Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Μηχανικών Πληροφορικής & Τηλεπικοινωνιών

Διανυσματική Αναπαράσταση Λέξεων Για Επεξεργασία Φυσικής Γλώσσας

Ζιάμπας Ευάγγελος

Επιβλέπων Καθηγητής : Μαλαματή Λούτα

1 Ιουνίου 2021

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Διανυσματική Αναπαράσταση Λέξεων Για Επεξεργασία Φυσικής Γλώσσας" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Νικόλαου Δημόκα και της κα. Μαλαματή Λούτα αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή:

Ευχαριστίες

Με την ολοκλήρωση της παρούσας διπλωματικής εργασίας θα ήθελα να ευχαριστήσω τον Καθηγητή μου κ. Νικόλαο Δημόκα που μου ανέθεσε την εργασία καθώς και για όλες τις συμβουλές που μου έδωσε κατά την διάρκεια εκπόνησης της διπλωματικής εργασίας. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου για όλη την στήριξη κατά την διάρκεια των σπουδών μου.

Περίληψη

Η συγκεκριμένη εργασία αναφέρεται στην μελέτη της επεξεργασίας της φυσικής γλώσσας καθώς και στην υλοποίηση αρκετών τεχνικών της, χρησιμοποιώντας διάφορες μεθόδους αναπαράστασης λέξεων με την μορφή διανυσμάτων. Πραγματοποιείται ανάλυση και εκτέλεση του αλγορίθμου word2vec για το συγκεκριμένο σκοπό. Δημιουργείται και αξιολογείται ένα μοντέλο γλώσσας στα ελληνικά. Τέλος παρουσιάζεται η υλοποίηση μιας εφαρμογής διεπαφής χρήστη μέσω της οποίας παρουσιάζεται το μοντέλο που κατασκευάστηκε στα ελληνικά.

Λέξεις κλειδιά: Επικοινωνία ανθρώπου και υπολογιστή, νευρωνικά δίκτυα, word2vec,διανυσματική αναπαράσταση λέξεων, επεξεργασία φυσικής γλώσσας

Abstract

This study refers to the analysis of the processing of natural language as well as the implementation of several techniques, using various methods of vector word representation. The word2vec algorithm is presents and implemented for the specific purpose. A language model for the Greek language is create and evaluated. Finally, the creation of a web application is presented, through which the model we built in the Greek language is presented.

Keywords: Human-computer communication, neural networks, word2vec, vector word representation, natural language processing

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1 Εισαγωγή

1.1 Ορισμός του προβλήματος

Η επεξεργασία φυσικής γλώσσας τις τελευταίες δεκαετίες, συνιστά ένα επιστημονικό πεδίο με εξέχουσα θέση στον τρόπο που επικοινωνούν άνθρωπος και υπολογιστής. Ορισμένες από τις κυριότερες εφαρμογές της αφορούν την μορφολογική ανάλυση, την ανάλυση του συντακτικού, το τρόπο που χωρίζονται οι προτάσεις καθώς και τους όρους που μπορούνα να εξαχθούν. Στη σημερινή εποχή έχει καταστεί εφικτή η αντίληψη του ύφους ενός κειμένου είτε αυτό είναι θετικό είτε αρνητικό. Η μηχανική μετάφραση επιτρέπει στην μετάφραση κειμένου να γίνεται αυτόματα με αξιοσημείωτη επιτυχία. Καταληκτικά, με την αντίληψη κειμένων γραμμένα στην φυσική γλώσσα, την φωνητική αναγνώριση και την δημιουργία φυσικής γλώσσας, είναι εφικτό να αναπτυχθούν βοηθοί σε ψηφιακή μορφή με σημαντικές προοπτικές. Κατανοούμε λοιπόν την αναγκαιότητα της αποτελεσματικής επεξεργασίας της φυσικής γλώσσας.

1.2 Στόχος της έρευνας

Στόχος της παρούσας εργασίας αποτελεί η ανάδειξη διάφορων μεθόδων υλοποίησης και επεξεργασίας της φυσικής γλώσσας με τη βοήθεια της απεικόνισης λέξεων με τη μορφή διανυσμάτων. Η ερευνητική δραστηριότητα της παρούσας εργασίας εστιάζεται στην παραγωγή ενός μοντέλου γλώσσας με τη βοήθεια των νευρωνικών δικτύων. Τέλος, αξιολογείται το μοντέλο αυτό προκειμένου να διαπιστωθεί ο βαθμός αποτελεσματικότητας του.

1.3 Προσέγγιση

Στα πλαίσια της παρούσας εργασίας δημιουργείται ένα μοντέλο στην ελληνική γλώσσα. Η ανάπτυξη του μοντέλου επιτυγχάνεται μέσω της εκπαίδευσης νευρωνικών δικτύων που αναπαριστούν λέξεις με την μορφή διανυσμάτων. Προκειμένου να το καταφέρουμε αυτό χρησιμοποιούμε ως δεδομένα διάφορα κείμενα της Wikipedia. Καταληκτικά το μοντέλο αυτό αξιολογείται.

1.4 Διάρθρωση κειμένου

Στο πρώτο κεφάλαιο γίνεται μία εισαγωγή στην συγκεκριμένη μελέτη. Στο δεύτερο κεφάλαιο αναλύονται όλες οι μέθοδοι που χρησιμοποιούνται ώστε να επιτευχθεί ο στόχος της παρούσας μελέτης. Το τρίτο κεφάλαιο σχετίζεται με τις πηγές που περιέχουν τις πληροφορίες που χρησιμοποιούνται ως είσοδος για την εκπαίδευση του μοντέλου. Στο τέταρτο κεφάλαιο παρουσιάζεται ο τρόπος υλοποίησης του μοντέλου word2vec στην γλώσσα της python, ενώ στο πέμπτο κεφάλαιο αναλύεται η εφαρμογή διεπαφής χρήστη που κατασκευάστηκε για την ευκολότερη πρόσβαση του χρήστη στο μοντέλο. Στο έκτο κεφάλαιο πραγματοποιείται η αξιολόγηση του μοντέλου που δημιουργήθηκε και αναλύονται τα αποτελέσματά του. Τέλος, στο έβδομο κεφάλαιο παρουσιάζονται τα συμπεράσματα της συγκεκριμένης μελέτης.

2 Βιβλιογραφική Επισκόπηση

2.1 Ανάκτηση Πληροφορίας

Εξαιτίας της μεγάλης ανάπτυξης των πηγών πληροφορίας που βρίσκονται στο διαδίκτυο τόσο σε ποσότητα όσο και σε πολυπλοκότητα, τα συστήματα ανάκτησης πληροφορίας είναι δυνατό να χρησιμοποιηθούν σε μεγάλο εύρος διεργασιών [1]. Για παράδειγμα μπορούν να χρησιμοποιηθούν τόσο σε απλές διεργασίες που σχετίζονται με την αποθήκευση όσο και σε διεργασίες που διαχειρίζονται την μεταφορά σημαντικων πληροφοριών. Προκειμένου να αντιμετωπιστεί αυτή η πολυπλοκότητα έχουν γίνει σημαντικές προσπάθειες ανάπτυξης προσεγγίσεων τα τελευταία χρόνια.

Η ανάκτηση πληροφορίας σχετίζεται με την ανάκτηση πληροφοριών από βάσεις δεδομένων. Πιο συγκεκριμένα, η ανάκτηση πληροφορίας βοηθάει σε μεγάλο βαθμό το χρήστη όταν πρέπει να διαχειριστεί μεγάλο όγκο πληροφοριών σε μορφή κειμένου.

Μία από τις πιο σημαντικές εφαρμογές της ανάκτησης πληροφορίας είναι η ανάκτηση πληροφορίας κειμένου. Με την συνεχώς εξελισσόμενη ανάπτυξη των τελευταίων χρόνων το διαδίκτυο αποτελεί κυρίαρχο μέσο επικοινωνίας με δεδομένα που σχετίζονται με πολλούς τομείς. Τέτοιο παράδειγμα αποτελεί ο τομέας των επιχειρήσεων καθώς και ο ακαδημαϊκός τομέας. Ιδιαίτερα χρήσιμο στους τομείς αυτούς είναι να υπάρχει η δυνατότητα εντοπισμού του σωστού κειμένου μέσα σε ένα μεγάλο όγκο κειμένων. Αυτός είναι ένας από τους πιο σημαντικούς λόγους που η δημιουργία συστημάτων ανάκτησης πληροφορίας αποτελεί επιτακτική ανάγκη.

2.2 Εξόρυξη δεδομένων

Η διαδικασία μέσω της οποίας πραγματοποιείται εξαγωγή σημαντικών δεδομένων από ένα μεγαλύτερο σύνολο δεδομένων που δεν έχουν υποστεί κάποιου είδους επεξεργασία καλείται εξόρυξη δεδομένων [2]. Η εξόρυξη δεδομένων χρησιμοποιείται σε πολλούς τομείς. Ένα παράδειγμα χρήσης είναι σε επιχειρήσεις, όπου διευκολύνει την εύρεση περισσότερων πελατών μαθαίνοντας πληροφορίες για αυτούς και εξελίσσοντας διάφορες στρατηγικές προκειμένου να τους δελεάσουν. Με τον τρόπο αυτό καταφέρνουν να πετυχαίνουν τους σκοπούς τους. Κύρια χαρακτηριστικά της εξόρυξης δεδομένων είναι η αυτόματη πρόβλεψη προτύπων, η πρόβλεψη βάση πιθανών αποτελεσμάτων, η εστίαση σε μεγάλους όγκους δεδομένων κ.α.

2.3 Εξόρυξη κειμένου

Η εξόρυξη κειμένου πρόκειται για μια διαδικασία παρόμοια με την ανάλυση κειμένου στην οποία δημιουργούνται ποιοτικές πληροφορίες από το κείμενο [3]. Οι υψηλής ποιότητας πληροφορίες προέρχονται συνήθως από τη δημιουργία προτύπων και τάσεων από μέσα, όπως είναι η στατιστική εκμάθηση προτύπων. Η εξόρυξη κειμένου συνήθως περιλαμβάνει τη διαδικασία διαμόρφωσης του κειμένου εισαγωγής, τη παραγωγή προτύπων στα δομημένα δεδομένα, την αξιολόγηση και την ερμηνεία του αποτελέσματος. Οι κλασικές εργασίες εξόρυξης κειμένου είναι η κατηγοριοποίηση κειμένου, η ομαδοποίηση κειμένου, η εξαγωγή οντοτήτων, η εξαγωγή σημαντικών λέξεων κλειδιών, η ανάλυση συναισθημάτων, η συνοπτική παρουσίαση εγγράφων και η μοντελοποίηση σχέσεων οντοτήτων.

2.4 Μηχανική Μάθηση

Η μηχανική μάθηση συνιστά τομέα της τεχνητής νοημοσύνης ο οποίος επιτρέπει στους υπολογιστές να εκπαιδεύονται [4]. Σκοπός της μηχανικής μάθησης είναι να επιτρέπει στους ηλεκτρονικούς υπολογιστές να επιλύουν αποδοτικά, προβλήματα που δεν έχουν αντιμετωπίσει προγενέστερα έχοντας ως σημείο αναφοράς την εμπειρία που διαθέτουν από άλλα προβλήματα. Ένα ευρύ φάσμα προσεγγίσεων έχουν προταθεί για τη μηχανική μάθηση. Οι πιο αξιοσημείωτες εξ΄αυτών είναι τα δέντρων αποφάσεων, η βαθιά μάθηση (deep learning), η ομαδοποίηση, η ενισχυτική μάθηση, τα νευρωνικά δίκτυα, τα δίκτυα Bayes, η ενιχυτική μάθηση και ο επαγωγικός λογικός προγραμματισμός [5].

Αλγόριθμος μάθησης καλείται ένας αλγόριθμος που είναι ικανός να εκπαιδεύεται από τις πληροφορίες που έχει επεξεργαστεί. Τα στοιχεία που τον χαρακτηρίζουν είναι η διεργασία την οποία καλείται να υλοποιήσει, η αποτελεσματικότητα του για τη συγκεκριμένη διεργασία καθώς και η γνώση που αποκόμισε από αυτό. Η διεργασία δεν σχετίζεται με τη διαδικασία της μάθησης αλλά με το αποτέλεσμα που περιμένουμε από αυτή. Παραδείγματος χάρη, για έναν αλγόριθμο ο οποίος αποσκοπεί στην επίλυση μιας μαθηματικής συνάρτησης, η διεργασία είναι το αποτέλεσμα της συνάρτησης και όχι ο τρόπος επίλυσης αυτής. Με σκοπό την αξιολόγηση της απόδοσης του αλγορίθμου απαιτείται ο ορισμός κλιμάκων μέτρησης ανάλογα με το εκάστοτε πρόβλημα. Οι κλίμακες μέτρησης καθορίζουν το πόσο αποδοτικός και αποτελεσματικός είναι ο αλγόριθμος. Κατά κύριο λόγο η αξιολόγηση πραγματοποιείται σε στοιχεία που διαχειρίζεται για πρώτη φορά ο αλγόριθμος. Η γνώση που απέκτησε ο αλγόριθμος σχετίζεται με την προηγούμενη

υλοποίηση του σε ανάλογες συνθήκες.

Γενίκευση καλείται η ιδιότητα του αλγορίθμου να είναι αποδοτικός σε πληροφορίες που δεν έχει διαχειριστεί ξανά. Ως σφάλμα εκπαίδευσης ορίζεται η απόδοση του αλγορίθμου στα δεδομένα που διαχειρίζεται την παρούσα χρονική στιγμή. Η αποτελεσματικότητα του αλγορίθμου σε δεδομένα που δεν έχει συναντήσει ξανά δίνει το σφάλμα γενίκευσης. Στόχος του αλγορίθμου μάθησης είναι να μπορεί να επιλέγει τις καλύτερες παραμέτρους με σκοπό την κατά το δυνατόν ελαχιστοποίηση των σφαλμάτων.

Επιπροσθέτως, ο τομέας της μηχανικής μάθησης αναπτύσσει την εξελικτική μάθηση (Evolutionary Learning), η οποία είναι παρόμοια με τις διαδικασίες φυσικής αναπαραγωγής σε έμβια όντα. Η βασική χρήση της εξελεγκτικής μάθησης είναι σε προβλήματα βελτιστοποίησης. Οι αλγόριθμοι που χρησιμοποιούνται κατά βάση στην εξελικτική μάθηση είναι οι γενετικοί.

2.4.1 Είδη Μηχανική Μάθησης

Η μηχανική μάθηση δημιουργεί τρεις τρόπους μάθησης οι οποίοι μοιάζουν σε μεγάλο βαθμό με τους τρόπους της ανθρώπινης μάθησης.

Ο πρώτος τρόπος είναι η επιβλεπόμενη μάθηση, στην περίπτωση αυτή ο αλγόριθμος δημιουργεί μια συνάρτηση που παρουσιάζει δεδομένες εισόδους σε γνωστές επιθυμητές εξόδους. Η διαδικασία αυτή έχει ως στόχο η συνάρτηση να μπορεί να γενικευθεί και για εισόδους που η έξοδος δεν είναι γνωστή. Η κύρια χρήση αυτού του τρόπου μάθησης αφορά προβλήματα ταξινόμησης,πρόγνωσης και διερμηνείας.

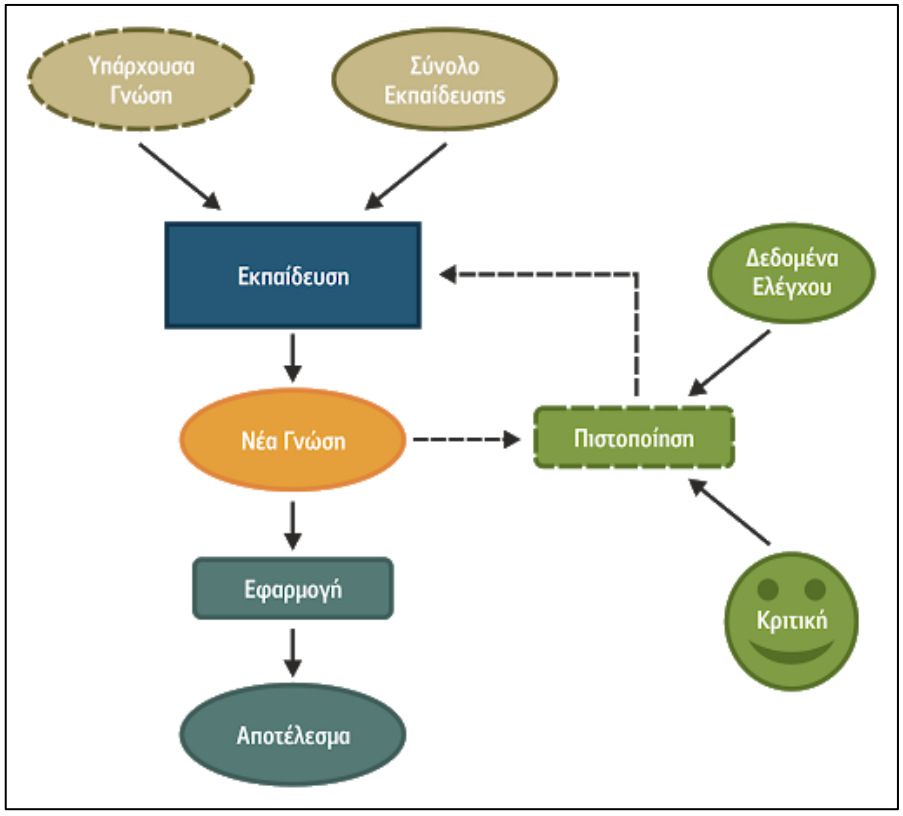
Ο δεύτερος τρόπος είναι η μη επιβλεπόμενη μάθηση, εδώ ο αλγόριθμος δημιουργεί ένα μοντέλο με την μορφή παρατηρήσεων για ένα συγκεκριμένο αριθμό εισόδων χωρίς να έχει γνώση των επιθυμητών εξόδων. Η μη επιβλεπόμενη μάθηση χρησιμοποιείται κατά κύριο λόγο σε προβλήματα ανάλυσης συσχετίσεων και ομαδοποίησης.

Τέλος, ο τρίτος και τελευταίος τρόπος είναι η ενισχυτική μάθηση, στην προκειμένη περίπτωση ο αλγόριθμος αλληλεπιδρώντας με το περιβάλλον μαθαίνει μια στρατηγική ενεργειών. Η ενισχυτική μάθηση χρησιμοποιείται κατα βάση σε προβλήματα σχεδιασμού, παραδείγματος χάρη ελέγχει την κίνηση ενός ρομπότ.

Για κάθε πρόβλημα που επιζητά λύση στον τομέα της μηχανικής μάθησης υπάρχει ένας κατάλληλος τρόπος μάθησης και για να υλοποιηθεί ο τρόπος αυτός υπάρχει και ένας κατάλληλος αλγόριθμος.

Υπάρχουν αρκετοί αλγόριθμοι μάθησης που διαφοροποιούνται μεταξύ τους ανάλογα με τα

δεδομένα που δέχονται σαν είσοδο, για παράδειγμα άλλοι αλγόριθμοι χρησιμοποιούν σαν είσοδο μόνο παρατηρήσεις ενώ άλλοι δίνουν μεγάλη βαρύτητα στην γνώση που υπάρχει ήδη. Έχοντας ως κριτήριο τον τρόπο μάθησης ανάλογα με το πόσο χρησιμοποιεί την γνώση που προϋπάρχει μπορούν να καταταγούν οι αλγόριθμοι σε δύο κατηγορίες. Η πρώτη κατηγορία αφορά τους πλούσιους σε χρήση της γνώσης και η δεύτερη τους φτωχούς σε χρήση της γνώσης. Την πρώτη κατηγορία απαρτίζουν οι αλγόριθμοι που σχετίζονται με την μάθηση βασισμένη σε επεξηγήσεις, την μάθηση βασισμένη σε περιπτώσεις και την μάθηση από επαγωγές με την βοήθεια παραδειγμάτων. Η δεύτερη κατηγορία αποτελείται από τους αλγόριθμους που σχετίζονται με την υλοποίηση νευρωνικών δικτύων.



Εικόνα 1: Φάσεις μηχανικής μάθησης [6]

Στην Εικόνα 1, απεικονίζεται ο τρόπος λειτουργίας των αλγορίθμων μηχανικής μάθησης. Η πιο σημαντική φάση του αλγορίθμου είναι η εκπαίδευση, στην οποία ο αλγόριθμος χρησιμοποιεί σαν είσοδο ένα σύνολο δεδομένων προκειμένου να επιτύχει το σκοπό του. Επίσης, ο κάθε αλγόριθμος μπορεί να χρησιμοποιεί την προϋπάρχουσα είτε στο εκατό τις εκατό είτε και καθόλου.

2.5 Τεχνητά Νευρωνικά δίκτυα

Τις περισσότερες φορές προκειμένου να αναφερθούμε στα τεχνητά νευρωνικά δίκτυα χρησιμοποιούμε τον όρο νευρωνικά δίκτυα, τα οποία είναι μια προσέγγιση της επιστήμης για να μοντελοποιήσουν τον τρόπο με τον οποίο λειτουργεί ο εγκέφαλος του ανθρώπου [7]. Ο εγκέφαλος του ανθρώπου είναι ιδιαίτερα σύνθετος και δουλεύει με εντελώς διαφορετικό τρόπο σε σχέση με αυτό των ηλεκτρονικών υπολογιστών.

Καθημερινά ο άνθρωπος χρησιμοποιεί διάφορες λειτουργίες, όπως για παράδειγμα η όραση, στις οποίες καταφέρνει και επεξεργάζεται μεγάλους όγκους δεδομένων, σε πολύ μεγαλύτερες ταχύτητες με αυτές που θα επιτύγχανε και ο πιο γρήγορος ηλεκτρονικός υπολογιστής.

Ορισμένες από τις ιδιότητες του ανθρώπινου εγκεφάλου περιέχουν την εκτέλεση πολλών υπολογισμών ταυτοχρόνως, τη δυνατότητα να μαθαίνει και να προσαρμόζεται στο εκάστοτε περιβάλλον καθώς και την επιρρέπεια σε λάθη, χωρίς όμως να σπαταλά μεγάλα ποσοστά ενέργειας. Τα χαρακτηριστικά αυτά αποτέλεσαν την αφορμή για την έναρξη ενεργειών μοντελοποίησης του ανθρώπινου εγκεφάλου, διότι οι ηλεκτρονικοί υπολογιστές μειονεκτούν σε πολλά από τα πεδία που προαναφέρθηκαν ενώ το σημαντικότερο πλεονέκτημα τους αφορά την κατηγορία των μαθηματικών υπολογισμών.

Προκειμένου να μοντελοποιηθεί αποτελεσματικά ο εγκέφαλος του ανθρώπου, πρέπει να γίνει απόλυτα κατανοητός η μέθοδος λειτουργίας ολόκληρου του νευρικού συστήματος του ανθρώπου. Αυτό απαρτίζεται από πολλούς νευρώνες, που έχουν διαφορετικό μέγεθος, οι νευρώνες αυτοί καλύπτουν ένα μεγάλο μέρος του ανθρώπινου σώματος [8].

