό είναι», «είναι ένα», «ένα ωραίο», «ωραίο τραγούδι»

Στην περίπτωση των 3-gram μοντέλων η ίδια πρόταση θα γίνει:

«Αυτό είναι ένα», «ένα ωραίο τραγούδι» κ.ο.κ.

Χρησιμοποιώντας τη τεχνική των n-gram οι ερευνητές προσπαθούν να προβλέψουν την τελευταία λέξη του n-gram μοντέλου με δεδομένες τις προηγούμενες λέξεις του μοντέλου. Στην **Εικόνα 2** βλέπουμε την εφαρμογή ενός 2-gram μοντέλου στην πρόταση «The quick brown fox jumped over the lazy dog.»



Εικόνα 2: Εφαρμογή τεχνικής n-gram

https://www.depends-on-the-definition.com/introduction-n-gram-language-models/

2.2 Θεωρητικό υπόβαθρο

2.2.1 Μηχανική μάθηση

Σύμφωνα με τη βικιπαίδεια μηχανική μάθηση είναι υποπεδίο της επιστήμης των υπολογιστών που αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Το 1959, ο Άρθουρ Σάμουελ ορίζει τη μηχανική μάθηση ως "Πεδίο μελέτης που δίνει στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν ρητά προγραμματιστεί". Η μηχανική μάθηση διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικές με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις βασιζόμενες στα δεδομένα ή να εξάγουν αποφάσεις που εκφράζονται ως το αποτέλεσμα.

Η μηχανική μάθηση είναι στενά συνδεδεμένη και συχνά συγχέεται με την υπολογιστική στατιστική, έναν κλάδο, που επίσης επικεντρώνεται στην πρόβλεψη μέσω της χρήσης των υπολογιστών. Έχει επίσης ισχυρούς δεσμούς με την μαθηματική βελτιστοποίηση. Η Μηχανική μάθηση εφαρμόζεται σε μια σειρά από υπολογιστικές εργασίες, όπου τόσο ο σχεδιασμός όσο και ο ρητός προγραμματισμός των αλγορίθμων είναι ανέφικτος. Παραδείγματα εφαρμογών αποτελούν τα φίλτρα spam (spam filtering), η οπτική αναγνώριση χαρακτήρων (OCR), οι μηχανές αναζήτησης και η υπολογιστική όραση. Η Μηχανική μάθηση μερικές φορές συγχέεται με την εξόρυξη δεδομένων, όπου η τελευταία επικεντρώνεται περισσότερο στην εξερευνητική ανάλυση των δεδομένων, γνωστή και ως μη επιτηρούμενη μάθηση. Πηγή

Επιτηρούμενη μάθηση

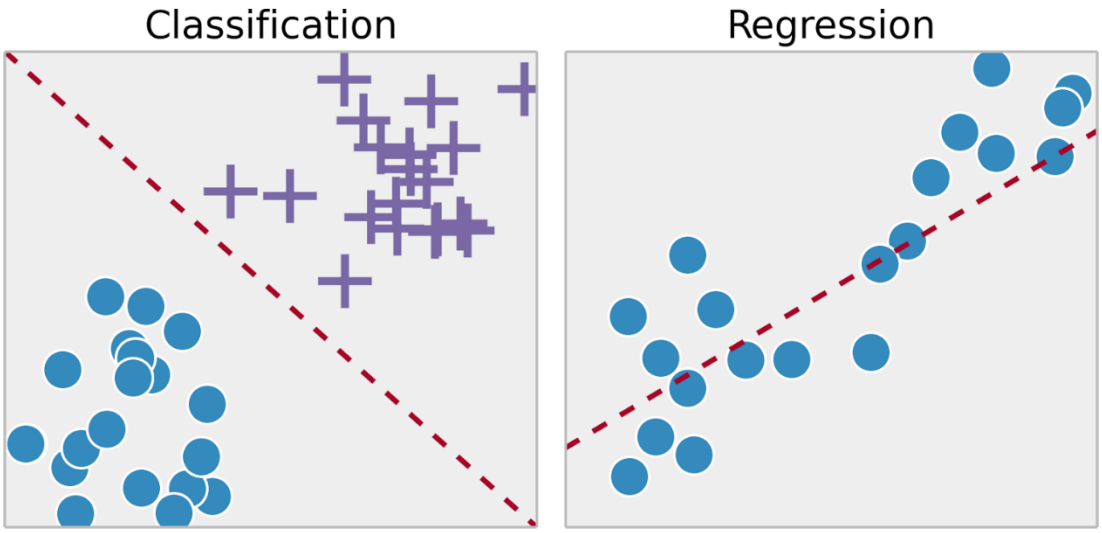
Η επιτηρούμενη μάθηση συνήθως βρίσκει εφαρμογή σε προβλήματα ταξινόμησης. Οι πιο συνηθισμένοι αλγόριθμοι επιτηρούμενης μάθησης περιλαμβάνουν την γραμμική παλινδρόμηση, τη naive bayes, τα τυχαία δάση κ.α. Τόσο στην παλινδρόμηση όσο και στην ταξινόμηση, ο στόχος είναι να βρεθούν συγκεκριμένες σχέσεις ή δομές στα δεδομένα εισόδου ώστε να παραχθούν ακριβή δεδομένα εξόδου. Η ακριβής έξοδος καθορίζεται εξολοκλήρου από τα δεδομένα εισόδου, καθώς ο στατιστικός θόρυβος μειώνει σημαντικά την αποτελεσματικότητα του μοντέλου.

Κατά τη διεξαγωγή επιτηρούμενης μάθησης, οι κύριοι προβληματισμοί είναι η πολυπλοκότητα του μοντέλου και η συρρίκνωση του δείκτη bias.

Το σωστό επίπεδο πολυπλοκότητας του μοντέλου καθορίζεται από τη φύση των δεδομένων εισόδου. Στη περίπτωση που τα δεδομένα εισόδου δεν είναι αρκετά ή δεν είναι ομοιόμορφα κατανεμημένα, θα πρέπει το μοντέλο να είναι χαμηλής πολυπλοκότητας.

Ο δείκτης bias σχετίζεται με τη γενίκευση του μοντέλου. Σε οποιοδήποτε μοντέλο, υπάρχει μια ισορροπία μεταξύ του bias που είναι ο σταθερός όρος σφάλματος, και της διακύμανσης η οποία είναι το διάστημα στο οποίο το σφάλμα μπορεί να διακυμανθεί. Έτσι, με υψηλό τον δείκτη bias και χαμηλό δείκτη διακύμανσης, θα προέκυπτε ένα εντελώς λανθασμένο μοντέλο.

Ο δείκτης bias και η διακύμανση συνήθως κινούνται σε αντίθετες κατευθύνσεις μεταξύ τους. Με άλλα λόγια, η αύξηση του δείκτη bias οδηγεί συνήθως σε χαμηλότερη διακύμανση και αντίστροφα. Επιπλέον, για να θεωρηθούν τα μοντέλα σχετικά ακριβή, η διακύμανση του μοντέλου θα πρέπει να κλιμακώνεται με το μέγεθος και την πολυπλοκότητα των δεδομένων εισόδου. Με άλλα λόγια, τα μικρά, απλά σύνολα δεδομένων πρέπει συνήθως να ακολουθούνται από απλούστερα μοντέλα και τα μεγάλα σύνολα δεδομένων από πολυπλοκότερα μοντέλα. Στην **Εικόνα 3** βλέπουμε μια ταξινόμηση της μορφής classification όπου οι κύκλοι έχουν διαχωριστεί από τους σταυρούς και μια γραμμική παλινδρόμηση όπου οι κύκλοι τείνουν να ακολουθούν μια ευθεία γραμμή.

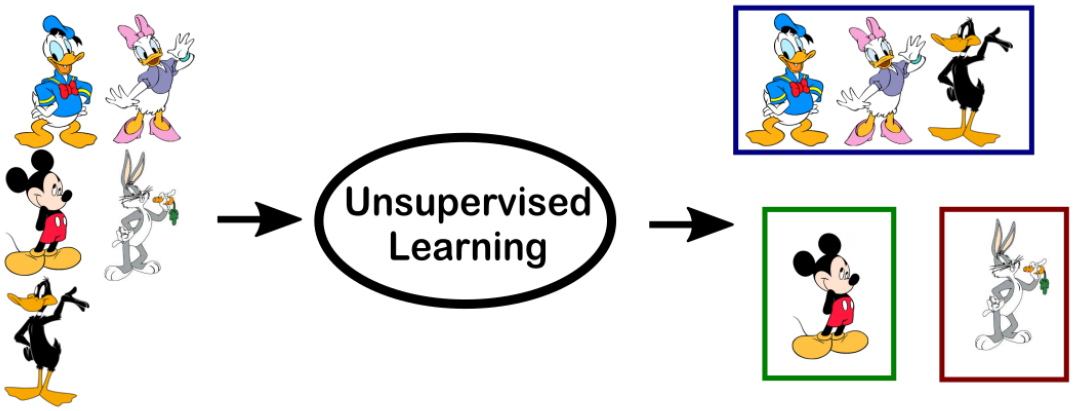


Εικόνα 3: Ταξινόμηση και παλινδρόμηση
https://www.fcodelabs.com/2018/12/13/Machine-Learning-Intro/

Μη επιτηρούμενη μάθηση

Οι συνηθέστερες εφαρμογές της μη επιτηρούμενης μάθησης είναι η συσπείρωση (clustering), η representation learning και η density estimation. Σε όλες αυτές τις περιπτώσεις, στόχος είναι η εκμάθηση της εγγενούς δομής των δεδομένων. Οι συχνότερα εμφανιζόμενοι αλγόριθμοι στη μη επιτηρούμενη μάθηση περιλαμβάνουν τους k-means clustering, principal component analysis και τους auto-encoders.

Η μη επιτηρούμενη μάθηση είναι πολύ χρήσιμη στη διερευνητική ανάλυση επειδή μπορεί να αναγνωρίσει αυτόματα τη δομή των δεδομένων. Για παράδειγμα, εάν ένας αναλυτής προσπαθούσε να κατατάξει τους καταναλωτές με βάση κάποια μη αριθμητικά χαρακτηριστικά, οι μη επιτηρούμενες μέθοδοι ομαδοποίησης θα αποτελούσαν μια εξαιρετική αφετηρία για την ανάλυσή τους. Σε περιπτώσεις όπου είναι αδύνατο ή δεν είναι πρακτικό να παρατηρήσει ένας άνθρωπος τάσεις στα δεδομένα, η μη επιτηρούμενη μάθηση μπορεί να παρέχει αυτές τις τάσεις προς ανάλυση. Στην **Εικόνα 4** φαίνεται ότι μέσω της μη επιτηρούμενης μάθησης, ένας αλγόριθμος μηχανικής μάθησης μπορεί να ξεχωρίσει τους ήρωες της Disney ανάλογα με το είδος τους.



Εικόνα 4: Μη επιτηρούμενη μάθηση
https://www.fcodelabs.com/2018/12/13/Machine-Learning-Intro/

Πλεονεκτήματα μηχανικής μάθησης

i. Εντοπίζει εύκολα τις τάσεις και τα πρότυπα

Η Μηχανική Μάθηση μπορεί να διαχειριστεί μεγάλους όγκους δεδομένων και να ανακαλύψει συγκεκριμένες τάσεις και πρότυπα τα οποία δεν θα ήταν προφανή στον άνθρωπο. Για παράδειγμα, για έναν ιστότοπο ηλεκτρονικού εμπορίου, όπως η Amazon, η μηχανική μάθηση χρησιμεύει ως ένα εργαλείο για την κατανόηση των συμπεριφορών περιήγησης και των ιστορικών αγοράς των χρηστών του, ώστε να τους παρέχει προτάσεις όπως τα προϊόντα, τις προσφορές και τις υπενθυμίσεις που τους αφορούν.

ii. Δεν απαιτείται ανθρώπινη παρέμβαση (αυτοματοποίηση)

Με τη μηχανική μάθηση, δεν χρειάζεται να παρακολουθεί κανείς το έργο του σε κάθε βήμα. Το γεγονός ότι οι μηχανές έχουν τη δυνατότητα να μάθουν, τους επιτρέπει να κάνουν προβλέψεις και να βελτιώσουν τους αλγόριθμους τους μόνες τους. Ένα κοινό παράδειγμα αυτού είναι τα λογισμικά προστασίας από ιούς, καθώς μαθαίνουν να φιλτράρουν και να αναγνωρίζουν νέες απειλές. Ένα άλλο παράδειγμα είναι η αναγνώριση των spam.

iii. Συνεχής Βελτίωση

Καθώς οι αλγόριθμοι μηχανικής μάθησης είναι σε θέση να αποκτούν εμπειρία, μπορούν και γίνονται ολοένα καλύτεροι και πιο αποτελεσματικοί κι αυτό με τη σειρά του οδηγεί στη λήψη καλύτερων αποφάσεων. Για παράδειγμα, σε ένα μοντέλο πρόγνωσης καιρού, καθώς αυξάνεται η ποσότητα των δεδομένων, οι αλγόριθμοι μαθαίνουν να κάνουν τις προβλέψεις τους με μεγαλύτερη ακρίβεια και ταχύτητα.

iv. Χειρισμός πολυδιάστατων και πολυποίκιλων δεδομένων

Οι αλγόριθμοι εκμάθησης μηχανών είναι ικανοί στο χειρισμό δεδομένων που είναι πολυδιάστατα, πολλαπλών ποικιλιών και μπορούν να το κάνουν σε δυναμικά και μη καθορισμένα περιβάλλοντα.

Μειονεκτήματα μηχανικής μάθησης

i. Συλλογή μεγάλου όγκου δεδομένων

Στη μηχανική μάθηση απαιτούνται πολύ μεγάλα σε όγκο σύνολα δεδομένων, ώστε να μπορέσει να εκπαιδευτεί το εκάστοτε μοντέλο μηχανικής μάθησης. Τα σύνολα αυτά, πρέπει να είναι συνεκτικά αμερόληπτα και καλής ποιότητας. Εάν τα σύνολα δεδομένων δεν είναι έτοιμα, τα μοντέλα μηχανικής μάθησης παραμένουν ανενεργά έως ότου να δημιουργηθούν νέα δεδομένα.

ii. Χρόνος και πόροι

Οι αλγόριθμοι μηχανικής μάθησης χρειάζονται αρκετό χρόνο για να μάθουν και να αναπτυχθούν αρκετά ώστε να εκπληρώσουν το σκοπό τους με σημαντική ακρίβεια και αποτελεσματικότητα. Επίσης, χρειάζονται έναν τεράστιο αριθμό πόρων για να λειτουργήσουν, πράγμα το οποίο σημαίνει πρόσθετες απαιτήσεις σε όλα τα επίπεδα.

iii. Ερμηνεία των αποτελεσμάτων

Μια άλλη σημαντική πρόκληση είναι η ικανότητα να μπορέσουν να ερμηνευθούν με ακρίβεια τα αποτελέσματα που παράγονται από τους αλγόριθμους. Επιπλέον, κανείς θα πρέπει να επιλέξει με προσοχή ανάμεσα σε μια πληθώρα αλγορίθμων το ποιος ταιριάζει καλύτερα στις απαιτήσεις του.

iv. Υψηλή ευαισθησία σφάλματος

Η μηχανική μάθηση είναι αυτόνομη αλλά εξαιρετικά ευαίσθητη σε σφάλματα. Για παράδειγμα, εάν εκπαιδεύσει κανείς έναν αλγόριθμο με ένα αρκετά μικρό σύνολο δεδομένων ώστε να μην είναι περιεκτικό, θα καταλήξει σε λανθασμένες προβλέψεις που προέρχονται από ένα ανεπαρκές σύνολο δεδομένων εκπαίδευσης. Τέτοιου είδους σφάλματα μπορούν να ενεργοποιήσουν μια αλυσίδα σφαλμάτων τα οποία θα είναι δύσκολο να εντοπιστούν και να διορθωθούν για μεγάλο χρονικό διάστημα.

Εφαρμογές μηχανικής μάθησης στην επεξεργασία φυσικής γλώσσας

i. Εικονικοί Προσωπικοί Βοηθοί

Siri, Alexa, Google Now είναι μερικά από τα δημοφιλή παραδείγματα εικονικών προσωπικών βοηθών. Όπως υποδηλώνει η ονομασία τους, βοηθούν στην εύρεση πληροφοριών ύστερα από φωνητικές εντολές. Το μόνο που χρειάζεται να κάνει κανείς, είναι να τους ενεργοποιήσει και για παράδειγμα να ρωτήσει "Ποιο είναι το πρόγραμμά μου για σήμερα;", "Ποιες είναι οι πτήσεις από Γερμανία προς Λονδίνο" ή παρόμοιες ερωτήσεις. Για να απαντήσει, ο προσωπικός βοηθός, αναζητά τις πληροφορίες, ανακαλώντας τα σχετικά ερωτήματα ή στέλνοντας εντολή σε άλλους πόρους (όπως εφαρμογές τηλεφώνου) για τη συλλογή των πληροφοριών. Είναι επίσης δυνατή η καθοδήγηση των βοηθών για ορισμένα καθήκοντα, όπως για παράδειγμα ρύθμιση αφύπνισης, υπενθυμίσεις, κ.α.

ii. Μηχανική μετάφραση

Στις μέρες μας είναι χρήσιμη για αρκετούς ανθρώπους η δυνατότητα μετάφρασης πληροφοριών από μια γλώσσα σε μια άλλη. Η ιδέα πίσω από τη μηχανική μετάφραση είναι η ανάπτυξη αλγορίθμων οι οποίοι επιτρέπουν αυτόματη μετάφραση χωρίς να χρειάζεται ανθρώπινη παρέμβαση. Η πιο γνωστή εφαρμογή είναι η μετάφραση της Google.

