ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΑΝΥΣΜΑΤΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΛΕΞΕΩΝ ΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Διπλωματική Εργασία

του

Μπερμπέρογλου Ιωάννη

Θεσσαλονίκη, 06/2018

ΔΙΑΝΥΣΜΑΤΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΛΕΞΕΩΝ ΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Μπερμπέρογλου Ιωάννης

Πτυχίο Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας 2015

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέποντες Καθηγητές
Μαργαρίτης Κωνσταντίνος
Σακελλαρίου Ηλίας

Εγκρίθηκε από την τετραμελή εξεταστική επιτροπή την 26/06/2018

Μαργαρίτης Κωνσταντίνος	Σακελλαρίου Ηλίας	Ρεφανίδης Ιωάννης	Σαμαράς Νικόλαος
.....	.....	.....	.....

Μπερμπέρογλου Ιωάννης

.....

Περίληψη

Η παρούσα έρευνα αφορά την ανάλυση της επεξεργασίας φυσικής γλώσσας και την υλοποίηση διαφόρων τεχνικών της, με τη χρήση μεθόδων διανυσματικής αναπαράστασης λέξεων. Γίνεται παρουσίαση και υλοποίηση του αλγορίθμου word2vec για αυτό το σκοπό. Με την βιβλιοθήκη gensim δημιουργούνται και αξιολογούνται μοντέλα γλωσσών για την ελληνική και την αγγλική γλώσσα. Αυτά τα μοντέλα συγκρίνονται και αναλύονται οι παράγοντες σύγκρισης διαφορετικών γλωσσών. Παρουσιάζεται τελικά η δημιουργία μιας εφαρμογής διαδικτύου για την παρουσίαση των μοντέλων της ελληνικής γλώσσας.

Λέξεις Κλειδιά: Επεξεργασία φυσικής γλώσσας, νευρωνικά δίκτυα, διανυσματική αναπαράσταση λέξεων, word2vec, gensim

Abstract

This research tackles some aspects of natural language processing and some implementation methods with the use of word vector representations. The algorithm word2vec, providing value in natural language processing, is presented and analysed. Several models for the greek and english languages, are created and evaluated with the library gensim. These models are then compared and several aspects on language comparison are analysed. Finally, a web application is developed to handle the demonstration for the greek language models.

Keywords: Natural language processing, neural networks, word vector representations, word2vec, gensim

Ευχαριστίες

Αυτή η διπλωματική είναι αφιερωμένη στην οικογένεια μου που με στήριξε από την αρχή της ζωής μου μέχρι και σήμερα, όπως επίσης και στη γιαγιά μου Ελένη, η οποία ανέλαβε μετά χαράς όλες τις οικονομικές απαιτήσεις αυτού του μεταπτυχιακού προγράμματος. Θέλω επίσης να ευχαριστήσω τους επιβλέποντες καθηγητές μου, κ. Μαργαρίτη Κωνσταντίνο και κ. Σακελλαρίου Ηλία, οι οποίοι ήταν δίπλα μου κατά τη διάρκεια της έρευνας και συνεισέφεραν σε αυτή με τις γνώσεις που μου προσέφεραν.

Περιεχόμενα

Κατάλογος Εικόνων

1 Εισαγωγή

1.1 Το πρόβλημα

Στη σημερινή εποχή, ένας τομέας που παρουσιάζει μεγάλο ενδιαφέρον στην επικοινωνία ανθρώπου υπολογιστή, είναι η επεξεργασία φυσικής γλώσσας. Μερικές από τις σημαντικότερες εφαρμογές του περιλαμβάνουν την ανάλυση της μορφολογίας λέξεων, τη συντακτική ανάλυση, το διαχωρισμό προτάσεων και την εξαγωγή όρων. Πλέον υπάρχει η δυνατότητα ανάλυσης συναισθημάτων, όπου για παράδειγμα, μπορεί να γίνει αντιληπτός ο θετικός ή αρνητικός χαρακτήρας ενός κειμένου. Με τη μηχανική μετάφραση είναι δυνατή η αυτόματη μετάφραση κειμένου με μεγάλα ποσοστά επιτυχίας. Τελικά, με την αναγνώριση φωνής, την κατανόηση κειμένου φυσικής γλώσσας και την παραγωγή φυσικής γλώσσας, είναι δυνατή η δημιουργία ψηφιακών βοηθών με εντυπωσιακές δυνατότητες. Γίνεται κατανοητή έτσι η ανάγκη για αποτελεσματική επεξεργασία φυσικής γλώσσας.

1.2 Σκοπός της έρευνας

Σκοπός της έρευνας είναι η παρουσίαση τεχνικών υλοποίησης επεξεργασίας φυσικής γλώσσας με τη χρήση της διανυσματικής αναπαράστασης λέξεων. Θα παρουσιαστεί η δημιουργία μοντέλων γλωσσών με τη χρήση νευρωνικών δικτύων (neural networks) και ο τρόποι αξιολόγησης αυτών. Τέλος θα συγκριθούν μοντέλα της ελληνικής και της αγγλικής για να εμφανιστούν πιθανές διαφοροποιήσεις στην απόδοση ανάλογα με την πολυπλοκότητα της κάθε γλώσσας.

1.3 Προσέγγιση

Για την επίτευξη των στόχων της διπλωματικής θα δημιουργηθούν μοντέλα στην ελληνική και την αγγλική γλώσσα. Για τη δημιουργία των μοντέλων θα εκπαιδευτούν νευρωνικά δίκτυα με την προσέγγιση της αναπαράστασης λέξεων στο διανυσματικό

χώρο και τη μέθοδο word2vec. Τα δεδομένα που θα χρησιμοποιηθούν για την εκπαίδευση θα προέρχονται από στιγμιότυπα της wikipedia. Τέλος, τα μοντέλα αυτά θα αξιολογηθούν και θα συγκριθούν.

1.4 Συνεισφορά

Το πεδίο της επεξεργασίας φυσικής γλώσσας αποτελεί σημαντικό μέρος της επιστημονικής έρευνας που πραγματοποιείται τα τελευταία χρόνια. Η διπλωματική θα προσθέσει επιστημονικό έργο σε αυτόν τον τομέα. Επίσης θα δημιουργηθούν μοντέλα και κριτήρια αξιολόγησης μοντέλων στην ελληνική γλώσσα όπου αντίστοιχες προσεγγίσεις είναι περιορισμένες.