Τα κυριότερα σημεία που απαρτίζουν ένα τυπικό νευρώνα είναι τα ακόλουθα:

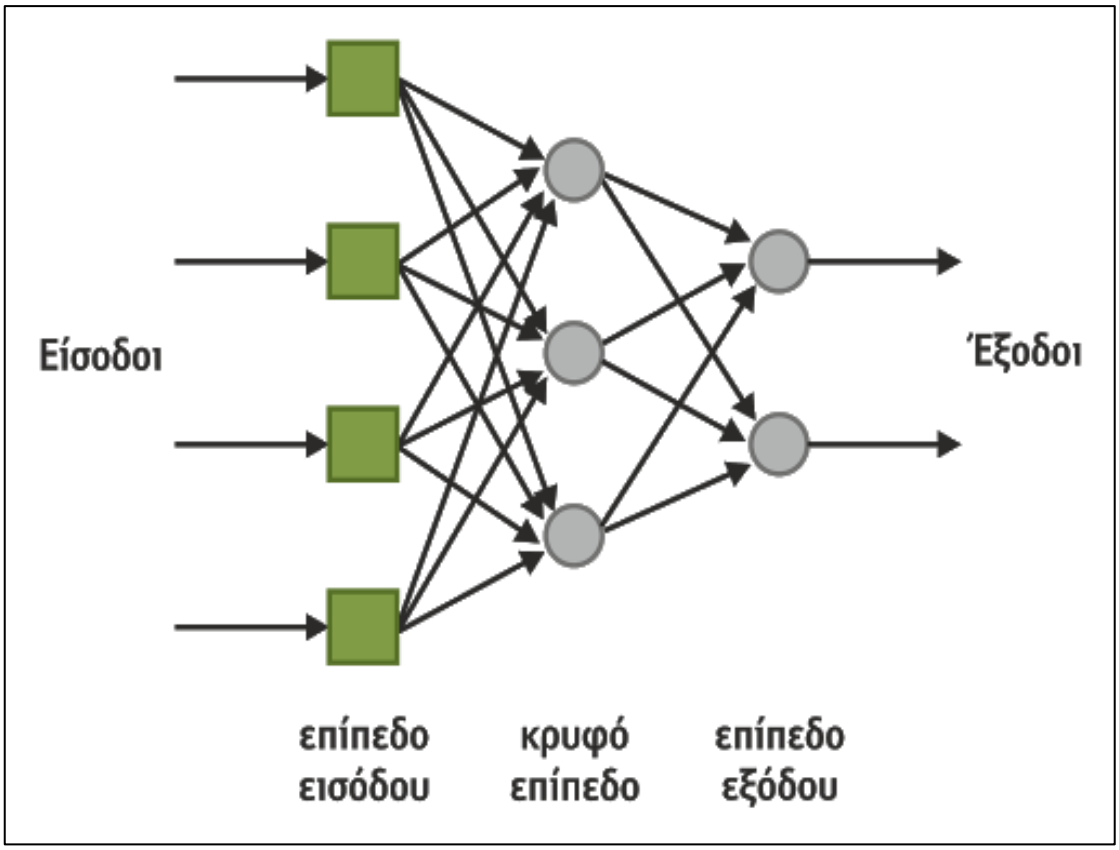
- Το κυτταρικό σώμα
- Οι δενδρίτες
- Ο άξονας του

Το κυτταρικό σώμα είναι αυτό το οποίο περιέχει τον πυρήνα του κυττάρου και εντοπίζεται μεταξύ του άξονα και των δενδριτών. Ως δενδρίτες ορίζεται το σημείο από το οποίο εισάγονται τα σήματα στον νευρώνα και πρόκειται για μικρές προεξοχές. Ο άξονας πρόκεται για μία λεπτή ίνα, με μεγάλο μήκος και η βασική λειτουργία του αφορά την έξοδο σημάτων από το νευρώνα. Ο τρόπος με τον οποίο επικοινωνούν οι νευρώνες μεταξύ τους γίνεται με τη μεταφορά σημάτων από τις σχέσεις που έχουν δημιουργήσει με άλλους νευρώνες. Η μεταφορά των σημάτων αυτών γίνεται από τον άξονα ενός νευρώνα στους δενδρίτες ενός άλλου.

Όπως απεικονίζεται και στην επόμενη εικόνα, η βασική δομή ενός νευρωνικού δικτύου αποτελείται από κόμβους οι οποίοι καλούνται συχνά ως νευρώνες συστήματος. Οι κόμβοι αυτοί λαμβάνουν ένα αριθμό εισόδων από διαφορετικές πηγές, υλοποιεί υπολογισμούς και έχει μία έξοδο. Οι βασικές κατηγορίες των κόμβων αυτών είναι οι:

- Νευρώνες εισόδου
- Νευρώνες εξόδου
- Υπολογιστικοί νευρώνες

Ως νευρώνες εισόδου ορίζονται αυτοί που χρησιμοποιεί το σύστημα για την είσοδο των δεδομένων και θα μπορούσαν να συγκριθούν με τους δενδρίτες που αναφέρθηκαν προηγουμένως. Αντιθέτως ως νευρώνες εξόδου του συστήματος ορίζονται αυτοί που απαρτίζουν το επίπεδο εξόδου αποτελεσμάτων από το δίκτυο. Υπολογιστικοί νευρώνες είναι αυτοί που είναι υπεύθυνοι για την ενεργοποίηση του νευρώνα και εξαρτώνται από τα δεδομένα εισόδου που δέχετια. Οι νευρώνες αυτοί αποτελούνται από μία συνάρτηση ενεργοποίησης και μπορούν να χαρακτηριστούν ως το κυρφό επίπεδο του δικτύου.



Εικόνα 2: Ένα τυπικό νευρωνικό δίκτυο [9]

Ένα βασικό χαρακτηριστικό των νευρωνικών δικτύων είναι η ικανότητά τους να εκπαιδεύονται. Προκειμένου να γίνει αυτό εφικτό αρκεί να αλλάξουν τα μεγέθη των τιμών και των βαρών του κάθε κόμβου του δικτύου. Υπάρχουν αρκετοί αλγόριθμοι που είναι ικανοί να κάνουν αυτές τις αλλαγές, έτσι ώστε το δίκτυο να καταφέρει το στόχο του με τον πιο αποτελεσματικό τρόπο. Δύο είναι οι βασικές κατηγορίες των αλγορίθμων εκπαίδευσης, η μάθηση με επίβλεψη και η μάθηση χωρίς επίβλεψη. Η πρώτη κατηγορία είναι δυνατό να υλοποιηθεί μόνο εάν υπάρχει κάποιος εξωτερικός εκπαιδευτής, διότι αυτός είναι υπεύθυνος να δώσει στο δίκτυο τις τιμές τόσο για την είσοδο όσο και για την έξοδο προκειμένου το δίκτυο να αλλάξει τα βάρη του κάθε κόμβου.

Αντιθέτως στην δεύτερη κατηγορία δεν χρειάζεται εξωτερικός εκπαιδευτής. Χωρίς να εισάγονται δεδομένα το δίκτυο είναι ικανό να κατανοεί μόνο του τον τρόπο που δομούνται τα δεδομένα εισόδου και να αλλάζει τις τιμές από τα βάρη του κάθε κόμβου ανάλογα.

Τα βαθιά νευρωνικά δίκτυα αποτελούν μια υποενότητα των νευρωνικών δικτύων, πιο συγκεκριμένα πρόκειται για νευρωνικά δίκτυα που έχουν πολλά κρυφά επίπεδα [10].

Αξίζει να σημειωθεί πως εάν αυξήσουμε τα κρυφά επίπεδα σε ένα νευρωνικό δίκτυο αυξάνεται και η αποτελεσματικότητα του σε διεργασίες με μεγάλη πολυπλοκότητα. Ωστόσο με το να αυξηθούν τα κρυφά επίπεδα υπάρχει ένα σημαντικό μειονέκτημα, αυτό είναι ότι αυξάνονται σε μεγάλο βαθμό οι υπολογιστικοί πόροι που απαιτούνται προκειμένου να εκπαιδευτεί ένα δίκτυο.

Τα νευρωνικά δίκτυα χρησιμοποιούνται συχνά σε διάφορους επιστημονικούς τομείς. Στον τομέα της ιατρικής χρησιμοποιούνται για να αναγνωρίζονται πρότυπα των ασθενών. Σε μία μηχανή αυτοκινήτου, η χρήση νευρωνικού δικτύου μπορεί να βελτιώσει την κατανάλωση καυσίμου. Ένα ακόμη παράδειγμα, αφορά το χρηματιστήριο και πιο συγκεκριμένα τον τρόπο που θα κινηθούν οι μετοχές του καθώς και άλλων οικονομικών θεμάτων.

2.6 Δημιουργία ενός τυπικού νευρωνικού δικτύου

Προκειμένου να υλοποιηθεί ένα τυπικό νευρωνικό δίκτυο στη γλώσσα της Python θα γίνει χρήση της βιβλιοθήκης tensorflow της Google [11]. Το tensorflow πρόκειται για μια συμβολική βιβλιοθήκη που χρησιμοποιείται συχνά για υπολογισμό αριθμητικών πράξεων αλλά βρίσκει εφαρμογές και στους τομείς της μηχανικής μάθησης και των νευρωνικών δικτύων. Το tensorflow δημιουργήθηκε χάρη στην ανάγκη της Google να δώσει εντολή σε ένα σύστημα για να προσομοιώσει την λειτουργία του ανθρώπινου εγκεφάλου κατά την διάρκεια της μάθησης. Χρησιμοποιώντας την απλή κανονική ανθρώπινη γλώσσα και μια

βαριά απλοποίηση, μια πλευρά του tensorflow θα μπορούσε να είναι η προηγμένη τεχνολογία φιλτραρίσματος [12]. Παραδείγματος χάρη, αν ένας χρήστης ανεβάσει μια φωτογραφία στο Google photos, το tensorflow θα συγκρίνει τις παραμέτρους της εικόνας στη βάση δεδομένων που διαθέτει και θα αποφασίσει αν απεικονίζεται άνθρωπος ή ζώο.

Θα υλοποιηθεί ένα μοντέλο γραμμικής παλινδρόμησης στην γλώσσα της Python προκειμένου να κατανοηθούν οι βασικές αρχές λειτουργίας του. Στην ακόλουθη Εικόνα παρουσιάζεται η υλοποίηση ενός απλού νευρωνικού δικτύου σε Python. Στην πρώτη γραμμή γίνεται εισαγωγή της βιβλιοθήκης σε μια συγκεκριμένη έκδοση και στην γραμμή 5 δημιουργείται μία συνεδρία προκειμένου να καλούμε μέσω αυτής την tensorflow. Έπειτα γίνεται ορισμός του γραμμικού μοντέλου το οποίο τοποθετείται στην μεταβλητή linear_model και ισούται με 'W*x+b'. Τα W και b είναι οι τιμές που θα καθορίσουν το μοντέλο. Ως x ορίζονται τα δεδομένα που θα δοθούν σαν είσοδος και ως y το αποτέλεσμα. Στόχος της εκπαίδευσης που θα γίνει, είναι να γίνει όσο το δυνατό πιο αποτελεσματική προσέγγιση των W και b προκειμένου το νευρωνικό δίκτυο που θα υλοποιηθεί να αναπαριστά με τον καλύτερο τρόπο την παλινδρόμηση.

```
1   import tensorflow.compat.v1 as tf
2   tf.disable_v2_behavior()
3
4   #sess = tf.Session()
5   sess = tf.compat.v1.Session()
6
7   W = tf.Variable([0], dtype=tf.float32)
8   b = tf.Variable([0], dtype=tf.float32)
9   x = tf.placeholder(tf.float32)
10
11  #Define the linear model
12  linear_model = W * x + b
```

Εικόνα 3: Μοντέλο γραμμικής παλινδρόμησης 1

Στις γραμμές 7 και 8 ορίζονται οι μεταβλητές οι οποίες αναλογούν στα μεγέθη των βαρών των ενδιάμεσων κόμβων. Στην επόμενη γραμμή ορίζεται μια μεταβλητή στην οποία μπαίνουν τα δεδομένα εισόδου. Αξίζει να σημειωθεί ότι στο tensorflow οι τιμές που έχουν τα βάρη οι οποίες θα υπολογιστούν στην εκπαίδευση αναφέρονται ως Variable και placeholder είναι οι θέσεις που τοποθετούνται τα εξωτερικά δεδομένα. Καταληκτικά στην γραμμή 12 τοποθετείται στην μεταβλητή linear_model το μοντέλο παλινδρόμησης.

```
14    init = tf.global_variables_initializer()
15    sess.run(init)
16
17    #y is the deisred outcome
18    y = tf.placeholder(tf.float32)
19
20    #Sum the squared the deltas from the model and this is the loss function
21    squared_deltas = tf.square(linear_model - y)
22    loss = tf.reduce_sum(squared_deltas)
23
24    #Use the GradientDescentOptimizer to minimize the loss function
25    optimizer = tf.train.GradientDescentOptimizer(0.01)
26    train = optimizer.minimize(loss)
```

Εικόνα 4: Μοντέλο γραμμικής παλινδρόμησης 2

Έπειτα, στις γραμμές 14 και 15 παίρνουν αρχικές τιμές οι μεταβλητές από το tensorflow, στη γραμμή 18 προσδιορίζεται το placeholder y, το οποίο σχετίζεται με την προβλεπόμενη έξοδο της παλινδρόμησης. Στην μεταβλητή της γραμμής 21 τοποθετούνται τα τετράγωνα της διαφοράς των y που έχουν υπολογισθεί με τα αναμενόμενα. Έπειτα υπολογίζεται το άθροισμα των τετραγώνων αυτών και τοποθετείται σε μία μεταβλητή με την μορφή της συνάρτησης απώλειας. Στόχος της εκπαίδευσης είναι να ελαχιστοποιηθεί η συνάρτηση απώλειας.

Στη γραμμή 25 ορίζεται η συνάρτηση βελτιστοποίησης, δηλαδή ο τρόπος με τον οποίο θα αλλάζουν οι μεταβλητές της παλινδρόμησης. Τέλος στην γραμμή 26, στην μεταβλητή train τοποθετείται το βήμα εκπαίδευσης για κάθε επανάληψη.

Επεξηγηματικά, κατά την διάρκεια της εκπαίδευσης οι τιμές των μεταβλητών W και b θα αλλάζουν τιμές με τέτοιο τρόπο ώστε αν προσθέσουμε τα τετράγωνα της διαφοράς των αναμενόμενων τιμών από αυτά των πραγματικών να περιορίζεται συγκριτικά με το προηγούμενο.

```
28    x_values = [1, 2, 3, 4, 5]
29    y_values = [7, 12, 17, 22, 27]
30
31    #Iterate and train the model
32    for i in range(10000):
33        sess.run(train, {x: x_values, y: y_values})
34
35    #Print the results
36    trained_W = sess.run(W) [0]
37    trained_b = sess.run(b) [0]
38    print("Output ->Trained function: ", trained_W, "x+", trained_b)
```

Εικόνα 5: Μοντέλο γραμμικής παλινδρόμησης 3

Στις γραμμές 28 και 29, της Εικόνας 5 ορίζονται οι αρχικές τιμές εισόδου για τα x και y. Έπειτα, στην γραμμή 32 σε ένα βρόγχο 1000 επαναλήψεων διενεργείται η εκπαίδευση του μοντέλου. Στις γραμμές 36 και 37 δίνεται πρόσβαση στις μεταβλητές W και b οι οποίες έχουν λάβει τιμές μετά το πέρας της εκπαίδευσης. Τέλος στην γραμμή 38 τυπώνεται το αποτέλεσμα της παλινδρόμησης και στην προκειμένη περίπτωση οι τιμές των μεταβλητών W και b.

Στην Εικόνα 6, απεικονίζεται μετά το πέρας της εκπαίδευσης. Μετά την εκπαίδευση το W ισούται με 5.0001 και η μεταβλητή b ισούται με 1.999969, αυτό μας δείχνει ότι δεν υπάρχει απόλυτη ακρίβεια στο αποτέλεσμα και αυτό οφείλεται στην συνάρτηση βελτιστοποίησης. Συμπερασματικά, είναι εμφανές ότι το αποτέλεσμα υπολογίζεται κατά προσέγγιση.

```
Output ->Trained function:  5.000001 x+ 1.9999969
```

Εικόνα 6: *Μοντέλο γραμμικής παλινδρόμησης* 4

2.7 Επεξεργασία φυσικής γλώσσας

Η μέθοδος με την οποία ένας ηλεκτρονικός υπολογιστής καταφέρνει να επεξεργάζεται κείμενο σε φυσική γλώσσα, ονομάζεται επεξεργασία φυσικής γλώσσας [13]. Αυτή μπορεί να σχετίζεται τόσο με την αντίληψη του κειμένου όσο και με την δημιουργία αυτού. Ιδιαίτερο ενδιαφέρον έχουν τα χαρακτηριστικά και οι ιδιαιτερότητες του φυσικού λόγου που δυσκολεύουν αυτή τη διεργασία.

Η επεξεργασία φυσικής γλώσσας προσφέρει πολλά πλεονεκτήματα και βρίσκει εφαρμογή σε διάφορους τομείς. Αρχικά, η κύρια χρήση της αφορά τον τομέα της επικοινωνίας ανθρώπου και υπολογιστή. Σε αυτό το τομέα δίνεται η δυνατότητα στους χρήστες να επικοινωνούν χρησιμοποιώντας της φυσική τους γλώσσα και όχι κάποια γλώσσα προγραμματισμού. Ένας ακόμα τομέας που διευκολύνετε με την χρήση της επεξεργασίας φυσικής γλώσσας είναι η διαχείριση της πληροφορίας. Στον τομέα αυτό, η επεξεργασία φυσικής γλώσσας προσφέρει να την δυνατότητα οι πληροφορίες να διαχειρίζονται και να επεξεργάζονται αυτόματα ανάλογα με την ερμηνεία τους. Παραδείγματος χάρη, αν υπήρχε ένα σύστημα που θα μπορούσε να αντιληφθεί τη σημασία ενός εγγράφου, θα μπορούσε να το ομαδοποιήσει μαζί με άλλα έγγραφα παρόμοιας σημασίας. Επίσης, ο τομέας της διαχείρισης πληροφορίας επωφελείται της χρήσης της επεξεργασίας φυσικής γλώσσας. Κατά την αναζήτηση της πληροφορίας με την χρήση της τεχνητής γλώσσας οι χρήστες

είναι απαραίτητο να γνωρίζουν το πως δομείται η βάση δεδομένων, ενώ στη φυσική γλώσσα η αναζήτηση γίνεται αρκετά πιο εύκολη διότι μπορεί να περιοριστεί απλά ως πρός το περιεχόμενο αναζήτησης.

Μία από τις πιο σημαντικές ιδιαιτερότητες της φυσικής γλώσσας είναι η μέθοδος με την οποία γίνεται η μετάδοση της πληροφορίας. Στους ηλεκτρονικούς υπολογιστές, η πληροφορία απαιτείται να έχει σωστή δομή και να υφίσταται σε όλο το φάσμα της προκειμένου να γίνει κατανοητή. Για παράδειγμα, σε μία γλώσσα προγραμματισμού μέχρι και ένα απλό σύμβολο να λείπει μπορεί να οδηγήσει σε σφάλμα και γίνει ανέφικτο στον υπολογιστή να το κατανοήσει. Αντιθέτως, στη φυσική γλώσσα δεν είναι απαραίτητη η υπάρξη μιας συγκεκριμένης δομής του λόγου και ένα μεγάλο κομμάτι της πληροφορίας που επηρεάζει την επικοινωνία παραλείπεται.

Αυτό συμβαίνει διότι οι άνθρωποι με την πάροδο των χρόνων αποκτούν εμπειρία. Μάλιστα, αρκετές φορές ορισμένες λέξεις χρησιμοποιούνται με διαφορετικούς τρόπους από τους ανθρώπους και αυτό προκαλεί πολλαπλές σημασίες, οι οποίες όμως γίνονται κατανοητές χάρη στην εμπειρία που προαναφέρθηκε.

Προκειμένου να καταστεί εφικτή η επεξεργασία της φυσικής γλώσσας, βασική προϋπόθεση είναι να κωδικοποιηθούν οι λέξεις. Αυτό πρέπει να γίνει με τέτοιο τρόπο ώστε να μπορέσουν να αναδειχθούν οι ομοιότητες και οι διαφορές της κάθε λέξης. Η αποτελεσματικότερη αντιμετώπιση είναι να οριστούν διανύσματα για την κάθε λέξη.

2.7.1 Απεικόνιση λέξεων με μορφή διανυσμάτων

Η σημαντικότερη παράμετρος για την αποδοτικότερη επεξεργασία φυσικού λόγου, είναι η αντίληψη του νοήματος των λέξεων. Είναι σχεδόν ανέφικτο να μοντελοποιηθεί πλήρως μια γλώσσα από ένα υπολογιστή έτσι όπως αυτή γίνεται αντιληπτή από ένα άνθρωπο. Ωστόσο, είναι εφικτή η μοντελοποίηση πολλών πτυχών της. Υπάρχουν αρκετές μέθοδοι για την αναπαράσταση μιας γλώσσας σε ένα διανυσματικό χώρο με πολλές διαστάσεις [14]. Η κάθε λέξη, αναπαριστάται από ένα διάνυσμα που αποτελείται από πραγματικούς αριθμούς. Ο τρόπος με τον οποίο αντιστοιχίζονται τα διανύσματα με τις λέξεις, είναι τέτοιος ώστε λέξεις με παρόμοια σημασία να έχουν κοντινές διανυσματικές τιμές, στην αντίστοιχη διάσταση.

Με τη βοήθεια των νευρωνικών δικτύων μπορούν να βελτιστοποιηθούν οι διανυσματικές τιμές της κάθε λέξης, προκειμένου να γίνεται πιο αποδοτική απεικόνιση της κάθε λέξης

συγκριτικά με τις άλλες στον διανυσματικό χώρο.

Επίσης, υπάρχουν μέθοδοι που δημιουργούν διανύσματα υπολογίζοντας το πόσο φορές εμφανίστηκε μια λέξη στο κείμενο [15]. Προκειμένου να υλοποιηθεί αυτή η διεργασία, βασική προϋπόθεση είναι να υπάρχουν κείμενα ποιοτικά δομημένα.

Η αποδοτικότητα του μοντέλου καθορίζεται σε μεγάλο βαθμό από τον όγκο των δεδομένων τα οποία θα ληφθούν ως είσοδος. Προκειμένου να είναι ικανοποιητικά τα αποτελέσματα, βασική προϋπόθεση είναι να υπάρχουν κείμενα με μεγάλο όγκο λέξεων. Με σκοπό την καλύτερη απόδοση του μοντέλου, είναι επιτακτική ανάγκη να γίνει εξ΄αρχής μια βελτιστοποίηση του κειμένου.

Για παράδειγμα, χρήσιμο είναι να αφαιρεθούν λέξεις που δεν εμφανίζονται αρκετά συχνά, αυτό γίνεται για να αποφευχθεί η λανθασμένη αναπαράσταση αυτών των λέξεων στο μοντέλο και η αλλοίωση των διανυσμάτων των κοντινών τους λέξεων. Επιπρόσθετα, καλό θα ήταν να αφαιρεθούν ακόμα και τα σύμβολα, τα σημεία στίξης, οι αριθμοί, τα άρθρα, οι αντωνυμίες και λέξεις που η χρήση τους συμβάλλει μόνο στο συντακτικό του κειμένου. Έπειτα από όλες αυτές τις ενέργειες, μπορεί να θεωρηθεί ότι έχει γίνει μια καλή βελτιστοποίηση του κειμένου και έχει μείνει μόνο το ουσιαστικό κομμάτι του και μπορεί να αρχίσει η παραγωγή των διανυσμάτων.

2.7.2 Μοντέλα συχνότητων και πρόβλεψης

Μια σχετικά απλή μέθοδος για να δημιουργηθούν διανύσματα από λέξεις είναι να μετράται το πόσο συχνά εμφανίζεται η κάθε λέξη στο κείμενο. Πιο συγκεκριμένα, έχοντας κάποια διαφορετικά κείμενα μπορεί να κατασκευαστεί για κάθε μία λέξη ένα διάνυσμα, έτσι ώστε κάθε αριθμός που θα περιέχει το διάνυσμα να αναπαριστά το σύνολο των εμφανίσεων της λέξης στο κάθε κείμενο. Παραδείγματος χάρη, εάν υπάρχουν πέντε κείμενα με τη λέξη αυτοκίνητο να εμφανίζεται έξι φορές στο πρώτο κείμενο, μία φορά στο δεύτερο, τρείς φορές στο τρίτο, ούτε μία φορά στο τέταρτο και δύο φορές στο πέμπτο, το διάνυσμα της λέξης θα ισούται με [6, 1, 3, 0, 2] και η κάθε λέξη θα αναπαρασταθεί με ένα διάνυσμα πέντε διαστάσεων. Όπως είναι προφανές ένα μοντέλο αυτού του είδους στηρίζεται εξ' ολοκλήρου στην μέτρηση των λέξεων στο κείμενο.