Η εφαρμογή Google Translate βασίζεται στη στατιστική μηχανική μετάφραση. Η εφαρμογή αυτή δεν επαναπαύεται μόνο στη μετάφραση ενός κειμένου αντικαθιστώντας μία μία τις λέξεις. Η μετάφραση Google, συγκεντρώνει όσα περισσότερα κείμενα μπορεί μεταξύ δύο γλωσσών ώστε να κάνει ακριβείς παραλληλισμούς των εννοιών των δύο γλωσσών. Ο τρόπος με τον οποίο λειτουργεί η εφαρμογή δεν διαφέρει από τον τρόπο που μαθαίνει να μιλάει ένας άνθρωπος, αφού από μικρή ηλικία μαθαίνει λέξεις και σταδιακά αρχίζει να αποδίδει σημασιολογική αξία σε αυτές, ώσπου τελικά να εξάγει σημασιολογικές αξίες με δεδομένους συνδυασμούς λέξεων.

iii. Συναισθηματική ανάλυση

Η ανάλυση συναισθημάτων (γνωστή και ως εξόρυξη γνώμης ή συναισθηματική τεχνητή νοημοσύνη) είναι ένα ενδιαφέρον είδος εξόρυξης δεδομένων που μετρά την κλίση διαφόρων ανθρώπινων απόψεων. Σκοπός αυτής της ανάλυσης είναι να προσδιοριστούν οι υποκειμενικές απόψεις κάποιου σε ένα κείμενο. Για παράδειγμα, τέτοιες απόψεις μπορεί να είναι η κριτική μιας ταινίας ή μια συναισθηματική κατάσταση που αναρτά κανείς στα social media. Η ανάλυση τέτοιων απόψεων βοηθά τις εταιρείες να ελέγξουν εάν οι πελάτες είναι ικανοποιημένοι με τα αγαθά ή τις υπηρεσίες που τους παρέχουν.

iv. Εξαγωγή περίληψης κειμένου

Αυτή είναι η διαδικασία δημιουργίας μιας σύντομης, ακριβούς και ευανάγνωστης περίληψης ενός εγγράφου με μεγάλο κείμενο. Το σημαντικότερο πλεονέκτημα της χρήσης μιας περίληψης είναι ότι μειώνει το χρόνο ανάγνωσης.

v. Chatbots

Ορισμένοι ιστότοποι προσφέρουν σήμερα τη δυνατότητα συνομιλίας με εκπρόσωπο υποστήριξης πελατών καθώς οι χρήστες πλοηγούνται εντός του ιστότοπου. Ωστόσο, δεν έχουν όλοι οι ιστότοποι διαθέσιμο ένα στέλεχος για να απαντήσουν στα ερωτήματα των χρηστών. Γι' αυτές τις περιπτώσεις, χρησιμοποιείται ένα chatbot. Αυτά τα bots τείνουν να αποσπάσουν πληροφορίες από τον ιστότοπο και να τις παρουσιάζουν στους πελάτες. Τα chatbots γίνονται καλύτερα με τη πάροδο του χρόνου και τείνουν να κατανοούν καλύτερα τα ερωτήματα των χρηστών και να δίνουν καλύτερες απαντήσεις, κάτι που είναι δυνατό λόγω των αλγορίθμων μηχανικής μάθησης.

vi. Καλύτερα αποτελέσματα μηχανών αναζήτησης

Η Google και άλλες μηχανές αναζήτησης χρησιμοποιούν μηχανική μάθηση για να βελτιώσουν τα αποτελέσματα αναζήτησης τους. Κάθε φορά που ένας χρήστης εκτελεί μια αναζήτηση, οι αλγόριθμοι στο back-end παρατηρούν την αντίδρασή του στα αποτελέσματα. Αν ο χρήστης ανοίξει μια ιστοσελίδα που εμφανίζεται στα κορυφαία αποτελέσματα και παραμείνει εκεί για μεγάλο χρονικό διάστημα, η μηχανή αναζήτησης υποθέτει ότι τα αποτελέσματα που εμφανίζονται ήταν σύμφωνα με το ερώτημα. Ομοίως, αν ο χρήστης φτάσει στη δεύτερη ή την τρίτη σελίδα των αποτελεσμάτων αναζήτησης αλλά δεν ανοίξει κανένα από τα αποτελέσματα, η μηχανή αναζήτησης εκτιμά ότι τα αποτελέσματα που προβάλλονται δεν αντιστοιχούν στην απαίτηση. Με αυτό τον τρόπο, οι αλγόριθμοι που λειτουργούν στο back-end βελτιώνουν τα αποτελέσματα αναζήτησης.

vii. Συστάσεις προϊόντος

Οι χρήστες ενός ιστότοπου λαμβάνουν μηνύματα ηλεκτρονικού ταχυδρομείου που έχουν ως περιεχόμενο προτάσεις αγορών, για τις επόμενες μέρες μετά την αγορά ενός προϊόντος online. Με βάση τη συμπεριφορά των χρηστών όσον αφορά τον ιστότοπο / εφαρμογή δηλαδή τυχόν προηγούμενες αγορές, αντικείμενα που τους αρέσουν ή προστίθενται στο καλάθι, οι αλγόριθμοι μηχανικής μάθησης κάνουν συστάσεις στους χρήστες για τα προϊόντα αυτά.

viii. Ταξινόμηση κειμένου

Η ταξινόμηση κειμένου μπορεί να χρησιμοποιηθεί για την οργάνωση, τη δομή και την κατηγοριοποίηση σχεδόν οποιασδήποτε μορφής κειμένου. Ας υποθέσουμε ότι διανέμουμε έγγραφα σε ορισμένες κατηγορίες. Για παράδειγμα αν δεν προλαβαίνουμε να ταξινομήσουμε εμείς ένα αρχείο κειμένου μπορούμε να το δώσουμε σε μια εφαρμογή που χρησιμοποιεί μηχανική μάθηση να το κάνει για μας. Με τη χρήση τεχνικών επεξεργασίας φυσικής γλώσσας, οι ταξινομητές κειμένου μπορούν να αναλύσουν αυτόματα το κείμενο και στη συνέχεια να αντιστοιχίσουν ένα σύνολο προκαθορισμένων ετικετών ή κατηγοριών με βάση το περιεχόμενό του.

ix. Αναγνώριση χαρακτήρων

Τα συστήματα αναγνώρισης χαρακτήρων έχουν επίσης πολλές εφαρμογές κυρίως για τον καθορισμό ενός εγγράφου ως νόμιμου η μη.

2.2.2 Δείκτης Bias

Ο όρος Bias πρακτικά σημαίνει παρεμβολή στα αποτελέσματα της έρευνας λόγω προκαθορισμένων ιδεών, προκατάληψης ή επιρροής προς μια συγκεκριμένη κατεύθυνση. Τα δεδομένα, καθώς και τα άτομα που αναλύουν τα δεδομένα μπορεί να είναι προκατειλημμένα. Όταν τα δεδομένα είναι προκατειλημμένα, εννοούμε ότι το δείγμα δεν είναι αντιπροσωπευτικό του συνόλου του πληθυσμού. Για παράδειγμα, εξαγωγή συμπερασμάτων για ολόκληρο τον πληθυσμό της Ελλάδας με βάση έρευνα που έγινε σε 10 μαθητές θα οδηγήσει σε παραπλανητικά αποτελέσματα. Όταν οι άνθρωποι που αναλύουν δεδομένα λέμε ότι είναι προκατειλημμένοι, σημαίνει ότι εσκεμμένα ωθούν τα αποτελέσματα της ανάλυσής τους σε μια συγκεκριμένη κατεύθυνση.

Οι πέντε πιο συνηθισμένες μορφές του δείκτη bias είναι:

i. Δείκτης bias επιβεβαίωσης

Παρουσιάζεται όταν το άτομο που πραγματοποιεί την ανάλυση δεδομένων θέλει να αποδείξει μια προκαθορισμένη παραδοχή. Ο αναλυτής προσαρμόζει τα δεδομένα ώστε να αποδειχθεί αυτή η υπόθεση, εξαιρώντας σκόπιμα συγκεκριμένες μεταβλητές από την ανάλυσή του. Αυτό συμβαίνει όταν οι αναλυτές δεδομένων χειραγωγούνται εκ των προτέρων ώστε να υποστηρίξουν ένα συγκεκριμένο συμπέρασμα.

ii. Δείκτης bias επιλογής

Αυτό συμβαίνει όταν τα δεδομένα επιλέγονται υποκειμενικά με αποτέλεσμα, το δείγμα που χρησιμοποιείται να μην αποτελεί μια σωστή αντανάκλαση του πληθυσμού. Αυτό το σφάλμα παρατηρείται συχνά στην επιλογή ομάδων ατόμων ώστε να διεξαχθούν έρευνες.

Επομένως, καλό θα είναι το δείγμα που θα χρησιμοποιηθεί για έρευνα να είναι αρκετά περιεκτικό ώστε να αντανακλά όσο γίνεται καλύτερα ολόκληρο τον πληθυσμό.

iii. Ακραίες τιμές

Οι ακραίες τιμές θα πρέπει να εξαιρούνται από ένα σύνολο δεδομένων καθώς αποκλίνουν αρκετά του μέσου όρου του δείγματος και δημιουργούν ανωμαλίες στην εξαγωγή αποτελεσμάτων. Ο αναλυτής θα πρέπει να ελέγχει αν το δείγμα το οποίο χρησιμοποιεί εμπεριέχει ακραίες τιμές.

iv. Υπερφόρτωση/Υποφόρτωση

Η υπερφόρτωση συμβαίνει όταν ένα μοντέλο μαθαίνει από τον θόρυβο των δεδομένων εισόδου, στο βαθμό που επηρεάζει αρνητικά την απόδοση του μοντέλου. Αυτό σημαίνει ότι ο θόρυβος, ή οι τυχαίες διακυμάνσεις στα δεδομένα εκπαίδευσης, προωθούνται και «μαθαίνονται» από το μοντέλο ως δεδομένα χρήσιμα για την εξαγωγή αποτελέσματος.

Η υποφόρτωση αναφέρεται σε ένα μοντέλο που δεν μπορεί να επεξεργαστεί τα δεδομένα εισόδου. Ένα υποφορτωμένο μοντέλο μηχανικής μάθησης έχει κακή απόδοση στην εξαγωγή αποτελέσματος.

v. Σύγχυση μεταβλητών

Το φαινόμενο της σύγχυσης μεταβλητών εμφανίζεται όταν θεωρούμε λανθασμένα ότι μια μεταβλητή εξαρτάται από μια άλλη, ενώ στην πραγματικότητα μια «κρυφή» μεταβλητή προκαλεί τη συσχέτιση των δυο πρώτων. Για παράδειγμα, αν μια έρευνα δείξει ότι καθώς αυξάνονται οι πωλήσεις παγωτών αυξάνεται και ο αριθμός των ανθρώπων που πνίγονται στο νερό δεν σημαίνει ότι τα παγωτά σχετίζονται με τους πνιγμούς. Μια «κρυφή» μεταβλητή, όπως για παράδειγμα ο καιρός, ευθύνεται γι' αυτή τη συσχέτιση καθώς εκείνος είναι που ωθεί τους ανθρώπους να αγοράζουν παγωτά και να πηγαίνουν για κολύμπι.

2.2.3 Νευρωνικό δίκτυο

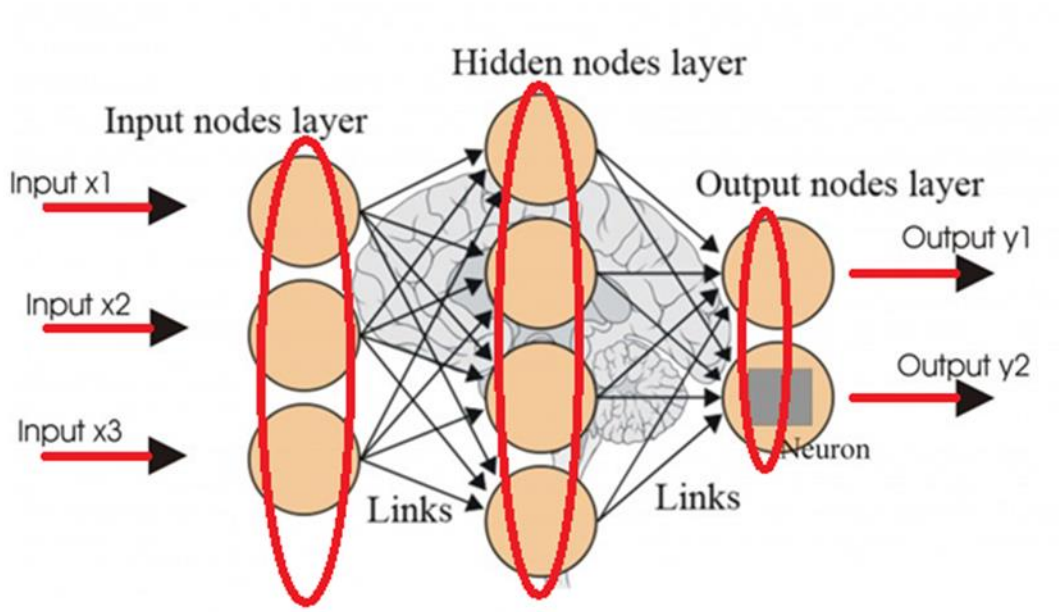
Νευρωνικό δίκτυο ονομάζεται ένα κύκλωμα διασυνδεδεμένων νευρώνων. Στην περίπτωση βιολογικών νευρώνων, πρόκειται για ένα τμήμα νευρικού ιστού. Στη περίπτωση των τεχνητών νευρώνων πρόκειται για ένα αφηρημένο αλγοριθμικό κατασκεύασμα το οποίο εμπίπτει στον τομέα της υπολογιστικής νοημοσύνης, όταν στόχος του νευρωνικού δικτύου είναι η επίλυση κάποιου υπολογιστικού προβλήματος, ή της υπολογιστικής νευροεπιστήμης, όταν στόχος είναι η υπολογιστική προσομοίωση της λειτουργίας των βιολογικών νευρωνικών δικτύων με βάση κάποιο μαθηματικό μοντέλο τους.

Το νευρωνικό δίκτυο είναι ένα δίκτυο από απλούς υπολογιστικούς κόμβους (νευρώνες, νευρώνια), διασυνδεδεμένους μεταξύ τους. Είναι εμπνευσμένο από το Κεντρικό Νευρικό Σύστημα (ΚΝΣ), το οποίο προσπαθεί να προσομοιώσει.

Οι νευρώνες είναι τα δομικά στοιχεία του δικτύου. Κάθε τέτοιος κόμβος δέχεται ένα σύνολο αριθμητικών εισόδων από διαφορετικές πηγές (είτε από άλλους νευρώνες, είτε από το περιβάλλον), επιτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο. Η εν λόγω έξοδος είτε κατευθύνεται στο περιβάλλον, είτε τροφοδοτείται ως είσοδος σε άλλους νευρώνες του δικτύου. Υπάρχουν τρεις τύποι νευρώνων: οι νευρώνες εισόδου, οι νευρώνες εξόδου και οι υπολογιστικοί νευρώνες ή κρυμμένοι νευρώνες.

Οι νευρώνες εισόδου δεν επιτελούν κανέναν υπολογισμό, μεσολαβούν απλώς ανάμεσα στις περιβαλλοντικές εισόδους του δικτύου και στους υπολογιστικούς νευρώνες. Οι νευρώνες εξόδου διοχετεύουν στο περιβάλλον τις τελικές αριθμητικές εξόδους του δικτύου. Οι υπολογιστικοί νευρώνες πολλαπλασιάζουν κάθε είσοδό τους με το αντίστοιχο συναπτικό βάρος και υπολογίζουν το ολικό άθροισμα των γινομένων. Το άθροισμα αυτό τροφοδοτείται ως όρισμα στη συνάρτηση ενεργοποίησης, την οποία υλοποιεί εσωτερικά κάθε κόμβος. Η τιμή που λαμβάνει η συνάρτηση για το εν λόγω όρισμα είναι και η έξοδος του νευρώνα για τις τρέχουσες εισόδους και βάρη. Πηγή

Στην **Εικόνα 5** απεικονίζεται μια απλή μορφή νευρωνικού δικτύου, όπου οι κύκλοι αποτελούν τους νευρώνες και τα μαύρα μικρά βέλη τις διασυνδέσεις μεταξύ τους. Τα κόκκινα βέλη αποτελούν τις εισόδους και εξόδους του δικτύου και τα κόκκινα οβάλ σχήματα που περικλείουν τους νευρώνες ονομάζονται στρώματα.



Εικόνα 5: Νευρωνικό δίκτυο

https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/

Πλεονεκτήματα νευρωνικών δικτύων

- **Αποθήκευση πληροφοριών σε ολόκληρο το δίκτυο**

Οι πληροφορίες, σε αντίθεση με τον παραδοσιακό προγραμματισμό αποθηκεύονται σε ολόκληρο το δίκτυο και όχι σε κάποια βάση δεδομένων. Η απώλεια ορισμένων πληροφοριών δεν εμποδίζει τη λειτουργία του δικτύου.

- **Δυνατότητα λειτουργίας με ελλιπή γνώση**

Αφού εκπαιδευτεί το νευρωνικό δίκτυο, μπορεί να παράγει αποτελέσματα ακόμη και με ελλιπή πληροφόρηση. Η απώλεια της απόδοσης εξαρτάται από τη σημαντικότητα των πληροφοριών που λείπουν.

- **Έχουν ανοχή σφάλματος**

Η καταστροφή ενός ή περισσότερων κυττάρων ενός νευρωνικού δικτύου δεν το εμποδίζει να παράγει αποτελέσματα. Αυτό το χαρακτηριστικό καθιστά τα δίκτυα ανεκτικά στα σφάλματα.

- **Παράλληλη ικανότητα επεξεργασίας**

Τα τεχνητά νευρωνικά δίκτυα έχουν υπολογιστική ισχύ η οποία μπορεί να πραγματοποιήσει περισσότερες από μία εργασίες ταυτόχρονα.

Μειονεκτήματα νευρωνικών δικτύων

- **Εξάρτηση από το hardware**

Τα τεχνητά νευρωνικά δίκτυα απαιτούν επεξεργαστές με μεγάλη ισχύ επεξεργασίας.

- **Αδυναμία επεξήγησης του αποτελέσματος που παρέχουν**

Αυτό είναι το σημαντικότερο πρόβλημα των τεχνητών νευρωνικών δικτύων. Ένα νευρωνικό δίκτυο δεν δίνει καμία επεξήγηση για το πώς παρήχθησαν τα αποτελέσματα του, πράγμα το οποίο μειώνει την εμπιστοσύνη προς στο δίκτυο.

- **Αδυναμία προσδιορισμού σωστής διάρθρωσης του δικτύου**

Δεν υπάρχει κανένας ειδικός κανόνας για τον προσδιορισμό της δομής των τεχνητών νευρωνικών δικτύων. Η κατάλληλη διάρθρωση του δικτύου επιτυγχάνεται μέσω εμπειρίας, δοκιμών και σφαλμάτων.