1.5 Διάρθρωση του κειμένου

Το πρώτο κεφάλαιο περιλαμβάνει την εισαγωγή της έρευνας. Στο δεύτερο κεφάλαιο γίνεται βιβλιογραφική επισκόπηση όλων των τεχνολογιών που θα χρησιμοποιηθούν όπως και οι αρχές λειτουργίας αυτών. Το τρίτο κεφάλαιο αφορά πηγές δεδομένων που μπορούν να χρησιμοποιηθούν στην εκπαίδευση νευρωνικών δικτύων για τη δημιουργία μοντέλων για την ελληνική και την αγγλική γλώσσα. Στο τέταρτο κεφάλαιο θα παρουσιαστεί η δημιουργία αυτών των μοντέλων με την υλοποίηση του word2vec σε python. Στο πέμπτο κεφάλαιο θα παρουσιαστεί η διεπαφή διαδικτύου που δημιουργήθηκε για την παρουσίαση του ελληνικού μοντέλου. Στο έκτο κεφάλαιο, θα εφαρμοστούν κριτήρια αξιολόγησης στα μοντέλα και θα συγκριθούν οι επιδόσεις τους. Στο έβδομο και τελευταίο κεφάλαιο βρίσκονται τα συμπεράσματα της έρευνας.

2 Βιβλιογραφική Επισκόπηση

2.1 Μηχανική μάθηση

Η μηχανική μάθηση (machine learning) είναι ένας τομέας της τεχνητής νοημοσύνης και είναι το πεδίο που προσφέρει στους υπολογιστές την ικανότητα να μαθαίνουν [1]. Ο στόχος της είναι να προσφέρει στους ηλεκτρονικούς υπολογιστές τη δυνατότητα να αντιμετωπίζουν αποτελεσματικά προβλήματα που δεν έχουν ξανασυναντήσει, με γνώμονα την προηγούμενη εμπειρία τους σε άλλα προβλήματα. Έχουν προταθεί πολλές προσεγγίσεις για τη μηχανική μάθηση. Οι σημαντικότερες από αυτές είναι η χρήση δέντρων αποφάσεων, ο επαγωγικός λογικός προγραμματισμός, η ομαδοποίηση, τα δίκτυα Bayes, η ενισχυτική μάθηση, οι γενετικοί αλγόριθμοι, τα νευρωνικά δίκτυα και η βαθιά μάθηση [2].

Ένας αλγόριθμος μάθησης ορίζεται ως ένας αλγόριθμος ο οποίος έχει τη δυνατότητα να μαθαίνει από τα δεδομένα που συναντά. Τα σημεία που τον ορίζουν είναι το έργο που καλείται να παράξει, η επίδοση του πάνω σε αυτό το έργο και η εμπειρία που έχει αποκτήσει από αυτό. Το έργο δεν αφορά τη διαδικασία της μάθησης αλλά το αναμενόμενο αποτέλεσμα αυτής. Για παράδειγμα, σε έναν αλγόριθμο που αποσκοπεί στην εκμάθηση ενός παιδιού να περπατάει, το έργο είναι το περπάτημα και όχι η διαδικασία που ακολουθείται για να επιτευχθεί η εκμάθηση αυτού. Για την αξιολόγηση της επίδοσης του αλγορίθμου είναι αναγκαίος ο ορισμός μετρικών, ανάλογα με το κάθε πρόβλημα, οι οποίες κρίνουν την αποτελεσματικότητα και την επίδοση του αλγορίθμου. Συνήθως η αξιολόγηση γίνεται σε δεδομένα που δεν έχει ξανασυναντήσει ο αλγόριθμος. Η εμπειρία αφορά την προηγούμενη εφαρμογή του αλγορίθμου σε παρόμοιες καταστάσεις.

Η δυνατότητα ενός αλγορίθμου να είναι αποτελεσματικός σε δεδομένα που δεν έχει ξανασυναντήσει ονομάζεται γενίκευση. Κατά τη διαδικασία εκπαίδευσης ενός μοντέλου σε ένα σύνολο δεδομένων, υπολογίζεται το σφάλμα εκπαίδευσης, δηλαδή η επίδοση στα υπάρχοντα δεδομένα. Από την επίδοση του αλγορίθμου σε νέα δεδομένα προκύπτει το σφάλμα γενίκευσης. Σκοπός ενός αλγορίθμου μάθησης είναι η κατάλληλη επιλογή των παραμέτρων που χρησιμοποιεί, έτσι ώστε να ελαχιστοποιούνται αυτά τα σφάλματα.

2.2 Νευρωνικά δίκτυα

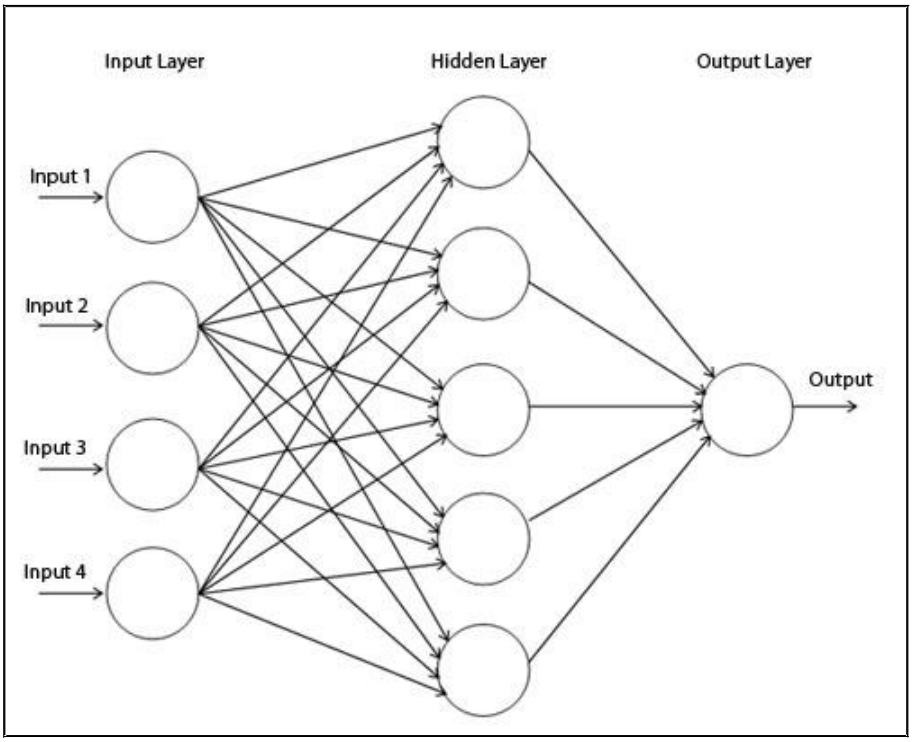
Τα τεχνητά νευρωνικά δίκτυα, για τα οποία χρησιμοποιείται συνήθως ο όρος νευρωνικά δίκτυα, αποτελούν μία προσπάθεια της επιστήμης για την μοντελοποίηση του τρόπου λειτουργίας του ανθρώπινου εγκεφάλου [3]. Ο ανθρώπινος εγκέφαλος είναι εξαιρετικά πολύπλοκος και λειτουργεί με αρκετά διαφορετικό τρόπο από αυτό που λειτουργούν οι συμβατικοί ηλεκτρονικοί υπολογιστές. Στις λειτουργίες για τις οποίες χρησιμοποιείται καθημερινά από τον άνθρωπο, όπως είναι η όραση, μπορεί και επεξεργάζεται τεράστιους όγκους πληροφορίας, σε ταχύτητες πολλαπλάσιες από εκείνες που θα πετύχαινε ακόμη και ο ταχύτερος ηλεκτρονικός υπολογιστής σήμερα.