Ένα γνωστό μοντέλο παρόμοιας λειτουργίας με το προηγούμενο είναι το Term Frequency–Inverse Document Frequency. Το μοντέλο αυτό λειτουργεί με γνώμονα το πόσο συχνά εμφανίζεται μία λέξη στο κάθε κείμενο, αλλά ταυτόχρονα υπολογίζει και το πόσο συχνά εμφανίζεται σε άλλα κείμενα. Παραδείγματος χάρη, όταν μία λέξη εντοπίζεται αρκετές

φορές στο σύνολο των κειμένων, είναι πολύ πιθανό να πρόκειται για κάποιο σύνδεσμο ή άρθρο που εμφανίζεται παντού και έχει ως αποτέλεσμα να μην είναι εφικτή η δημιουργία διανύσματος για την συγκεκριμένη λέξη.

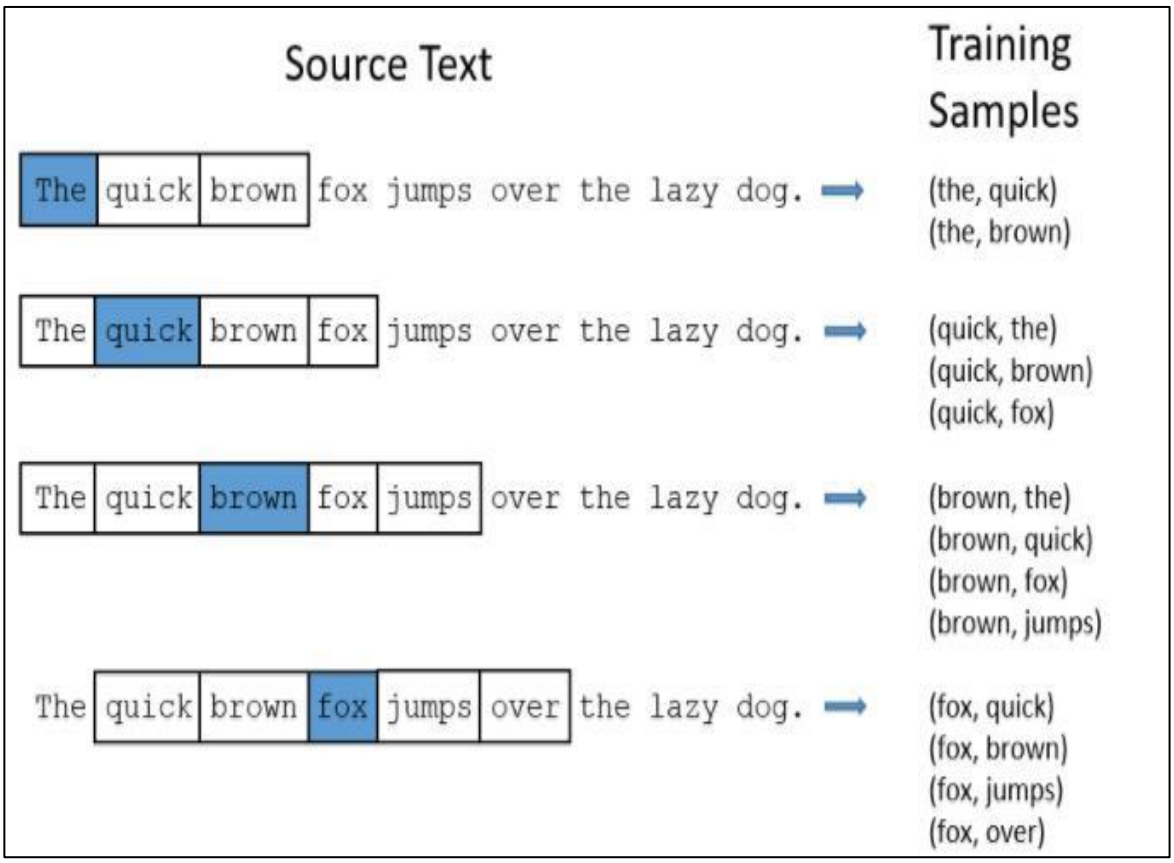
Υπάρχουν και ορισμένα μοντέλα που υλοποιούνται με πίνακες που περιέχουν λέξεις που εμφανίζονται δίπλα σε άλλες. Τα μοντέλα αυτά προϋποθέτουν να υπάρχει ένας πίνακας που να αποτελείται από μία στήλη, μία σειρά για την κάθε λέξη, στον πίνακα αυτό απεικονίζονται οι εμφανίσεις λέξεων που βρίσκονται κοντά ανάλογα με το παράθυρο αναζήτησης. Η συγκεκριμένη αντιμετώπιση έχει καλύτερα αποτελέσματα διότι λέξεις που βρίσκονται δίπλα σε άλλες συνήθως έχουν κάποιου είδους σχέση. Ωστόσο για την υλοποίηση των συγκεκριμένων μοντέλων υπάρχουν πολύ μεγάλες υπολογιστικές απαιτήσεις [16].

Το πρόβλημα των απαιτήσεων αυτών λύνεται με την αξιοποίηση των μοντέλων πρόβλεψης, κατορθώνοντας ταυτόχρονα αρκετά αποτελεσματικότερη απεικόνιση των λέξεων. Στην παρούσα έρευνα θα μελετηθεί ο αλγόριθμος word2vec ο οποίος κατατάσσεται στην ενότητα αυτών των μοντέλων, τα οποία σε συνδυασμό με τα νευρωνικά δίκτυα αναλμβάνουν την διανυσματική παρουσία των λέξεων. Έπειτα θα μελετηθούν οι δύο βασικότερες αρχιτεκτονικές των μοντέλων πρόβλεψης, πρόκειται για την Skip-gram και την CBOW.

2.7.3 Αρχιτεκτονικές μοντέλων πρόβλεψης

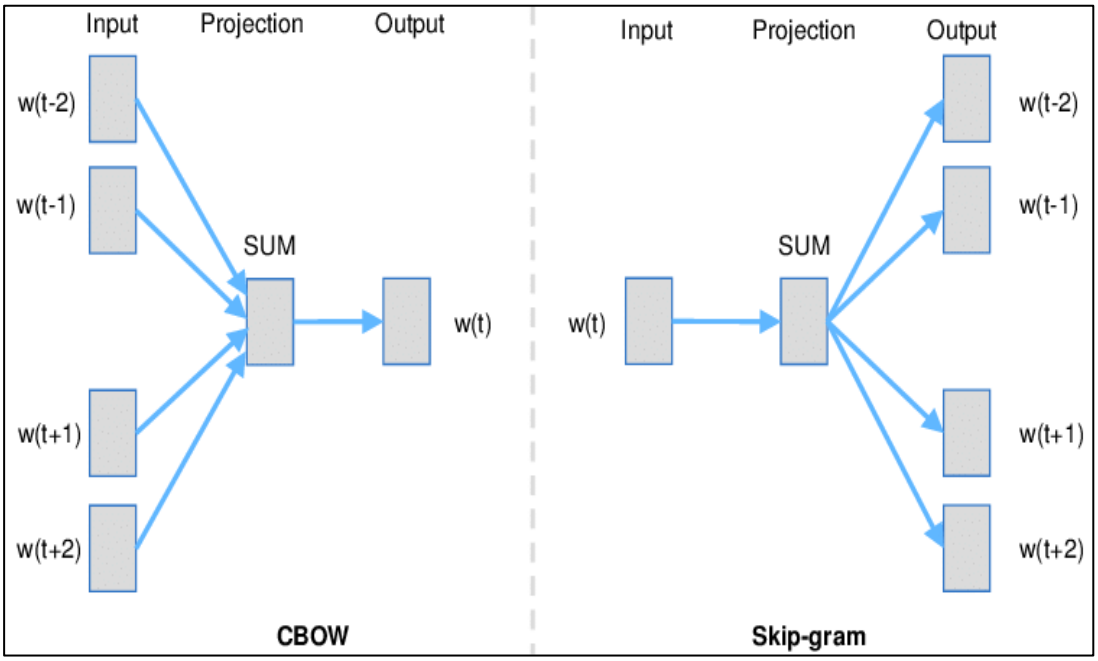
Μία συχνή μέθοδος, είναι να παρακολουθείται το πόσο συχνά εμφανίζεται μία λέξη δίπλα σε άλλες. Αυτή είναι η μέθοδος που ένας άνθρωπος κατανοεί την γλώσσα. Παραδείγματος χάρη, εάν οι λέξεις που εντοπίζονται περισσότερες φορές δίπλα στην λέξη αυτοκίνητο, είναι οι λέξεις οδηγώ και όχημα, φαίνεται ότι σε μία διάσταση οι λέξεις αυτές έχουν κάποια σχέση. Σύμφωνα με την αντίληψη αυτή έχουν δημιουργηθεί δύο πολύ αποδοτικές αρχιτεκτονικές που είναι ικανές να δημιουργούν διανύσματα από λέξεις. Αυτές είναι η Skip-gram και η CBOW.

Στην αρχιτεκτονική Skip-gram, για τη κάθε λέξη που εντοπίζεται μέσα σε ένα κείμενο, αποθηκεύονται τα δεδομένα για τις λέξεις που εντοπίζονται πριν και μετά από αυτή. Το σύνολο των λέξεων δεν είναι σταθερό και προσαρμόζεται στις εκάστοτε ανάγκες εφαρμογής. Το ίδιο ισχύει και για τα βάρη των λέξεων, πιο συγκεκριμένα οι λέξεις που εντοπίζονται πλησιέστερα σε μία λέξη είναι μεγαλύτερης σημασίας συγκριτικά με εκείνες που εντοπίζονται στο τέλος του παραθύρου ελέγχου.



Εικόνα 7: Παράδειγμα αρχιτεκτονικής Skip-gram [17]

Όπως απεικονίζεται και στο παράδειγμα της Εικόνας 7, κατασκευάζονται ζευγάρια εκπαίδευσης για όλες τις λέξεις που εντοπίζονται στο κείμενο. Αυτά τα δεδομένα, μπορούν να αξιοποιηθούν για να δημιουργηθεί και να μπορεί να μεταβληθεί το διάνυσμα της κάθε λέξης. Από την άλλη, η αρχιτεκτονική CBOW χρησιμοποιεί μια εντελώς διαφορετική μέθοδο. Για κάθε λέξη που εντοπίζεται στο κείμενο ανανεώνεται το διάνυσμα της κάθε λέξης που βρίσκεται κοντά της, σε αντίθεση με την αρχιτεκτονική Skip-gram που ανανεώνεται το διάνυσμα της συγκεκριμένης λέξης. Οι διαφορές των δύο αυτών αρχιτεκτονικών απεικονίζονται στην Εικόνα 8.



Εικόνα 8: Σύγκριση αρχιτεκτονικών [18]

Ακόμη μία διαφορά των δύο αρχιτεκτονικών είναι η μέθοδος με την οποία γίνεται η πρόβλεψη του μοντέλου. Με την Skip-gram με γνώμονα μία λέξη, γίνεται πρόβλεψη των λέξεων που είναι πιθανό να εντοπίζονται παραπλήσια της ενώ με την CBOW γίνεται πρόβλεψη της λέξης που υπάρχει πιθανότητα να εντοπίζεται ανάμεσα σε κάποιες λέξεις. Με βάση τις αρχιτεκτονικές που μόλις αναφέρθηκαν παράχθηκαν μέθοδοι που είναι υπεύθυνοι για να δημιουργήσουν και να αξιοποιήσουν το μοντέλο. Μία από τις πιο γνωστές είναι η word2vec.

2.8 Ο αλγόριθμός word2vec

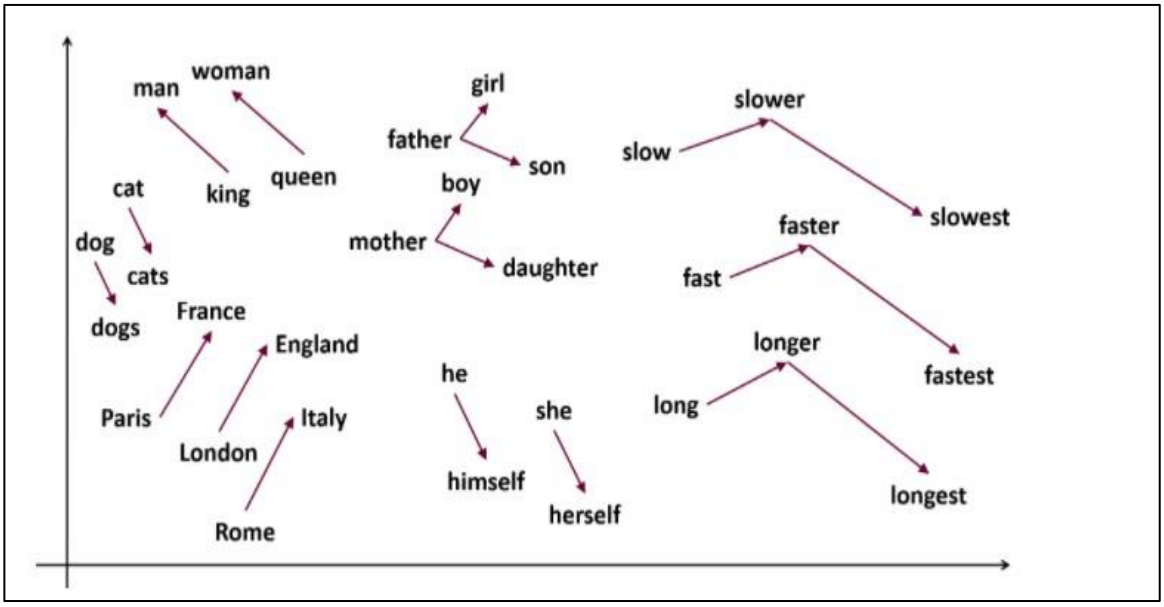
2.8.1 Προσέγγιση του αλγορίθμου

Ο αλγόριθμος word2vec αποτελείται από μια επαναληπτική μέθοδο η οποία παράγει διανύσματα λέξεων και δημιουργεί μοντέλα που προτάθηκαν από την Google [19]. Ό αλγόριθμος word2vec υλοποιείται χρησιμοποιώντας την μέθοδο της επανάληψης. Πιο συγκεκριμένα κατά την εκπαίδευση του μοντέλου ενημερώνει το μοντέλο μετά από κάθε κείμενο που επεξεργάζεται, ενώ άλλοι αλγόριθμοι διαχειρίζονται το κείμενο σαν ένα σύνολο κατά τη διάρκεια δημιουργίας του μοντέλου.

Κατά τη διάρκεια παραγωγής ενός μοντέλου δέχεται σαν δεδομένα εισόδου ένα μεγάλο αριθμό κειμένων και παράγει ένα διανυσματικό χώρο, στον οποίο για την κάθε μία λέξη

του κειμένου παράγεται και ένα μοναδικό διάνυσμα. Οι διαστάσεις συνηθώς αγγίζουν το μέγεθος των εκατοντάδων.

Η διεργασία δημιουργίας του μοντέλου είναι εφικτό να χαρακτηριστεί από τις παραμέτρους του, δηλαδή από την αρχιτεκτονική που θα χρησιμοποιηθεί, το πλήθος των διαστάσεων και το μέγεθος του παραθύρου. Όπως προαναφέρθηκε οι αρχιτεκτονικές που μπορούν να χρησιμοποιηθούν είναι οι Skip-gram και CBOW. Ως παράθυρο ελέγχου ορίζεται το πλήθος των λέξεων που θα χρησιμοποιηθούν στο μοντέλο και βρίσκονται πρίν και μετά από την κάθε λέξη. Επίσης, το πλήθος των διαστάσεων καθορίζει το πόσο μεγάλο θα είναι το μοντέλο καθώς και την πολυπλοκότητα του.



Εικόνα 9: Απεικόνιση με word2vec [20]

Με τη μέθοδο word2vec κατά τη δημιουργία των διανυσμάτων λέξεων δίνεται η δυνατότητα στις λέξεις να κρατούν τόσο τη σημασία τους όσο και το συντακτικό τους, γεγονός που τους επιτρέπει να μπορούν να τις χρησιμοποιηθούν με την βοήθεια αριθμητικών πράξεων. Όπως φαίνεται και στην Εικόνα 9, υπάρχουν αναλογίες όσον αφορά τον τρόπο με τον οποίο τοποθετούνται οι λέξεις στον διανυσματικό χώρο. Με μαθηματικές πράξεις είναι δυνατόν να επιλυθούν αναλογίες της μορφής: 'Η λέξη Netherland σχετίζεται με τη λέξη Amsterdam, όπως η λέξη Italy με τη λέξη Rome'. Επιπρόσθετα, φαίνεται πως λέξεις με παρόμοια σημασία, εντοπίζονται σε κοντινές θέσεις στο διανυσματικό χώρο.

Βασική προϋπόθεση προκειμένου να αξιοποιηθεί με τον καλύτερο δυνατό τρόπο ένα μοντέλο αυτού του είδους είναι να κατανοηθεί πλήρως το πως αυτό δημιουργήθηκε. Όταν δύο λέξεις εντοπίζονται παραπλήσια στο διανυσματικό χώρο υποδηλώνει ότι μέσα στο

κείμενο εντοπίζονται κοντά σε όμοιες λέξεις. Παραδείγματος χάρη, οι λέξεις που ορίζουν την πρόγνωση του καιρού είναι πιθανό να εντοπίζονται σε κοντινές θέσεις, καθώς και οι λέξεις που χαρακτηρίζουν ένα άνθρωπο. Η λέξη αγόρι είναι πιθανό να εντοπίζεται κοντά με τη λέξη κορίτσι σε μία διάσταση, αφού οι λέξεις αυτές ορίζουν τον άνθρωπο. Ακόμα η λέξη αγόρι υπάρχει περίπτωση να εμφανίζεται κοντά στην λέξη άντρας. Αυτό δεν υποδηλώνει βέβαια ότι η λέξη άντρας θα βρίσκεται πλησιέστερα στην λέξη αγόρι από τη λέξη κορίτσι, αφού είναι πιθανό η διάσταση που ορίζει το ανθρώπινο φύλο να μην είναι τόσο δυνατή και η λέξη άντρας να εντοπίζεται πιο πολλές φορές κοντά σε λέξεις που είναι κοινές με τη λέξη κορίτσι.

Όπως έχει προαναφερθεί, χρησιμοποιώντας πολύ μεγάλα κείμενα κατά την εκπαίδευση word2vec μοντέλων τα αποτελέσματα είναι πολύ καλύτερα, με κάποιες εξαιρέσεις. Στην περίπτωση που δεν έχει παραχθεί κάποιο διάνυσμα για ορισμένες άγνωστες λέξεις, δεν είναι εφικτό να ταυτιστούν με άλλες λέξεις. Λύση στο πρόβλημα αυτό έχει δώσει το μοντέλο Fast Text το οποίο έχει παρόμοια λογική με την υλοποίηση του αλγορίθμου word2vec. Το Fast Text χρησιμοποιήθηκε πρώτη φορά από την εφαρμογή του Facebook και η κύρια διαφορά του από τον αλγόριθμο word2vec είναι ότι τα διανύσματα παράγονται σε με βάση τις συλλαβές και όχι με βάση τις λέξεις [21].

Με αυτό το τρόπο είναι εφικτό να συσχετιστούν οι άγνωστες λέξεις, με λέξεις που έχουν πανομοιότυπες συλλαβές. Ωστόσο, ένα σημαντικό μειονέκτημα είναι οι πολλές υπολογιστικές απαιτήσεις που χρειάζονται προκειμένου να χειριστούν όλες οι συλλαβές.

2.8.2 Η υλοποίηση word2vec με χρήση της Python

Για την δημιουργία του αλγορίθμου word2vec συνίσταται να χρησιμοποιηθεί η βιβλιοθήκη gensim διότι θα επεξεργαστεί ένας μεγάλος όγκος πληροφοριών και δεν προτείνεται η υλοποίηση του εξ΄αρχής χωρίς να υπάρχει κάποιος συγκεκριμένος λόγος. Στην παρούσα εργασία πραγματοποιήθηκε μία πολύ απλή υλοποίηση προκειμένου να αναλυθούν οι βασικές αρχές λειτουργίας του.

Η υλοποίηση αυτή, έγινε με Python με τη χρήση του αλγορίθμου word2vec με τη βοήθεια της γνωστής βιβλιοθήκης της Google την tensorflow, βασισμένο στην αρχιτεκτονική Skip-gram. Επιπρόσθετα, γίνεται χρήση της βιβλιοθήκης numpy [22], προκειμένου να μας παρέχεται η απαραίτητη λειτουργικότητα για υπολογισμούς.

Στην Εικόνα 10, στις δύο πρώτες γραμμές εισάγονται οι βιβλιοθήκες tensorflow και numpy, στην γραμμή 4 γίνεται χρήση της κατάλληλης έκδοσης των βιβλιοθηκών και στις

υπόλοιπες γραμμές δημιουργείται η μέθοδος create_dict προκειμένου να δημιουργηθεί ένα σύνολο μοναδικών λέξεων από ένα κείμενο. Πιο συγκεκριμένα, στις γραμμές 7-9 εισάγονται σε ένα άδειο πίνακα οι λέξεις του κειμένου και τροποποιούνται όλα τα γράμματα σε πεζά, μετατρέπονται οι τελείες και τα κόμματα σε κενά και έπειτα χωρίζονται οι λέξεις. Τέλος, στη γραμμή 11 χρησιμοποιώντας την μέθοδο set εμφανίζονται όλες οι λέξεις του αποκλειστικά μία φορά.

```
1   import tensorflow.compat.v1 as tf
2   import numpy as np
3
4   tf.disable_v2_behavior()
5
6   #Create a dictionary from the corpus
7   def create_dict(corpus):
8       words = []
9       for word in corpus.lower().replace('.',' ').split():
10          words.append(word)
11      dictionary = set(words)
12      return dictionary
```

Εικόνα 10: Εκτέλεση word2vec 1

Στην Εικόνα 11, εμφανίζονται ακόμα δύο μέθοδοι που χρησιμοποιήθηκαν προκειμένου να γίνει η υλοποίηση του αλγορίθμου. Στην γραμμή 15, ορίζεται η get_sentences που είναι υπεύθυνη να διαχωρίζει το κείμενο σε προτάσεις και τα τοποθετεί σε μία λίστα. Ο διαχωρισμός του κειμένου γίνεται με την χρήση της τελείας, αυτός είναι και ο τρόπος που ξεχωρίζει η μία πρόταση από τις άλλες.

Έπειτα, δημιουργείται ένας άδειος πίνακας στον οποίο μετά τοποθετούνται οι λέξεις της κάθε πρότασης με πεζά γράμματα και στην γραμμή 19 επιστρέφεται σαν αποτέλεσμα ο πίνακας αυτός.