- **Αδυναμία επεξήγησης του προβλήματος στο δίκτυο**

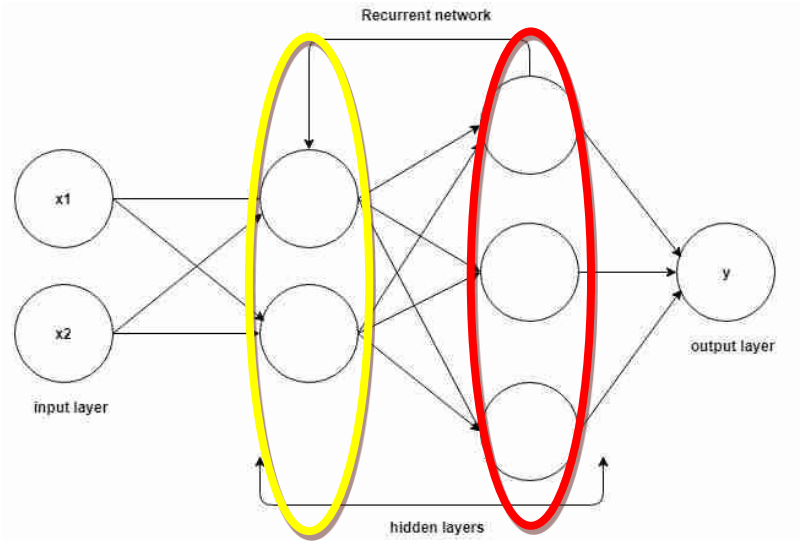
Τα τεχνητά νευρωνικά δίκτυα μπορούν να λειτουργήσουν με αριθμητικές πληροφορίες. Τα προβλήματα θα πρέπει να μεταφραστούν σε αριθμητικές τιμές προτού εισαχθούν σε ένα νευρωνικό δίκτυο. Ο μηχανισμός απεικόνισης του προβλήματος επηρεάζει άμεσα την απόδοση του δικτύου. Η δυνατότητα απεικόνισης του προβλήματος εξαρτάται από την ικανότητα του χρήστη.

2.2.4 Αναδρομικό νευρωνικό δίκτυο

Ένα Αναδρομικό Νευρωνικό Δίκτυο ή Recurrent Neural Network (RNN) είναι μια υποκατηγορία τεχνητού νευρωνικού δικτύου το οποίο χρησιμοποιείται κατά βάση σε εφαρμογές που αφορούν την αναγνώριση ομιλίας και την επεξεργασία φυσικής γλώσσας. Τα RNNs έχουν σχεδιαστεί να αναγνωρίζουν «χαρακτηριστικά» του συνόλου δεδομένων τους και να χρησιμοποιούν πρότυπα με στόχο τη καλύτερη δυνατή πρόβλεψη.

Τα RNNs χρησιμοποιούνται στη Βαθιά Μάθηση (Deep Learning) και στην ανάπτυξη μοντέλων που εξομοιώνουν τις λειτουργίες των νευρώνων του ανθρώπινου εγκεφάλου. Είναι πολύ ακριβή στις προβλέψεις τους και ξεχωρίζουν από τις υπόλοιπες υποκατηγορίες τεχνητών νευρωνικών δικτύων διότι χρησιμοποιούν βρόγχους ανατροφοδότησης κατά την επεξεργασία μιας ακολουθίας δεδομένων μαζί με το αρχικό σύνολο δεδομένων, ώστε να ενισχύσουν τις προβλέψεις τους. Αυτοί οι βρόγχοι ανατροφοδότησης στην περίπτωση των RNNs ονομάζονται μνήμη.

Οι περιπτώσεις εφαρμογής των RNNs αφορούν συχνά προβλήματα ταξινόμησης κειμένου όπου η επόμενη λέξη προβλέπεται από το νευρωνικό δίκτυο με βάση τα δεδομένα που προηγούνται. Όπως φαίνεται στην **Εικόνα 6** εφαρμόζεται μια επαναληπτική λειτουργία όπου δίνει την έξοδο του δεύτερου στρώματος (κόκκινο) ξανά προς τροφοδότηση στο προηγούμενο στρώμα (κίτρινο).



Εικόνα 6: Αναδρομικό νευρωνικό δίκτυο
https://towardsdatascience.com/machine-learning-recurrent-neural-networks-and-long-short-term-memory-lstm-python-keras-example-86001ceaaebc?gi=1d76502a03b9

2.2.5 Δίκτυα ανατροφοδοτούμενης μονάδας με πύλη

Το δίκτυο Ανατροφοδοτούμενης Μονάδας με Πύλη ή αλλιώς Gated Recurrent Unit (GRU), το οποίο χρησιμοποιήθηκε και στη παρούσα εργασία, είναι μια υποκατηγορία των Αναδρομικών Νευρωνικών Δικτύων.

Τα δίκτυα αυτού του τύπου στοχεύουν στην επίλυση του προβλήματος της εξαφάνισης κλίσης που έχει συνήθως ένα τυποποιημένο Αναδρομικό Νευρωνικό Δίκτυο. Στα Αναδρομικά Νευρωνικά Δίκτυα, κάθε ένα από τα βάρη του νευρωνικού δικτύου λαμβάνει μια ενημέρωση αναλογική προς το μερικό παράγωγο της συνάρτησης σφάλματος σε σχέση με το τρέχον βάρος σε κάθε επανάληψη της εκπαίδευσης.

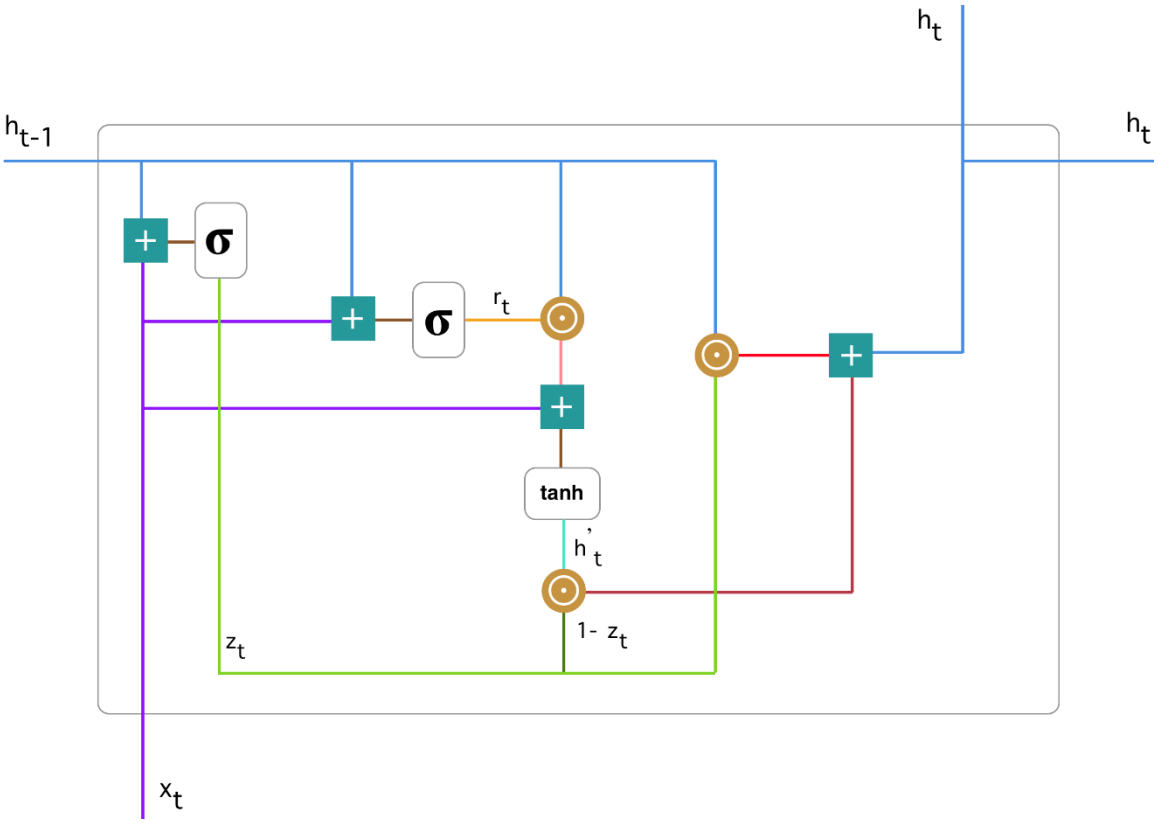
Με απλά λόγια, οι τιμές που δίνουν τα επιπλέον συναπτικά βάρη που δημιουργούνται από την εφαρμογή ενός επαναλαμβανόμενου δικτύου, τείνουν να μην επηρεάζουν καθόλου το δίκτυο στο σύνολό του με συνέπεια να το καθιστούν αναποτελεσματικό.

Τα δίκτυα GRU μπορούν επίσης να θεωρηθούν ως μια παραλλαγή των LSTM, επειδή και τα δύο έχουν παρόμοιο σχεδιασμό και σε ορισμένες περιπτώσεις, παράγουν εξίσου εξαιρετικά αποτελέσματα.

Αυτό που ξεχωρίζει τα δίκτυα GRU από τα δίκτυα LSTM, είναι ότι τα δίκτυα αυτού του τύπου, έχουν τη δυνατότητα να αποθηκεύουν ένα κομμάτι των πληροφοριών που προέρχεται από προηγούμενα στρώματα του δικτύου, για κάποιο χρονικό διάστημα. Έπειτα, την κατάλληλη στιγμή, διοχετεύουν το αποθηκευμένο κομμάτι πληροφοριών στο δίκτυο, μαζί με την ήδη υπάρχουσα ροή πληροφοριών ώστε να καλυτερεύσουν την ποιότητα των πληροφοριών για τα επόμενα στρώματα.

Λειτουργία GRU δικτύου

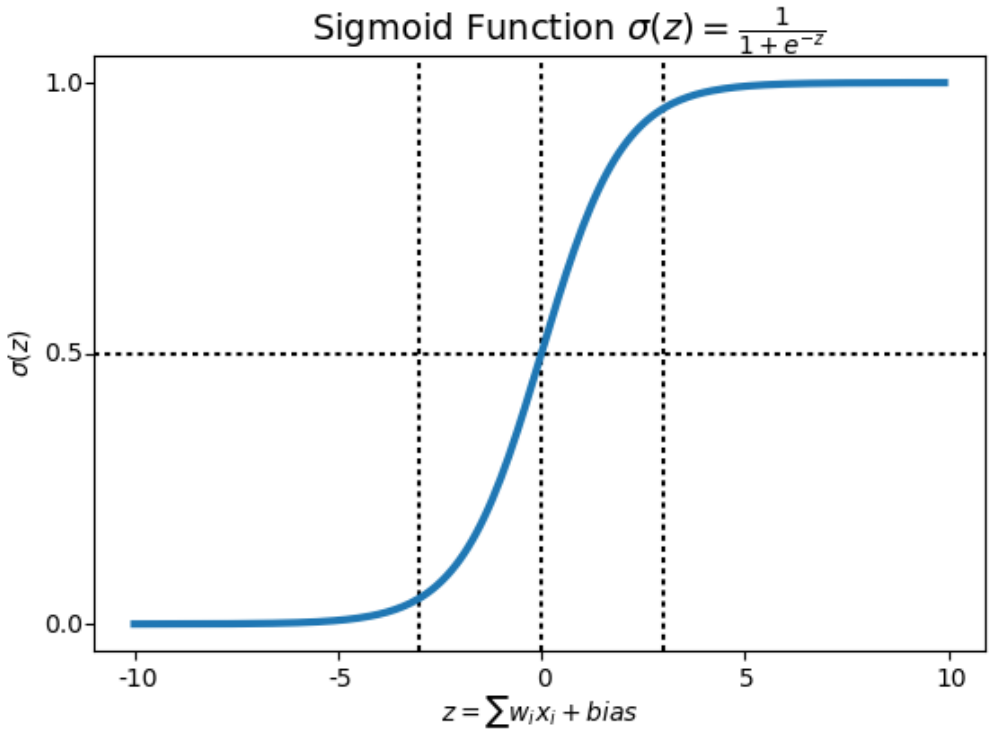
Στην **Εικόνα 7** φαίνεται η αρχιτεκτονική ενός GRU δικτύου. Το σύμβολο (+) είναι το σύμβολο της πρόσθεσης, το (σ) αποτελεί την σιγμοειδή λειτουργία, ο κίτρινος κύκλος τη λειτουργία "Hadamard Product" και το σύμβολο (tanh) τη λειτουργία (tanh).



Εικόνα 7: Αρχιτεκτονική GRU δικτύου
https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

- Σιγμοειδής λειτουργία:

Η σιγμοειδής λειτουργία είναι μια μαθηματική συνάρτηση που έχει χαρακτηριστική καμπύλη σχήματος "S" ή αλλιώς σιγμοειδή καμπύλη και κυμαίνεται στο διάστημα (0,1). Η συχνότερη μορφή σιγμοειδούς λειτουργίας είναι η logistic sigmoid function η οποία παρουσιάζεται στην **Εικόνα 8** και καθορίζεται από τον τύπο που φαίνεται στην εν λόγω εικόνα.



Εικόνα 8: Logistic sigmoid function
https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

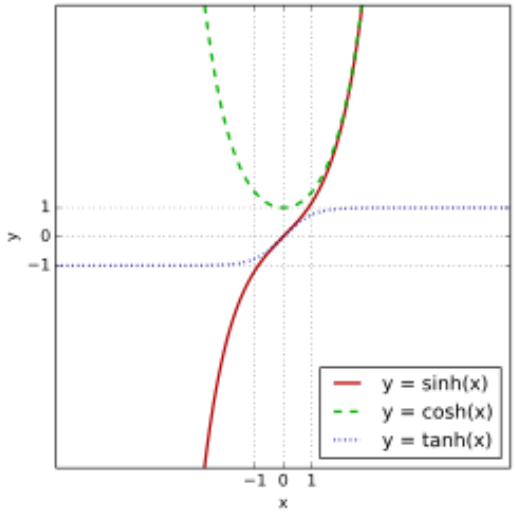
- Hadamard Product:

Ως Hadamard Product ορίζουμε μια λειτουργία η οποία λαμβάνει στοιχεία από δύο μήτρες ίδιων διαστάσεων και παράγει μια τρίτη μήτρα ίδιας διάστασης όπου κάθε της στοιχείο είναι το παράγωγο των στοιχείων i, j των αρχικών δύο μητρών.

- Tanh function:

Η tanh function (TANH(x)) επιστρέφει την υπερβολική εφαπτομένη της γωνίας x. Στην **Εικόνα 9** απεικονίζεται η υπερβολική εφαπτομένη, η οποία ορίζεται ως το υπερβολικό ημίτονο της γωνίας x προς το υπερβολικό συνημίτονο της γωνίας x και κυμαίνεται στο διάστημα (-1,1):

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$



Εικόνα 9: Tanh function

https://en.wikipedia.org/wiki/Hyperbolic_functions

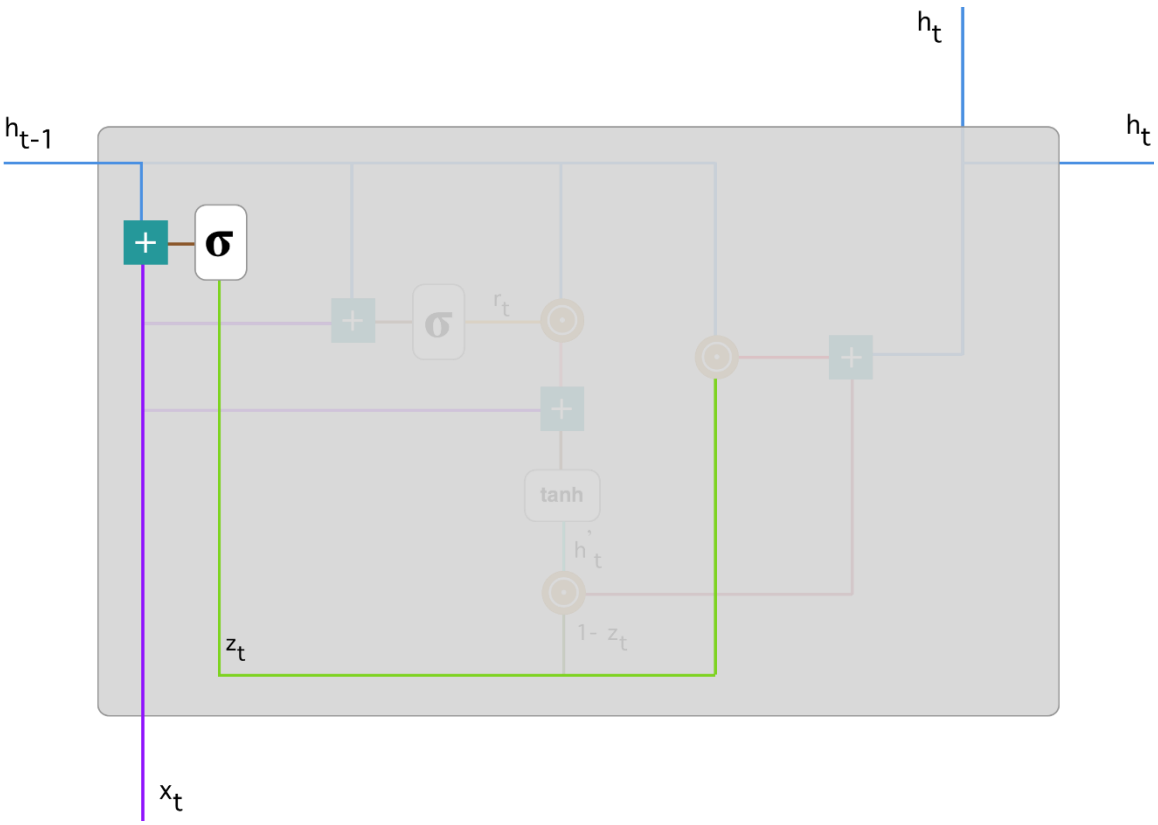
Βήματα που ακολουθεί το δίκτυο:

- ✓ Πύλη ενημέρωσης

Αρχικά το δίκτυο υπολογίζει την πύλη ενημέρωσης z_t για το χρονικό σημείο t χρησιμοποιώντας τον τύπο:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