Μερικά από τα χαρακτηριστικά που διαθέτει ο ανθρώπινος εγκέφαλος περιλαμβάνουν την εκτέλεση υπολογισμών με μεγάλο παραλληλισμό, τη δυνατότητα μάθησης και προσαρμογής στο περιβάλλον όπως και την ανοχή σε λάθη, διατηρώντας παράλληλα εξαιρετικά χαμηλή κατανάλωση ενέργειας. Αυτά τα χαρακτηριστικά οδήγησαν σε προσπάθειες μοντελοποίησης του, καθώς οι υπολογιστές υστερούν σε αρκετά από αυτά τα πεδία και το μοναδικό σημαντικό πλεονέκτημα που έχουν, βρίσκεται στον τομέα των αριθμητικών υπολογισμών.

Για την αποτελεσματική μοντελοποίηση του ανθρώπινου εγκεφάλου, πρέπει να ληφθεί υπόψη ο τρόπος που λειτουργεί ολόκληρο το νευρικό σύστημα του ανθρώπου. Το τελευταίο αποτελείται από δισεκατομμύρια νευρώνες, διαφόρων μεγεθών, οι οποίοι διανύουν ολόκληρο το ανθρώπινο σώμα [4]. Τα βασικά σημεία ενός τυπικού νευρώνα είναι οι δενδρίτες, το κυτταρικό σώμα και ο άξονας του. Οι δενδρίτες αποτελούν μικρές προεξοχές και είναι συνήθως το σημείο εισαγωγής σημάτων στο νευρώνα. Το κυτταρικό σώμα βρίσκεται ανάμεσα στους δενδρίτες και τον άξονα και περιλαμβάνει τον πυρήνα του κυττάρου. Ο άξονας είναι μία λεπτή ίνα, μεγάλου μήκους, η οποία λειτουργεί ως έξοδος σημάτων από το νευρώνα. Η επικοινωνία μεταξύ νευρώνων επιτυγχάνεται με τη μεταφορά σημάτων από τις συνάψεις που διατηρούν με άλλους νευρώνες, με τα σήματα να μεταφέρονται από τον άξονα του ενός στους δενδρίτες του άλλου.

Ένα νευρωνικό δίκτυο, όπως παρουσιάζεται και στην Εικόνα 1, απαρτίζεται από κόμβους οι οποίοι μπορούν θεωρηθούν ως νευρώνες του συστήματος. Ο κάθε κόμβος δέχεται ένα σύνολο εισόδων από διαφορετικές πηγές, εκτελεί κάποιου είδους υπολογισμό και έχει μία έξοδο. Η έξοδος μπορεί να είναι και είσοδος σε άλλους κόμβους. Οι κόμβοι χωρίζονται σε τρεις κατηγορίες, τους νευρώνες εισόδου, τους νευρώνες εξόδου και τους

υπολογιστικούς νευρώνες. Οι νευρώνες εισόδου είναι αυτοί που αποτελούν το επίπεδο εισόδου δεδομένων του συστήματος και μπορούν να συγκριθούν με τους δενδρίτες του νευρικού συστήματος. Οι υπολογιστικοί νευρώνες αποτελούν το κρυφό επίπεδο του δικτύου. Ο καθένας έχει μία συνάρτηση ενεργοποίησης η οποία εξαρτάται από τις εισόδους που δέχεται και ορίζει την ενεργοποίηση ή μη του νευρώνα. Τέλος, οι νευρώνες εξόδου αποτελούν το επίπεδο εξόδου αποτελεσμάτων από το δίκτυο.



Εικόνα 1: Ένα απλό νευρωνικό δίκτυο

Μία ιδιαιτερότητα των νευρωνικών δικτύων είναι η δυνατότητα εκπαίδευσης τους. Η εκπαίδευση αυτή επιτυγχάνεται μέσω της αλλαγής των τιμών και βαρών που περιέχει που κάθε ο κόμβος του δικτύου. Έχουν προταθεί πολλοί αλγόριθμοι που αναλαμβάνουν τις αλλαγές αυτές, με σκοπό το δίκτυο να εκπληρώνει όσο το δυνατό καλύτερα το σκοπό του. Οι αλγόριθμοι εκπαίδευσης μπορούν να χωριστούν σε δύο βασικές κατηγορίες, τη μάθηση με επίβλεψη και τη μάθηση χωρίς επίβλεψη. Η μάθηση με επίβλεψη απαιτεί την ύπαρξη κάποιου εξωτερικού εκπαιδευτή. Για παράδειγμα, ένα δίκτυο μπορεί να δεχθεί τις τιμές εισόδου όπως και τις τιμές εξόδου από τον εκπαιδευτή και να μεταβάλλει τα βάρη των κόμβων του ανάλογα. Η μάθηση χωρίς επίβλεψη δεν απαιτεί την ύπαρξη εξωτερικού εκπαιδευτή. Χωρίς να παρέχονται άλλες πληροφορίες το

δίκτυο μπορεί να εντοπίζει μόνο του τη δομή των δεδομένων εισόδου και να μεταβάλλει τα βάρη των κόμβων του αντίστοιχα.