Στις υπόλοιπες γραμμές, δημιουργείται μία ακόμη βοηθητική μέθοδος με την οποία τροποποιούμε έναν αριθμό σε διάνυσμα. Στην γραμμή 23, παράγεται ένα διάνυσμα ίσο με το μηδέν το μέγεθος του οποίου ορίζεται από το μήκος του λεξικού. Τέλος, στην γραμμή 24 ορίζεται ως είσοδος του διανύσματος ο δείκτης 1 και επιστρέφεται το διάνυσμα.

```
15      def get_sentences(corpus):
16          sentences = []
17          for sentence in corpus.split('.'):
18              sentences.append(sentence.lower().split())
19          return sentences
20
21      #Create a vector an index eq. 2=> [0,0,1,0,0,0]
22      def create_vector(index, dictionary_size):
23          vector = np.zeros(dictionary_size)
24          vector[index] = 1
25          return vector
```

Εικόνα 11: Εκτέλεση word2vec 2

Ακολούθως, αρχίζει η υλοποίηση του word2vec. Στην γραμμή 27, δίνεται σαν είσοδος του αλγορίθμου ένα ενδεικτικό κείμενο. Στις γραμμές 30 και 31 τοποθετείται στις μεταβλητές integerForWord και wordForInteger ένα λεξικό για την κάθε μία προκειμένου να απεικονισθεί κάθε λέξη με ένα αριθμό και το ανάποδο. Έπειτα, παράγεται ένα λεξικό από το κείμενο που δόθηκε σαν είσοδος. Στη γραμμή 35, αποθηκεύεται το μέγεθος του λεξικού που δημιουργήθηκε προηγουμένως. Στη συνέχεια, στον βρόχο επανάληψης γεμίζουν με αριθμούς και λέξεις τα λεξικά που δημιουργήθηκαν προηγουμένως. Παραδείγματος χάρη, αν μία λέξη εντοπίζεται στην θέση 5 στο λεξικό, στην γραμμή 39 εισάγεται ο αριθμός 5 σε αντίθεση με την γραμμή 30 που εισάγεται η λέξη.

```
27  corpus_raw = 'A boy goes to school. He takes a lesson. A girl takes a lesson. ' \
28               'She walks into the library. The girl meets the boy'
29
30  integerForWord = {}
31  wordForInteger = {}
32
33  dictionary = create_dict(corpus_raw)
34
35  vocabulary_size = len(dictionary)
36
37  #Map words to ints and ints to words
38  for i, word in enumerate(dictionary):
39      integerForWord[word] = i
40      wordForInteger[i] = word
41
42  sentences = get_sentences(corpus_raw)
```

Εικόνα 12: Εκτέλεση word2vec 3

```
44  WINDOW_SIZE = 2
45
46  #Create pairs for each word with WINDOW_SIZE in mind
47  pairs = []
48  for sentence in sentences:
49      for j, word in enumerate(sentence):
50          leftBoundary = max(j - WINDOW_SIZE, 0)
51          rightBoundary = min(j + WINDOW_SIZE, len(sentence)) + 1
52          for nearby in sentence[leftBoundary : rightBoundary]:
53              if nearby != word:
54                  pairs.append([word, nearby])
```

Εικόνα 13: Εκτέλεση word2vec 4

Στην Εικόνα 13, παρουσιάζεται η παραγωγή ζευγαριών από λέξεις που εντοπίζονται στην κάθε πρόταση. Αρχικά, αρχικοποιείται το μέγεθος του παραθύρου να είναι ίσο με 2 και δημιουργείται ένα κενός πίνακας με το όνομα pairs.

Ακολούθως, στον βρόχο επανάληψης παράγονται ζευγάρια λέξεων για την κάθε πρόταση ανάλογα με το μέγεθος του παραθύρου, δηλαδή αν αυτό ισούται με 2 θα δημιουργηθεί από ένα ζευγάρι με την λέξη αυτή και τις διπλανές της. Παραδείγματος χάρη, για το κείμενο που έχουμε σαν είσοδο στην προκειμένη περίπτωση, για ‘goes’ με τη χρήση του μεγέθους του παραθύρου που ισούται με 2, οι λέξεις είναι οι ‘a boy goes to school’. Επομένως, φτιάχνονται τέσσερα ζευγάρια με πρώτη λέξη τη goes και δεύτερη μία από τις υπόλοιπες για κάθε ζευγάρι. Με αυτό τον τρόπο παρουσιάζεται η πληροφορία των λέξεων που βρίσκονται δίπλα στην λέξη goes.

```
56  input = []
57  output = []
58
59  #Map the pairs to vectors
60  for pair in pairs:
61      input.append(create_vector(integerForWord[pair[0]], vocabulary_size))
62      output.append(create_vector(integerForWord[pair[1]], vocabulary_size))
63
64  #Numpy arrays for input and output
65  x_train = np.asarray(input)
66  y_train = np.asarray(output)
```

Εικόνα 14: Εκτέλεση word2vec 5

Όπως φαίνεται στην Εικόνα 14, ξεκινούν να προετοιμάζονται τα δεδομένα που θα χρειαστούν για να υλοποιηθεί η εκπαίδευση. Αρχικά, ορίζονται δύο πίνακες στους οποίους μέσα στο βρόχο επανάληψης τοποθετούνται τα διανύσματα για κάθε ζευγάρι λέξεων. Στην γραμμή 61, τοποθετούνται οι λέξεις για τις οποίες πραγματοποιείται η αναζήτηση ενώ στη γραμμή 62 τοποθετούνται οι πιθανές λέξεις που βρίσκονται σε διπλανές θέσεις.

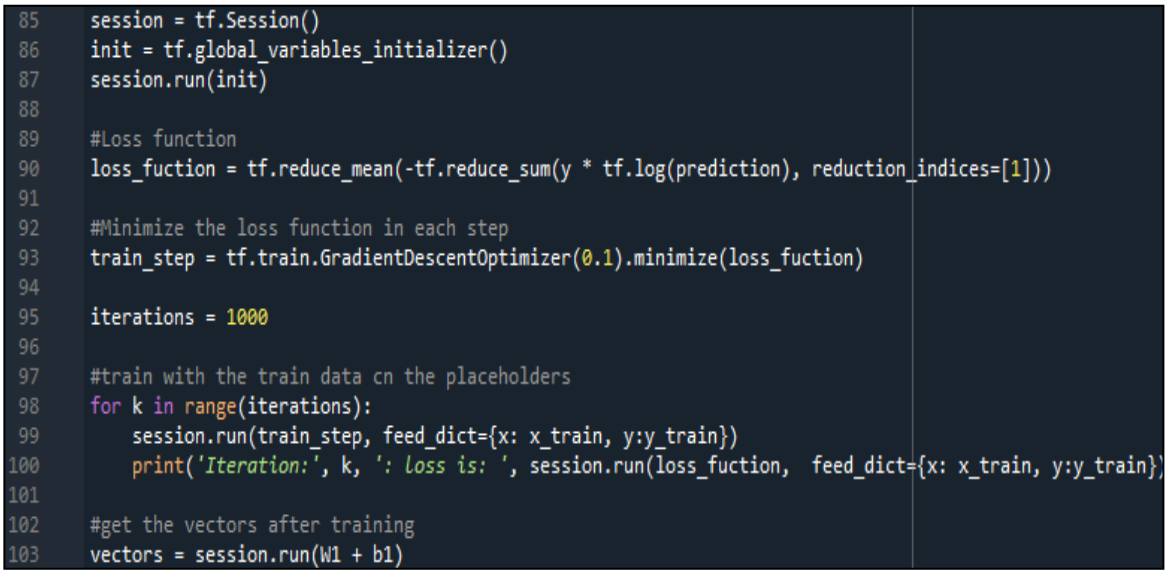
Τέλος, γίνεται μετατροπή των πινάκων input και output σε numpy μορφή προκειμένου να αποθηκευτούν σε μία μεταβλητή ο κάθε ένας από αυτούς και να αποτελέσουν στην συνέχεια την είσοδο του μοντέλου.

```
69  #Input and output vectors
70  x = tf.placeholder(tf.float32, shape=(None, vocabulary_size))
71  y = tf.placeholder(tf.float32, shape=(None, vocabulary_size))
72
73  DIMENSIONS = 5
74  #Tensorflow variables for input generated by training
75  W1 = tf.Variable(tf.random_normal([vocabulary_size, DIMENSIONS]))
76  b1 = tf.Variable(tf.random_normal([DIMENSIONS]))
77  representation = tf.add(tf.matmul(x, W1), b1)
78
79  #Tensorflow prediction
80  W2 = tf.Variable(tf.random_normal([DIMENSIONS, vocabulary_size]))
81  b2 = tf.Variable(tf.random_normal([vocabulary_size]))
82  prediction = tf.nn.softmax(tf.add(tf.matmul(representation, W2), b2))
```

Εικόνα 15: Εκτέλεση word2vec 6

Όπως απεικονίζεται στην Εικόνα 15, ξεκινά η προεργασία για την εκτέλεση της εκπαίδευσης με τη βοήθεια του tensorflow. Τα δύο κυρίαρχα είδη τιμών στο tensorflow είναι τα variable και τα placeholder. Στην γραμμή 70 και 71, παράγονται δύο placeholder, το ένα αφορά την λέξη που έχει δοθεί σαν είσοδος και το άλλο αφορά την λέξη που θα επιστραφεί σαν έξοδος δηλαδή μια κοντινή λέξη για την προηγούμενη.

Επιπρόσθετα, στις γραμμές 75 και 76 δημιουργούνται δύο μεταβλητές οι οποίες θα αλλάζουν τιμή κατά την διάρκεια της εκπαίδευσης με σκοπό να επιτευχθεί η αποτελεσματικότερη αναπαράσταση των δεδομένων. Έπειτα, παρουσιάζεται η συνάρτηση η οποία θα χρησιμοποιηθεί για να απεικονίσουμε το μοντέλο σε πέντε διαστάσεις για το διάνυσμα της κάθε λέξης του κειμένου. Στο word2vec η συνάρτηση αυτή όπως φαίνεται και στην γραμμή 77 είναι η: ‘x* W1 + b1’. Το x είναι το διάνυσμα για την κάθε λέξη. Το W1 πρόκειται για ένα πίνακα που περιλαμβάνει τα διανύσματα της κάθε λέξης. Τέλος το b1 είναι μια απλή σταθερά. Στις γραμμές 80-82, καθορίζεται η συνάρτηση πρόβλεψης της κάθε λέξης και καλείται η softmax [23]. Η διαδικασία υπολογισμού αποτελείται από τον πολλαπλασιασμό του πίνακα W2 με την συνάρτηση και της σταθεράς b2.

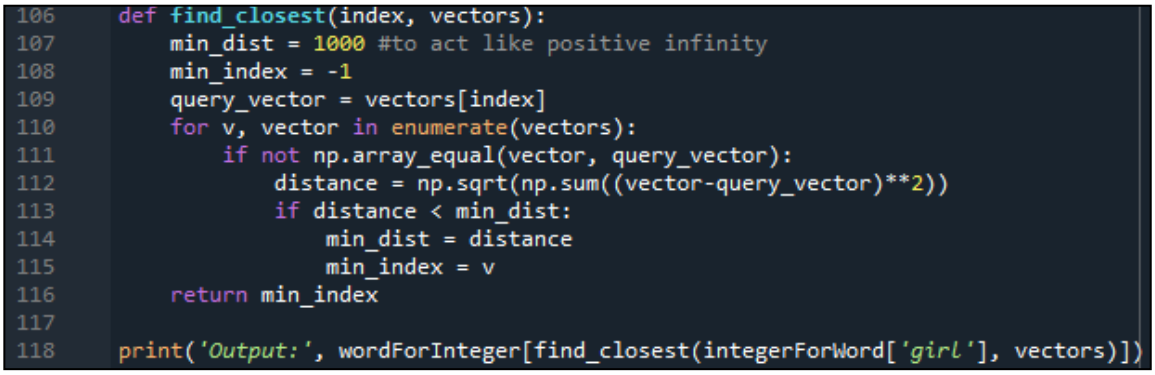


```
session = tf.Session()
init = tf.global_variables_initializer()
session.run(init)

#Loss function
loss_fuction = tf.reduce_mean(-tf.reduce_sum(y * tf.log(prediction), reduction_indices=[1]))

#Minimize the loss function in each step
train_step = tf.train.GradientDescentOptimizer(0.1).minimize(loss_fuction)

iterations = 1000

#train with the train data cn the placeholders
for k in range(iterations):
    session.run(train_step, feed_dict={x: x_train, y:y_train})
    print('Iteration:', k, ': loss is: ', session.run(loss_fuction,  feed_dict={x: x_train, y:y_train})

#get the vectors after training
vectors = session.run(W1 + b1)
```

Εικόνα 16: Εκτέλεση word2vec 7

Στην προηγούμενη Εικόνα, απεικονίζεται η έναρξη της διεργασίας του tensorflow που θα χρησιμοποιηθεί στην εκπαίδευση του μοντέλου. Στις γραμμές 85-87, αρχικοποιούνται οι παράμετροι που θα χρησιμοποιηθούν. Έπειτα, στην μεταβλητή loss_function ορίζεται η συνάρτηση απώλειας. Στην γραμμή 93, στην μεταβλητή train_step εισάγεται το βήμα της εκπαίδευσης με στόχο την ελαχιστοποίηση της συνάρτησης loss_function προκειμένου να αναπαρασταθούν αποτελεσματικότερα τα δεδομένα.

Στις επόμενες γραμμές, ορίζεται στην μεταβλητή iterations ο αριθμός των επαναλήψεων ο οποίος ισούται με 1000. Ο βρόγχος επανάληψης της γραμμής 98, δέχεται ως είσοδο δύο πίνακες που έχουν ορισθεί προηγουμένως και πραγματοποιείται η τελική εκπαίδευση του μοντέλου. Τέλος, η μεταβλητή vectors χρησιμοποιείται για την αποθήκευση των τελικών αποτελεσμάτων του μοντέλου.



```
def find_closest(index, vectors):
    min_dist = 1000 #to act like positive infinity
    min_index = -1
    query_vector = vectors[index]
    for v, vector in enumerate(vectors):
        if not np.array_equal(vector, query_vector):
            distance = np.sqrt(np.sum((vector-query_vector)**2))
            if distance < min_dist:
                min_dist = distance
                min_index = v
    return min_index

print('Output:', wordForInteger[find_closest(integerForWord['girl'], vectors)])
```

Εικόνα 17: Εκτέλεση word2vec 8

Στην Εικόνα 17, δημιουργήθηκε μια συνάρτηση για την εφαρμογή του μοντέλου. Πιο

συγκεκριμένα, στις γραμμές 107-116 γίνεται υπολογισμός της απόστασης μεταξύ δύο διανυσμάτων προκειμένου να βρεθεί το κοντινότερο διάνυσμα. Τέλος, στην γραμμή 118 εμφανίζεται με την εντολή print το πιο κοντινό διάνυσμα της λέξης που δόθηκε σαν είσοδος δηλαδή της λέξης girl.

Εικόνα 18: Αποτέλεσμα υλοποίησης word2vec

Όπως φαίνεται και στην Εικόνα 18, το αποτέλεσμα της διεργασίας επέστρεψε σαν αποτέλεσμα την λέξη boy γεγονός που υποδηλώνει ότι το διάνυσμα της λέξης girl βρίσκεται κοντά στο διάνυσμα της λέξης boy.

2.9 Αξιολόγηση word2vec μοντέλων

2.9.1 Η χρησιμότητα της αξιολόγησης

Ένα σοβαρό θέμα στην παραγωγή μοντέλων στον τομέα της επεξεργασίας φυσικής γλώσσας είναι ο τρόπος με τον οποίο θα αξιολογηθούν. Είναι ιδιαίτερα δύσκολο να αξιολογηθούν από τον άνθρωπο τέτοιου είδους μοντέλα διότι διαχειρίζονται μεγάλο όγκο δεδομένων και παράγουν διανύσματα αρκετά μεγάλων διαστάσεων. Προκειμένου, να καταστεί εφικτό να αξιολογείται αποτελεσματικά ένα μοντέλο και να παρουσιάζεται με αριθμητική μορφή για να μπορούν να συγκριθούν τα μοντέλα, είναι επιτακτική ανάγκη να υπάρχουν τεκμηριωμένες διαδικασίες. Ακολούθως θα εξεταστούν οι σημαντικότερες μέθοδοι αξιολόγησης.

Οι δύο κυριότερες μέθοδοι αξιολόγησης μοντέλων είναι η εσωτερική και η εξωτερική αξιολόγηση [24]. Ως μέθοδοι εσωτερικής αξιολόγησης καλούνται αυτές που στηρίζονται στην ανάλυση του μοντέλου μέσω μιας συλλογής ερωτημάτων και αποτελεσμάτων που σχετίζονται με την σημασιολογική και συντακτική σχέση που έχουν οι λέξεις μεταξύ τους. Η μέθοδος αυτή πραγματοποιεί προβλέψεις για τα ερωτήματα και συγκρίνει τα αποτελέσματα με τα αναμενόμενα. Στόχος της εσωτερικής αξιολόγησης είναι να δοθεί σαν αποτέλεσμα μία αριθμητική μέτρηση που θα αντιπροσωπεύει το πόσο αποτελεσματικό είναι ένα μοντέλο.

Ως μέθοδοι εξωτερικής αξιολόγησης καλούνται αυτοί που εκμεταλλεύονται το μοντέλο

προκειμένου να εκτελούν διάφορες εφαρμογές που βασίζονται κατά κύριο λόγο στην επεξεργασία της φυσικής γλώσσας. Ο τρόπος με τον οποίο αξιολογείται αυτή η μέθοδος εξαρτάται αποκλειστικά από την αποδοτικότητα των εφαρμογών που προαναφέρθηκαν. Ιδιαίτερα σημαντικό είναι το γεγονός πως το αποτέλεσμα της αξιολόγησης ενός μοντέλου δεν είναι δεδομένο και πως μπορεί να αλλάζει αν μεταβάλλουμε την μέθοδο της αξιολόγησης. Παραδείγματος χάρη, τα ποσοστά επιτυχίας ενός μοντέλου μπορεί να πολύ καλύτερα με την χρήση μιας μεθόδου εξωτερικής αξιολόγησης παρά μιας μεθόδου εσωτερικής αξιολόγησης. Για το κάθε μοντέλο πρέπει να γίνεται σωστή επιλογή της μεθόδου αξιολόγησης σύμφωνα με όσα προαναφέρθηκαν καθώς και των αναγκών του κάθε μοντέλου.

2.9.2 Εσωτερική αξιολόγηση

Στην εσωτερική αξιολόγηση τα μέτρα με τα οποία θα γίνει η αξιολόγηση μπορεί να είναι είτε απόλυτα είτε συγκριτικά. Με την χρήση των απόλυτων μέτρων το μοντέλο δοκιμάζεται με έτοιμα δεδομένα που δεν αλλάζουν. Αντιθέτως, με την χρήση των συγκριτικών μέτρων η αξιολόγηση πραγματοποιείται με την βοήθεια εξωτερικών παρατηρητών. Παραδείγματος χάρη, ζητείται από τους εξωτερικούς χρήστες η συμπλήρωση ερωτηματολογίου, όπου για κάποια είσοδο πρέπει να δώσουν την αναμενόμενη έξοδο του μοντέλου. Η εσωτερική αξιολόγηση μπορεί να χρησιμοποιήσει ορισμένες κατηγορίες ελέγχων. Οι κατηγορίες αυτές περιέχουν την αναλογία, την συνοχή, την σχετικότητα και την εκλεκτική προτίμηση [25].

Αναλογία καλείται η ικανότητα που έχει το μοντέλο να λύνει προβλήματα σχέσεων λέξεων. Αυτό το καταφέρνει το μοντέλο μέσω μαθηματικών υπολογισμών στα διανύσματα των λέξεων, δηλαδή αν υπάρχει μία σχέση μεταξύ μιας λέξης με μία άλλη το μοντέλο είναι ικανό να προβλέψει μία τρίτη λέξη η οποία να έχει σχέση με παρόμοιο τρόπο με μία τέταρτη λέξη.

Παραδείγματος χάρη, η λέξη ‘μεγάλος’ σχετίζεται με την λέξη ‘μικρός’ με τον ίδιο τρόπο που σχετίζονται οι λέξεις ‘κοντά’ και ‘μακριά’ μεταξύ τους.

Στην συγκεκριμένη περίπτωση, η αξιολόγηση σχετίζεται με τα ακριβή ποσοστά επιτυχίας των ερωτημάτων. Ωστόσο πρέπει να δοθεί ιδιαίτερη προσοχή όταν επιλέγονται τα ερωτήματα διότι υπάρχει μεγάλο πλήθος λέξεων που παρόλο που είναι ίδιες έχουν διαφορετική σημασία και όπως είναι εύκολα κατανοητό σε περίπτωση λάθος επιλογής θα επηρεαστούν τα αποτελέσματα του μοντέλου.

Η εκλεκτική προτίμηση πρόκειται για την προδιάθεση κάποιων λέξεων να εντοπίζονται

δίπλα με ένα σταθερό συντακτικό τρόπο. Ορισμένα ουσιαστικά ενδέχεται να παρουσιάζονται συνήθως με την μορφή υποκειμένου ενός ρήματος και όχι τόσο συχνά με την μορφή αντικειμένου του. Παραδείγματος χάρη, τέτοιες λέξεις είναι οι λέξεις ‘παιδί’ και ‘μελετώ’. Το πιθανότερο είναι η λέξη παιδί, να παρουσιαστεί ως υποκείμενο της λέξης μελετώ και όχι ως αντικείμενο. Αυτό γίνεται διότι πιο πιθανό είναι ένα παιδί να μελετά για παράδειγμα ένα βιβλίο παρά ένα άλλο υποκείμενο να μελετά ένα παιδί. Ελέγχωντας τέτοιου είδους παραδείγματα είναι εφικτό να μετρηθεί με ακρίβεια η απόδοση του μοντέλου και να αξιολογηθεί.

Σχετικότητα ονομάζεται ο βαθμός συγγένειας που εντοπίζεται μεταξύ δύο λέξεων. Στο μοντέλο word2vec η σχετικότητα μπορεί να εκφραστεί με αριθμητικό τρόπο. Πιο συγκεκριμένα, αν η σχετικότητα ανάμεσα σε δύο λέξεις ισούται με μηδέν αυτό δηλώνει ότι δεν έχουν καθόλου συγγένεια ενώ αν ισούται με ένα δηλώνει απόλυτη συγγένεια. Παραδείγματος χάρη, η σχετικότητα της λέξης ‘αυτοκίνητο’ και της λέξης ‘όχημα’ θα είχε τιμή που προσεγγίζει το ένα, αντιθέτως η σχετικότητα της λέξης ‘αυτοκίνητο’ και της λέξης ‘ζώο’ θα είχε τιμή που θα προσεγγίζει το μηδέν. Εδώ προκειμένου να είναι εφικτή η αξιολόγηση, χρησιμοποιούνται παραδείγματα ανάλογα του προηγουμένου και υπολογίζεται το πόσο απόκλιση το μοντέλο από τα παραδείγματα αυτά.

Η συνοχή υποδηλώνει ότι λέξεις που έχουν παρόμοια σημασία, οφείλουν να εμφανίζονται κοντά και στο διανυσματικό χώρο. Ένα αποδοτικό μοντέλο οφείλει να διατηρεί την συνοχή των γειτονικών λέξεων. Παραδείγματος χάρη, αν δοθούν σαν είσοδο λέξεις οι οποίες όλες εκτός από μία είναι ονόματα, η λέξη που δεν είναι όνομα πρέπει να εντοπίζεται αμέσως από το μοντέλο. Έτσι, αξιολογείται η συνοχή του μοντέλου.

Αν τα μέτρα αξιολόγησης είναι απόλυτα, τα ερωτήματα οφείλουν να πληρούν ορισμένα χαρακτηριστικά προκειμένου να εξασφαλιστεί μία ποιοτική αξιολόγηση. Οι λέξεις είναι απαραίτητο να εμφανίζονται με σταθερή συχνότητα, αν δηλαδή σκοπός είναι να αξιολογηθεί το μοντέλο στην αποτελεσματικότητα του σε λέξεις που εμφανίζονται συχνά, είναι λογικό να επιλέγονται λέξεις με μεγάλη συχνότητα. Εν κατακλείδι, οι λέξεις οφείλουν να έχουν συγκεκριμένη σημασία και όχι αφηρημένη.

Αξίζει να σημειωθεί ότι οι μέθοδοι εσωτερικής αξιολόγησης δεν δημιουργούν αποτελέσματα που μπορούν να συγκριθούν μεταξύ τους. Επεξηγηματικά, εάν σε ένα έλεγχο αναλογιών το ποσοστό επιτυχίας ισούται με 80%, αυτό δεν υποδηλώνει ότι το ποσοστό θα είναι το ίδιο και στους υπόλοιπους ελέγχους. Αυτό σημαίνει, ότι ανάλογα με τις απαιτήσεις του μοντέλου πρέπει να δίνεται βαρύτητα στις μεθόδους που αντιστοιχούν στο εκάστοτε μοντέλο.