Όταν το αποτέλεσμα x_t ενός στρώματος νευρώνων είναι έτοιμο, πολλαπλασιάζεται με το δικό του βάρος W (z). Το ίδιο ισχύει και για το h_ (t-1) το οποίο είναι το αποτέλεσμα των προηγούμενων στρωμάτων νευρώνων και πολλαπλασιάζεται με το δικό του βάρος U (z). Τα δύο αποτελέσματα προστίθενται μαζί και εφαρμόζεται μια σιγμοειδής λειτουργία στο άθροισμα ώστε να μετατραπεί το άθροισμα σε μια διακύμανση μεταξύ 0 και 1. Ακολουθώντας το παραπάνω συλλογισμό, έχουμε:



Εικόνα 10: Πύλη ενημέρωσης
https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

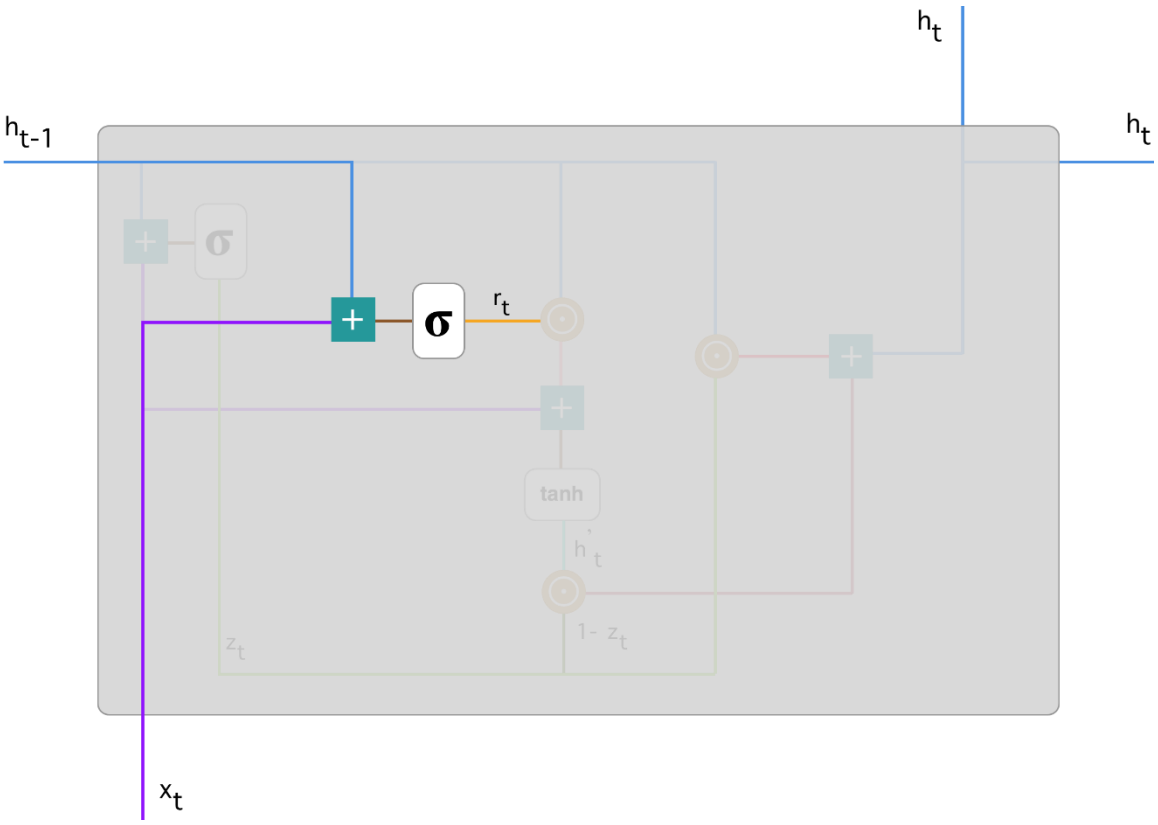
Όπως φαίνεται στην **Εικόνα 10** η πύλη ενημέρωσης βοηθά το μοντέλο να καθορίσει πόσα από τα στοιχεία του προηγούμενου στρώματος πρέπει να μεταφερθούν στο επόμενο στρώμα.

- ✓ Πύλη επαναφοράς

Στη συνέχεια το δίκτυο αποφασίζει πόσες από τις πληροφορίες που προέρχονται από προηγούμενα στρώματα δε θα πρέπει να λάβει υπόψιν. Για να το υπολογίσει αυτό χρησιμοποιεί τον τύπο:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

Αυτός ο τύπος είναι ο ίδιος με αυτόν του προηγούμενου βήματος. Η **Εικόνα 11** δείχνει πού βρίσκεται στο δίκτυο η πύλη επαναφοράς:



Εικόνα 11: Πύλη επαναφοράς
https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Όπως και πριν, το δίκτυο προσθέτει τα αποτελέσματα των προηγούμενων στρωμάτων h_ (t-1) - μπλε γραμμή και του επικείμενου στρώματος x_t - μοβ γραμμή, και τα πολλαπλασιάζει με τα αντίστοιχα βάρη τους. Στη συνέχεια, αθροίζει τα αποτελέσματα και εφαρμόζει τη λειτουργία sigmoid.

- ✓ Τρέχον περιεχόμενο μνήμης

Στο βήμα αυτό εισάγονται τα αποτελέσματα του προηγούμενου βήματος και αλληλεπιδρούν με τα αποτελέσματα του τρέχοντος στρώματος ως εξής:

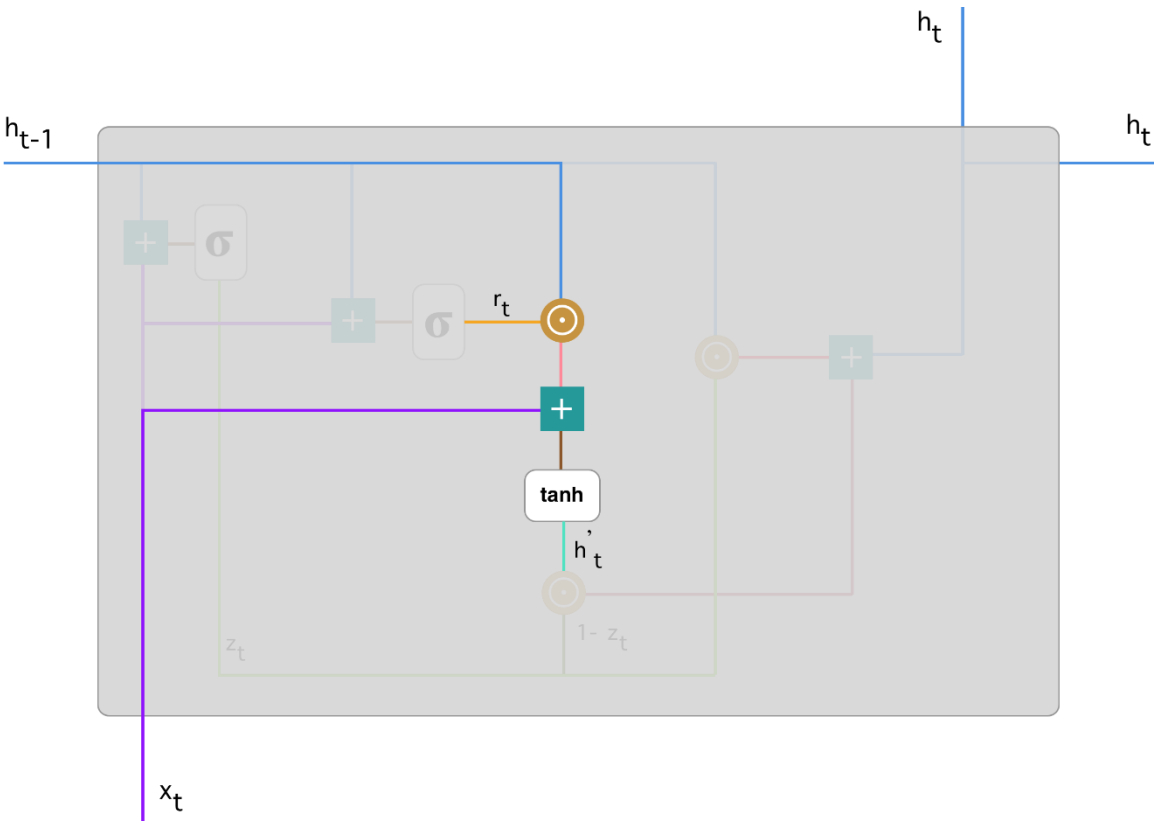
$$h_t^{'} = \tanh(Wx_t + r_t \odot Uh_{t-1})$$

- ❖ Πολλαπλασιάζονται τα προηγούμενα αποτελέσματα x_t με βάρος W και τα αποτελέσματα του τρέχοντος στρώματος h_ (t-1) με βάρος U.

- ❖ Υπολογίζεται το Hadamard product μεταξύ της πύλης επαναφοράς r_t και Uh_ (t-1). Σ' αυτό το σημείο καθορίζεται τι πρέπει να αφαιρεθεί από τα αποτελέσματα των προηγούμενων στρωμάτων.
  Για να γίνει πιο κατανοητή η λειτουργία του δικτύου ας υποθέσουμε ότι έχουμε ως σύνολο δεδομένων εισόδου ένα κείμενο το οποίο αρχίζει με τη φράση «Αυτό είναι ένα βιβλίο φαντασίας που παρουσιάζει...» και τελειώνει με τη φράση «Το βιβλίο αυτό δε μου άρεσε καθόλου». Σε περίπτωση που θέλουμε να κάνουμε μια συναισθηματική ανάλυση όπως αναφέραμε παραπάνω, μέσω τις λειτουργίας GRU το νευρωνικό δίκτυο είναι σε θέση να παραβλέψει την αρχική πρόταση και να εστιάσει στη δήλωση αρεσκείας του βιβλίου.

- ❖ Αθροίζονται τα δύο παραπάνω αποτελέσματα

- ❖ Εφαρμόζεται μια tanh function επί των αποτελεσμάτων.



Εικόνα 12: Τρέχον περιεχόμενο μνήμης
https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

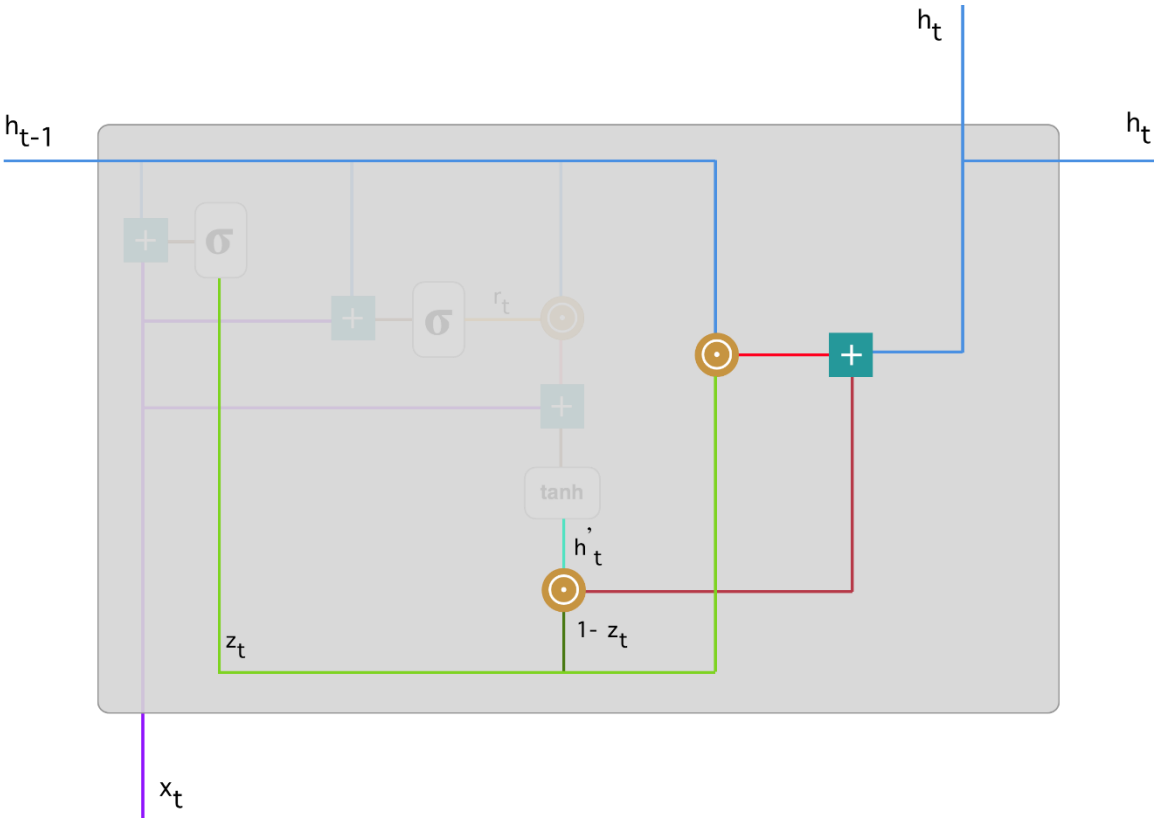
Στην **Εικόνα 12** φαίνονται όσα περιγράφηκαν παραπάνω. Πολλαπλασιάζεται το στοιχείο h_ (t-1) - μπλε γραμμή με το στοιχείο r_t - πορτοκαλί γραμμή και στη συνέχεια αθροίζεται το παράγωγο - ροζ γραμμή με την είσοδο x_t - μοβ γραμμή. Τέλος, αφού εφαρμοστεί η λειτουργία tanh παράγεται η γραμμή h'_t - ανοιχτό πράσινο.

- ✓ Τελική επιλογή αποτελέσματος για το επόμενο στρώμα

Ως τελευταίο βήμα, το δίκτυο πρέπει να υπολογίσει τον δείκτη h_t ο οποίος ουσιαστικά καθορίζει ποιες πληροφορίες από τα προηγούμενα στρώματα και ποιες από τα αποτελέσματα των προηγούμενων βημάτων θα προχωρήσουν στα επόμενα στρώματα. Αυτό γίνεται ως εξής:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'}$$

- Υπολογίζεται το Hadamard product μεταξύ των στοιχείων της πύλης ενημέρωσης με τα στοιχεία των προηγούμενων στρωμάτων z_t και h_ (t-1).
- Υπολογίζεται το Hadamard product μεταξύ των στοιχείων (1-z_t) και του τρέχοντος περιεχομένου μνήμης h'_t.
- Αθροίζονται τα δύο αυτά αποτελέσματα.



Εικόνα 13: Τελική επιλογή αποτελέσματος
https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Σύμφωνα με την **Εικόνα 13** φαίνεται πώς η z_t - πράσινη γραμμή χρησιμοποιείται για τον υπολογισμό 1-z_t που, σε συνδυασμό με την h'_t - ανοιχτή πράσινη γραμμή, παράγει ένα αποτέλεσμα που απεικονίζεται με τη σκούρα κόκκινη γραμμή. Το z_t αλληλεπιδρά με το h_ (t-1) - μπλε γραμμή σε έναν πολλαπλασιασμό στοιχείων Hadamard product. Τέλος, η h_t - μπλε γραμμή είναι το αποτέλεσμα του αθροίσματος των εξόδων που αντιστοιχούν στις, φωτεινή κόκκινη και σκούρα κόκκινη, γραμμές. Πηγή

2.2.6 Συνάρτηση softmax

Στα μαθηματικά, η συνάρτηση softmax, γνωστή και ως softargmax ή κανονικοποιημένη εκθετική συνάρτηση, είναι μια συνάρτηση που εισάγει ως είσοδο ένα διάνυσμα πραγματικών αριθμών K και την ομαλοποιεί, σε μια κατανομή πιθανότητας που αποτελείται από K πιθανότητες, ανάλογες προς τους εκθέτες των αριθμών εισόδου. Η **Εικόνα 14** δείχνει συνοπτικά τον τύπο. Πριν από την εφαρμογή της softmax, ορισμένα στοιχεία του διανύσματος θα μπορούσαν να έχουν αρνητικές τιμές ή μεγαλύτερες του ενός αλλά μετά την εφαρμογή της softmax, κάθε στοιχείο θα ανήκει στο διάστημα (0,1) έτσι ώστε να μπορούν να ερμηνευτούν ως πιθανότητες. Επιπλέον, τα στοιχεία εισόδου με τις μεγαλύτερες τιμές θα αντιστοιχούν σε μεγαλύτερες πιθανότητες. Η Softmax χρησιμοποιείται συχνά στα νευρωνικά δίκτυα για να απεικονίσει το μη κανονικοποιημένο παράγωγο ενός δικτύου ως μια κατανομή πιθανότητας. Πηγή

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$

Εικόνα 14: Συνάρτηση softmax

https://en.wikipedia.org/wiki/Softmax_function

2.2.7 Αλγόριθμος βελτιστοποίησης Adam

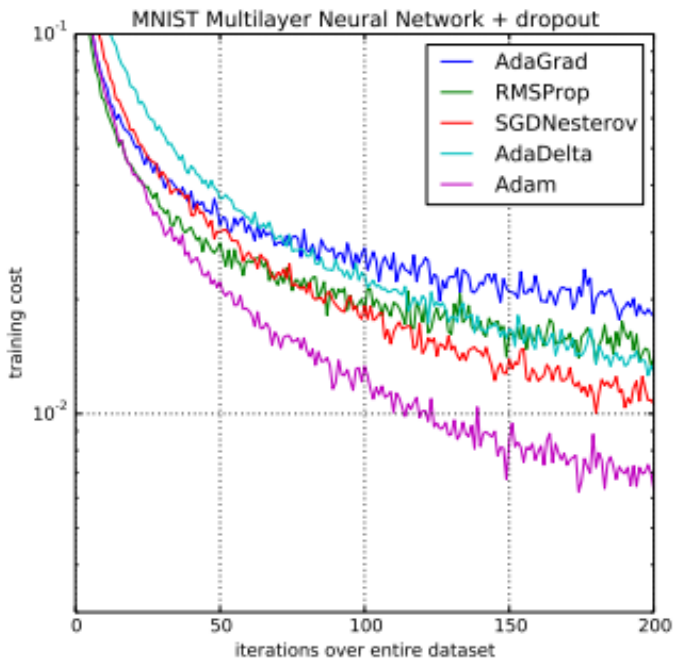
Ο Adam είναι ένας αλγόριθμος βελτιστοποίησης που μπορεί να χρησιμοποιηθεί αντί της κλασικής διαδικασίας στοχαστικής κλίσης για την ενημέρωση των βαρών του δικτύου με βάση τα δεδομένα εκπαίδευσης. Για να είναι σε θέση να μαθαίνουν τα νευρωνικά δίκτυα, θα πρέπει τα συναπτικά βάρη μεταξύ των νευρώνων να ενημερώνονται καθώς προχωράει η διαδικασία διαβίβασης των δεδομένων στο δίκτυο. Ο αλγόριθμος Adam είναι αρωγός στην ενημέρωση των συναπτικών βαρών των νευρώνων του δικτύου, η οποία αποσκοπεί στο να προσαρμοστούν οι προβλέψεις του δικτύου με τα πραγματικά δεδομένα τα οποία εισάγονται στο δίκτυο.