Μία υποκατηγορία των νευρωνικών δικτύων είναι τα βαθιά νευρωνικά δίκτυα [5] τα οποία είναι νευρωνικά δίκτυα με πολλαπλά κρυφά επίπεδα. Έχει παρατηρηθεί πως με την αύξηση των κρυφών επιπέδων ενός νευρωνικού δικτύου αυξάνεται η επίδοση του σε προβλήματα υψηλής πολυπλοκότητας. Το βασικό μειονέκτημα της αύξησης των κρυφών επιπέδων, είναι η μεγάλη αύξηση στις υπολογιστικές απαιτήσεις για την εκπαίδευση του δικτύου.

Τα νευρωνικά δίκτυα έχουν ήδη πολλές εφαρμογές στην επιστήμη. Στην ιατρική χρησιμοποιούνται για αναγνώριση προτύπων στους ασθενείς. Σε έναν κινητήρα αυτοκινήτου, ένα νευρωνικό δίκτυο μπορεί να ορίζει τη βέλτιστη εισροή καυσίμου για την ελαχιστοποίηση της κατανάλωσης. Ένα άλλο παράδειγμα, είναι η πρόβλεψη της κίνησης των μετοχών του χρηματιστηρίου όπως και άλλων οικονομικών μεγεθών.

2.2.1 Υλοποίηση νευρωνικών δικτύων στην Python

Για την υλοποίηση ενός απλού νευρωνικού δικτύου στην Python θα χρησιμοποιηθεί η βιβλιοθήκη tensorflow [6] της Google. Το tensorflow είναι μία βιβλιοθήκη ανοικτού κώδικα η οποία προσφέρει λειτουργικότητα για αριθμητικές πράξεις όπως και για τη μηχανική μάθηση και τα νευρωνικά δίκτυα. Παρέχει πολλές τεχνικές για πολλές διαφορετικές εφαρμογές. Για τη συγκεκριμένη έρευνα, θα γίνει αναφορά στη βασική χρήση του tensorflow και σε όσα στοιχεία χρησιμοποιηθούν στη συνέχεια. Θα γίνει μία υλοποίηση ενός μοντέλου γραμμικής παλινδρόμησης σε Python για την μελέτη των αρχών λειτουργίας του.

Στην Εικόνα 2 στη γραμμή 1 γίνεται η εισαγωγή του πακέτου και στην γραμμή 3 δημιουργείται μία σύνοδος, μέσω της οποίας θα γίνονται κλήσεις στη βιβλιοθήκη. Στη συνέχεια ορίζεται το γραμμικό μοντέλο της παλινδρόμησης το οποίο είναι $y = Wx+b$. Τα W και b αποτελούν τις τιμές οι οποίες θα ορίσουν το μοντέλο. Το x αφορά τα δεδομένα εισόδου και το y την έξοδο. Σκοπός της εκπαίδευσης που θα πραγματοποιηθεί στη συνέχεια, είναι ο καλύτερος δυνατός προσδιορισμός των W και b έτσι ώστε το νευρωνικό δίκτυο που θα δημιουργηθεί να αντιπροσωπεύει καλύτερα την παλινδρόμηση.

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

Εικόνα 2: Γραμμική παλινδρόμηση με tensorflow 1

Στις γραμμές 5 και 6 γίνεται ο ορισμός των δύο μεταβλητών που αντιστοιχούν στα βάρη των ενδιάμεσων κόμβων του δικτύου. Στη γραμμή 7 ορίζεται το x, στο οποίο θα τοποθετούνται τα δεδομένα εισόδου. Σημειώνεται πως στο tensorflow ως Variable ορίζονται οι τιμές των βαρών που θα υπολογιστούν κατά την εκπαίδευση ενώ ως placeholder οι θέσεις στις οποίες τοποθετούνται εξωτερικά δεδομένα. Τελικά στη γραμμή 10 ορίζεται το μοντέλο της παλινδρόμησης.

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

Εικόνα 3: Γραμμική παλινδρόμηση με tensorflow 2

Στη συνέχεια, στις γραμμές 12 και 13 γίνεται η αρχικοποίηση των μεταβλητών από το tensorflow ενώ στη γραμμή 16 ορίζεται το placeholder y, το οποίο θα αφορά τα αναμενόμενα αποτελέσματα από την παλινδρόμηση. Στη γραμμή 19 ορίζεται στη μεταβλητή squared_deltas, ο υπολογισμός των τετράγωνων των διαφορών των υπολογισμένων y με τα αναμενόμενα. Το άθροισμα αυτών των τετραγώνων ορίζεται ως συνάρτηση απώλειας για το μοντέλο και αποθηκεύεται στη μεταβλητή loss. Σκοπός της

εκπαίδευσης είναι η ελαχιστοποίηση της συνάρτησης απώλειας. Στη γραμμή 23 ορίζεται ο τρόπος με τον οποίο θα μεταβάλλονται οι μεταβλητές της παλινδρόμησης, δηλαδή η συνάρτηση βελτιστοποίησης, ενώ στη γραμμή 24 το βήμα εκπαίδευσης για κάθε επανάληψη. Σε κάθε επανάληψη δηλαδή της εκπαίδευσης, θα μεταβάλλονται οι μεταβλητές για το W και το b, έτσι ώστε, το άθροισμα των τετραγώνων των διαφορών των αναμενόμενων από τα πραγματικά αποτελέσματα να μειώνεται κάθε φορά σε σχέση με το προηγούμενο.

Στην Εικόνα 4, στις γραμμές 26 και 27 ορίζονται τα δεδομένα εισόδου για τα x και τα y. Στη συνέχεια, στη γραμμή 31 πραγματοποιείται η εκπαίδευση του μοντέλου σε 10000 επαναλήψεις. Μετά την εκπαίδευση έχουν οριστεί τιμές για τις μεταβλητές W και b στις οποίες αποκτάται πρόσβαση στις γραμμές 34 και 35. Στη γραμμή 36, τυπώνεται η συνάρτηση της παλινδρόμησης. Όπως φαίνεται, μετά την εκπαίδευση του μοντέλου, η μεταβλητή W έχει την τιμή 5.000001 και η μεταβλητή b την τιμή 1.9999969. Με γυμνό μάτι φαίνεται από τα δεδομένα ότι η συνάρτηση θα έπρεπε να είναι 5x + 2. Ο λόγος για τον οποίο δεν εντοπίστηκε με απόλυτη ακρίβεια η συνάρτηση έχει να κάνει με τη συνάρτηση βελτιστοποίησης που χρησιμοποιείται σε κάθε βήμα για την ελαχιστοποίηση της συνάρτησης απώλειας. Η προσέγγιση είναι προσεγγιστική και όχι απόλυτη.

```
26      x_values = [1, 2, 3, 4, 5]
27      y_values = [7, 12, 17, 22, 27]
28
29      # Iterate and train the model
30      for i in range(10000):
31          sess.run(train, {x: x_values, y: y_values})
32
33      # Print the results
34      trained_W = sess.run(W)[0]
35      trained_b = sess.run(b)[0]
36      print("Output -> Trained function: ", trained_W, "x +", trained_b)
Output -> Trained function:  5.000001 x + 1.9999969
```