2.9.3 Εξωτερική αξιολόγηση

Ως μέθοδοι εξωτερικής αξιολόγησης ορίζονται αυτοί που πραγματοποιούν έλεγχο της αποτελεσματικότητας ενός μοντέλου σε μία συγκεκριμένη εφαρμογή επεξεργασίας φυσικής γλώσσας. Οι εφαρμογές αυτές περιλαμβάνουν τις ακόλουθες:

- Αναγνώριση οντοτήτων
- Ανάλυση συναισθημάτων
- Αντιστοίχιση λέξεων
- Ανάλυση εξαρτήσεων

Η αναγνώριση οντοτήτων πρόκειται για μία εφαρμογή η οποία βρίσκει τις οντότητες σε μία πρόταση. Βασική προϋπόθεση είναι να καθοριστεί η δομή της πρότασης, αυτό σημαίνει ότι πρέπει να οριστούν οι οντότητες που υπάρχει πιθανότητα να περιέχει η πρόταση και έπειτα να γίνει εύρεση αυτών σε άλλες προτάσεις. Η αναγνώριση οντοτήτων εφαρμόζεται προκειμένου να κατανοηθεί το κείμενο και στους ψηφιακούς βοηθούς, αφού βοηθάει στην αντίληψη των εντολών που δίνει ο χρήστης. Παραδείγματος χάρη, είναι δυνατό να ειπωθεί ότι μία πρόταση περιλαμβάνει ένα τομέα μιας εφαρμογής (π.χ. φώτα σε ένα 'έξυπνο' αυτοκίνητο), μία ενέργεια (π.χ. άνοιγμα ή κλείσιμο) και μία τοποθεσία (π.χ. αυτοκίνητο).

Η εφαρμογή οφείλει να αντιληφθεί ότι στην πρόταση 'άνοιξε τα φώτα στο αυτοκίνητο' τομέας είναι η λέξη φώτα, ενέργεια είναι η λέξη άνοιξε και η τοποθεσία είναι η λέξη αυτοκίνητο. Χρησιμοποιώντας τέτοιου είδους παραδείγματα είναι εφικτό να υπολογιστεί η απόδοση του μοντέλου.

Η ανάλυση συναισθημάτων σχετίζεται με την εύρεση του χαρακτήρα μιας πρότασης. Αυτό υλοποιείται με τον εντοπισμό των θετικών και των αρνητικών λέξεων που υπάρχουν σε μία πρόταση, αυτό καθιστά δυνατό τον χαρακτηρισμό μίας πρότασης ως θετική, αρνητική ή αδιάφορη. Με αυτό το τρόπο η αξιολόγηση του μοντέλου γίνεται με τη χρήση μίας εφαρμογής ανάλυσης συναισθημάτων, βρίσκοντας έτσι το πόσο αποδοτική είναι.

Η αντιστοίχιση λέξεων σχετίζεται με τον ορθό τρόπο σύνδεσης λέξεων με άλλες λέξεις σε μία πρόταση. Παραδείγματος χάρη, στην πρόταση 'Ο Γιώργος προσκάλεσε τον Γιάννη στο σπίτι του', οφείλει να προσδιοριστεί ότι το άρθρο 'του' αναφέρεται στον Γιώργο και όχι στο Γιάννη. Χρησιμοποιώντας ένα μοντέλο σε μία παρόμοια εφαρμογή είναι εφικτό υλοποιηθεί η αξιολόγηση του ως πρός την αποτελεσματικότητα του.

Η ανάλυση εξαρτήσεων προσδιορίζει τις εξαρτήσεις μεταξύ λέξεων σε μία πρόταση. Παραδείγματος χάρη, μία τέτοια εφαρμογή έχει την δυνατότητα να εντοπίζει το υποκείμενο και το αντικείμενο ενός ρήματος. Η ανάλυση εξαρτήσεων υλοποιείται συνήθως με την

χρήση ενός δέντρου στο οποίο απεικονίζονται οι εξαρτήσεις της πρότασης. Με αυτό το τρόπο είναι δυνατό να αξιολογηθεί ένα μοντέλο μιας τέτοιας εφαρμογής ως προς το ποσοστό επιτυχίας του.

Αξίζει να σημειωθεί ότι ένα μοντέλο δεν μπορεί να αξιολογηθεί χρησιμοποιώντας ως γενικό κριτήριο τις μεθόδους εξωτερικής αξιολόγησης διότι οι μέθοδοι αυτοί αξιολογούν το μοντέλο σε καθορισμένες συνθήκες.

Για παράδειγμα, ορθό είναι να χρησιμοποιηθεί η αντιστοίχιση λέξεων σαν βασικό κριτήριο, για ένα μοντέλο που έχει αυτό το σκοπό, αλλά όχι και σε ένα μοντέλο που η χρήση του αφορά την αναγνώριση οντοτήτων. Εν κατακλείδι, για την καλύτερη αποτελεσματικότητα μιας αξιολόγησης πρέπει να επιλεγεί η αντίστοιχη μέθοδος αξιολόγησης που καλύπτει τις ανάγκες της εκάστοτε περίπτωσης.

3 Δεδομένα εκπαίδευσης

3.1 Πηγές δεδομένων εκπαίδευσης

Ο κυριότερος παράγοντας που καθορίζει την αποτελεσματικότητα ενός συστήματος που έχει σαν στόχο την επεξεργασία φυσικής γλώσσας, είναι το πόσο μεγάλα είναι τα δεδομένα εισόδου του μοντέλου αλλά και πόσο ποιοτικά είναι. Παρόλο που υπάρχει μια συνεχόμενη βελτίωση της απόδοσης των αλγορίθμων που χειρίζονται τη παραγωγή μοντέλων, χρησιμοποιώντας όλο και περισσότερα ποιοτικά δεδομένα βελτιώνεται ακόμα περισσότερο η αποτελεσματικότητα του συστήματος [26].

Ο όγκος των δεδομένων που μπορούν να χρησιμοποιηθούν στην επεξεργασία φυσικής γλώσσας είναι αρκετά μεγάλος και βρίσκεται στο διαδίκτυο. Οι περισσότερες από αυτές σχετίζονται με συλλογές λογοτεχνικού περιεχομένου, δεδομένα από εφημερίδες, εγκυκλοπαίδειες όπως και περιεχόμενο από το διαδίκτυο. Οι πιο πολλές πηγές αφορούν δεδομένα για την αγγλική γλώσσα, ωστόσο πολλές από αυτές διαθέτουν και υλικό σε αρκετές άλλες γλώσσες, αλλά σε πολύ λιγότερη ποσότητα σε σχέση με την αγγλική γλώσσα. Στην συγκεκριμένη εργασία η δημιουργία μοντέλων θα γίνει στην ελληνική γλώσσα.

Η προτιμότερη επιλογή δεδομένων στην συγκεκριμένη περίπτωση είναι η χρήση δεδομένων από την Wikipedia. Η Wikipedia διαθέτει μεγάλη πληθώρα άρθρων σε διάφορες γλώσσες [27]. Για τα ελληνικά αυτή τη στιγμή υπάρχουν πάνω από 100.000 διαθέσιμα άρθρα στην Wikipedia. Αν και ο αριθμός αυτός είναι αρκετά μικρότερος σε σχέση με άλλες γλώσσες είναι επαρκής για τις ανάγκες της παρούσας εργασίας.

3.2 Πηγές δεδομένων της Wikipedia

Η Wikipedia παρέχει τη δυνατότητα σε όλους τους χρήστες να λάβουν από την βάση δεδομένων της οποιοδήποτε αντίγραφο κειμένου επιθυμούν από διάφορες χρονικές στιγμές [28]. Αυτό γίνεται μέσω της Wikimedia [29], η οποία είναι μία από τις πολλές υποκατηγορίες που προσφέρει η Wikipedia.

Οι χρήστες μπορούν να λάβουν τα αντίγραφα κειμένου σε μια μεγάλη ποικιλία γλωσσών. Επίσης, παρέχεται μια μεγάλη πληθώρα άρθρων, περιλήψεων καθώς και αρκετών ακόμα στοιχείων που μπορούν να καλύψουν τις ανάγκες των χρηστών.

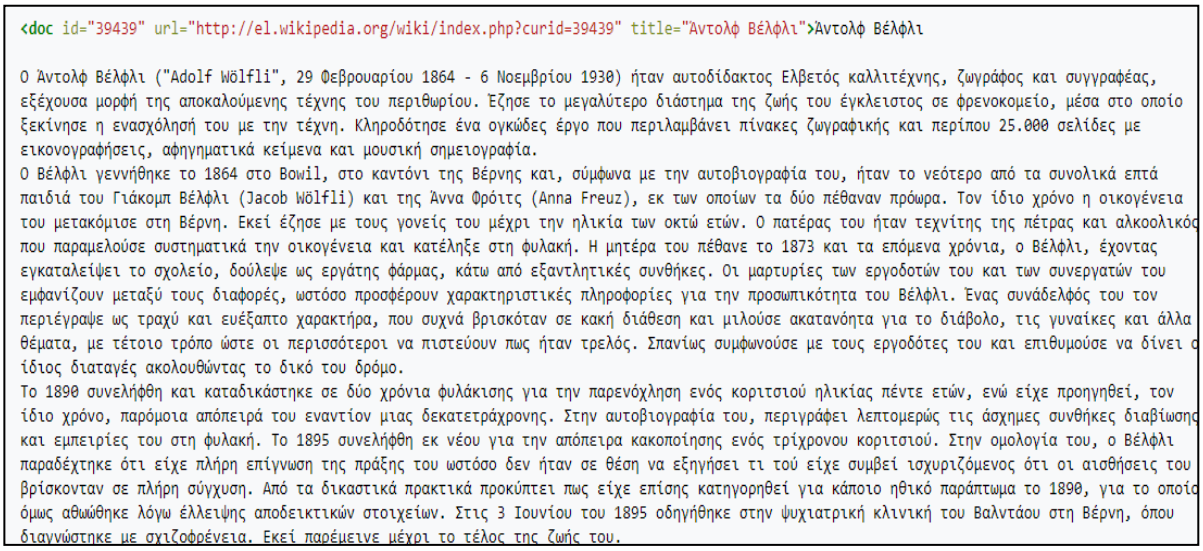
Στην ακόλουθη Εικόνα εμφανίζεται η σελίδα μέσω της οποίας πραγματοποιήθηκε η λήψη

του ελληνικού αντιγράφου που θα χρησιμοποιηθεί στην παρούσα έρευνα [30].

Εικόνα 19: Λήψη ελληνικού αντιγράφου Wikipedia [31]

3.3 Επεξεργασία δεδομένων

Έπειτα από την λήψη των απαιτούμενων δεδομένων από την Wikipedia είναι αναγκαίο να επεξεργαστούν προκειμένου να είναι έτοιμα για χρήση από το μοντέλο. Με την λήψη των δεδομένων από την Wikipedia επιστρέφεται ένα μεγάλο αρχείο xml το οποίο περιέχει όλα τα άρθρα. Στην ακόλουθη Εικόνα απεικονίζεται ο τρόπος καταγραφής των άρθρων.



Εικόνα 20:Το άρθρο για το λήμμα Άντολφ Βέλφι

Για το κάθε άρθρο περιέχονται αρκετές πληροφορίες όπως ο τίτλος, ο χρόνος δημοσίευσης, το id, τον υπερσύνδεσμο στον οποίο υπάρχει το άρθρο, το άτομο που το δημοσίευσε, μία περίληψη και το βασικό κείμενο του άρθρου. Το σημαντικό μέρος που χρησιμοποιείται από το μοντέλο word2vec είναι το κείμενο.

Αξίζει να σημειωθεί ότι τα κείμενα των άρθρων που εξάγονται από την Wikipedia περιέχουν αρκετά μέρη τα οποία είναι περιττά κατά την εκτέλεση του μοντέλου. Παραδείγματος χάρη, διάφορα σημεία στίξης, σύμβολα και ορισμένα σημεία σήμανσης που καθορίζουν την μορφή και το περιεχόμενο των άρθρων δεν χρειάζονται κατά την εκπαίδευση του μοντέλου. Για το λόγο αυτό είναι αναγκαίο τα δεδομένα πριν χρησιμοποιηθούν, να επεξεργαστούν προκειμένου να αφαιρεθούν τα μη απαραίτητα στοιχεία. Στην συγκεκριμένη εργασία, θα χρησιμοποιηθεί μια γνωστή βιβλιοθήκη της γλώσσας Python η οποία διαθέτει πολλά εργαλεία χρήσιμα στην επεξεργασία φυσικής γλώσσας. Η βιβλιοθήκη αυτή ονομάζεται gensim και είναι αυτή που θα χρησιμοποιηθεί για την προ-επεξεργασία των κειμένων πριν την παραγωγή του μοντέλου [32].

Η βιβλιοθήκη gensim περιλαμβάνει ένα μεγάλο όγκο πακέτων τα οποία χρησιμεύουν για την παραγωγή συλλογής δεδομένων. Ένα τέτοιο πακέτο είναι το WikiCorpus, το οποίο διαχειρίζεται την επεξεργασία δεδομένων από τη Wikipedia. Το πακέτο αυτό χρησιμοποιεί σαν είσοδο ένα αρχείο με άρθρα της Wikipedia και ορίζοντας τις κατάλληλες παραμέτρους είναι ικανό να δώσει σαν έξοδο μόνο το χρήσιμο κείμενο.

```
1  from gensim.corpora import WikiCorpus
2
3  if __name__ == '__main__':
4      inputFileName = 'elwiki-20210301-pages-articles-multistream.xml.bz2'
5      outputFileName = 'wiki.el.text'
6
7      print("Preparing wikipedia corpus")
8      wiki = WikiCorpus(inputFileName, dictionary={})
9      numberOfArticles = 0
10     output = open(outputFileName, 'wb')
11
12     print("Process initiated")
13     for text in wiki.get_texts():
14         numberOfArticles += 1
15         output.write(' '.join(text).encode('utf-8') + b'\n')
16         if numberOfArticles % 1000 == 0:
17             print("Currently at " + str(numberOfArticles) + 'articles')
18     output.close()
19     print('Finished\n Saved' + str(numberOfArticles) + 'articles to file' + outputFileName)
```

Εικόνα 21: Υλοποίηση WikiCorpus

Στην προηγούμενη Εικόνα, εμφανίζεται η χρήση του WikiCorpus στην γλώσσα Python για την προ επεξεργασία των κειμένων. Αρχικά, προκειμένου να γίνει χρήση του WikiCorpus είναι απαραίτητο να εισαχθεί στο περιβάλλον εργασίας της Python με την βοήθεια της βιβλιοθήκης gensim. Έπειτα, δηλώνεται στην μεταβλητή inputFileName το αρχείο εισόδου της Wikipedia και αντίστοιχα στην outputFileName το αρχείο στο οποίο θα αποθηκευτεί το αποτέλεσμα. Στην γραμμή 8, καλείται το πακέτο που θα αναλύσει το κείμενο.

Μετά το πέρας της ανάλυσης, ανοίγει το αρχείο εξόδου κειμένου για να γίνει η εγγραφή. Στη γραμμή 13, στον βρόχο επανάληψης με την χρήση της μεθόδου wiki.get_text παραχωρείται πρόσβαση στο επεξεργασμένο κείμενο. Έπειτα, γίνεται εγγραφή του αρχείου εξόδου στην γραμμή 15 και κλείνει το αρχείο στην γραμμή 18.

Στο τέλος αυτής της διαδικασίας, το αρχείο που δημιουργήθηκε είναι έτοιμο για να χρησιμοποιηθεί διότι δεν περιέχει καθόλου σύμβολα. Επίσης, στο αρχείο που παράχθηκε κάθε γραμμή αποτελείται μόνο από μία πρόταση. Το τελικό αρχείο, θα χρησιμοποιηθεί για τη δημιουργία και την εκπαίδευση μοντέλων με τον αλγόριθμο word2vec στην γλώσσα της Python.

4 Μοντέλο word2vec

4.1 Παραγωγή μοντέλου με χρήση της gensim

4.1.1 Δημιουργία σε Python

Προκειμένου να επιτευχθούν οι στόχοι της παρούσας εργασίας θα γίνει χρήση της βιβλιοθήκης gensim [24]. Για να δημιουργηθεί εξ΄αρχής ένα νέο μοντέλο πρέπει να χρησιμοποιηθεί το πακέτο Word2Vec το οποίο περιέχεται στην βιβλιοθήκη της gensim. Στην επόμενη Εικόνα, εμφανίζεται ο κώδικας στον οποίο καλείται ο αλγόριθμος Word2Vec με τη βοήθεια του οποίου θα δημιουργηθεί το μοντέλο που θα μελετηθεί στην παρούσα εργασία.

```
1  import os
2  import sys
3  import logging
4  import multiprocessing
5
6  from gensim.models import Word2Vec
7  from gensim.models.word2vec import LineSentence
8
9
10 if __name__ == '__main__':
11     program = os.path.basename(sys.argv[0])
12     logger = logging.getLogger(program)
13
14     logging.basicConfig(format= '%(asctime)s : %(levelname)s : %(message)s')
15     logging.root.setLevel(level=logging.INFO)
16     logger.info("running %s" % ''.join(sys.argv))
17
18
19     inp = "wiki.el.text"
20     outp = "wiki.el.skip.model"
21
22     model = Word2Vec(LineSentence(inp), sg=1, size=300, iter=1, window=5, min_count=10, workers=multiprocessing.cpu_count())
23
24
25     model.init_sims(replace=True)
26
27     model.save(outp)
```

Εικόνα 22: Μοντέλο Word2vec

Στις γραμμές 1-4, γίνεται εισαγωγή διάφορων πακέτων που είναι απαραίτητα για την σωστή λειτουργία του κώδικα. Έπειτα, εισάγεται το πακέτο Word2Vec από την βιβλιοθήκη gensim.models. Στην γραμμή 7 εισάγεται το πακέτο μέσω του οποίου θα διαχωριστούν οι προτάσεις από το αρχείο εισόδου που παράχθηκε προηγουμένως. Στο αρχείο που δημιουργήθηκε προηγουμένως η κάθε γραμμή περιέχει μία πρόταση και τώρα γίνεται το ανάποδο. Παράγεται δηλαδή μία λίστα από προτάσεις διαβάζοντας το κείμενο ανά γραμμή. Με αυτό το τρόπο εισάγονται τα δεδομένα στον αλγόριθμο Word2Vec.

Στις γραμμές 19 και 20, εισάγονται σε δύο ξεχωριστές μεταβλητές το αρχεία εισόδου και το αρχείο εξόδου αντίστοιχα. Πιο συγκεκριμένα, η μεταβλητή inp είναι αυτή στην οποία αποθηκεύονται τα αρχεία εισόδου ενώ στην μεταβλητή outp τοποθετείται το αρχείο εξόδου, δηλαδή το μοντέλο που έχει δημιουργηθεί. Έπειτα, καλείται το μοντέλο Word2Vec το οποίο στη συγκεκριμένη περίπτωση δέχεται επτά ορίσματα. Το πρώτο εξ' αυτών είναι οι προτάσεις που θα χρησιμοποιηθούν ως είσοδο για την εκπαίδευση. Το δεύτερο είναι το sg, το οποίο αντιπροσωπεύει την αρχιτεκτονική με την οποία θα παραχθεί το μοντέλο. Έαν θέλουμε να χρησιμοποιήσουμε την Skip-gram το sg παίρνει την τιμή ένα ενώ για την CBOW παίρνει την τιμή μηδέν. Το τρίτο όρισμα καθορίζει τις διαστάσεις σύμφωνα με τις οποίες θα παραχθεί το μοντέλο. Το τέταρτο όρισμα καθορίζει το πόσες επαναλήψεις θα γίνουν κατά τη διάρκεια της εκπαίδευσης. Το window αντιπροσωπεύει το μέγεθος του παραθύρου, δηλαδή τον αριθμό των λέξεων οι οποίες θα εξετάζονται πριν και μετά από κάθε λέξη. Το έκτο όρισμα παρέχει την δυνατότητα να επιλεχθούν οι λέξεις που θα χρησιμοποιηθούν στην εκπαίδευση. Παραδείγματος χάρη, αν το συγκεκριμένο όρισμα ισούται με 5, οι λέξεις που εμφανίζονται λιγότερες από πέντε φορές στο κείμενο δεν θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου.Το τελευτάιο όρισμα ορίζει τον αριθμό των νημάτων που χρησιμοποιηθούν στην εκπαίδευση. Τέλος, αξίζει να σημειωθεί ότι υπάρχει μεγάλη πληθώρα ορισμάτων ωστόσο αυτά που εξηγήθηκαν είναι τα κυριότερα.

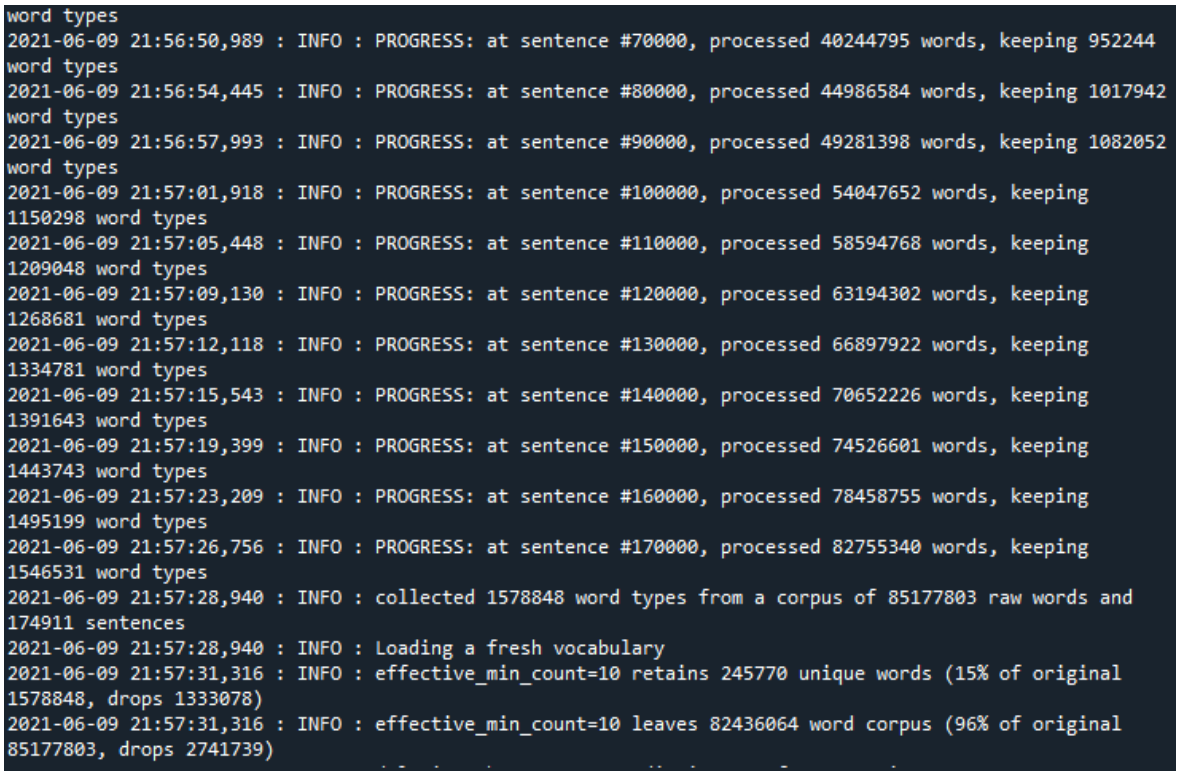
Οι δύο τελευταίες γραμμές του κώδικα αφορούν την καλύτερη χρήση της μνήμης RAM κατά τη διάρκεια εκπαίδευσης του μοντέλου και την αποθήκευση του αρχείο εξόδου που περιέχει το μοντέλο που δημιουργήθηκε.

Παρακάτω, θα αναλυθεί η εκτέλεση του μοντέλου και θα παρουσιαστούν τα στάδια εκτέλεσης του κώδικα. Η αρχιτεκτονική που επιλέχθηκε για την εκτέλεση του μοντέλου είναι η Skip-gram, άρα το όρισμα sg ισούται με ένα. Το όρισμα που καθορίζει τις διαστάσεις ισούται με 300, ενώ θα γίνει μία μόνο επανάληψη με μέγεθος παραθύρου ίσο με 5. Αρχικά, πραγματοποιείται μια προ επεξεργασία και έπειτα γίνεται η εκπαίδευση του μοντέλου.