Ο Adam είναι διαφορετικός από την κλασική διαδικασία στοχαστικής κλίσης, καθώς η στοχαστική κλίση διατηρεί ένα μοναδικό ρυθμό εκμάθησης για όλες τις ενημερώσεις βάρους και ο ρυθμός εκμάθησης δεν αλλάζει κατά τη διάρκεια της εκπαίδευσης. Στη περίπτωση του Adam ο ρυθμός εκμάθησης διατηρείται για κάθε βάρος του δικτύου ξεχωριστά και προσαρμόζεται ως εκμάθηση στην συνολική εκμάθηση του δικτύου. Οι συγγραφείς περιγράφουν τον Adam ως ένα συνδυασμό των πλεονεκτημάτων των δύο άλλων επεκτάσεων της στοχαστικής κλίσης, τους αλγορίθμους Adaptive Gradient Algorithm και RMSProp.

- **Adaptive Gradient Algorithm (AdaGrad)**
  Ο AdaGrad ενώ διατηρεί έναν ανά παράμετρο ρυθμό εκμάθησης, ο ρυθμός αυτός τείνει να συρρικνώνεται κατά τη διαδικασία εκπαίδευσης του δικτύου, το οποίο σημαίνει ότι το νευρωνικό δίκτυο παύει να αποκτά επιπλέον γνώση.

- **RMSProp**
  Ο RMSProp διατηρεί επίσης ποσοστά εκμάθησης ανά παράμετρο τα οποία όμως προσαρμόζονται με βάση το μέσο όρο μόνο των αρχικών συναπτικών βαρών του δικτύου.

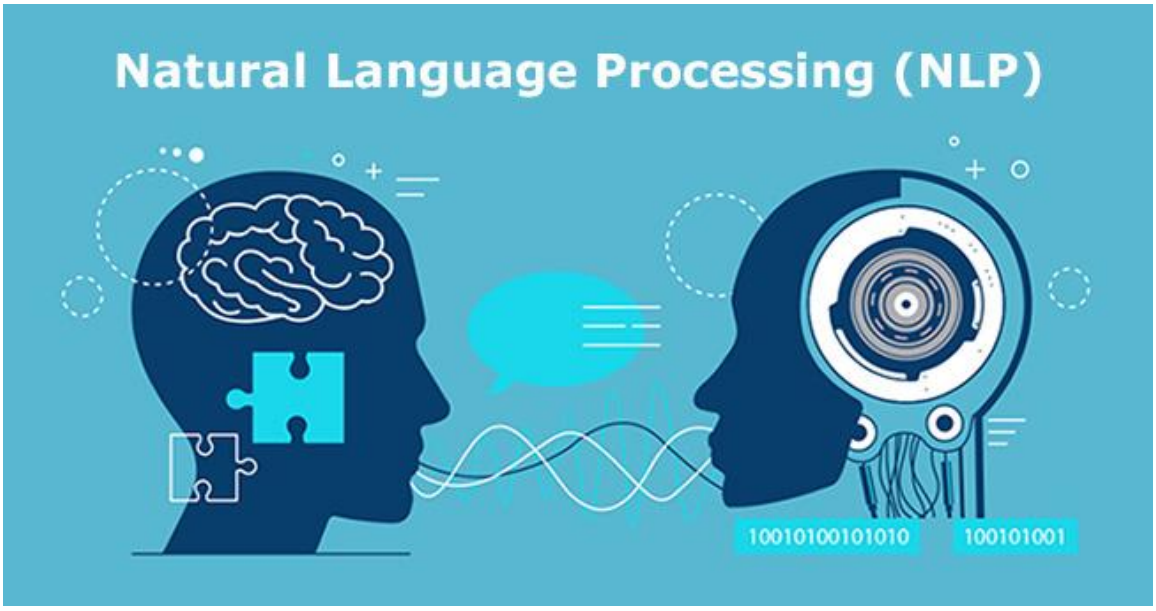
Ο Adam συγκεντρώνει τα οφέλη τόσο του AdaGrad όσο και του RMSProp. Αντί να προσαρμόζει μόνο τα ποσοστά εκμάθησης των παραμέτρων βάσει των αρχικών συναπτικών βαρών του δικτύου, όπως στη περίπτωση του RMSProp, ο Adam χρησιμοποιεί επίσης τον μέσο όρο και των δεύτερων συναπτικών βαρών του δικτύου, δηλαδή αφού ήδη έχουν ενημερωθεί μια φορά τα συναπτικά βάρη των νευρώνων του δικτύου. Στην **Εικόνα 15** φαίνεται η υπεροχή του Adam έναντι των άλλων αλγορίθμων καθώς μειώνει το «κόστος» εκπαίδευσης πολύ γρηγορότερα το οποίο με τη σειρά του σημαίνει ότι το δίκτυο είναι σε θέση να μαθαίνει ταχύτερα. Πηγή



Εικόνα 15: Σύγκριση Adam με άλλους αλγόριθμους
https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

2.2.8 Επεξεργασία φυσικής γλώσσας

Η επεξεργασία φυσικής γλώσσας (ΕΦΓ) είναι ένας διεπιστημονικός κλάδος της επιστήμης της πληροφορικής, της τεχνητής νοημοσύνης και της υπολογιστικής γλωσσολογίας και ασχολείται με τις αλληλεπιδράσεις μεταξύ των υπολογιστών και των ανθρώπινων (φυσικών) γλωσσών. Κατά συνέπεια, η ΕΦΓ συνδέεται στενά με την αλληλεπίδραση ανθρώπου-υπολογιστή. Προκλήσεις στην ΕΦΓ περιλαμβάνουν την κατανόηση φυσικής γλώσσας, δηλαδή την προσπάθεια να καταστούν ικανοί οι υπολογιστές να εξάγουν νοήματα από ανθρώπινα ή γλωσσικά δεδομένα, αλλά και την παραγωγή φυσικής γλώσσας. Η **Εικόνα 16** παρομοιάζει τη μετάδοση ενός μηνύματος από έναν άνθρωπο και την αποκωδικοποίηση του μηνύματος από έναν υπολογιστή. Πηγή



Εικόνα 16: Επεξεργασία φυσικής γλώσσας
https://thinkpalm.com/blogs/natural-language-processing-nlp-artificial-intelligence/

3. Απαιτήσεις της εφαρμογής που πρόκειται να αναπτυχθεί

3.1 Σκοπός της εφαρμογής

Σκοπός της εφαρμογής είναι η πρόβλεψη των tokens (λέξεις, σύμβολα) που ακολουθούν την εισαγωγή ενός αρχικού token με βάση τη γλώσσα προγραμματισμού Java. Για παράδειγμα, έστω ότι ο χρήστης της εφαρμογής εισάγει ως αρχικό, το token «import». Η εφαρμογή θα πρέπει να είναι σε θέση να συμπληρώσει τα επόμενα tokens που ακολουθούν με βάση τις προβλέψεις της π.χ.:

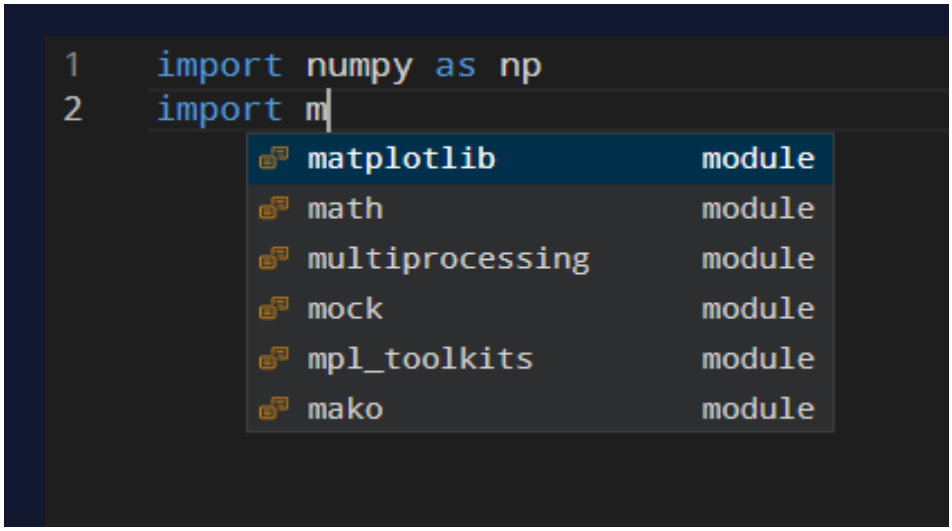
«import java.util.scanner»

Φυσικά, τα tokens «java.util.scanner» δεν είναι τα μοναδικά που μπορούν να ακολουθούν το token import, όμως είναι μια αρκετά στοχευμένη πρόβλεψη του τι θα ήθελε να συντάξει ο προγραμματιστής μετά την εισαγωγή του token «import».

Σε συνέχεια των παραπάνω, για να μπορέσει ο χρήστης να δει τις προβλέψεις της εφαρμογής, αναπτύχθηκε ένα αναδυόμενο παράθυρο το οποίο θα εμφανίζεται τη στιγμή που η εφαρμογή θα είναι σε θέση να παράγει τις προβλέψεις της.

Με αυτόν τον τρόπο γίνεται μια προσπάθεια δημιουργίας ενός εργαλείου αυτόματης συμπλήρωσης κώδικα, διευκόλυνσης των προγραμματιστών κατά τη συγγραφή κώδικα, καθώς και να δοθεί τροφή για σκέψη για μελλοντικές επεκτάσεις ώστε να αξιοποιηθούν στο μέγιστο οι δυνατότητες της μηχανικής μάθησης στα πλαίσια της ανάπτυξης λογισμικού.

Στην **Εικόνα 17** φαίνεται ένα παράδειγμα εργαλείου αυτόματης συμπλήρωσης κώδικα βασισμένο στη μηχανική μάθηση.



Εικόνα 17: Εργαλείο αυτόματης συμπλήρωσης κώδικα
https://kite.com/

3.2 Σε ποιους απευθύνεται η εφαρμογή

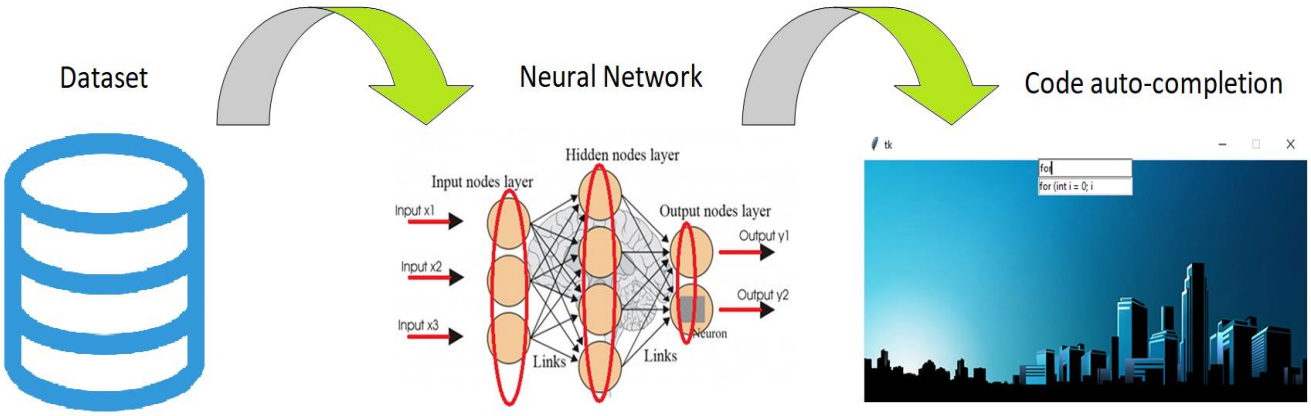
Η εφαρμογή απευθύνεται σε ερευνητές οι οποίοι είναι πρόθυμοι να ερευνήσουν και να πειραματιστούν με τις δυνατότητες των νευρωνικών δικτύων στα πλαίσια του κλάδου της επεξεργασίας φυσικής γλώσσας. Παρατηρώντας το μοντέλο νευρωνικού δικτύου που αναπτύχθηκε, οι ερευνητές μπορούν να προσπαθήσουν να αυξήσουν την ακρίβειά του, να δοκιμάσουν άλλα μοντέλα νευρωνικών δικτύων ή ακόμη και να συνδυάσουν μοντέλα μεταξύ τους. Τέλος θα μπορούσαν να καλυτερεύσουν τη λειτουργία του αναδυόμενου παραθύρου της εφαρμογής, ή να εφαρμόσουν άλλες μεθόδους παρουσίασης των αποτελεσμάτων στον χρήστη.

3.3 Χαρακτηριστικά της εφαρμογής

- ✓ Για να λειτουργήσει η εφαρμογή χρειάζεται, όπως και κάθε μοντέλο νευρωνικού δικτύου, ένα σύνολο δεδομένων ως είσοδο επί του οποίου το νευρωνικό δίκτυο θα εκπαιδευτεί και θα παράγει τις προβλέψεις του. Στη περίπτωση της εφαρμογής, χρησιμοποιήθηκε ένα αρχείο κειμένου το οποίο περιέχει κώδικα με βάση τη γλώσσα προγραμματισμού Java. Φυσικά, ο κάθε ενδιαφερόμενος, μπορεί να χρησιμοποιήσει ένα διαφορετικό σύνολο δεδομένων ως είσοδο στο νευρωνικό δίκτυο, ανάλογα με τις απαιτήσεις του.

- ✓ Στη συνέχεια θα πρέπει να σχεδιαστεί το μοντέλο νευρωνικού δικτύου, το οποίο θα εκπαιδευτεί και θα παρέχει τις προβλέψεις του στον χρήστη. Η επιλογή του μοντέλου νευρωνικού δικτύου προς ανάπτυξη είναι κρίσιμη, καθώς δεν έχουν σχεδιαστεί όλα τα μοντέλα νευρωνικών δικτύων για τους ίδιους σκοπούς. Στην εφαρμογή αυτή αναπτύχθηκε ένα μοντέλο αναδρομικού νευρωνικού δικτύου – ανατροφοδοτούμενης μονάδας με πύλη (RNN-GRU) καθώς κρίθηκε η καταλληλότερη επιλογή για τις προβλέψεις των tokens που προσπαθεί να επιτύχει η εφαρμογή. Επομένως, το σύνολο δεδομένων εισέρχεται στο νευρωνικό δίκτυο επεξεργάζεται από τα στρώματα νευρώνων του δικτύου (εκπαίδευση δικτύου) και τέλος παράγει τις προβλέψεις του.

- ✓ Τέλος, είναι χρήσιμη η περαιτέρω ανάπτυξη μιας εφαρμογής, η οποία πραγματεύεται τις δυνατότητες των νευρωνικών δικτύων, ώστε να είναι κανείς σε θέση να δει τα αποτελέσματα που του παρείχε το νευρωνικό δίκτυο. Για το λόγο αυτό λοιπόν σχεδιάστηκε ένα αναδυόμενο παράθυρο, το οποίο θα εμφανίζεται στο χρήστη μόλις το μοντέλο πραγματοποιήσει τις προβλέψεις του, όπου ο χρήστης θα μπορεί να δει τα αποτελέσματα του νευρωνικού δικτύου. Στην **Εικόνα 18** αντικατροπτίζεται η λειτουργία της εφαρμογής.



Εικόνα 18: Λειτουργία εφαρμογής

3.4 Απαιτήσεις λογισμικού-Python

Σύμφωνα με τη βικιπαίδεια η Python είναι μια διερμηνευόμενη (interprited), γενικού σκοπού (general-purpose) και υψηλού επιπέδου, γλώσσα προγραμματισμού. Ανήκει στις γλώσσες προστακτικού προγραμματισμού (Imperative programming) και υποστηρίζει τόσο το διαδικαστικό (procedural programming) όσο και το αντικειμενοστρεφές (object-oriented programming) προγραμματιστικό υπόδειγμα (programming paradigm). Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (garbage collection ή GC). Πηγή

Python και μηχανική μάθηση

Η Python προσφέρει συνοπτικό και ευανάγνωστο κώδικα. Ενώ βρίσκονται περίπλοκοι αλγόριθμοι πίσω από τη μηχανική μάθηση και την τεχνητή νοημοσύνη, η απλότητα της Python επιτρέπει στους προγραμματιστές να γράφουν αξιόπιστο κώδικα. Οι προγραμματιστές μπορούν να επικεντρωθούν στην επίλυση ενός προβλήματος μηχανικής μάθησης αντί να εστιάζουν στις τεχνικές απαιτήσεις της γλώσσας. Επιπλέον, η Python απευθύνεται και σε μη εξοικειωμένους προγραμματιστές καθώς είναι εύκολη στην εκμάθηση. Ο κώδικας σε Python είναι εύκολα κατανοητός από τον άνθρωπο, γεγονός που καθιστά ευκολότερη τη δημιουργία μοντέλων μηχανικής μάθησης.

3.5 Λογισμικό Natural Language Toolkit (NLTK)

Το NLTK είναι η κορυφαία πλατφόρμα για την ανάπτυξη προγραμμάτων σε Python για την επεξεργασία δεδομένων που σχετίζονται με την ανθρώπινη γλώσσα. Παρέχει μια σουίτα βιβλιοθηκών επεξεργασίας κειμένου για ταξινόμηση, tokenization, starking, tagging και parsing.