Εικόνα 4: Γραμμική παλινδρόμηση με tensorflow 3

2.3 Επεξεργασία φυσικής γλώσσας

Ως επεξεργασία φυσικής γλώσσας, ορίζεται ο τρόπος με τον οποίο ένας ηλεκτρονικός υπολογιστής μπορεί να επεξεργαστεί κείμενο σε φυσική γλώσσα [7]. Αυτή η επεξεργασία μπορεί να αφορά είτε την κατανόηση κειμένου είτε την παραγωγή αυτού. Ενδιαφέρον προκαλεί η μελέτη των χαρακτηριστικών και ιδιαιτεροτήτων της φυσικής γλώσσας που καθιστούν το έργο αυτό δύσκολο.

Η σημαντικότερη ιδιαιτερότητα της φυσικής γλώσσας είναι ο τρόπος που μεταδίδεται η πληροφορία. Στον κόσμο των υπολογιστών, η πληροφορία πρέπει να είναι δομημένη και να υπάρχει στο σύνολο της για να γίνει αντιληπτή. Σε μία γλώσσα προγραμματισμού, για παράδειγμα, ακόμη και η έλλειψη ενός συμβόλου είναι αρκετή, για τη μη κατανόηση του συνόλου του προγράμματος από τον υπολογιστή. Αντίθετα, στη φυσική γλώσσα, δεν υπάρχει απαραίτητα κάποια συγκεκριμένη δομή στο λόγο και το μεγαλύτερο μέρος της πληροφορίας που ορίζει την επικοινωνία παραλείπεται. Αυτό είναι δυνατό, χάρη στην εμπειρία των ανθρώπων στην εκάστοτε εποχή. Πολλές φορές στην καθημερινή επικοινωνία οι άνθρωποι χρησιμοποιούν τις ίδιες λέξεις με διαφορετικούς τρόπους, προκαλώντας διαφορετικές ερμηνείες, οι οποίες γίνονται αντιληπτές από τα συμφραζόμενα και από την κοινή γνώση που υπάρχει.

Για να είναι εφικτή η επεξεργασία της φυσικής γλώσσας, είναι απαραίτητη η κωδικοποίηση των λέξεων. Οι λέξεις πρέπει να να λάβουν κάποιο συμβολισμό ο οποίος να μπορεί να δείξει ομοιότητες και διαφορές στις λέξεις. Η καλύτερη αντιμετώπιση είναι ο ορισμός διανυσμάτων για κάθε λέξη.

2.3.1 Αναπαράσταση λέξεων στο διανυσματικό χώρο

Ο βασικός παράγοντας για την αποτελεσματική επεξεργασία φυσικής γλώσσας, είναι η κατανόηση της σημασίας των λέξεων. Είναι σχετικά αδύνατο να πραγματοποιηθεί πλήρης μοντελοποίηση μιας γλώσσας από ένα ηλεκτρονικό υπολογιστή με τον τρόπο που ένας άνθρωπος αντιλαμβάνεται τη γλώσσα. Παρόλα αυτά, είναι δυνατή η μερική μοντελοποίηση αρκετών συνιστωσών της. Έχουν προταθεί πολλές προσεγγίσεις στις οποίες οι λέξεις μιας γλώσσας αναπαριστώνται σε ένα διανυσματικό χώρο πολλών διαστάσεων [8].

Η κάθε λέξη, συγκεκριμένα, περιγράφεται από ένα διάνυσμα πραγματικών αριθμών. Η επιλογή των διανυσμάτων είναι τέτοια ώστε, λέξεις που σχετίζονται σημασιολογικά σε διάφορες διαστάσεις να έχουν και κοντινές τιμές στα διανύσματα τους, στις αντίστοιχες διαστάσεις. Με τη χρήση νευρωνικών δικτύων μπορεί να επιτευχθεί η βελτιστοποίηση των τιμών των διανυσμάτων για κάθε λέξη, με σκοπό την καλύτερη αναπαράσταση της εκάστοτε λέξης σε σχέση με τις υπόλοιπες. Υπάρχουν επίσης τεχνικές δημιουργίας διανυσμάτων με τη μέτρηση των εμφανίσεων των λέξεων μέσα στο κείμενο [9].

Για να είναι εφικτή αυτή η διαδικασία, είναι απαραίτητη η ύπαρξη ενός συνόλου ποιοτικών κειμένων. Το μέγεθος των δεδομένων είναι ο βασικός παράγοντας που επηρεάζει την αποτελεσματικότητα του μοντέλου. Για να επιτευχθούν ικανοποιητικά αποτελέσματα, είναι απαραίτητα κείμενα που περιέχουν από εκατομμύρια έως και δισεκατομμύρια λέξεις. Διάφορες βελτιστοποιήσεις είναι επίσης απαραίτητες πριν τη δημιουργία του μοντέλου, όπως είναι η αφαίρεση λέξεων με μικρή συχνότητα εμφάνισης. Αν για παράδειγμα, σε ένα κείμενο 50 εκατομμυρίων λέξεων, μία λέξη εμφανίζεται μόνο 3 φορές θα ήταν λογικό να αφαιρεθεί. Οι λόγοι αφαίρεσης της, είναι η κακή αναπαράσταση της στο μοντέλο, όπως επίσης και η διαστρέβλωση των διανυσμάτων των κοντινών της λέξεων. Επίσης, ανάλογα με τις ανάγκες της εφαρμογής που αναπτύσσεται, μπορεί να αφαιρεθούν σημεία στίξης, σύμβολα, αριθμοί, άρθρα, αντωνυμίες και άλλες λέξεις που επιτελούν συντακτικούς ή γραμματικούς σκοπούς. Αφού ολοκληρωθούν οι βελτιστοποιήσεις και παραμείνει ένα ποιοτικό κείμενο μπορεί να ξεκινήσει η δημιουργία αυτών των διανυσμάτων.