4.1.2 Εκτέλεση κώδικα και παραγωγή μοντέλων

Το αρχικό στάδιο της εκτέλεσης, αφορά όπως αναφέρθηκε και προηγουμένως την προ επεξεργασία του κειμένου. Πιο συγκεκριμένα, ο κάθε χρήστης επιλέγει τον αριθμό εμφάνισης της κάθε λέξης, προκειμένου όσες εμφανίζονται λιγότερες φορές από τον αριθμό που έχει οριστεί να απομακρύνονται. Στην συγκεκριμένη περίπτωση ο αριθμός

αυτός ισούται με δέκα. Μετά το πέρας της ανάγνωσης ολόκληρου του κειμένου, παράγεται το λεξικό από το οποίο έχουν αφαιρεθεί οι λέξεις που δεν είναι αναγκαίες για τα δεδομένα του κάθε χρήστη.



Εικόνα 23: Προ επεξεργασία μοντέλου 1

Στην προηγούμενη Εικόνα, εμφανίζεται η προ επεξεργασία του αλγορίθμου. Το αρχικό κείμενο περιέχει περίπου 58 εκατομμύρια λέξεις με 1.4 εκατομμύρια εξ' αυτών να είναι μοναδικές. Εφαρμόζοντας το κριτήριο εμφάνισης των λέξεων, μειώνονται σε αρκετά μεγάλο βαθμό οι λέξεις που εμφανίζονται μόνο μία φορά, οι οποίες θα είναι το λεξικό με το οποίο θα γίνει η εκπαίδευση του μοντέλου.

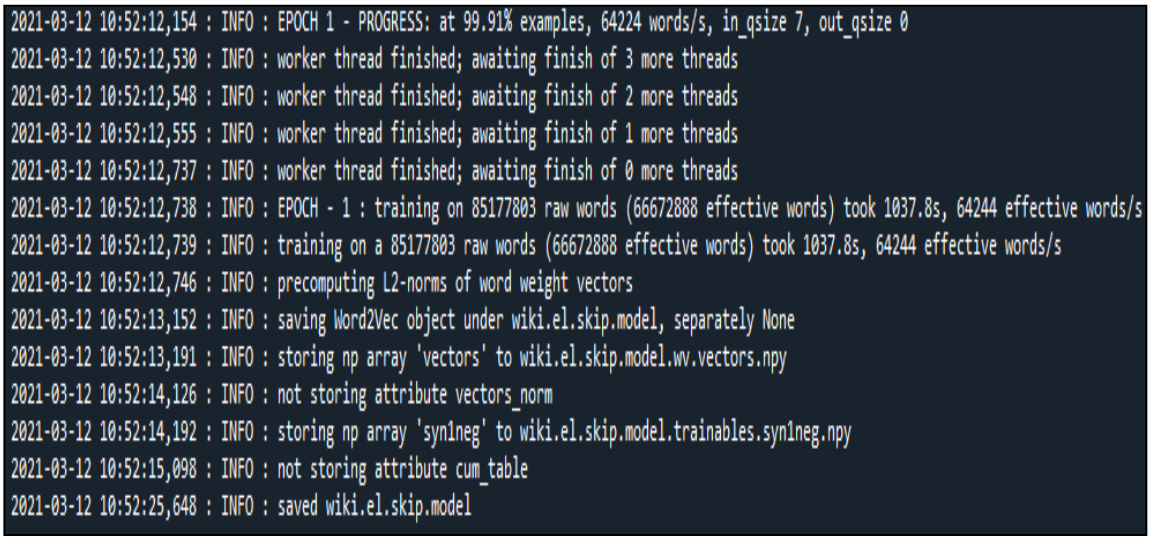
Στην Εικόνα 23, απεικονίζεται πως οι λέξεις που εμφανίζονται πάνω από δέκα φορές αποτελούν μόνο ένα 15%. Με τη μείωση αυτή, ελαχιστοποιείται σε αρκετά μεγάλο βαθμό ο όγκος που χρειάζεται το μοντέλο και βελτιώνεται η αποτελεσματικότητα του στο ποσοστό αυτό. Αξίζει να σημειωθεί, ότι παρόλο που απορρίπτεται ένα ικανό μέρος των μοναδικών λέξεων διατηρείται σχεδόν όλο το κείμενο και αυτό φαίνεται από το ποσοστό που φαίνεται και στην προηγούμενη εικόνα. Εξίσου σημαντικό είναι το γεγονός ότι στο ποσοστό αυτό, το 83% των λέξεων που απορρίπτονται είναι όροι ή και λέξεις που δεν συμπεριλαμβάνονται στην γλώσσα που χρησιμοποιεί το κείμενο. Αυτό σημαίνει ότι αποβάλλεται η πληροφορία που δεν είναι αναγκαία για την καλή λειτουργία του μοντέλου.

Ακολούθως πραγματοποιείται ακόμα μεγαλύτερη μείωση του όγκου του κειμένου αφαιρώντας τις λέξεις που εμφανίζονται πολλές φορές. Το πόσο συχνά εμφανίζονται οι λέξεις αυτές καθορίζεται από το όρισμα sample κατά τη δήλωση του Word2Vec, η προτεινόμενη τιμή για το όρισμα αυτό ισούται με 0.001 στις περισσότερες περιπτώσεις. Αυτό υποδηλώνει ότι κάθε λέξη δεν πρέπει να ξεπερνα το 0.001 του κειμένου. Στην προκειμένη περίπτωση, οι λέξεις αυτές είναι 41 και με την απόρριψη τους το κείμενο ελαχιστοποιείται κατά 30%. Τέτοιες λέξεις που εμφανίζονται τόσο συχνά δεν αποτελούν χρήσιμη πληροφορία κατά τη δημιουργία του μοντέλου διότι μπορεί να εμφανίζονται δίπλα σε αρκετές λέξεις με τις οποίες να μην έχουν καμία σχέση. Παραδείγματος χάρη η λέξη 'ή' μπορεί να έχει μεγάλη συχνότητα εμφάνισης όμως αυτό δεν σημαίνει ότι θα σχετίζεται με τις κοντινές της λέξεις.

```
2021-06-10 11:29:29,416 : INFO : deleting the raw counts dictionary of 1578848 items
2021-06-10 11:29:29,511 : INFO : sample=0.001 downsamples 30 most-common words
2021-06-10 11:29:29,512 : INFO : downsampling leaves estimated 66672672 word corpus (80.9% of prior
82436064)
2021-06-10 11:29:30,714 : INFO : estimated required memory for 245770 words and 300 dimensions:
712733000 bytes
2021-06-10 11:29:30,715 : INFO : resetting layer weights
```

Εικόνα 24: Προ επεξεργασία μοντέλου 2

Στην Εικόνα 24, απεικονίζεται το τελικό στάδιο της προ επεξεργασίας του μοντέλου. Ο συνολικός αριθμός των λέξεων ισούται με 66 εκατομμύρια και το λεξικό των λέξεων που εμφανίζονται μόνο μία φορά περιέχει 254.770 λέξεις. Για τις λέξεις που συγκεκριμένου λεξικού που δημιουργήθηκε θα παραχθεί ένα διάνυσμα. Γνωρίζοντας το πλήθος των διαστάσεων και το εύρος του λεξικού είναι εφικτό να υπολογιστεί το μέγεθος του τελικού μοντέλου.



Εικόνα 25: Τελικό στάδιο εκτέλεσης

Στην Εικόνα 25, απεικονίζεται η ολοκλήρωση της εκτέλεσης του κώδικα για την εκπαίδευση του μοντέλου. Όλη η διεργασία χρειάστηκε 1037.8 δευτερόλεπτα έχοντας μέσο όρο λέξεων ανα δευτερόλεπτο ίσο με 62.244 λέξεις. Ένας βασικός παράγοντας που καθορίζει την διάρκεια της διαδικασίας είναι ο αριθμός των επαναλήψεων που θα επιλέξει ο χρήστης. Στην προκειμένη περίπτωση, ο αριθμός των επαναλήψεων ισούται με ένα.

4.2 Λειτουργίες word2vec μοντέλου

Το μοντέλο word2vec παρέχει μεγάλη πληθώρα λειτουργιών στο χρήστη. Ορισμένες από αυτές είναι το πόσο όμοιες είναι δύο λέξεις, η εύρεση της πιο όμοιας λέξης, οι αναλογίες μεταξύ λέξεων καθώς και η αφαίρεση της λιγότερο όμοιας λέξης. Ωστόσο στην συγκεκριμένη εργασία θα μελετηθεί η λειτουργία του εντοπισμού της πιο όμοιας λέξης και θα παρουσιαστεί η υλοποίηση του στη συνέχεια.

4.2.1 Εντοπισμός πιο όμοιας λέξης

Μία σημαντική λειτουργία του word2vec όπως προαναφέρθηκε είναι η εύρεση της πιο όμοιας λέξης. Προκειμένου να επιτευχθεί αυτό γίνεται χρήση της most_similar, η οποία είναι μία μέθοδος του word2vec που προσφέρει αυτή την ικανότητα. Όπως φαίνεται και στην παρακάτω Εικόνα, αρχικά δηλώνονται τα απαραίτητα πακέτα και οι βιβλιοθήκες. Στην γραμμή 8, εισάγεται το μοντέλο στην αντίστοιχη μεταβλητή και στην γραμμή 10 στη μεταβλητή word1 εισάγεται η λέξη για την οποία θα γίνει η αναζήτηση. Στην γραμμή 12,

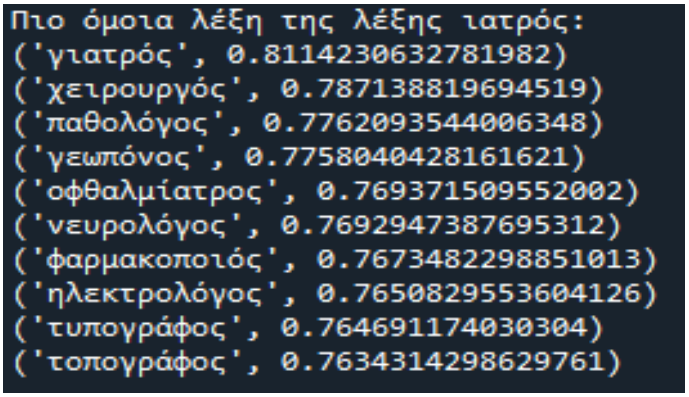
δημιουργείται ένας πίνακας που περιέχει τα αποτελέσματα της μεθόδου έχοντας σαν είσοδο την λέξη που δηλώθηκε προηγουμένως. Έπειτα, δημιουργείται για τις ανάγκες της εργασίας ένα αρχείο εξόδου με ονομασία την λέξη που δίνεται σαν είσοδος και περιέχει τα αποτελεσματα της μεθόδου. Τέλος, στις γραμμές 15-17 τυπώνεται το αποτέλεσμα της μεθόδου με τέτοιο τρόπο ώστε η πιο όμοια λέξη να βρίσκεται πρώτη.

```
1   from __future__ import unicode_literals
2   import gensim
3   import numpy as np
4
5   model = gensim.models.Word2Vec.load("wiki.el.skip.model")
6
7   word1 = 'ιατρός'
8
9   array = model.wv.most_similar([word1.lower()])
10  np.savetxt(f'result_{word1}.txt', array, fmt="%s")
11
12  print('Πιο όμοια λέξη της λέξης ' + word1 + ':')
13  for tupl in array:
14      print(tupl)
```

Εικόνα 26: Εντοπισμός πιο όμοιας λέξης

Στο παράδειγμα της Εικόνας 26, η λέξη που δόθηκε σαν είσοδος είναι η λέξη ιατρός και το αποτέλεσμα της μεθόδου έδειξε ως πιο όμοια αυτής, την λέξη γιατρός. Γεγονός το οποίο ήταν αναμενόμενο διότι οι λέξεις είναι ίδιας σημασίας και το μόνο που τις ξεχωρίζει είναι ο τρόπος γραφής τους. Οι λέξεις χειρούργος και παθολόγος είναι λέξεις που έχουν σχέση με την λέξη ιατρός.

Είναι συχνό φαινόμενο οι λέξεις που επιστρέφονται ως αποτέλεσμα της μεθόδου να μην είναι αυτές που θα περιμέναμε, διότι η αποτελεσματικότητα του μοντέλου καθορίζεται σε μεγάλο βαθμό από τα δεδομένα εισόδου. Αυτός είναι και ο λόγος που τα αποτελέσματα του μοντέλου διαφοροποιούνται σε σχέση με τα αποτελέσματα που θα έδινε ένας άνθρωπος καθώς η γνώση που διαθέτει είναι πολύ μεγαλύτερη.



Εικόνα 27: Αποτελέσματα μεθόδου

5 Διεπαφή παρουσίασης του μοντέλου

Προκειμένου το μοντέλο που παράχθηκε να μπορεί να είναι εύκολα προσβάσιμο και εύχρηστο στους χρήστες είναι ανάγκη η δημιουργία μιας εφαρμογής διαδικτύου. Μέσω αυτής, οι χρήστες θα μπορούν να κάνουν ερωτήματα και να λαμβάνουν άμεσα αποτελέσματα. Η υλοποίηση της εφαρμογής είναι σχετικά απλή αλλά καλύπτει όλες τις ανάγκες της εργασίας. Η εφαρμογή αποτελείται από δύο επίπεδα. Το πρώτο είναι η δημιουργία της εφαρμογής διαδικτύου η οποία έγινε με την γλώσσα Python και πιο συγκεκριμένα έγινε χρήση του Flask το οποίο είναι web framework της Python [33]. Προκειμένου η εφαρμογή να είναι όσο το δυνατόν πιο φιλική και όμορφη στους χρήστες έγινε ακόμα χρήση της γλώσσας Html [34], μέσω της οποίας κατασκευάζεται η διεπαφή του χρήστη και αποτελεί το δεύτερο επίπεδο της εφαρμογής.

5.1 Διακομιστής διαδικτύου

Η εφαρμογή μέσω της οποίας θα γίνεται η διαδικτυακή παρουσία της εφαρμογής θα αναπτυχθεί στη γλώσσα Python με τη χρήση του Flask το οποίο πρόκειται για ένα web framework της συγκεκριμένης γλώσσας. Το μέγεθος της εφαρμογής είναι σχετικά μικρό διότι γίνεται χρήση έτοιμων πακέτων και βιβλιοθηκών τα οποία διευκολύνουν σε μεγάλο βαθμό την δημιουργία τέτοιου είδους εφαρμογών.

Στην Εικόνα 28, παρουσιάζεται ο βασικός κώδικας της εφαρμογής διαδικτύου. Στην γραμμή 1, γίνεται εισαγωγή των απαραίτητων πακέτων. Πιο συγκεκριμένα, εισάγεται το Flask που όπως προαναφέρθηκε σε αυτό θα βασιστεί η δημιουργία της εφαρμογής, ακόμα εισάγεται το render_template το οποίο πρόκειται για ένα πακέτο μέσω του οποίου θα εισαχθούν στην συνέχεια τα δεδομένα στην διεπαφή χρήστη και το request το οποίο διαχειρίζεται τα αιτήματα POST μέσω του οποίου αποστέλλονται τα δεδομένα.

Στην γραμμή 2, εισάγεται η βιβλιοθήκη gensim που έχει αναφερθεί και προηγουμένως και θα μας βοηθήσει στην εκτέλεση του μοντέλου word2vec. Στη συνέχεια, στην μεταβλητή app δημιουργείται η εφαρμογή με τη χρήση του Flask. Στην γραμμή 6, ορίζεται στην μεταβλητή model το αρχείο εισόδου που δημιουργήθηκε προηγουμένως και περιέχει το σύνολο των κειμένων για το μοντέλο.

app.py > ...

```python
from flask import Flask , render_template, request
import gensim


app = Flask(__name__)
model = gensim.models.Word2Vec.load("wiki.el.skip.model")


@app.route('/')
def index():
    return render_template('index.html')

@app.route('/submit', methods=['POST'])
def submit():
    if request.method == 'POST':
        word = request.form['word']
        new = []
        if word == '' :
            return render_template('index.html', message='Παρακαλώ συμπληρώστε την λέξη')
        N=5;
        array = model.wv.most_similar(positive=[word.lower()],topn=N)

        labels = [row[0] for row in array]
        values = [row[1] for row in array]

        return render_template('index.html', result0 = 'Οι πιο όμοιες λέξεις είναι:',
        result1 = str(array[0][0]),  result2 = str(array[1][0]),  result3 = str(array[2][0]),
        result4 = str(array[3][0]),  result5 = str(array[4][0]),
        max=1 ,labels=labels, values=values)

if __name__ == '__main__':
    app.debug = True
    app.run()
```

Εικόνα 28: Εφαρμογή διακομιστή διαδικτύου

Στις γραμμές 9-10, στην κύρια διαδρομή της εφαρμογής επιστρέφεται η βασική μορφή της ιστοσελίδας μέσω της συνάρτησης index. Στην γραμμή 13, στην διαδρομή submit της εφαρμογής εισάγονται δεδομένα και αποστέλλονται με την μέθοδο POST. Στην γραμμή 14, ορίζεται η συνάρτηση submit στην οποία με την χρήση της request.method γίνεται έλεγχος της μεθόδου που ζητείται. Έπειτα, στην μεταβλητή word εισάγεται η λέξη που θα δώσει ο χρήστης σαν είσοδο.

Στις γραμμές 18-19, γίνεται ένας έλεγχος σε περίπτωση που ο χρήστης δεν δώσει κάποια λέξη σαν είσοδο να του επιστρέφεται ένα μήνυμα προκειμένου να του επισημάνει πως πρέπει να εισάγει μία λέξη. Στην συνέχεια, ορίζεται η μεταβλητή N η οποία στην συγκεκριμένη περίπτωση ισούται με 5 και καθορίζει των αριθμό των λέξεων που θα επιστρέψει το μοντέλο ως αποτέλεσμα. Στην γραμμή 21, δημιουργείται στην μεταβλητή array ένας πίνακας ο οποίος θα περιέχει τα αποτελέσματα του μοντέλου δηλαδή τις πέντε πιο κοινές λέξεις της λέξης που έδωσε σαν είσοδο ο χρήστης.

Στις γραμμές 23-24, τοποθετείται στην μεταβλητή labels η πρώτη σειρά του πίνακα array και στην μεταβλητή values η δεύτερη σειρά του πίνακα. Οι μεταβλητές αυτές θα χρησιμοποιηθούν στην συνέχεια για την δημιουργία διαγράμματος. Στις γραμμές 26-29, με την χρήση της μεθόδου render_template εισάγεται στην index εκεί δηλαδή όπου βρίσκεται η διεπαφή παρουσίασης του χρήστη, το σύνολο των αποτελεσμάτων που αποθηκεύτηκαν στην μεταβλητή array.

Ακολούθως στις γραμμές 31-33, γίνεται ένας εσωτερικός έλεγχος του προγράμματος για να δει αν η ενότητα αυτή κλήθηκε διαδραστικά και έπειτα καλεί την καθορισμένη συνάρτηση και την εκτέλεση του κώδικα. Δηλαδή, το app.run() θα εκτελεστεί μόνο αν η ενότητα αυτή κλήθηκε διαδραστικά και δεν έχει εισαχθεί σε άλλη μονάδα.

Τέλος, η εντολή της γραμμής 32 είναι μια παράμετρος εντοπισμού σφαλμάτων η οποία διευκολύνει τον προγραμματιστή σε περίπτωση λάθους.

5.2 Εφαρμογή διεπαφής χρήστη

Η εφαρμογή διεπαφής χρήστη έχει αναπτυχθεί με τη χρήση της γλώσσας Html και είναι σχετικά απλή. Η υλοποίηση του παρουσιάζεται στην επόμενη Εικόνα και θα αναλυθεί στην συνέχεια.

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="../static/style.css">
8      <title>Word2Vec</title>
9      <style type="text/css">
10         body {
11             background-image:url('static/Word2Vec.png');
12         ;}
13         </style>
14     <script src='https://cdnjs.cloudflare.com/ajax/libs/Chart.js/1.0.2/Chart.min.js'></script>
15 </head>
```

Εικόνα 29: Υλοποίηση διεπαφής χρήστη 1

Στην Εικόνα 29, απεικονίζεται η έναρξη της υλοποίησης της διεπαφής χρήστη. Στην

γραμμή 1, δηλώνεται ο τύπος του αρχείου που στην συγκεκριμένη περίπτωση είναι html. Στην επόμενη γραμμή, καθορίζεται η γλώσσα που είναι τα αγγλικά. Στις γραμμές 3-14, ορίζεται η επικεφαλίδα της διεπαφής χρήστη και οι διάφορες παράμετροι της.
Ορισμένες εξ' αυτών είναι η κωδικοποίηση, το όνομα και το είδος του περιεχομένου.
Στην γραμμή 7, καθορίζεται η μορφοποίηση της εφαρμογής μέσω του CSS που δεν αναλυθεί περαιτέρω διότι ξεφεύγει από τις ανάγκες της παρούσας εργασίας. Στην γραμμή 8, ορίζεται ο τίτλος της ιστοσελίδας. Επίσης στις γραμμές 9-13, ορίζεται η μορφοποίηση του βασικού κορμού της ιστοσελίδας και εισάγεται μια εικόνα ως φόντο.
Τέλος στην γραμμή 14, εισάγεται με την μορφή 'script' ένας υπερσύνδεσμος ο οποίος θα χρησιμοποιηθεί στην συνέχεια για την δημιουργία ενός διαγράμματος που θα απεικονίζει τα αποτελέσματα του μοντέλου.
Στην επόμενη Εικόνα, παρουσιάζεται η βασική δομή του κώδικα της διεπαφής χρήστη.

```
16  <body>
17      <div class="container" style="margin:0 auto" align = center>
18          <img src="../static/ex.png"  class="logo">
19          {% if message %}
20          <p class="message">{{ message | safe }}</p>
21          {% endif %}
22          <form action="/submit" method="POST">
23              <div class="form-group">
24                  <h3>Εισάγετε Λέξη</h3>
25                  <input
26                      type="text"
27                      name= "word"
28                      placeholder="Λέξη"
29                  />
30              </div>
31          <input type="submit" value="Αποστολή" class="btn" />
32          </form>
33          <form>
34              <div class="container" >
35                  <p style="color:black;">{{result0}}</p>
36                  <p style="color:black;">{{result1}}</p>
37                  <p style="color:black;">{{result2}}</p>
38                  <p style="color:black;">{{result3}}</p>
39                  <p style="color:black;">{{result4}}</p>
40                  <p style="color:black;">{{result5}}</p>
41              </div>
```

Εικόνα 30: Υλοποίηση διεπαφής χρήστη 2

Στην Εικόνα 30, αρχικά εισάγεται μία εικόνα με την μορφή logo και στη συνέχεια

δημιουργείται ένα πεδίο που θα εμφανίζει ένα μήνυμα στον χρήστη σε περίπτωση που δεν συμπληρώσει την λέξη που απαιτείται. Ακολούθως, στις γραμμές 21-32 δημιουργείται μία φόρμα τα δεδομένα της οποίας θα αποστέλλονται με την μέθοδο αιτημάτων POST στον διακομιστή διαδικτύου που δημιουργήθηκε προηγουμένως.

Στην γραμμή 23 ορίζεται ο τίτλος του πεδίου που θα δημιουργηθεί στις επόμενες γραμμές. Έπειτα δηλώνονται οι παράμετροι του πεδίου εισόδου που δημιουργείται όπως το όνομα καθώς και ο τύπος των δεδομένων που πρέπει να εισαχθούν. Στην γραμμή 30, δημιουργείται το κουμπί αποστολή το οποίο θα είναι υπεύθυνο για να αποστείλει τα δεδομένα που θα εισάγει ο χρήστης.

Τέλος, δημιουργείται μία φόρμα με πολλαπλά πεδία τα οποία εμφανίζουν στον χρήστη τα αποτελέσματα που θα επιστρέφει το μοντέλο και θα εμφανίζονται κάτω από το πεδίο που θα εισαχθούν τα δεδομένα.