Χάρη σε έναν πρακτικό οδηγό που εισάγει θεμελιώδη στοιχεία προγραμματισμού παράλληλα με τα θέματα της υπολογιστικής γλωσσολογίας, το NLTK είναι κατάλληλο για γλωσσολόγους, μηχανικούς, φοιτητές, εκπαιδευτικούς, ερευνητές κ.α. Το NLTK είναι διαθέσιμο σε Windows, MacOS και Linux και είναι ένα ελεύθερο, ανοικτού κώδικα, έργο. Στην **Εικόνα 19** φαίνονται οι δυνατότητες που παρέχει η NLTK. Πηγή

Εικόνα 19: Λογισμικό NLTK
https://data-flair.training/blogs/nltk-python-tutorial/

3.6 Keras API

Το Keras είναι ένα API, υψηλού επιπέδου, νευρωνικών δικτύων, γραμμένο σε Python και ικανό να τρέχει πάνω από βιβλιοθήκες όπως η TensorFlow, η CNTK και η Theano. Αναπτύχθηκε με έμφαση στον γρήγορο πειραματισμό. Πηγή

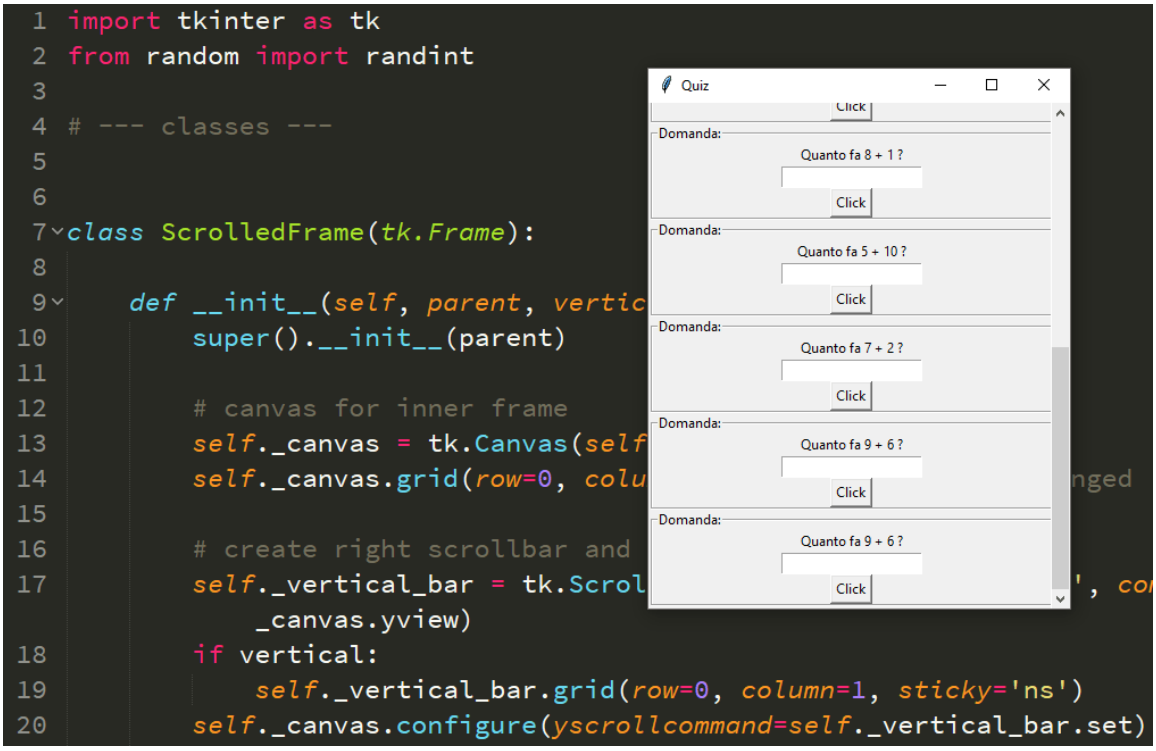
3.7 Βιβλιοθήκη Tensorflow

Το TensorFlow είναι μαθηματική βιβλιοθήκη την οποία ανέπτυξε η Google Brain, ομάδα τεχνητής νοημοσύνης της Google, αρχικά για εσωτερική χρήση. Σκοπός της δημιουργίας της ήταν η διευκόλυνση διαδικασιών όπως ο προγραμματισμός ροής δεδομένων, ενώ βρήκε χρήση και στην μηχανική μάθηση έχοντας ως παράδειγμα τα νευρωνικά δίκτυα. Η δημόσια διανομή του TensorFlow έγινε τον Νοέμβριο του 2015 κάτω από την άδεια ανοιχτού λογισμικού της Apache (Apache 2.0 Open Source License). Η **Εικόνα 20** εξηγεί τον συνδυασμό του Keras API και της βιβλιοθήκης TensorFlow. Πηγή

Εικόνα 20: Keras & Tensorflow
https://colab.research.google.com/github/GoogleCloudPlatform/cloudml-samples/blob/master/notebooks/tensorflow/getting-started-keras.ipynb

3.8 Βιβλιοθήκη Tkinter

Η Tkinter είναι η τυπική βιβλιοθήκη GUI για την Python. Η Python όταν συνδυάζεται με την Tkinter παρέχει έναν γρήγορο και εύκολο τρόπο δημιουργίας εφαρμογών GUI. Η Tkinter μπορεί να παρέχει μια ισχυρή αντικειμενοστραφή διεπαφή. Στην **Εικόνα 21** φαίνεται ένα αναδυόμενο παράθυρο, παράγωγο της βιβλιοθήκης TKinter το οποίο εμπεριέχει κάποιες λειτουργίες. Πηγή



Εικόνα 21: Βιβλιοθήκη TKinter
https://pythonprogramming.altervista.org/tkinter-an-example-of-an-app/?doing_wp_cron=1580488797.7344920635223388671875

4. Σχεδίαση και υλοποίηση της εφαρμογής

4.1 Γνωριμία με την εφαρμογή

Η παρακάτω εφαρμογή αφορά ένα εργαλείο που βασίζεται σε τεχνικές νευρωνικών δικτύων. Αφού ο χρήστης «τρέξει» την εφαρμογή και το νευρωνικό δίκτυο κάνει τις προβλέψεις του, θα του παρουσιαστεί ένα αναδυόμενο παράθυρο, όπου εκεί θα μπορεί να εισάγει ένα οποιοδήποτε token του συντακτικού της γλώσσας Java. Τη στιγμή που θα ξεκινήσει ο χρήστης να εισάγει το token της επιλογής του, το νευρωνικό δίκτυο θα προβλέψει και θα παρουσιάσει στον χρήστη τα εφτά επόμενα tokens που ακολουθούν το token που εισήγαγε ο χρήστης (σύμφωνα με τη πρόβλεψη του δικτύου). Για παράδειγμα εάν ο χρήστης εισάγει το token «public» το νευρωνικό δίκτυο αυτόματα θα του παρουσιάσει το αποτέλεσμα «public static void main(String[]» όπου ουσιαστικά είναι τα εφτά επικρατέστερα tokens να ακολουθούν το token «public».

Η είσοδος του δικτύου αποτελείται από ένα αρχείο κειμένου, με projects γραμμένα σε Java επιλεγμένα από τα αποθετήρια κώδικα GitHub και Kaggle, προσαρμοσμένο με τέτοιο τρόπο ώστε να αφαιρεθεί ο στατιστικός θόρυβος που εμπεριείχε. Επιπρόσθετα, χρησιμοποιήθηκε ένα δεύτερο σύνολο δεδομένων το οποίο αποτελείται από το ίδιο αρχείο κειμένου χωρίς όμως να αφαιρεθεί ο στατιστικός θόρυβος με στόχο να αποδειχθεί, όπως φαίνεται στη συνέχεια, η σημαντικότητα της προεπεξεργασίας των δεδομένων εισόδου. Η έξοδος του νευρωνικού δικτύου είναι ουσιαστικά οι προβλέψεις που παρέχει το δίκτυο στον χρήστη.

Εδώ θα πρέπει να αναφερθεί ότι, όπως σε όλα τα προβλήματα μηχανικής μάθησης, έτσι και σε αυτήν την περίπτωση, από το σύνολο των δεδομένων εισόδου θα συλλεχθεί ένα δείγμα που θα χρησιμοποιηθεί για την εκπαίδευση του νευρωνικού δικτύου.

4.2 Μέθοδος dataset_preparation

Όπως αναφέρθηκε στη προηγούμενη υποενότητα η είσοδος του δικτύου αποτελείται από ένα αρχείο κειμένου, με projects γραμμένα σε Java επιλεγμένα από τα αποθετήρια κώδικα GitHub και Kaggle, προσαρμοσμένο με τέτοιο τρόπο ώστε να αφαιρεθεί ο στατιστικός θόρυβος που εμπεριείχε.

Η μέθοδος dataset_preparation, η οποία φαίνεται στην **Εικόνα 22** μετατρέπει το αρχείο κειμένου σε μία μορφή την οποία μπορεί να κατανοήσει το νευρωνικό δίκτυο. Αρχικά παίρνει όλο το κείμενο και μετατρέπει τυχόν κεφαλαία γράμματα σε πεζά και χωρίζει το κείμενο σε προτάσεις. Στη συνέχεια με τη βοήθεια της εντολής tokenizer απομονώνει κάθε token ξεχωριστά. Έπειτα βρίσκει το πλήθος των tokens σε κάθε πρόταση και χωρίζει τα tokens ώστε να πάρει ένα δείγμα από αυτά (predictors, labels) και να εκπαιδευτεί επί αυτού του δείγματος.

Τέλος, επιστρέφει το συνολικό πλήθος των tokens (total_words) το πλήθος των tokens ανά πρόταση (max_sequence_len) και το δείγμα (predictors, label).

```
def dataset_preparation(data):
    # basic cleanup
    corpus = data.lower().split("\n")
    #print(corpus)

    # tokenization
    tokenizer.fit_on_texts(corpus)
    total_words = len(tokenizer.word_index) + 1

    # create input sequences using list of tokens
    input_sequences = []
    for line in corpus:
        token_list = tokenizer.texts_to_sequences([line])[0]
        for i in range(1, len(token_list)):
            n_gram_sequence = token_list[:i + 1]
            input_sequences.append(n_gram_sequence)

    # pad sequences
    max_sequence_len = max([len(x) for x in input_sequences])
    input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))

    # create predictors and label
    predictors, label = input_sequences[:, :-1], input_sequences[:, -1]
    label = ku.to_categorical(label, num_classes=total_words)

    return predictors, label, max_sequence_len, total_words
```

Εικόνα 22: Μέθοδος dataset_preparation

4.3 Μέθοδος generate_text

Η μέθοδος αυτή (**Εικόνα 23**) δημιουργεί κείμενο από τις προβλέψεις του νευρωνικού δικτύου και έχει οριστεί να προβλέπει τα εφτά επόμενα tokens που ακολουθούν το εισαγόμενο από το χρήστη token (input_txt). Χρησιμοποιεί την εντολή model.predict_classes ώστε να πάρει τις προβλέψεις του μοντέλου και τις ενσωματώνει στο εισαγόμενο token. Επιστρέφει το εισαγόμενο token μετά την ενσωμάτωση των προβλέψεων του μοντέλου.

```
347 #Prediction method
348 def generate_text(input_txt):
349
350
351
352     for _ in range(7):
353         token_list = tokenizer.texts_to_sequences([input_txt])[0]
354         token_list = pad_sequences([token_list], maxlen=max_sequence_len - 1)
355
356         predicted = model.predict_classes(token_list, verbose=0)
357
358         output_word = ""
359
360         for word, index in tokenizer.word_index.items():
361             if index == predicted:
362                 output_word = word
363                 break
364
365         input_txt += " " +output_word
366
367
368
369
370
371
372     return input_txt
```

Εικόνα 23: Μέθοδος generate_text

4.4 Κλάση combobox_autocomplete

Η κλάση αυτή δημιουργεί το αναδυόμενο παράθυρο στο οποίο θα εμφανίζονται οι προβλέψεις του νευρωνικού δικτύου. Η δήλωση και οι παράμετροι της κλάσης αυτής είναι η εξής:

Combobox_Autocomplete (root, list_of_items)

- root: Η παράμετρος που υποδηλώνει ότι θα γίνει χρήση της βιβλιοθήκης TKinter ώστε να δημιουργηθεί το αναδυόμενο παράθυρο.

- list_of_itmes: Τα πιθανά tokens που θα εισάγει ως αρχικά ο χρήστης.

4.5 Μέθοδος create_model

Αυτή είναι η μέθοδος (**Εικόνα 24**) που ουσιαστικά παρέχει προβλέψεις και παρουσιάζονται οι λειτουργίες τις αναλυτικά:

- Το μοντέλο είναι της μορφής Sequential η οποία μας δίνει την επιλογή να προσθέτουμε στρώματα νευρώνων στο νευρωνικό δίκτυο.

- Με την εντολή model.add προστίθενται τα στρώματα του νευρωνικού δικτύου. Το πλήθος των νευρώνων της εισόδου (input_dim) καθώς και του πρώτου στρώματος νευρώνων (output_dim), είναι ίσο με το πλήθος της μεταβλητής total_words που προέρχεται από τη μέθοδο dataset_preparation. Στη συνέχεια προστίθεται ένα στρώμα νευρώνων στο οποίο εφαρμόζεται η τεχνική GRU με πλήθος νευρώνων (units) ίσο με 150. Έπειτα προστίθεται ένα ακόμη στρώμα GRU με πλήθος νευρώνων ίσο με 50. Τέλος, το στρώμα εξόδου Dense έχει πλήθος όσο η μεταβλητή total_words.

- Στο μοντέλο εφαρμόζεται η συνάρτηση που περιγράφεται παραπάνω (softmax), και ο αλγόριθμος βελτιστοποίησης adam.

- Ως μετρική, το μοντέλο χρησιμοποιεί την categorical_crossentropy η οποία ειδικεύεται σε προβλήματα που επεξεργάζονται κείμενο. Η μετρική αυτή δείχνει το ποσοστό των tokens που δε μπόρεσαν να επεξεργαστούν σωστά από το μοντέλο.

- Το μοντέλο εκπαιδεύεται με τις μεταβλητές predictors και label.

- Το batch_size ορίζει τον αριθμό της παρτίδας των tokens που θα διαδοθούν μέσω του δικτύου από τον συνολικό αριθμό του δείγματος.
  Για παράδειγμα, υποθέτουμε ότι έχουμε 1050 tokens συνολικό αριθμό δείγματος και ρυθμίζουμε το batch_size ίσο με 100. Ο αλγόριθμος παίρνει τα πρώτα 100 tokens (από το 1ο έως το 100ο) από τον συνολικό αριθμό του δείγματος και εκπαιδεύει το δίκτυο. Στη συνέχεια, παίρνει τα επόμενα 100 δείγματα (από 101ο έως 200ο) και εκπαιδεύει πάλι το δίκτυο. Συνεχίζουμε να κάνουμε αυτήν τη διαδικασία μέχρι να διαδώσουμε όλες τις παρτίδες στο δίκτυο. Στη περίπτωση της εργασίας το batch_size είναι ίσο με 128.

- Ένα epoch ισοδυναμεί με μια πλήρη εκπαίδευση του νευρωνικού δικτύου, δηλαδή διάδοση όλων των δεδομένων εκπαίδευσης στο δίκτυο. Στη περίπτωση της εργασίας το epoch είναι ίσο με 3, ώστε να επαναλάβει τη διαδικασία εκπαίδευσής του το δίκτυο 3 φορές και να αυξήσει την ακρίβειά του.

- Στη συνέχεια με την εντολή model.summary() εκτυπώνονται τα αποτελέσματα του μοντέλου, τα οποία θα φανούν στη συνέχεια.

- Τέλος η μέθοδος επιστρέφει το μοντέλο.

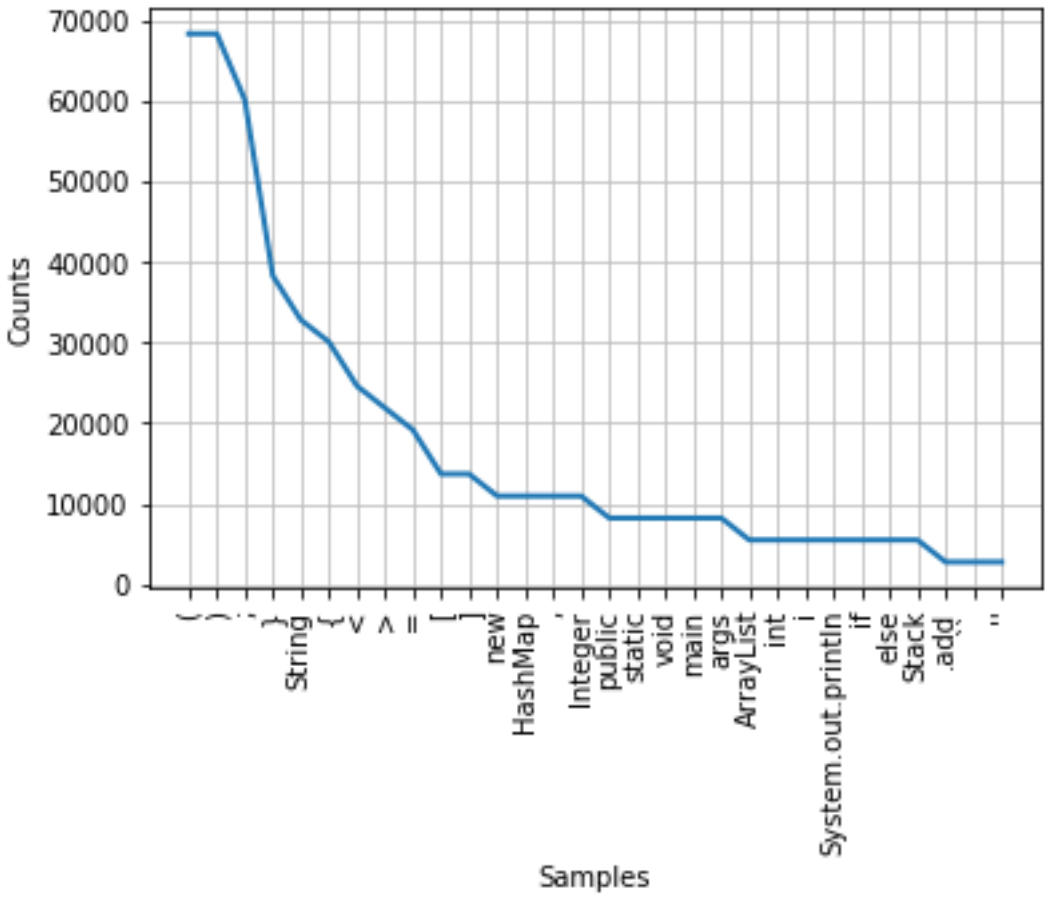
```
def create_model(predictors, label, max_sequence_len, total_words):



    #GRU
    model = Sequential()

    model.add(Embedding(input_dim=total_words,output_dim= total_words, input_length=max_sequence_len - 1))



    model.add(GRU(units=150, return_sequences=True ))

    model.add(GRU(units=50))


    model.add(Dense(units=total_words, activation="softmax"))
    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=['acc'])
    #earlystop = EarlyStopping(monitor='val_loss', min_delta=0, patience=5, verbose=1, mode='auto')
    model.fit(predictors, label, batch_size=128, epochs=3, verbose=1)
    print
    model.summary()
    return model
```

Εικόνα 24: Μέθοδος create_model

5. Παρουσίαση της εφαρμογής

5.1 Τα συχνότερα εμφανιζόμενα tokens στη μορφή διαγράμματος

Στον **Πίνακα 1** με τη χρήση της βιβλιοθήκης NLTK και συγκεκριμένα με την εντολή fdist.plot(30, cumulative = False) εμφανίζονται τα 30 συχνότερα tokens του συνόλου δεδομένων. Όπως ήταν αναμενόμενο σε ένα σύνολο δεδομένων το οποίο περιέχει συντακτικό της γλώσσας Java τα συχνότερα tokens είναι οι παρενθέσεις (). Ακολουθεί το ελληνικό ερωτηματικό ; οι αγκύλες, η λέξη String κ.λπ. Τα ευρήματα τα οποία παρουσιάζονται στους **Πίνακα 1** και **Πίνακα 2** έγιναν επί του πρώτου συνόλου δεδομένων, δηλαδή επί του αρχείου κειμένου, με projects γραμμένα σε Java επιλεγμένα από τα αποθετήρια κώδικα GitHub και Kaggle, προσαρμοσμένο με τέτοιο τρόπο ώστε να αφαιρεθεί ο στατιστικός θόρυβος που εμπεριείχε.