2.3.2 Μοντέλα συχνοτήτων και μοντέλα πρόβλεψης

Ένας σχετικά απλός τρόπος δημιουργίας διανυσμάτων λέξεων από ένα κείμενο είναι η μέτρηση των εμφανίσεων της κάθε λέξης μέσα σε αυτό. Συγκεκριμένα, με την διάθεση ενός αριθμού διαφορετικών κειμένων μπορεί να δημιουργηθεί ένα διάνυσμα για την κάθε λέξη, με κάθε αριθμό του διανύσματος να αντιπροσωπεύει το άθροισμα των εμφανίσεων της λέξης σε κάθε κείμενο. Για παράδειγμα, αν υπάρχουν 4 κείμενα και η λέξη σπίτι εμφανίζεται 5 φορές στο πρώτο, 2 φορές στο δεύτερο, καμία φορά στο τρίτο και 4 φορές στο τέταρτο, το διάνυσμα για τη λέξη θα ήταν [5, 2, 0, 4] και αντίστοιχα η

κάθε λέξη του λεξικού θα περιγραφεί από ένα διάνυσμα τεσσάρων διαστάσεων. Ένα τέτοιο μοντέλο βασίζεται στη μέτρηση και μόνο των λέξεων μέσα στο κείμενο.

Έχουν δημιουργηθεί πολλά μοντέλα που ακολουθούν παρόμοια λογική με διάφορες βελτιστοποιήσεις όπως είναι το TF-IDF. Το συγκεκριμένο μοντέλο μετρά και πάλι τη συχνότητα εμφάνισης της κάθε λέξης για κάθε κείμενο, λαμβάνει όμως υπόψη και τη συχνότητα εμφάνισης της λέξης στα διαφορετικά κείμενα. Για παράδειγμα, μία λέξη που εμφανίζεται πολλές φορές σε όλα τα κείμενα, μάλλον αφορά λέξη για την οποία δεν μπορεί να δημιουργηθεί διάνυσμα καθώς πιθανώς είναι κάποιος σύνδεσμος ή άρθρο και εμφανίζεται παντού.

Έχουν προταθεί και κάποια μοντέλα τα οποία βασίζονται σε πίνακες εμφάνισης λέξεων κοντά σε άλλες λέξεις. Αυτά τα μοντέλα απαιτούν ένα πίνακα με μία σειρά και μία στήλη για κάθε λέξη που υπάρχει στο λεξικό, στον οποίο καταγράφονται οι κοντινές εμφανίσεις των λέξεων με βάση ένα παράθυρο αναζήτησης. Μία τέτοια αντιμετώπιση προσφέρει καλύτερα αποτελέσματα καθώς λέξεις που εμφανίζονται κοντά σε άλλες λέξεις σχετίζονται συνήθως με κάποιο τρόπο. Το μεγάλο πρόβλημα είναι οι τεράστιες υπολογιστικές απαιτήσεις για τη δημιουργία και τη χρήση αυτών των μοντέλων.

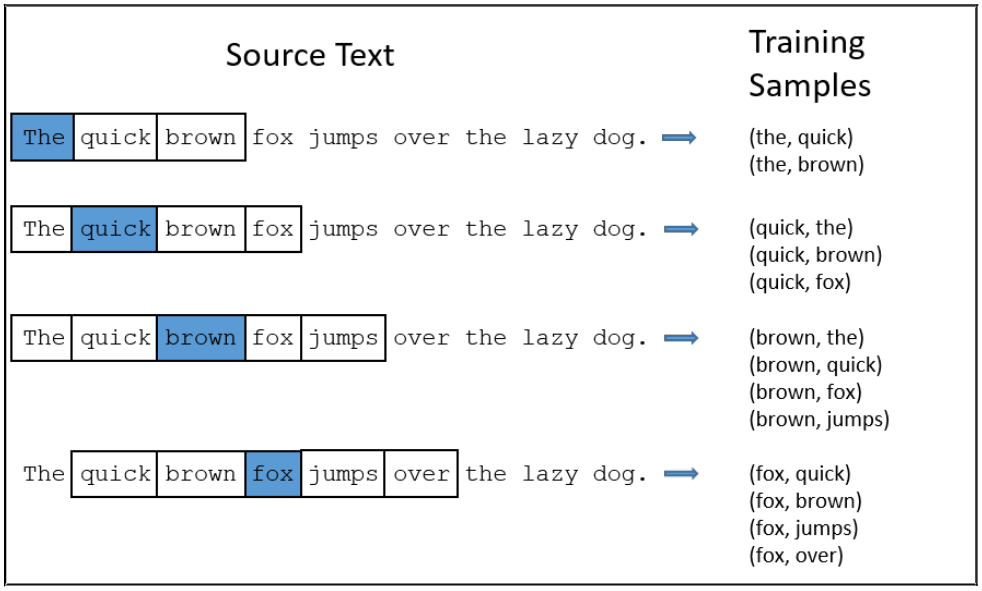
Τη λύση στο πρόβλημα των υπολογιστικών απαιτήσεων φέρουν τα μοντέλα πρόβλεψης, καταφέρνοντας παράλληλα πολύ καλύτερη αναπαράσταση των λέξεων [10]. Ο αλγόριθμος που θα μελετηθεί σε αυτήν την έρευνα είναι ο word2vec και ανήκει στην κατηγορία μοντέλων πρόβλεψης, τα οποία με τη χρήση των νευρωνικών δικτύων δημιουργούν τα διανύσματα των λέξεων. Στη συνέχεια θα αναλυθούν οι δύο κυρίαρχες αρχιτεκτονικές που χρησιμοποιούνται από τα μοντέλα πρόβλεψης, οι οποίες είναι η Skip-gram και η CBOW.

2.3.3 Οι αρχιτεκτονικές Skip-gram και CBOW

Μία συνήθης πρακτική, είναι η παρακολούθηση της εμφάνισης των λέξεων κοντά σε άλλες λέξεις μέσα στο κείμενο. Αυτός είναι και ο τρόπος που ένα παιδί μαθαίνει μία γλώσσα. Αν για παράδειγμα, οι πιο συχνά εμφανιζόμενες λέξεις κοντά στη λέξη *σπίτι*, είναι οι λέξεις *πηγαίνω* και *φεύγω*, υποδεικνύεται πως σε κάποια διάσταση αυτές οι δύο λέξεις σχετίζονται. Βασισμένοι σε αυτή τη λογική έχουν προταθεί δύο πολύ