```
42          </form>
43          <canvas id="chart" width="600" height="400"></canvas>
44          <script>
45            // bar chart data
46            var barData = {
47              labels : [
48                {% for item in labels %}
49                "{{ item }}",
50                {% endfor %}
51            ],
52
53              datasets : [{
54                fillColor: "rgba(151,187,205,0.2)",
55                strokeColor: "rgba(151,187,205,1)",
56                pointColor: "rgba(151,187,205,1)",
57                data : [
58                  {% for item in values %}
59                    "{{ item }}",
60                  {% endfor %}
61              ]
62                }
63              ]
64            }
```

Εικόνα 31: Υλοποίηση διεπαφής χρήστη 3

Στην Εικόνα 31, δημιουργείται με τη μορφή ‘canvas’ ένα πεδίο στην ιστοσελίδα στο οποίο θα τοποθετηθεί ένα διάγραμμα που θα απεικονίζει τα αποτελέσματα του μοντέλου. Στις επόμενες γραμμές εισάγονται στην μεταβλητή labels οι λέξεις που δίνονται σαν

αποτέλεσμα και θα χρησιμοποιηθούν σαν πεδία στον άξονα χ του διαγράμματος. Στις γραμμές 53-56, ορίζεται η μεταβλητή datasets η οποία περιέχει την μορφοποίηση του διαγράμματος και πιο συγκεκριμένα τα χρώματα του. Τέλος, στην μεταβλητή data εισάγονται τα διανύσματα των λέξεων που δόθηκαν σαν αποτέλεσμα του μοντέλου και θα χρησιμοποιηθούν σαν τιμές στον άξονα y.

Στην Εικόνα 32, δημιουργείται το διάγραμμα με τη μορφή μπάρας. Στις γραμμές 69-70, ορίζονται τα όρια του διαγράμματος που στην συγκεκριμένη περίπτωση θα ισούται με 1 καθώς δεν μπορεί να ξεπεραστεί αυτή η τιμή. Έπειτα στις γραμμές 72-84, ορίζονται διάφορες παράμετροι του διαγράμματος που δεν αναλυθούν διότι δεν αποτελούν αντικείμενο της παρούσας εργασίας.

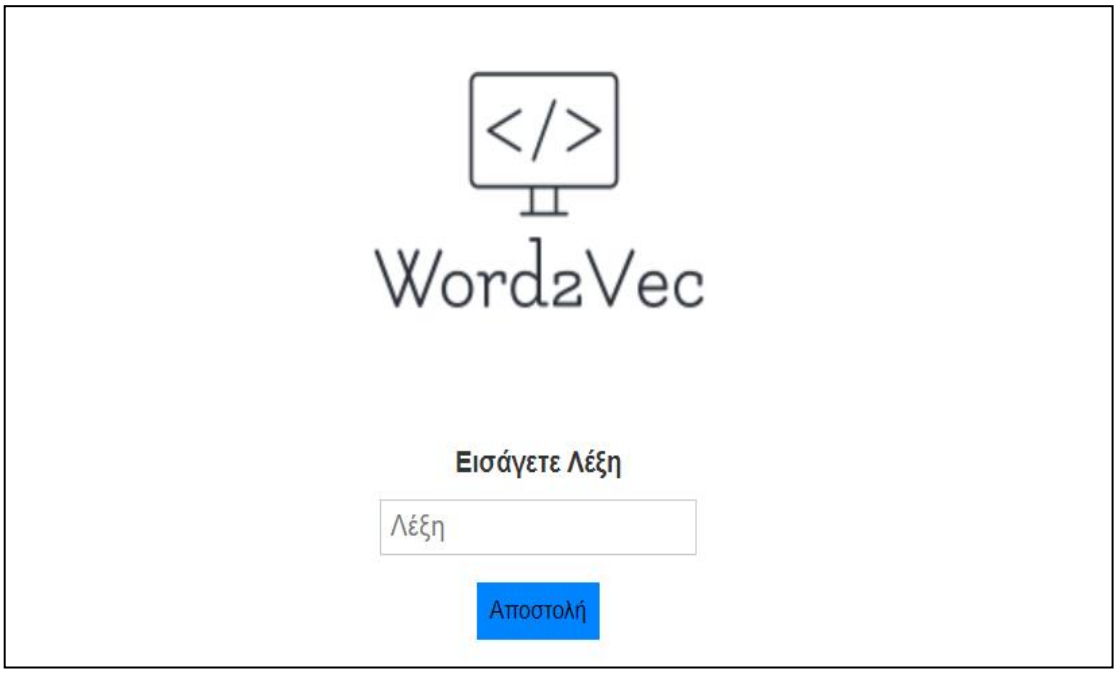
```
66          // get bar chart canvas
67          var mychart = document.getElementById("chart").getContext("2d");
68
69            steps = 1
70            max = {{ max }}
71
72          // draw bar chart
73          new Chart(mychart).Bar(barData, {
74            scaleOverride: true,
75            scaleSteps: steps,
76            scaleStepWidth: Math.ceil(max / steps),
77            scaleStartValue: 0,
78            scaleShowVerticalLines: true,
79            scaleShowGridLines : true,
80            barShowStroke : true,
81            scaleShowLabels: true
82            }
83          );
84
85        </script>
```

Εικόνα 32:Υλοποίηση διεπαφής χρήστη 4

5.3.1 Παρουσίαση διεπαφής χρήστη

Η εφαρμογή που δημιουργήθηκε εκτελεί τις λειτουργίες της τοπικά χωρίς να βρίσκεται σε κάποια διαδικτυακή διεύθυνση διότι δεν αποτελεί αντικείμενο της παρούσας έρευνας. Η εφαρμογή διεπαφής χρήστη προσφέρει πρόσβαση στην λειτουργία εύρεσης των πιο όμοιων

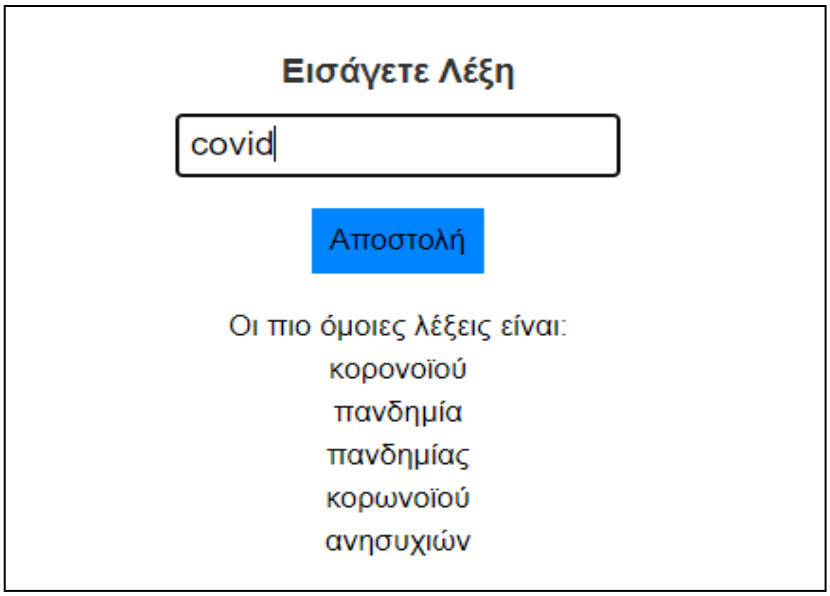
λέξεων που παρουσιάστηκε στο κεφάλαιο 4.2.1.



Εικόνα 33: Παρουσίαση ιστοσελίδας 1

Ο χρήστης μπορεί να εκτελέσει ερωτήματα στο μοντέλο μέσω της ιστοσελίδας.

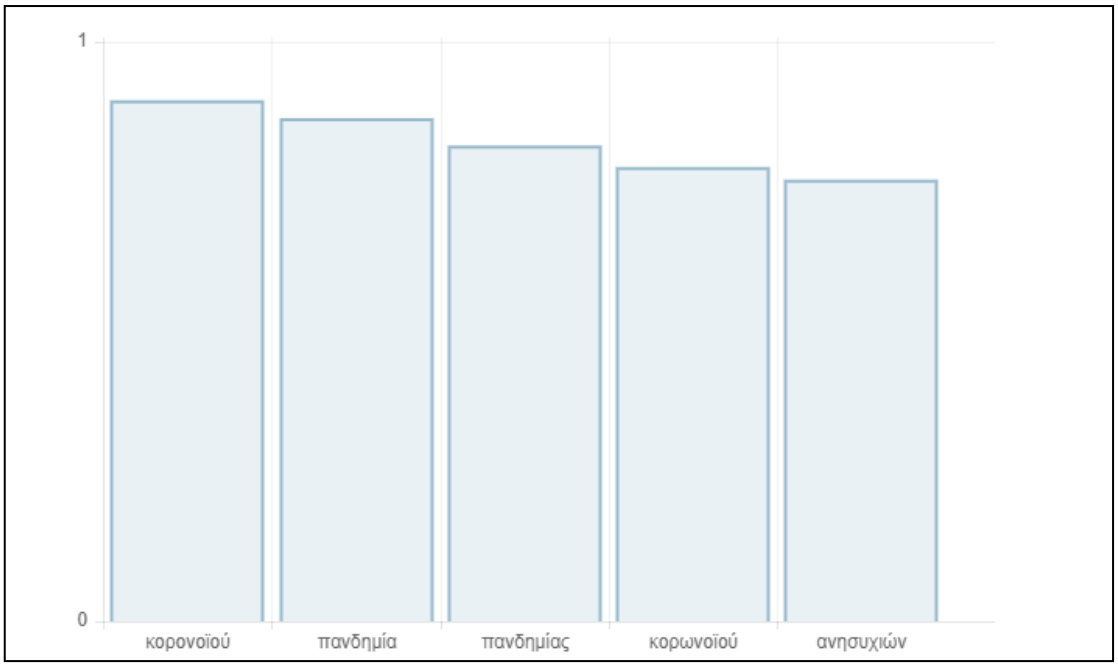
Πιο συγκεκριμένα όπως φαίνεται και στην Εικόνα 33, μπορεί να εισάγει στο πεδίο αναζήτησης την λέξη για την οποία επιθυμεί να βρεί τις πιο όμοιες της.



Εικόνα 34: Παρουσίαση ιστοσελίδας 2

Στην Εικόνα 34, απεικονίζεται ένα παράδειγμα εκτέλεσης της λειτουργίας της ιστοσελίδας.

Όπως φαίνεται και στο παράδειγμα, μετά το πάτημα του κουμπιού ‘Αποστολή’ εμφανίζονται ακριβώς από κάτω οι πέντε πιο όμοιες λέξεις για την λέξη που πραγματοποιήθηκε η αναζήτηση. Ακόμη όπως παρουσιάζεται και στην ακόλουθη Εικόνα, δημιουργείται και ένα διάγραμμα το οποίο απεικονίζει τα αποτελέσματα με την μορφή ραβδογράμματος.



Εικόνα 35: Παρουσίαση ιστοσελίδας 3

Το διάγραμμα της Εικόνας 35, παρουσιάζει τα αποτελέσματα της αναζήτησης και πιο συγκεκριμένα στον άξονα των χ έχουν τοποθετηθεί οι πέντε πιο όμοιες λέξεις που επέστρεψε το μοντέλο και στον άξονα των y έχουν τοποθετηθεί οι τιμές των διανυσμάτων της κάθε λέξης.

6 Αξιολόγηση

6.1 Διαδικασία αξιολόγησης

Μετά το πέρας της εκτέλεσης του μοντέλου επιτακτική ανάγκη αποτελεί η αξιολόγηση των αποτελεσμάτων του προκειμένου να ελεγχθεί η απόδοση του. Η αξιολόγηση δίνει στον χρήστη την δυνατότητα να εκτιμήσει αν η επιλογές του κατά την δημιουργία του μοντέλου ήταν οι καλύτερες δυνατές. Παραδείγματος χάρη, μετά το πέρας της αξιολόγησης πραγματοποιώντας πειράματα και χρησιμοποιώντας τα σωστά για την εκάστοτε περίπτωση κριτήρια, ο χρήστης μπορεί να διαπιστώσει ότι για το συγκεκριμένο μοντέλο η αρχιτεκτονική CBOW είναι πιο αποδοτική σε σχέση με την αρχιτεκτονική Skip-gram. Έπειτα θα πραγματοποιηθεί η αξιολόγηση του μοντέλου που δημιουργήθηκε προηγουμένως και θα εξεταστεί το αποτέλεσμα της.

Το είδος τη αξιολόγησης που υλοποιείται είναι εσωτερική αξιολόγηση διότι η χρήση του μοντέλου δεν έχει συγκεκριμένο σκοπό. Στη συνέχεια, θα παραχθούν κριτήρια αξιολόγησης για την μέθοδο της εύρεσης της πιο κοινής λέξης που δημιουργήθηκε σε προηγούμενο κεφάλαιο. Η υλοποίηση της διαδικασίας αξιολόγησης θα γίνει στην γλώσσα Python με την χρήση της βιβλιοθήκης gensim που χρησιμοποιήθηκε και κατά την δημιουργία του μοντέλου.

Η ερευνητική διαδικασία πραγματοποιήθηκε χρησιμοποιώντας δεδομένα μεγέθους 300mb και η εκτέλεση της διαδικασίας πραγματοποιήθηκε μόνο μία φορά. Τα χαρακτηριστικά του υπολογιστή στον οποίο υλοποιήθηκε η ερευνητική διαδικασία είναι τα ακόλουθα:

- Επεξεργαστής: Intel(R) Core(TM) i3-4030U CPU @ 1.90GHz 1.90 GHz
- Μνήμη Ram: 4GB
- Κάρτα γραφικών: Intel HD Graphics Family
- Λειτουργικό Σύστημα: Windows 10

6.2 Παράμετροι αξιολόγησης

Προκειμένου να γίνει η αξιολόγηση του μοντέλου, δημιουργήθηκε ένα αρχείο κειμένου το οποίο περιέχει έτοιμα ερωτήματα στην κατηγορία των αναλογιών. Το αρχείο αυτό θα αποτελέσει την είσοδο κατά την διαδικασία της αξιολόγησης. Το αρχείο που δημιουργήθηκε περιέχει πληροφορία από το αρχείο που παρέχει ελεύθερα σε όλους τους χρήστες η Google στην αρχική υλοποίηση της για το word2vec.

Το αρχείο που παρέχει η Google περιλαμβάνει δεδομένα για δεκατέσσερις κατηγορίες ερωτημάτων, τα οποίο είναι περίπου 20.000. Τα δεδομένα αυτά περιέχουν δεδομένα για πολλούς τομείς όπως κατηγορίες που αφορούν την γραμματική, πρωτεύουσες χωρών και πολλά άλλα. Πιο αναλυτικά οι κατηγορίες είναι οι ακόλουθες:

- Capital-common-countries

Περιέχει την πρωτεύουσα κάθε χώρας.

- Capital-world

Είναι παρόμοια με την κατηγορία που προαναφέρθηκε αλλά με μεγαλύτερο όγκο χωρών.

- Currency

Αντιστοίχιση της κάθε χώρας με το νόμισμα της.

- City-in-state

Αφορά τις πόλεις και τις πολιτείες των Ηνωμένων Πολιτειών

- Family

Περιλαμβάνει οικογενειακές σχέσεις.

- Gram 1-adjective-to-adverb

Περιέχει γραμματική και αντιστοίχιση επιρρημάτων με επίθετα.

- Gram 2-opposite

Περιέχει στοιχεία γραμματικής καθώς και συνδυασμούς αντιθέτων.

- Gram 3-comparative

Γραμματική, συνδυασμοί θετικού-συγκριτικού βαθμού.

- Gram 4-superlative

Αφορά την γραμματική και αντιστοίχιση θετικού και υπερθετικού βαθμού.

- Gram 5-present-participle

Αφορά την γραμματική και αντιστοίχιση θετικού και υπερθετικού βαθμού.

- Gram 6-nationality-adjective

Αφορά την γραμματική και την αντιστοίχιση χωρών και εθνικοτήτων.

- Gram 7- past-tense

Αφορά την γραμματική σε παρελθοντικό χρόνο

- Gram 8-plural

Σχετίζεται με την γραμματική και τον συνδυασμό ουσιαστικών στον πληθυντικό τους.

- Gram 9-plural-verbs

Αφορα την γραμματική και τον συνδυασμό ρημάτων στον πληθυντικό τους.

Στην συγκεκριμένη περίπτωση τους στόχους της παρούσας αξιολόγησης καλύπτουν μερικές κατηγορίες από αυτές που προαναφέρθηκαν. Πιο συγκεκριμένα, θα

χρησιμοποιηθούν οι κατηγορίες capital-common-countries, family, gram6-nationality-adjective, gram8-plural. Τα ερωτήματα που επιλέχθηκαν μεταφράστηκαν στα ελληνικά.

6.3 Υλοποίηση αξιολόγησης

Η αξιολόγηση θα εκτελεστεί με την βοήθεια της βιβλιοθήκης gensim που έχει χρησιμοποιηθεί και κατά την παραγωγή του μοντέλου προηγουμένως καθώς και της μεθόδου accuracy. Ο κώδικας που υλοποιεί την αξιολόγηση απεικονίζεται στην Εικόνα 36 και θα επεξηγηθεί στην συνέχεια.

Οπως παρουσίαζεται και στην ακόλουθη Εικόνα, στις δύο πρώτες γραμμές εισάγονται τα απαραίτητα πακέτα και η βιβλιοθήκη που θα χρησιμοποιηθεί για την υλοποίηση. Στην γραμμή 4, εισάγεται στην μεταβλητή model το μοντέλο word2vec που δημιουργήθηκε σε προηγούμενο κεφάλαιο και στην γραμμή 5 εισάγεται στην μεταβλητή questions το αρχείο με τα ερωτήματα που δημιουργήθηκε. Στην γραμμή 7, εκτελείται η αξιολόγηση για τα δεδομένα που δόθηκαν σαν είσοδο και τοποθετείται στην μεταβλητή result.

Τα ερωτήματα που χρησιμοποιήθηκαν σαν είσοδο αποτελούνται από δεδομένα της μορφής: ‘Πεκίνο Κίνα Βέρνη Ελβετία’. Οι πρώτες τρεις λέξεις είναι αυτές που καθορίζουν το ερώτημα που θα υλοποιηθεί ενώ η τελευταία πρόκειται για την αναμενόμενη απάντηση.

```
1   from __future__ import unicode_literals
2   import gensim
3
4   model = gensim.models.Word2Vec.load("wiki.el.skip.model")
5   questions = "questions.txt"
6
7   result =  model.wv.accuracy(questions)
8
9   for el in result:
10      corrects = len(el['correct'])
11      wrongs = len(el['incorrect'])
12      section = el['section']
13      ratio = 'Not available'
14      if corrects+wrongs > 0:
15          ratio = str(corrects*100/(corrects+wrongs)) + '%'
16      print ("Success rate: " + str(ratio) + " | Section: " +
17          section + ", Tests: " + str(corrects+wrongs))
```

Εικόνα 36: Αξιολόγηση μοντέλου

Στις γραμμές 9-17 της Εικόνας 36, υπολογίζεται το ποσοστό επιτυχίας για τις κατηγορίες που επιλέχθηκαν και τυπώνεται το αποτέλεσμα στο χρήστη. Στην γραμμή 7, τοποθετείται

στην result ένας πίνακας ο οποίος θα περιέχει για όλα τα ερωτήματα και για την κάθε κατηγορία ξεχωριστά ένα στοιχείο. Στις επόμενες γραμμές ορίζεται μία μέθοδος επανάληψης μέσα στην οποία ορίζονται τέσσερις διαφορετικές μεταβλητές στις οποίες αποθηκεύεται ένα πεδίο. Πιο συγκεκριμένα το πεδίο της γραμμής 12 περιέχει το όνομα της κατηγορίας των ερωτημάτων, το πεδίο της γραμμής 10 αποθηκεύονται τα ερωτήματα που επεξεργάστηκαν με επιτυχία και άλλο ένα πεδίο γι αυτά που απέτυχαν. Στην γραμμή 15, γίνεται υπολογισμός του ποσοστού επιτυχίας για το μοντέλο σε κάθε κατηγορία και έπειτα εμφανίζονται τα αποτελέσματα στην οθόνη.

Αξίζει να σημειωθεί ότι το αποτέλεσμα της αξιολόγησης σχετίζονται αποκλειστικά με τα ερωτήματα που οι λέξεις του βρέθηκαν στο λεξικό. Εφόσον το μοντέλο εκπαιδεύτηκε με δεδομένα που δεν έχουν μεγάλο μέγεθος, είναι αναμενόμενο αρκετές λέξεις να μην έχουν εμφανιστεί πολλές φορές προκειμένου να δημιουργηθούν διανύσματα για αυτές. Τέτοιου είδους ερωτήματα παραλείπονται και δεν λαμβάνονται υπόψη κατά τη διάρκεια της αξιολόγησης.

```
  result =  model.wv.accuracy(questions)
Success rate: 39.29618768328446% | Section: κοινές χώρες, Tests: 341
Success rate: 20.3125% | Section: κόσμος κεφαλαίου, Tests: 192
Success rate: 15.555555555555555% | Section: family, Tests: 90
Success rate: 31.41891891891892% | Section: gram6-nationality-adjective, Tests: 296
Success rate: 30.46789989118607% | Section: total, Tests: 919
```

Εικόνα 37: Αποτελέσματα αξιολόγησης στα ελληνικά

Στην Εικόνα 37, απεικονίζονται τα αποτελέσματα της αξιολόγησης για τα ερωτήματα που έχουν μεταφραστεί στην ελληνική γλώσσα. Μετά το πέρας της αξιολόγησης τυπώνεται και ο αριθμός των ερωτημάτων που εκτελέστηκαν, στην συγκεκριμένη περίπτωση ήταν 919 από ένα σύνολο περίπου 2000 ερωτημάτων. Ωστόσο, αν και εκτελέστηκε πολύ μικρότερος αριθμός ερωτημάτων είναι αρκετά προκειμένου να μπορούν να αξιολογηθούν τα αποτελέσματα. Το ποσοστό επιτυχίας του μοντέλου για τις κατηγορίες ερωτημάτων που επιλέχθηκαν είναι από 39,2% μέχρι 15,5% και ο μέσος όρος των αποτελεσμάτων ισούται με 30,4%.

```
result =  model.wv.accuracy(questions)
Success rate: 22.22222222222222% | Section: capital-common-countries, Tests: 72
Success rate: 18.840579710144926% | Section: capital-world, Tests: 69
Success rate: 15.555555555555555% | Section: family, Tests: 90
Success rate: 31.41891891891892% | Section: gram6-nationality-adjective, Tests: 296
Success rate: 25.806451612903224% | Section: total, Tests: 527
```

Εικόνα 38: Αποτελέσματα αξιολόγησης στα αγγλικά

Στην Εικόνα 38, απεικονίζονται τα αποτελέσματα της αξιολόγησης χρησιμοποιώντας σαν είσοδο τα ερωτήματα που δημιουργήθηκαν προηγουμένως χωρίς όμως να μεταφραστούν στα ελληνικά. Εδώ ο αριθμός των ερωτημάτων που εκτελέστηκαν είναι 527 και το ποσοστό επιτυχίας για τις τέσσερις κατηγορίες κυμαίνεται από 15,5-31%. Ο τελικός μέσος όρος ισούται με 25% και είναι προφανές ότι το ποσοστό επιτυχίας είναι πολύ μικρότερο σε σχέση με το προηγούμενο. Ωστόσο ήταν αναμενόμενο διότι το μοντέλο κατασκευάστηκε για την ελληνική γλώσσα.

Έπειτα από την εκτέλεση της αξιολόγησης τόσο για τα ελληνικά ερωτήματα όσο και για τα αγγλικά ερωτήματα, φαίνεται πως το ποσοστό επιτυχίας δεν είναι ιδιαίτερα υψηλό και αυτό οφείλεται στο μικρό όγκο των δεδομένων που χρησιμοποιήθηκαν σαν είσοδο κατά την εκπαίδευση του μοντέλου. Όπως φαίνεται και στην ακόλουθη Εικόνα, το μοντέλο word2vec της Google έχει ποσοστό επιτυχίας ίσο με 77%, αυτό συμβαίνει διότι το μέγεθος των δεδομένων που περιλαμβάνει αγγίζει περίπου τα 100 δισεκατομμύρια λέξεις.