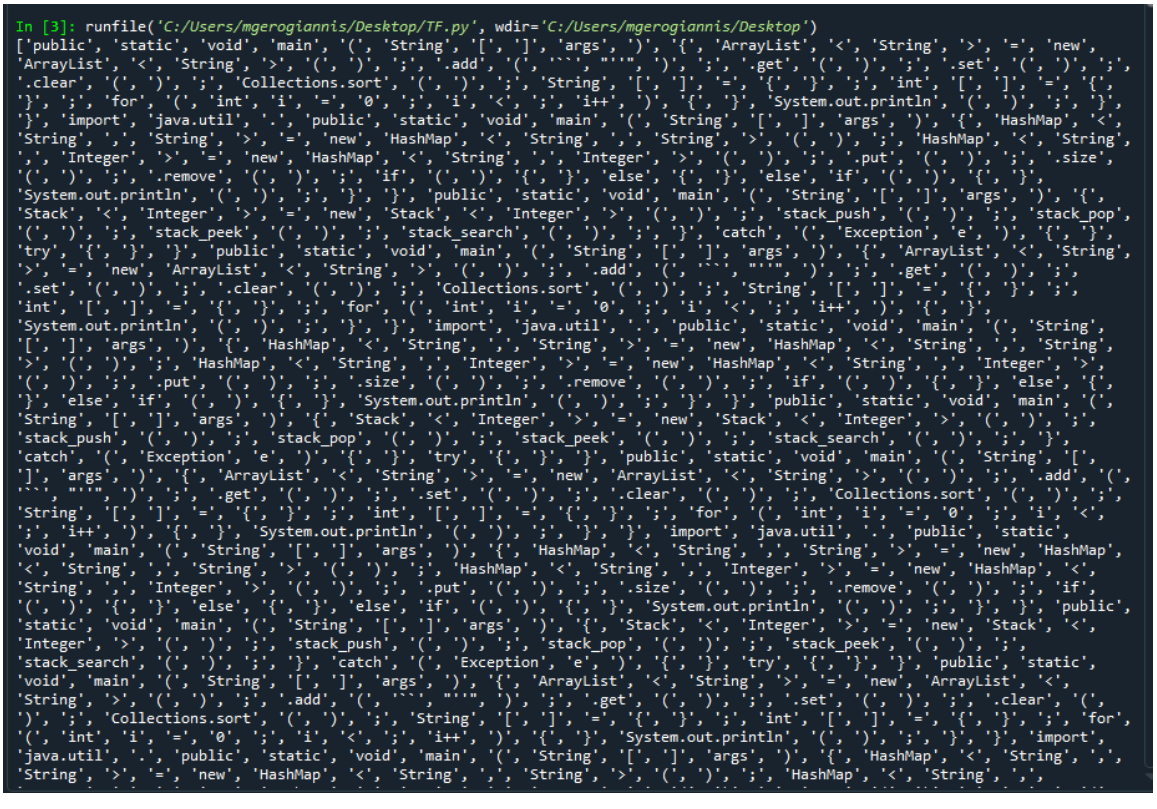
Πίνακας 1: Μέτρηση των συχνότερα εμφανιζόμενων token

5.2 Τα συχνότερα εμφανιζόμενα tokens στη μορφή πίνακα

Στον **Πίνακα 2** εμφανίζονται κατά σειρά ένα ένα όλα τα tokens του συνόλου δεδομένων. Το σύνολο δεδομένων ξεκινάει με τα tokens public, static, void, main, (, string κ.λπ. Η εμφάνιση των tokens γίνεται με τις εντολές:

- tokens = nltk.word_tokenize(data)

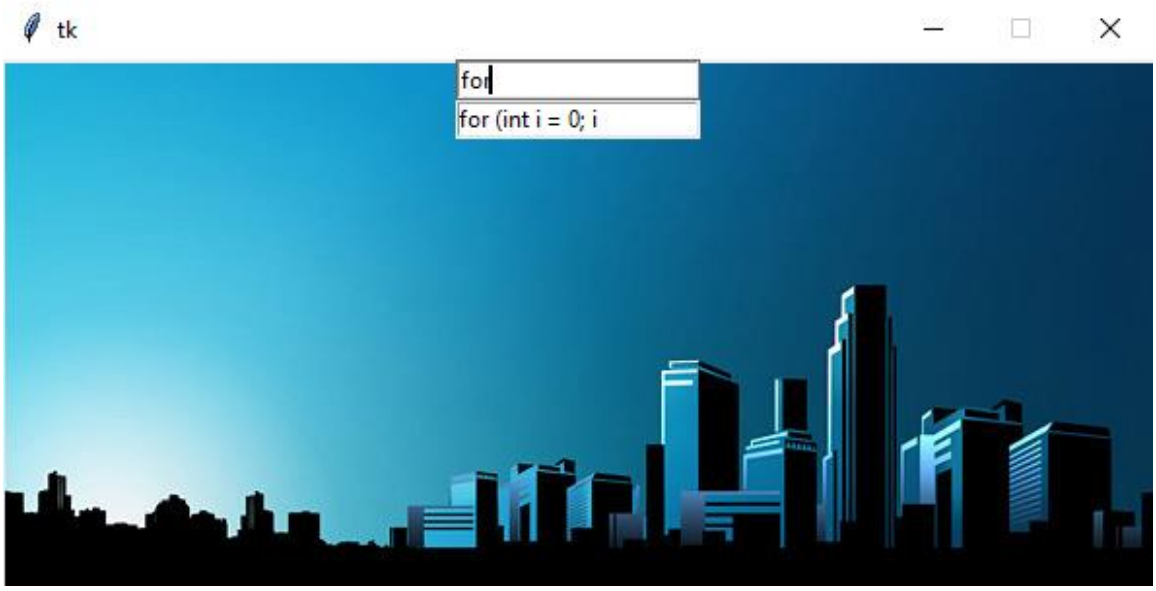
- print(tokens)



Πίνακας 2: Τα συχνότερα εμφανιζόμενα token

5.3 Πρόβλεψη νευρωνικού δικτύου με το token “for”

Στην παρακάτω **Εικόνα 25** βλέπουμε σε εφαρμογή το αναδυόμενο παράθυρο το οποίο θα εμφανίζεται στο χρήστη. Για την εμφάνιση του χρησιμοποιήθηκε η βιβλιοθήκη TKinter, ενώ όπως φαίνεται, η πρόβλεψη για τη λέξη for έγινε με σχετική ακρίβεια καθώς είναι πολύ πιθανό ο χρήστης μετά τη χρήση της for να θελήσει να συμπληρώσει for(int i=0; i, κ.λπ.

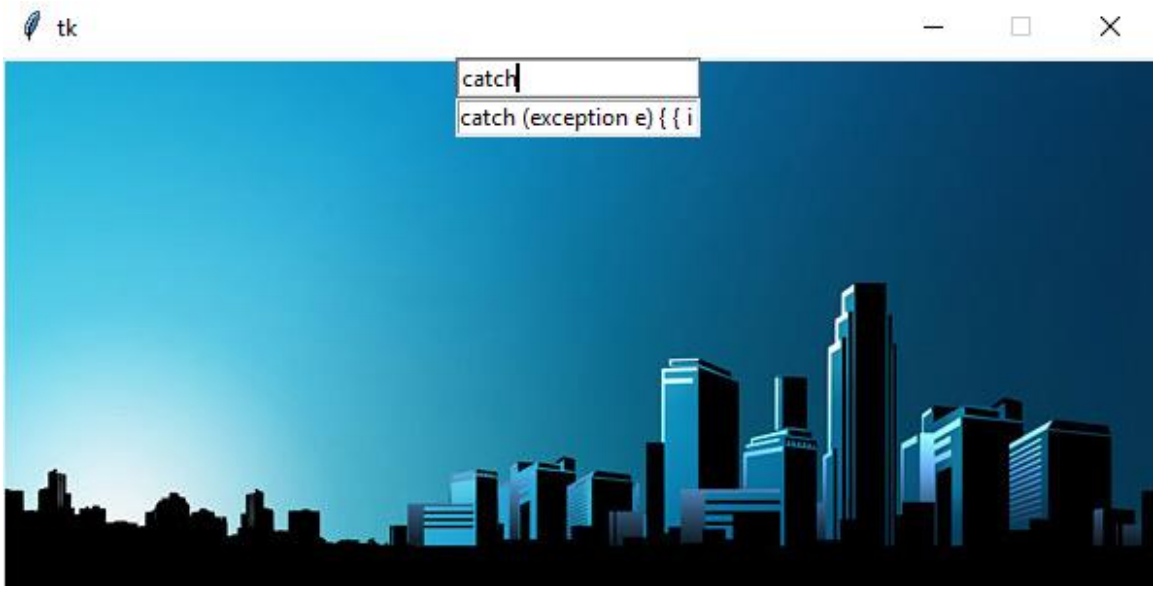


Εικόνα 25: Πρόβλεψη με τη “for”

5.4 Πρόβλεψη νευρωνικού δικτύου με το token “catch”

Εδώ στην **Εικόνα 26** βλέπουμε ένα παράδειγμα με τη λέξη catch. Το μοντέλο σε αυτή τη περίπτωση ενώ σωστά προέβλεψε τα 4 πρώτα tokens (exception e) στη συνέχεια συμπληρώνει κάτι το οποίο δεν έχει νόημα { { i.

Αυτό μπορεί να συμβαίνει για το λόγο ότι το σύνολο δεδομένων δεν παρέχει στο μοντέλο αρκετές εμφανίσεις του token “catch” έτσι ώστε το μοντέλο να είναι σε θέση να προβλέψει με περισσότερη ακρίβεια τη συνέχεια. Ένας δεύτερος λόγος είναι ότι το μοντέλο έχει περιθώρια βελτίωσης ώστε να παράγει ακόμη πιο ακριβή αποτελέσματα.



Εικόνα 26: Πρόβλεψη με τη “catch”

6. Αξιολόγηση της εφαρμογής

6.1 Ακρίβεια της εφαρμογής με το πρώτο σύνολο δεδομένων

Στον **Πίνακα 3** εμφανίζεται η ακρίβεια (accuracy) του μοντέλου η οποία είναι 89% και υπολογίζεται από την αφαίρεση της μετρικής categorical_crossentropy η οποία δείχνει το ποσοστό των tokens που δε μπόρεσαν να επεξεργαστούν σωστά από το μοντέλο.

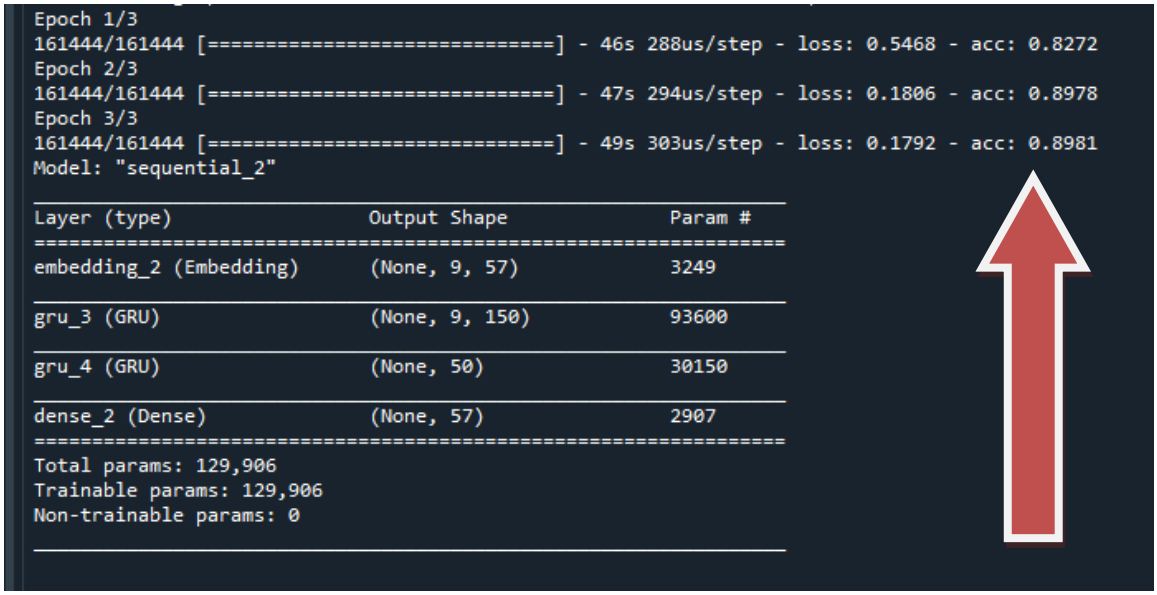
Η categorical_crossentropy μετρά την απόδοση ενός μοντέλου νευρωνικού δικτύου σε περιπτώσεις ταξινόμησης (classification), του οποίου η έξοδος είναι ουσιαστικά μια πιθανότητα μεταξύ 0 και 1. Η categorical_crossentropy αυξάνεται καθώς η προβλεπόμενη πιθανότητα ενός δείγματος αποκλίνει από την πραγματική τιμή.

Τα tokens τα οποία εισήχθησαν στο νευρωνικό δίκτυο ομαλοποιήθηκαν με τυχαία κατανομή δεικτών πιθανότητας σε κάθε ένα από αυτά. Για παράδειγμα έστω ότι στα tokens «for», «if», «import», έχουν ανατεθεί οι δείκτες πιθανότητας 0.55, 0.77, 0.43 αντίστοιχα κατά την εισαγωγή τους στο νευρωνικό δίκτυο. Εάν λοιπόν η πρόβλεψη του δείκτη πιθανότητας, που κάνει το νευρωνικό δίκτυο, για το token «for» είναι ίση με 0.03 τη στιγμή που η πραγματική τιμή του δείκτη είναι ίση με 0.55 έχουμε αυξημένη categorical_crossentropy. Σε αντίθετη περίπτωση η μετρική categorical_crossentropy μειώνεται. Στη περίπτωση της παρούσας εργασίας προσμετράται η συνολική categorical_crossentropy.

Επομένως έχουμε:

- 100% - 11%(categorical_crossentropy) = 89%(accuracy)

Η ακρίβεια του μοντέλου υπολογίστηκε επί του συνόλου δεδομένων που αποτελείται από ένα αρχείο κειμένου, με projects γραμμένα σε Java επιλεγμένα από τα αποθετήρια κώδικα GitHub και Kaggle, προσαρμοσμένο με τέτοιο τρόπο ώστε να αφαιρεθεί ο στατιστικός θόρυβος που εμπεριείχε.



Πίνακας 3: Ακρίβεια νευρωνικού δικτύου (1)

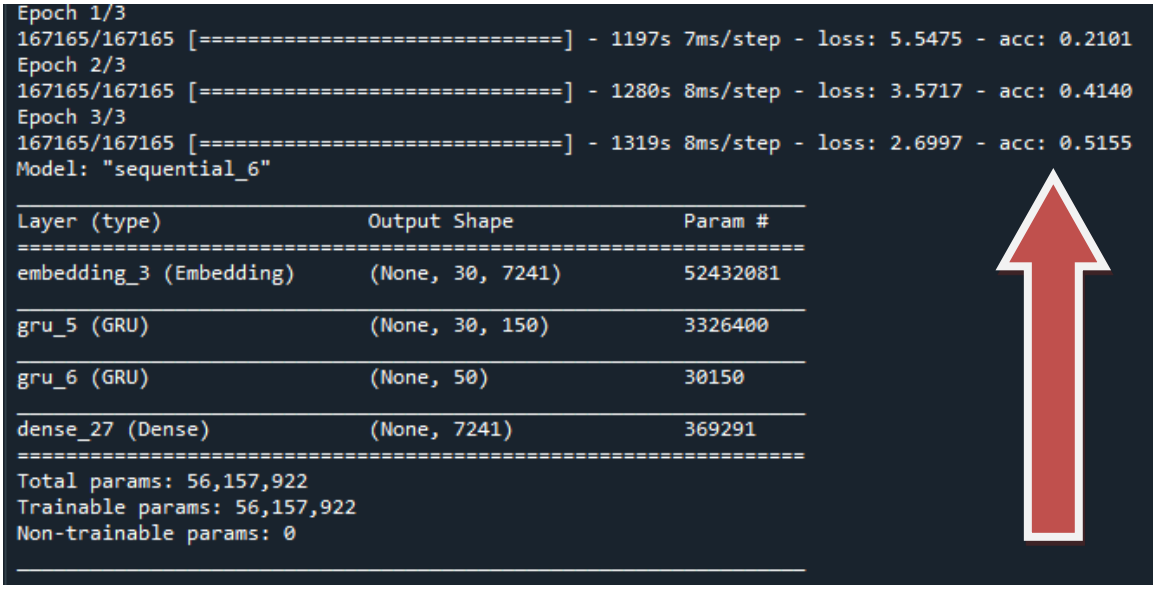
6.2 Ακρίβεια της εφαρμογής με το δεύτερο σύνολο δεδομένων

Στον **Πίνακα 4** βλέπουμε μια κατά πολύ μειωμένη ακρίβεια του μοντέλου (51%) στο σύνολο δεδομένων που αποτελείται από ένα αρχείο κειμένου με projects, γραμμένα σε Java, επιλεγμένα από τα αποθετήρια κώδικα GitHub και Kaggle αυτούσια (δίχως την αφαίρεση του στατιστικού θορύβου), ενώ οι υπόλοιπες συνθήκες του μοντέλου παρέμειναν ίδιες. Θα μπορούσαμε να αυξήσουμε την ακρίβεια του μοντέλου αυξάνοντας τον αριθμό των epochs, τον αριθμό των hidden layers, ή μειώνοντας το batch size, όμως εσκεμμένα έγινε με τις ίδιες συνθήκες για να αποδειχθεί η σημαντικότητα της προεπεξεργασίας του συνόλου δεδομένων.