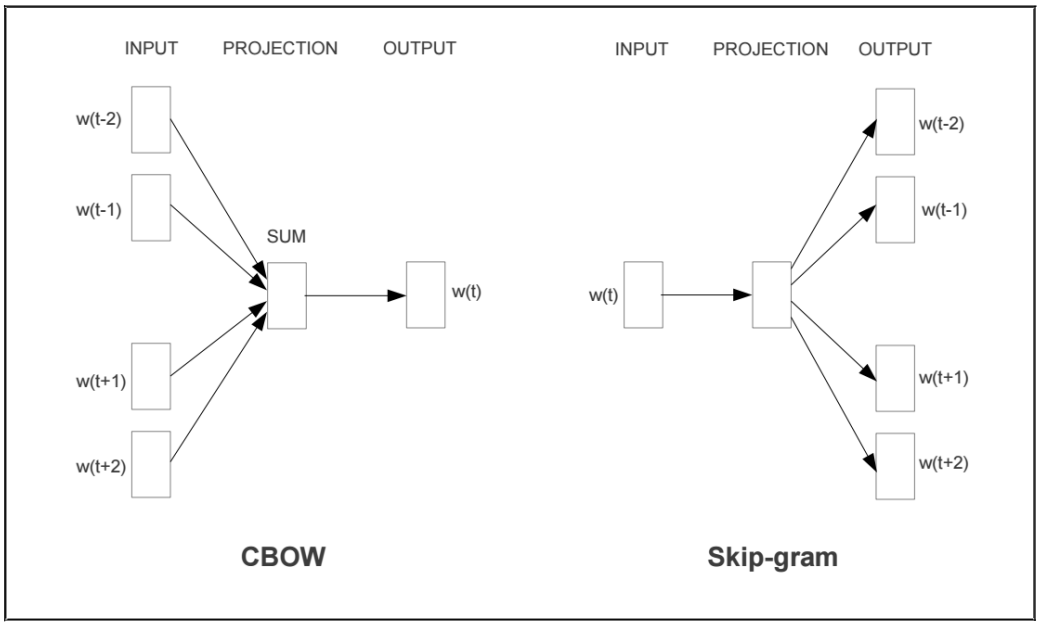
αποτελεσματικές αρχιτεκτονικές για τη δημιουργία διανυσμάτων λέξεων. Η πρώτη ονομάζεται Skip-gram και η δεύτερη CBOW (Continuous Bag of Words).

Με την αρχιτεκτονική Skip-gram, για κάθε λέξη που εμφανίζεται στο κείμενο, διατηρείται η πληροφορία για τις λέξεις που εμφανίζονται πριν και μετά από αυτή. Ο αριθμός των λέξεων είναι μεταβαλλόμενος και εξαρτάται από τις ανάγκες τις εφαρμογής. Το βάρος της κάθε λέξης μπορεί επίσης να είναι μεταβλητό. Για παράδειγμα, οι λέξεις που εμφανίζονται πιο κοντά σε μία λέξη έχουν μεγαλύτερη σημασία από αυτές που εμφανίζονται στο τέλος του παραθύρου ελέγχου.



Εικόνα 5: Εφαρμογή της αρχιτεκτονικής Skip-gram

Όπως φαίνεται και στην Εικόνα 5, δημιουργούνται ζευγάρια εκπαίδευσης για κάθε λέξη που εμφανίζεται στο κείμενο. Αυτή η πληροφορία στη συνέχεια, μπορεί να χρησιμοποιηθεί για τη δημιουργία και μεταβολή του διανύσματος της κάθε λέξης. Αντίθετα με την αρχιτεκτονική Skip-gram, η αρχιτεκτονική CBOW ακολουθεί την αντίθετη προσέγγιση. Για κάθε λέξη που εμφανίζεται στο κείμενο ενημερώνονται τα διανύσματα των λέξεων που βρίσκονται κοντά της και όχι το διάνυσμα της ίδιας της λέξης. Μία σύγκριση των δύο αρχιτεκτονικών παρουσιάζεται στην Εικόνα 6.



Εικόνα 6: Σύγκριση των αρχιτεκτονικών Skip-gram και CBOW

Εκτός από τον τρόπο δημιουργίας του μοντέλου αλλάζει και ο τρόπος πρόβλεψης από αυτό. Με τη CBOW προβλέπεται η λέξη που μπορεί να βρίσκεται ανάμεσα σε κάποιες λέξεις ενώ στη Skip-gram με βάση μία λέξη, προβλέπονται οι λέξεις που μπορεί να βρίσκονται κοντά της. Με γνώμονα αυτές τις αρχιτεκτονικές έχουν δημιουργηθεί μέθοδοι οι οποίοι αναλαμβάνουν τη δημιουργία του μοντέλου και την αξιοποίηση του. Η πιο γνωστή από αυτές είναι η μέθοδος word2vec.