```
Output -> Success rate: 83% | Section: capital-common-countries
          Success rate: 82% | Section: capital-world
          Success rate: 39% | Section: currency
          Success rate: 74% | Section: city-in-state
          Success rate: 90% | Section: family
          Success rate: 32% | Section: gram1-adjective-to-adverb
          Success rate: 50% | Section: gram2-opposite
          Success rate: 91% | Section: gram3-comparative
          Success rate: 88% | Section: gram4-superlative
          Success rate: 79% | Section: gram5-present-participle
          Success rate: 97% | Section: gram6-nationality-adjective
          Success rate: 66% | Section: gram7-past-tense
          Success rate: 85% | Section: gram8-plural
          Success rate: 68% | Section: gram9-plural-verbs
          Success rate: 77% | Section: total
```

Εικόνα 39: Αποτελέσματα αξιολόγησης Google

6.4 Πειράματα αξιολόγησης

Έπειτα, θα αναλυθεί η αποτελεσματικότητα των μοντέλων αλλάζοντας διάφορες παραμέτρους κατά την εκπαίδευση τους. Θα κατασκευαστούν μοντέλα και στις δύο αρχιτεκτονικές που υπάρχουν. Τα πειράματα που θα γίνουν θα μεταβάλλουν το παράθυρο αναζήτησης και το μέγεθος των διανυσμάτων. Στην επόμενη Εικόνα, παρουσιάζεται η δημιουργία μοντέλων με μεγέθη διανυσμάτων 100,200 και 300 και το παράθυρο αναζήτησης θα παίρνει τιμές ίσες με 5,7,9 και 11. Επίσης θα γίνει και αξιολόγηση των μοντέλων αυτών για να βρεθεί το ποσοστό επιτυχίας τους.

```
1   from __future__ import unicode_literals
2
3
4   from gensim.models import Word2Vec
5   from gensim.models.word2vec import LineSentence
6
7   questions = "questions-words.txt"
8   inp = "wiki.el.text"
9   lines = LineSentence(inp)
10  skip0rCBOW = ['CBOW', 'SKIP']
11
12  results = open("resultsGreek.txt", 'w')
13
14  for skip in range(0,2):
15      for dim in range (100, 301, 100):
16          for win in range(5, 12, 2):
17              results.write("wiki.el." + str(dim) + skip0rCBOW[skip] +str(win) + "\n")
18
19              model = Word2Vec(lines, sg=skip, size=dim, iter=3, window=win, min_count=5, workers=3)
20
21              model.init_sims(replace=True)
22
23              result =  model.wv.accuracy(questions)
24
25              for el in result:
26                  corrects = len(el['correct'])
27                  wrongs = len(el['incorrect'])
28                  section = el['section']
29                  ratio = 'Not available'
30                  if corrects+wrongs > 0:
31                      ratio = str(corrects*100/(corrects+wrongs)) + '%'
32                  print ("Success rate: " + str(ratio) + " | Section: " +
33                      section + ", Tests: " + str(corrects+wrongs))
34
35              results.write('\n')
36  results.close()
```

Εικόνα 40: Πειράματα μοντέλου

Στις γραμμές 1-5, γίνεται εισαγωγή των απαραίτητων πακέτων και βιβλιοθηκών. Στις γραμμές 7 και 8, εισάγεται στην μεταβλητή questions το αρχείο που περιέχει τα ερωτήματα και στην μεταβλητή inp το αρχείο με τα δεδομένα που χρησιμοποιήθηκε και στο προηγούμενο κεφάλαιο για την δημιουργία του μοντέλου. Στην επόμενη γραμμή, γίνεται

χρήση της απαραίτητης μεθόδου και εισάγονται σε μεταβλητή οι γραμμές ολόκληρου του κειμένου. Η γραμμή 10 σχετίζεται με τα ονόματα που θα δοθούν στα αποτελέσματα μετά την εκτέλεση. Στην γραμμή 12, γίνεται άνοιγμα του αρχείου στο οποίο θα τοποθετηθούν τα αποτελέσματα.

Στις γραμμές 14-16, υπάρχουν τρεις μέθοδοι επανάληψης οι οποίες είναι υπεύθυνες για να αλλάζουν τις παραμέτρους του μοντέλου. Στην γραμμή 21, κατασκευάζεται στην μεταβλητή model ένα μοντέλο με τις ήδη υπάρχουσες παραμέτρους. Στις γραμμές 25-33, γίνεται η αξιολόγηση του μοντέλου όπως έγινε και προηγουμένως μόνο που τώρα τα αποτελέσματα δεν τοποθετούνται στην κονσόλα αλλά αποθηκεύονται στο αρχείο που δημιουργήθηκε στην αρχή.

Μετά το πέρας της εκτέλεσης θα έχουν κατασκευαστεί 24 μοντέλα για τις παραμέτρους που αναφέρθηκαν προηγουμένως και θα έχουν αξιολογηθεί.

Το αρχείο που δημιουργείται περιέχει τις πληροφορίες για κάθε μοντέλο που αξιολογήθηκε, ένα τέτοιο παράδειγμα εμφανίζεται στην ακόλουθη Εικόνα. Όσον αφορά την ονομασία του κάθε μοντέλου, για παράδειγμα για το πρώτο μοντέλο της Εικόνας 41 το 200 καθορίζει τον αριθμό των διανυσμάτων, το SKIP καθορίζει την αρχιτεκτονική και ο αριθμός 5 καθορίζει το μέγεθος του παραθύρου αναζήτησης. Έπειτα, για όλα τα μοντέλα αναγράφεται στο αρχείο εξόδου το ποσοστό επιτυχίας του για όλες τις κατηγορίες ερωτημάτων.

```
wiki.el.200SKIP5
Success rate: 44.44444444444444% | Section: capital-common-countries, Tests: 72
Success rate: 37.68115942028985% | Section: capital-world, Tests: 69
Success rate: 27.77777777777778% | Section: family, Tests: 90
Success rate: 58.78378378378378% | Section: gram6-nationality-adjective, Tests: 296
Success rate: 48.76660341555977% | Section: total, Tests: 527
wiki.el.200SKIP7
Success rate: 45.833333333333336% | Section: capital-common-countries, Tests: 72
Success rate: 39.130434782608695% | Section: capital-world, Tests: 69
Success rate: 27.77777777777778% | Section: family, Tests: 90
Success rate: 64.1891891891892% | Section: gram6-nationality-adjective, Tests: 296
Success rate: 52.182163187855785% | Section: total, Tests: 527
wiki.el.200SKIP9
Success rate: 62.5% | Section: capital-common-countries, Tests: 72
Success rate: 55.072463768115945% | Section: capital-world, Tests: 69
Success rate: 28.88888888888889% | Section: family, Tests: 90
Success rate: 66.89189189189189% | Section: gram6-nationality-adjective, Tests: 296
Success rate: 58.25426944971537% | Section: total, Tests: 527
```

Εικόνα 41: Αποτέλεσμα πειραμάτων

Στην επόμενη ενότητα, προκειμένου να παρουσιαστούν και να αναλυθούν τα αποτελέσματα της αξιολόγησης του μοντέλου θα γίνει χρήση μόνο του συνολικού ποσοστού επιτυχίας και όχι της κάθε κατηγορίας ξεχωριστά.

6.5 Αποτελέσματα αξιολόγησης πειραμάτων

Η παρουσίαση των αποτελεσμάτων της αξιολόγησης θα αναπαρασταθούν με την μορφή πινάκων. Πιο συγκεκριμένα, θα δημιουργηθεί ένας πίνακας για κάθε μία αρχιτεκτονική που θα περιέχει τις τιμές όλες τις παραμέτρους που ορίστηκαν προηγουμένως.

Window Size / Vector Size	5	7	9	11
100	40%	44%	49%	53%
200	48%	52%	58%	58%
300	38%	52%	53%	56%

Πίνακας 1: Απεικόνιση αποτελεσμάτων αξιολόγησης Skip-Gram

Στον Πίνακα 1, απεικονίζονται τα αποτελέσματα της αξιολόγησης με την αρχιτεκτονική Skip-gram. Στον άξονα των y οι τιμές 100,200 και 300 αντιπροσωπεύουν το μέγεθος των διανυσμάτων, ενώ στο άξονα των χ οι τιμές 5,7,9 και 11 αντιπροσωπεύουν το μέγεθος του παραθύρου. Τα ποσοστά επιτυχίας κυμαίνονται από 38% έως 58%. Παρατηρείται ότι τα ποσοστά επιτυχίας είναι αρκετά υψηλά και σημειώνεται αύξηση του ποσοστού με την μεταβολή των διαστάσεων και του μεγέθους του παραθύρου. Αυτό σημαίνει ότι τα αποτελέσματα συνδέονται σε μεγάλο βαθμό με τις παραμέτρους εκπαίδευσης, πιο συγκεκριμένα όσο αυξάνεται το μέγεθος των διανυσμάτων και το μέγεθος του παραθύρου επιτυγχάνονται μεγαλύτερα ποσοστά αξιολόγησης.

Window Size / Vector Size	5	7	9	11
100	22%	27%	31%	35%
200	29%	29%	37%	38%
300	27%	35%	37%	40%

Πίνακας 2: Απεικόνιση αποτελεσμάτων αξιολόγησης CBOW

Στον Πίνακα 2, απεικονίζονται τα αποτελέσματα της αξιολόγησης με την χρήση της αρχιτεκτονικής CBOW. Στην συγκεκριμένη περίπτωση, τα ποσοστά επιτυχίας ξεκινούν από 22% και φτάνουν μέχρι 40%. Όπως φαίνεται και σε αυτή την αρχιτεκτονική υπάρχει μεταβολή των ποσοστών επιτυχίας της αξιολόγησης. Παρατηρείται, αν και σε μικρότερο βαθμό σε σχέση με την αρχιτεκτονική Skip-gram υπάρχει και εδώ αύξηση των ποσοστών επιτυχίας. Αξίζει να σημειωθεί πως και με την αρχιτεκτονική CBOW η αποτελεσματικότητα της αξιολόγησης μεταβάλλεται καθώς μεταβάλλονται και οι παράμετροι του μοντέλου.

7 Επίλογος

7.1 Συμπεράσματα

Μεγάλη ανάπτυξη τις τελευταίες δεκαετίες έχει παρατηρηθεί στον τομέα της επεξεργασίας φυσικής γλώσσας. Στις μέρες μας βρίσκει πολλές εφαρμογές σε διάφορους τομείς και διαρκώς αυξάνεται. Προκειμένου να μπορούν να υλοποιηθούν αυτές οι εφαρμογές καθώς και όσες παρουσιαστούν μελλοντικά αποτελεί επιτακτική ανάγκη η βέλτιστη χρήση της επεξεργασία φυσικής γλώσσας.

Για να υλοποιηθεί η επεξεργασία φυσικής γλώσσας παράγονται μοντέλα γλωσσών στα οποία βρίσκονται τα δεδομένα για ολόκληρη την γλώσσα. Με την χρήση των νευρωνικών δικτύων και διαφόρων διαδικασιών που αυτά προσφέρουν, γίνεται η εκπαίδευση των μοντέλων προκειμένου να επιτύχουν το επιθυμητό αποτέλεσμα. Η αποτελεσματικότητα των μοντέλων εξαρτάται από πολλούς παράγοντες. Για παράδειγμα, ιδιαίτερα σημαντική είναι η ποιότητα και η ποσότητα των δεδομένων που θα δοθούν σαν είσοδος κατά την εκπαίδευση. Επίσης εξίσου σημαντικό ρόλο παίζει και η σωστή επιλογή του αλγορίθμου που θα αναλάβει την διεργασία.

Με την χρήση των νευρωνικών δικτύων έχουν αναπτυχθεί αρκετές κατηγορίες εφαρμογών. Τέτοιες κατηγορίες είναι η επεξεργασία φυσικής γλώσσας, η αναγνώριση προτύπων και η αναγνώριση ομιλίας. Ωστόσο στις κατηγορίες που προαναφέρθηκαν δεν υπήρχαν σημαντικά αποτελέσματα στο παρελθόν. Η συγκεκριμένα εργασία εστιάζει στην επεξεργασία φυσικής γλώσσας.

Η επεξεργασία φυσικής γλώσσας περιέχει πολλές δυσκολίες και αυτό οφείλεται στην δομή της. Στην φυσική γλώσσα το μεγαλύτερο μέρος της απαραίτητης πληροφορίας προκειμένου να επιτευχθεί η επικοινωνία παραλείπεται γεγονός το οποίο καθιστά αρκετά δύσκολο να κατανοηθεί από τον υπολογιστή. Στην συγκεκριμένη μελέτη η παραγωγή και η εκπαίδευση του μοντέλου θα γίνει με τον ίδιο τρόπο που θα εκπαιδευόταν στην γλώσσα και ένας άνθρωπος.

Σε τέτοια μοντέλα είναι αρκετά μεγάλα σε μέγεθος τα κείμενα, τα οποία πρέπει να κατασκευάσουν διανύσματα για να αναπαραστήσουν την κάθε λέξη του κειμένου. Η δημιουργία των διανυσμάτων γίνονται με βάση το γεγονός ότι λέξεις που έχουν παρόμοια σημασία πρέπει να έχουν κοντινά διανύσματα. Υπάρχουν αρκετοί αλγόριθμοι ικανοί να ικανοποιήσουν τα δεδομένα που προαναφέρθηκαν με τον επικρατέστερο να είναι ο word2vec.

Ο word2vec πρόκειται για ένα αλγόριθμο της Google ο οποίος παράγει διανύσματα λέξεων. Λειτουργεί χρησιμοποιώντας ως δεδομένα εισόδου ένα σύνολο κειμένων και κατασκευάζει ένα διάνυσμα για κάθε μία λέξη του κειμένου. Το μοντέλο word2vec παρέχει μεγάλη πληθώρα λειτουργιών στο χρήστη. Ορισμένες από αυτές είναι η ομοιότητα μεταξύ δύο λέξεων, ο εντοπισμός της πιο όμοιας λέξης, οι αναλογίες μεταξύ λέξεων καθώς και ο αποκλεισμός της λιγότερο όμοιας λέξης. Ωστόσο στην συγκεκριμένη εργασία μελετήθηκε η λειτουργία του εντοπισμού των πιο όμοιων λέξεων.

Στην παρούσα μελέτη κατασκευάστηκε μοντέλο για την ελληνική γλώσσα με την χρήση του αλγορίθμου word2vec. Τα δεδομένα εισόδου για την εκπαίδευση του μοντέλου ήταν από τη Wikipedia. Αναλύθηκε ολόκληρη η διαδικασία της εκπαίδευσης και κατασκευής του μοντέλου. Ακόμα, κατασκευάστηκε και μία εφαρμογή διαδικτύου που δίνει πρόσβαση στο χρήστη να κάνει ερωτήματα στο μοντέλο στην ελληνική γλώσσα.

Ιδιαίτερα σημαντική είναι και η αξιολόγηση του μοντέλου που δημιουργήθηκε. Αν δεν γίνει αξιολόγηση δεν είναι δυνατό να υπολογισθεί με ακρίβεια το ποσοστό επιτυχίας του μοντέλου. Η αξιολόγηση μπορεί να διακριθεί σε εσωτερική και εξωτερική. Στην εσωτερική αξιολόγηση γίνεται έλεγχος του μοντέλου με ερωτήματα για να αξιολογηθεί το ποσοτό επιτυχίας που αφορά την διανυσματική αναπαράσταση των λέξεων. Στην εξωτερική αξιολόγηση γίνεται χρήση του μοντέλου για μία διεργασία και ελέγχεται η απόδοση του σε αυτή.

Στην παρούσα μελέτη έγινε χρήση της εσωτερικής αξιολόγησης χρησιμοποιώντας την μέθοδο των αναλογιών προκειμένου να εντοπιστούν οι πιο αποδοτικές παράμετροι κατά την εκπαίδευση του μοντέλου.

7.2 Περιορισμοί μελέτης

Η μελέτη περιορίζεται διότι η εκπαίδευση του μοντέλου είναι στην ελληνική γλώσσα. Αυτό συμβαίνει διότι η επεξεργασία φυσικής γλώσσας επηρεάζεται αρκετά από τον όγκο των δεδομένων που θα χρησιμοποιηθούν ως είσοδος του μοντέλου κατά την εκπαίδευση, και στην ελληνική γλώσσα το μέγεθος των δεδομένων είναι αρκετά περιορισμένο.

Τα δεδομένα που απαιτούνται προκειμένου να ληφθούν αποτελέσματα τα οποία ικανοποιούν τις ανάγκες της επεξεργασίας φυσικής γλώσσας είναι πολύ μεγαλύτερα από αυτά που μπορούν να εντοπιστούν στα ελληνικά. Ωστόσο, στόχος της συγκεκριμένης μελέτης είναι να παρουσιαστούν εργαλεία και διαδικασίες για της κατασκευή μοντέλων και η αξιοποίηση αυτών.

Ακόμα ένας περιορισμός σχετίζεται με την διαδικασία με την οποία υλοποιήθηκε η αξιολόγηση του μοντέλου. Αυτό συμβαίνει εξαιτίας του μικρού όγκου δεδομένων που χρησιμοποιήθηκαν σαν είσοδος του μοντέλου. Παρόλα αυτά, σκοπός είναι η παρουσίαση της διαδικασίας.

7.3 Πιθανές επεκτάσεις

Η συγκεκριμένη μελέτη είναι εφικτό επεκταθεί σε τρεις τομείς. Μια πρώτη επέκταση σχετίζεται με την χρήση της συγκεκριμένη προσέγγισης για να μελετηθεί και η αποτελεσμάτικοτητα και άλλων γλωσσών στην επεξεργασία φυσικής γλώσσας. Ακόμα προκειμένου να παραχθούν ακόμα πιο αποδοτικά μοντέλα θα μπορούσε να χρησιμοποιηθεί μεγαλύτερο πλήθος δεδομένων στα ελληνικά. Τέλος μπορεί να γίνει μελέτη της απόδοσης διαφορετικών αλγορίθμων δημιουργίας διανυσμάτων από αυτόν που χρησιμοποιήθηκε στην παρούσα έρευνα.

Βιβλιογραφία

[1] "Ανάκτηση Πληροφορίας.Wikipedia.2021.[ONLINE]." https://el.wikipedia.org/wiki/Ανάκτηση_πληροφοριών (accessed Jun. 10, 2021).

[2] "Data mining.Wikipedia.2021.[ONLINE]." https://en.wikipedia.org/wiki/Data_mining (accessed May 20, 2021).

[3] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Automatic Keyword Extraction from Individual Documents," *Text Min. Appl. Theory*, no. March, pp. 1–20, 2010, doi: 10.1002/9780470689646.ch1.

[4] "Machine learning - Wikipedia.2021 [ONLINE]." https://en.wikipedia.org/wiki/Machine_learning (accessed May 25, 2021).

[5] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015, doi: 10.1038/nature14539.

[6] "Machine Learning.2021.[ONLINE]." http://repfiles.kallipos.gr/html_books/93/04a-main.html (accessed Jun. 20, 2021).

[7] "Artificial neural network.Wikipedia.2021.[ONLINE]." https://en.wikipedia.org/wiki/Artificial_neural_network (accessed May 22, 2021).

[8] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer (Long. Beach. Calif).*, vol. 29, no. 3, pp. 31–44, 1996, doi: 10.1109/2.485891.

[9] "Neural Networks.2021.[ONLINE]." http://repfiles.kallipos.gr/html_books/93/04a-main.html (accessed Jun. 15, 2021).

[10] "Deep learning.Wikipedia.2021.[ONLINE]." https://en.wikipedia.org/wiki/Deep_learning (accessed May 20, 2021).

[11] "Tensorflow.2021.[ONLINE]." https://www.tensorflow.org/.

[12] "Tensorflow.2021.[ONLINE]." https://el.gadget-info.com/20764-everything-you-need-to-know-about-google-brains-tensorflow (accessed May 15, 2021).

[13] "Natural language processing.2021.[ONLINE]." https://en.wikipedia.org/wiki/Natural_language_processing (accessed May 15, 2021).

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, no. October, 2013.

[15] "An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec.2021.[ONLINE]." https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/.

[16] M. Baroni, G. Dinu, and G. Kruszewski, “Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors,” *52nd Annu. Meet. Assoc. Comput. Linguist. ACL 2014 - Proc. Conf.*, vol. 1, pp. 238–247, 2014, doi: 10.3115/v1/p14-1023.

[17] “Skip-Gram.2021.[ONLINE].” http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/ (accessed Jun. 12, 2021).

[18] “Skip-gram vs CBOW.2021.[ONLINE].” https://www.researchgate.net/figure/Continuous-Bag-of-words-CBOW-CB-and-Skip-gram-SG-training-model-illustrations_fig1_326588219 (accessed Jun. 05, 2021).

[19] “Word2vec.Wikipedia.2021.[ONLINE].” https://en.wikipedia.org/wiki/Word2vec (accessed Apr. 20, 2021).

[20] “Word2Vec.2021.[ONLINE].” https://medium.com/analytics-vidhya/implementing-word2vec-in-tensorflow-44f93cf2665f (accessed Jun. 15, 2021).

[21] “Fast Text.Wikipedia.2021.[ONLINE].” https://en.wikipedia.org/wiki/FastText (accessed Jun. 01, 2021).

[22] “NumPy.Wikipedia.2021.[ONLINE].” https://en.wikipedia.org/wiki/NumPy (accessed May 12, 2021).

[23] “Softmax.Wikipedia.2021.[ONLINE].” https://en.wikipedia.org/wiki/Softmax_function (accessed Mar. 01, 2021).

[24] T. Schnabel, I. Labutov, D. Mimno, and T. Joachims, “Evaluation methods for unsupervised word embeddings,” *Conf. Proc. - EMNLP 2015 Conf. Empir. Methods Nat. Lang. Process.*, no. September, pp. 298–307, 2015, doi: 10.18653/v1/d15-1036.

[25] O. Melamud, D. McClosky, S. Patwardhan, and M. Bansal, “The role of context types and dimensionality in learning word embeddings,” *2016 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. NAACL HLT 2016 - Proc. Conf.*, no. January, pp. 1030–1040, 2016, doi: 10.18653/v1/n16-1118.

[26] A. G. Lipscomb, “Ungelöste Probleme bei der Herstellung von Nahrungsmitteln: Schokolade,” *Fette, Seifen, Anstrichm.*, vol. 56, no. 10, pp. 803–809, 1954, doi: 10.1002/lipi.19540561008.

[27] “List of Wikipedias.Wikipedia.2021.” https://en.wikipedia.org/wiki/List_of_Wikipedias (accessed Mar. 04, 2021).

[28] Wikipedia, “Wikipedia Database.Wikipedia.2021.[ONLINE],” *Wikipedia.Com*, 2013. http://en.wikipedia.org/wiki/Wikipedia_database#Offline_wikipedia_reader (accessed Mar. 01, 2021).

[29] “Wikimedia.2021.[ONLINE].” https://dumps.wikimedia.org/elwiki/ (accessed Mar. 02, 2021).

[30] “elwiki dump on progress on 20210120.2021.[ONLINE].” https://dumps.wikimedia.org/elwiki/20210120/ (accessed Feb. 28, 2021).

[31] “Wikipedia.2021.[ONLINE].” https://dumps.wikimedia.org/elwiki/20210201/ (accessed May 20, 2021).

[32] “How to Develop Word Embeddings in Python with Gensim.2021.[ONLINE].” https://machinelearningmastery.com/develop-word-embeddings-python-gensim/ (accessed Mar. 20, 2021).

[33] “Flask(web framework).Wikipedia.2021.[ONLINE].” https://en.wikipedia.org/wiki/Flask_(web_framework) (accessed May 12, 2021).

[34] “HTML.Wikpedia.2021.[ONLINE].” https://en.wikipedia.org/wiki/HTML (accessed May 20, 2021).