Όπως αναφέρθηκε στην αρχή του κεφαλαίου 4 χρησιμοποιήθηκαν δύο σύνολα δεδομένων για να εκπαιδευτεί το μοντέλο. Στο πρώτο σύνολο η ακρίβεια του μοντέλου ήταν 89% ενώ στο δεύτερο 51% τη στιγμή που οι συνθήκες εκπαίδευσης του μοντέλου παρέμειναν οι ίδιες. Αυτό συμβαίνει διότι το δεύτερο σύνολο δεδομένων περιείχε αρκετό στατιστικό θόρυβο με αποτέλεσμα να δυσχεραίνει το μοντέλο. Σε αυτή τη περίπτωση ο στατιστικός θόρυβος ενδεχομένως να αφορούσε σχόλια τα οποία ήταν γραμμένα στα project για παράδειγμα:

```
/**
        * Hello world!
        *
        */
```

Αυτό το σχόλιο θα λέγαμε ότι αποτελεί θόρυβο για το μοντέλο διότι δε λαμβάνεται υπόψιν για την εξαγωγή αποτελέσματος πλην όμως υπάρχει στο σύνολο δεδομένων που εισέρχεται στο δίκτυο και το επηρεάζει.



Πίνακας 4: Ακρίβεια νευρωνικού δικτύου (2)

6.3 Σύγκριση ακρίβειας της παρούσας έρευνας με άλλες παρόμοιες

Έρευνα	Μέθοδος	Ακρίβεια μοντέλου
Παρούσα έρευνα	GRU	89%
Hellendoorn & Devanbu [2017]	n-gram	86.2%
Bhoopchand et al. [2016]	Sparse pointer networks	84.5%
Li et al. [2017]	LSTM	81%
Tran et al. [2016]	LSTM	69.2%

Πίνακας 5: Σύγκριση μεθόδων

Στον **Πίνακα 5** φαίνεται η υπεροχή, ως προς την ακρίβεια, που παρέχουν τα νευρωνικά δίκτυα GRU. Βέβαια, όπως προαναφέρθηκε σημαντικό ρόλο για την ακρίβεια του δικτύου παίζει το σύνολο δεδομένων που χρησιμοποιεί κανείς για να εκπαιδεύσει το μοντέλο νευρωνικού δικτύου που έχει δημιουργήσει.

6.4 Η σημαντικότητα της προεπεξεργασίας των δεδομένων

Η προεπεξεργασία αναφέρεται στους μετασχηματισμούς που εφαρμόζονται στα δεδομένα μας πριν την τροφοδοσία αυτών στο μοντέλο νευρωνικού δικτύου. Η προεπεξεργασία δεδομένων είναι μια τεχνική που χρησιμοποιείται για τη μετατροπή των πρωτογενών δεδομένων σε ένα καθαρό σύνολο δεδομένων. Με άλλα λόγια, κάθε φορά που συλλέγονται τα δεδομένα από διαφορετικές πηγές βρίσκονται σε μια ακατέργαστη μορφή που δεν καθιστά εφικτή την ανάλυση τους.

Για την επίτευξη καλύτερων αποτελεσμάτων του μοντέλου μηχανικής μάθησης, η μορφή των δεδομένων πρέπει να είναι η κατάλληλη. Ορισμένα μοντέλα μηχανικής μάθησης χρειάζονται πληροφορίες σε συγκεκριμένη μορφή, για παράδειγμα, ο αλγόριθμος Random Forest δεν υποστηρίζει μηδενικές τιμές, επομένως για να εκτελέσουμε έναν τέτοιο αλγόριθμο, οι τιμές null θα πρέπει να εξαλειφθούν.

6.5 Τεχνικές προεπεξεργασίας των δεδομένων

i. **Επανακλιμάκωση των δεδομένων**

   Όταν τα δεδομένα μας αποτελούνται από χαρακτηριστικά με διαφορετικές κλίμακες, καλό θα είναι για τους αλγόριθμους να μετατρέψουμε τα δεδομένα αυτά έχοντας ως γνώμονα την ίδια κλίμακα.

ii. **Δυαδικοποίηση των δεδομένων**

   Μπορούμε να μετατρέψουμε τα δεδομένα μας χρησιμοποιώντας μια δυαδική μορφή. Όλες οι τιμές πάνω από ένα προκαθορισμένο όριο θα παίρνουν την τιμή 1 και όλες οι ίσες ή χαμηλότερες θα παίρνουν την τιμή 0.

iii. **Τυποποίηση των δεδομένων**

   Η τυποποίηση είναι μια χρήσιμη τεχνική για τη μετατροπή χαρακτηριστικών με διαφορετικούς μέσους όρους και τυπικές αποκλίσεις, σε μια τυπική Gaussian κατανομή με μέσο όρο 0 και τυπική απόκλιση 1. Πηγή

7. Επίλογος

7.1 Σύνοψη και συμπεράσματα

Στις μέρες μας, οι εφαρμογές που συνδυάζουν τη μηχανική μάθηση και την επεξεργασία φυσικής γλώσσας, χρησιμοποιούνται αρκετά συχνά τόσο από άτομα όσο κι από οργανισμούς.

Καθώς λοιπόν αυξάνονται οι χρήσεις και οι δυνατότητες αυτών των εφαρμογών, είναι φυσικό επακόλουθο να εξεταστεί η χρήση τέτοιων εφαρμογών και σε πιο ειδικά πεδία. Η συγγραφή κώδικα σε σύγχρονα περιβάλλοντα προγραμματισμού αποτελεί ένα από αυτά τα ειδικά πεδία. Όπως συμπεραίνεται από την εργασία αυτή, τα νευρωνικά δίκτυα είναι αρκετά κοντά στο να παρέχουν ακριβείς προτάσεις συμπλήρωσης κώδικα, καθώς έχει αυξηθεί σημαντικά η ισχύς των υπολογιστών και είναι σε θέση να επεξεργαστούν μεγάλο όγκο δεδομένων στον οποίο θα βασίζουν τις προβλέψεις τους.

Το ιδανικό σενάριο θα ήταν, τη στιγμή που γράφει κώδικα ένας προγραμματιστής να συμπληρώνει αυτόματα το νευρωνικό δίκτυο το υπόλοιπο των όσων ήθελε να συμπληρώσει ο προγραμματιστής. Αυτό θα εξοικονομούσε πολύτιμο χρόνο κατά τη συγγραφή κώδικα καθώς επίσης θα βοηθούσε στην αποφυγή σφαλμάτων του προγραμματιστή κατά τη συγγραφή κώδικα.

Ένα δεύτερο σενάριο θα ήταν, το νευρωνικό δίκτυο να παρέχει στον προγραμματιστή προτάσεις σχετικά με τη συμπλήρωση του κώδικά του. Έτσι ακόμη κι ένας αρχάριος προγραμματιστής σε κάποια σημεία στα οποία ενδεχομένως δε γνωρίζει πως να συνεχίσει τον κώδικά του, με τις προτάσεις που θα του παρέχει το νευρωνικό δίκτυο να μπορέσει να συνεχίσει τη συγγραφή.

Βέβαια για να εφαρμοστούν τα παραπάνω θα χρειαστεί να εκπαιδευτεί και να εξεταστεί η αποτελεσματικότητα του νευρωνικού δικτύου αρκετές φορές και με ποικίλους τρόπους, ώστε να θεωρηθεί ικανό να χρησιμοποιηθεί ως βοήθημα κατά τη συγγραφή κώδικα.

7.2 Όρια και περιορισμοί της έρευνας

Ο κύριος περιορισμός των σύγχρονων τεχνολογιών επεξεργασίας φυσικής γλώσσας είναι η εξάρτησή τους από την τεράστια υπολογιστική ισχύ. Τα νευρωνικά δίκτυα για να λειτουργήσουν χρειάζονται μεγάλο αριθμό υπολογιστικών πόρων και η απόδοσή τους αυξάνεται καθώς αυξάνεται ο αριθμός αυτός.

Ένας ακόμη περιορισμός είναι η ακρίβεια ενός μοντέλου νευρωνικού δικτύου, ειδικότερα όταν το μοντέλο αυτό προσπαθεί να λύσει προβλήματα επεξεργασίας φυσικής γλώσσας. Ο λόγος είναι διότι θα πρέπει κανείς μόνος του να προσαρμόσει τις λέξεις σε αριθμούς ώστε να τους επεξεργαστεί το νευρωνικό δίκτυο και να παράγει αποτελέσματα.

7.3 Μελλοντικές Επεκτάσεις

Ένα σενάριο μελλοντικής επέκτασης στα πλαίσια της επεξεργασίας φυσικής γλώσσας για τους σκοπούς της συμπλήρωσης κώδικα σε περιβάλλοντα προγραμματισμού, θα ήταν η εφαρμογή ενός chatbot στο περιβάλλον. Το chatbot θα μπορούσε να χρησιμοποιηθεί ως βοηθός συμπλήρωσης κώδικα όπου κανείς θα μπορούσε να του θέτει ένα ερώτημα κι αυτό αυτόματα να εκτελεί μια αναζήτηση σε διάφορα αποθετήρια κώδικα ή σε πλατφόρμες οι οποίες παρέχουν απαντήσεις σε θέματα προγραμματισμού πχ StackOverflow. Το chatbot θα μπορούσε να κάνει την αναζήτηση στο διαδίκτυο για τον προγραμματιστή και να του παρέχει αξιόπιστες προτάσεις για την αντιμετώπιση ενός προβλήματος γλιτώνοντάς τον από χρόνο και κόπο.

Μια άλλη επέκταση, θα μπορούσε να είναι η αναγνώριση ομιλίας εντός του περιβάλλοντος προγραμματισμού. Με την αναγνώριση ομιλίας ο προγραμματιστής δε θα χρειάζεται πλέον να γράφει κώδικα αλλά να δίνει φωνητικά εντολές στο περιβάλλον προγραμματισμού που βρίσκεται κι εκείνο να τις εκτελεί.

7.4 Βιβλιογραφία

[1] [Allamanis et al., 2018] Mitliadis Allamanis, Earl T. Barr, Premkumar Devanbu and Charles Sutton. A Survey of Machine Learning for Big Code and Naturalness. arXiv preprint arXiv: 1709.06182, 2018.

[2] [Aye & Kaiser 2019] Gareth Ari Aye and Gail E. Kaiser. Sequence Model Design for Code Completion in the Modern IDE. arXiv preprint arXiv: 1711.09573, 2019.

[3] [Bahdanau et al., 2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translationby jointly learning to align and translate. arXiv preprint arXiv:1409.0473, 2014.

[4] [Bhoopchand et al., 2016] Avishka rBhoopchand, Tim Rocktaschel, Earl Barr, and Sebastian Riedel. Learning python code suggestion with a sparse pointer network. arXiv preprint arXiv:1611.08307, 2016.

[5] [Cheng et al., 2016] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory networks for machine reading. arXiv preprint arXiv:1601.06733, 2016.

[6] [Cho et al., 2014] Kyunghyun Cho, Bart van Merrienboer, ÇaglarGülçehre, FethiBo ugares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv preprint arXiv:1406.1078, 2014.

[7] [Hellendoorn & Devanbu 2017] Vincent J. Hellendoorn and Premkumar Devanbu. Are Deep Neural Networks the Best Choice for Modeling Source Code? pages 763-773, 2017.

[8] [Hindle et al., 2012] Abram Hindle, Earl T Barr, Zhendong Su, Mark Gabel, and Premkumar Devanbu. On the naturalness of software. In ICSE'12: Proc. of the International Conference on Software Engineering, pages 837–847,2012.

[9] [Hinton et al., 2012] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation offeature detectors. arXiv preprint arXiv: 1207.0580, 2012.

[10] [Karampatsis & Sutton 2019] Rafael-Michael Karampatsis and Charles Sutton. Maybe Deep Neural Networks are the Best Choice for Modeling Source Code arXiv preprint arXiv:1903.05734, 2019.

[11] [Li et al., 2017] Jian Li, Yue Wang, Michael R. Lyu, and Irwin King. Code Completion with Neural Attention and Pointer Networks. arXiv preprint arXiv:1711.09573, 2017.

[12] [Liu et al., 2016] Chang Liu, Xin Wang, Richard Shin,
Joseph E Gonzalez, and Dawn Song. Neural code completion. 2016.

[13] [Tran et al., 2016] Ke Tran, Arianna Bisazza, and ChristofMonz. Recurrent memory networks for language modeling. arXiv preprint arXiv:1601.01272, 2016.

[14] [Zaremba et al., 2014] Wojciech Zaremba, Ilya Sutskeverand Oriol Vinyals. Recurrent neural network regularization. arXiv preprint arXiv:1409.2329, 2014.

7.5 Διαδικτυακές πηγές

[1] Budhiraja, A, 2016, Dropout in (Deep) Machine learning, Budhiraja, A, viewed 05 February 2020, https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5

[2] Sterbak, T, 2019, Introduction To N-Gram Language Models, Sterbak, T, viewed 05 February 2020, https://www.depends-on-the-definition.com/introduction-n-gram-language-models/

[3] Ariely, D, 2013, Introduction to Machine Learning, Ariely, D, viewed 05 February 2020, https://www.fcodelabs.com/2018/12/13/Machine-Learning-Intro/

[4] Wikipedia, 2020, Μηχανική μάθηση, Wikipedia, viewed 05 February 2020, https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7

[5] Ali, S, & Ahuja, R, 2016, The Evolution and Core Concepts of Deep Learning & Neural Networks, Ali, S, & Ahuja, R, viewed 05 February 2020, https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/

[6] Maklin, C, 2019, LSTM Recurrent Neural Network Keras Example, Maklin ,C, viewed 05 February 2020, https://towardsdatascience.com/machine-learning-recurrent-neural-networks-and-long-short-term-memory-lstm-python-keras-example-86001ceaaebc

[7] Kostadinov, S, 2017, Understanding GRU Networks, Kostadinov, S, viewed 05 February 2020, https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[8] Kang, N, 2017, Multi-Layer Neural Networks with Sigmoid Function— Deep Learning for Rookies (2), Kang, N, viewed 05 February 2020, https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f

[9] Wikipedia, 2019, Softmax function, Wikipedia, viewed 05 February 2020, https://en.wikipedia.org/wiki/Softmax_function

[10] Brownlee, J, 2019, Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, Brownlee, J, viewed 05 February 2020, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[11] Cecil, S, 2019, NATURAL LANGUAGE PROCESSING (NLP) FOR ARTIFICIAL INTELLIGENCE, Cecil, S, viewed 05 February 2020, https://thinkpalm.com/blogs/natural-language-processing-nlp-artificial-intelligence/

[12] Wikipedia, 2019, Επεξεργασία φυσικής γλώσσας, Wikipedia, viewed 05 February 2020,https://el.wikipedia.org/wiki/%CE%95%CF%80%CE%B5%CE%BE%CE%B5%CF%81%CE%B3%CE%B1%CF%83%CE%AF%CE%B1_%CF%86%CF%85%CF%83%CE%B9%CE%BA%CE%AE%CF%82_%CE%B3%CE%BB%CF%8E%CF%83%CF%83%CE%B1%CF%82

[13] DATAFLAIR TEAM, 2018, NLTK Python Tutorial (Natural Language Toolkit), DATAFLAIR TEAM, viewed 05 February 2020, https://data-flair.training/blogs/nltk-python-tutorial

[14] Giovanni, G, 2018, Tkinter – An example of an App, Giovanni, G, viewed 05 February 2020, https://pythonprogramming.altervista.org/tkinter-an-example-of-an-app/

[15] https://www.python.org/, 2020, Tkinter — Python interface to Tcl/Tk, https://www.python.org/, viewed 05 February 2020, < https://docs.python.org/2/library/tkinter.html

[16] Abhishek, S, 2018, Data Preprocessing for Machine learning in Python, Abhishek, S, viewed 05 February 2020, https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/

[17] Wikipedia, 2020, Hyperbolic functions, Wikipedia, viewed 05 February 2020, https://en.wikipedia.org/wiki/Softmax_function

[18] Wikipedia, 2020, Νευρωνικό δίκτυο, Wikipedia, viewed 05 February 2020, https://el.wikipedia.org/wiki/%CE%9D%CE%B5%CF%85%CF%81%CF%89%CE%BD%CE%B9%CE%BA%CF%8C_%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

[19] Wikipedia, 2020, Python, Wikipedia, viewed 05 February 2020, https://el.wikipedia.org/wiki/Python

[20] https://www.nltk.org/, 2020, Natural Language Toolkit https://www.nltk.org/, viewed 05 February 2020 https://www.nltk.org/

[21] https://www.keras.io/, 2020, Keras: The Python Deep Learning library https://www.keras.io/, viewed 05 February 2020 https://keras.io/

[22] Wikipedia, 2018, Tensor Flow, Wikipedia, viewed 05 February 2020, https://el.wikipedia.org/wiki/Tensor_Flow

[23] GeeksforGeeks, 2018, Data Preprocessing for Machine learning in Python, GeeksforGeeks, viewed 05 February 2020, https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/

[24] Wikipedia, 2020, Hyperbolic functions, Wikipedia, viewed 05 February 2020, https://en.wikipedia.org/wiki/Hyperbolic_functions

[25] https://www.kite.com/, 2020, Keras: Code Faster in Python https://www.kite.com, viewed 05 February 2020 https://kite.com/