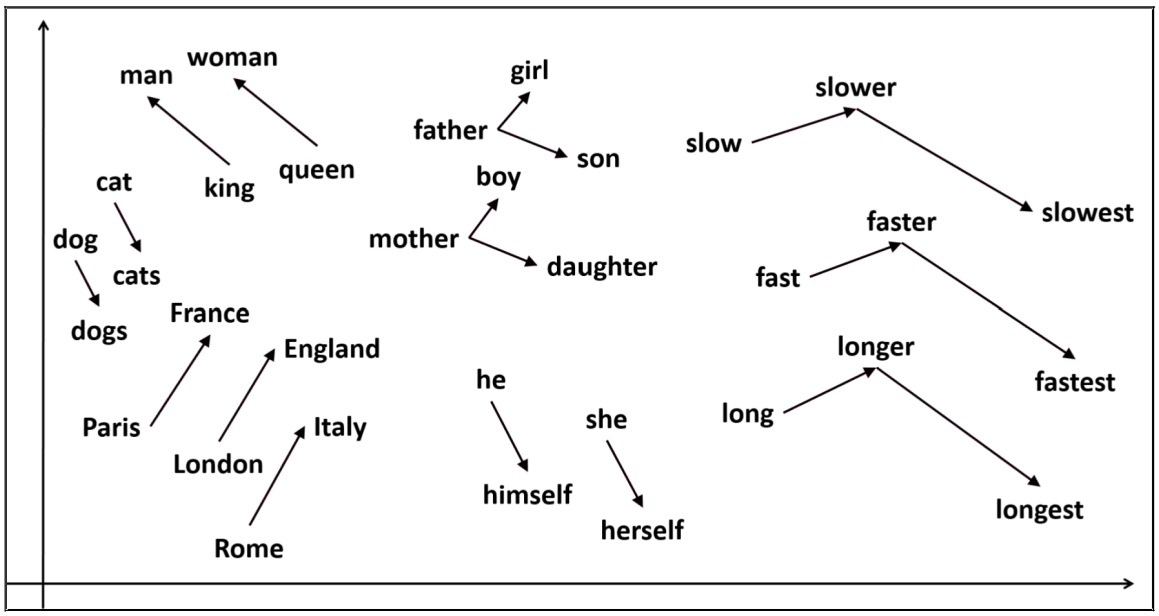
2.4 Ο αλγόριθμος word2vec

2.4.1 Η προσέγγιση

Ο αλγόριθμος word2vec περιλαμβάνει μία επαναληπτική μέθοδο παραγωγής διανυσμάτων λέξεων και δημιουργίας μοντέλων που προτάθηκε από την Google [11]. Σε αντίθεση με άλλες μεθόδους, οι οποίες αντιμετωπίζουν το κείμενο σαν σύνολο κατά την παραγωγή του μοντέλου, το word2vec λειτουργεί επαναληπτικά ενημερώνοντας το μοντέλο για κάθε κείμενο που πρέπει να επεξεργαστεί. Κατά τη δημιουργία ενός μοντέλου λαμβάνει ως είσοδο ένα μεγάλο σύνολο κειμένων και δημιουργεί ένα διανυσματικό χώρο μεταβλητών διαστάσεων, όπου για κάθε λέξη δημιουργείται και ένα

διαφορετικό διάνυσμα. Το μέγεθος των διαστάσεων συνηθίζεται να είναι της τάξης των μερικών εκατοντάδων.

Η διαδικασία της παραγωγής του μοντέλου μπορεί να παραμετροποιηθεί ως προς την αρχιτεκτονική που θα χρησιμοποιηθεί, το μέγεθος παραθύρου ελέγχου και τον αριθμό των διαστάσεων. Οι διαθέσιμες αρχιτεκτονικές είναι οι Skip-gram και CBOW που αναλύθηκαν νωρίτερα. Το παράθυρο ελέγχου ορίζει τον αριθμό των λέξεων που θα λαμβάνονται υπόψη, πριν και μετά από την κάθε λέξη. Όσον αφορά τον αριθμό των διαστάσεων, αυτός ορίζει την πολυπλοκότητα και το μέγεθος του μοντέλου.



Εικόνα 7: Αναπαράσταση λέξεων με το word2vec

Ο τρόπος με τον οποίο το word2vec δημιουργεί τα διανύσματα λέξεων επιτρέπει στις λέξεις να διατηρούν σημασιολογικές και συντακτικές πληροφορίες, οι οποίες μπορούν να αναπαραχθούν με τη χρήση αριθμητικών πράξεων. Για παράδειγμα, όπως φαίνεται στην Εικόνα 7, μπορούν παρατηρηθούν αναλογίες στην τοποθέτηση των λέξεων στο διανυσματικό χώρο. Με αλγεβρικές πράξεις είναι δυνατή η επίλυση αναλογιών του τύπου: *"*Η λέξη *France* συσχετίζεται με τη λέξη *Paris*, όπως η λέξη *England* με τη λέξη ***London****"*. Επίσης είναι εμφανές ότι λέξεις με κοντινή σημασιολογία, βρίσκονται κοντά και στο διανυσματικό χώρο.

Είναι σημαντικό να γίνει αντιληπτός ο τρόπος δημιουργίας ενός τέτοιου μοντέλου για να μπορέσει να αξιοποιηθεί σωστά. Το γεγονός ότι δύο λέξεις βρίσκονται κοντά στο

διανυσματικό χώρο σημαίνει ότι αυτές οι λέξεις βρίσκονται κοντά σε κοινές λέξεις μέσα στο κείμενο. Αυτό πιθανόν να σημαίνει ότι σχετίζονται σε κάποια διάσταση μεταξύ τους. Για παράδειγμα, οι λέξεις που δηλώνουν χρώματα μπορεί να βρίσκονται κοντά μεταξύ τους, όπως επίσης και οι λέξεις που ορίζουν έναν άνθρωπο. Η λέξη *άντρας* μπορεί να βρίσκεται κοντά με τη λέξη *γυναίκα* σε μία διάσταση, καθώς αυτές οι λέξεις ορίζουν ανθρώπους. Επίσης η λέξη *κορίτσι* μπορεί βρίσκεται κοντά στη λέξη *γυναίκα*. Δεν σημαίνει όμως αυτό απαραίτητα ότι η λέξη *κορίτσι* θα είναι πιο κοντά στη λέξη *γυναίκα* από τη λέξη *άντρας*, καθώς ίσως η διάσταση που αφορά το φύλο του ανθρώπου να μην είναι αρκετά ισχυρή και η λέξη άντρας να εμφανίζεται πιο συχνά κοντά σε κοινές λέξεις με τη λέξη γυναίκα.

Είναι γεγονός, πως με τη χρήση πολύ μεγάλων κειμένων για την εκπαίδευση μοντέλων word2vec επιτυγχάνονται καλά αποτελέσματα με κάποιες όμως εξαιρέσεις. Στις περιπτώσεις που αφορούν άγνωστες λέξεις, για τις οποίες δεν έχει δημιουργηθεί κάποιο διάνυσμα, δεν υπάρχει δυνατότητα να συσχετιστούν με άλλες λέξεις. Μία παρόμοια υλοποίηση με αυτή του word2vec είναι αυτή του FastText, η οποία λύνει το πρόβλημα με τις άγνωστες λέξεις. Το FastText έχει προταθεί από το Facebook και η βασική διαφοροποίηση που έχει από το word2vec είναι ότι τα διανύσματα δεν δημιουργούνται σε επίπεδο λέξεων αλλά σε επίπεδο συλλαβών. Αυτό επιτρέπει το συσχετισμό άγνωστων λέξεων με άλλες που έχουν παρόμοιες συλλαβές. Το αρνητικό είναι η αύξηση των υπολογιστικών απαιτήσεων για το χειρισμό όλων των συλλαβών.

2.4.2 Η υλοποίηση του αλγορίθμου

Για την εφαρμογή του αλγορίθμου word2vec προτείνεται η χρήση της βιβλιοθήκης gensim καθώς θα γίνει επεξεργασία τεράστιου όγκου δεδομένων και δεν συνίσταται υλοποίηση του αλγορίθμου εξ' αρχής χωρίς κάποιο συγκεκριμένο λόγο. Για τις ανάγκες της διπλωματικής θα γίνει μία σχετικά απλή υλοποίηση του αλγορίθμου με την αρχιτεκτονική Skip-gram στη γλώσσα Python με τη χρήση της βιβλιοθήκης tensorflow της Google για να μελετηθεί ο τρόπος λειτουργίας του. Η υλοποίηση του word2vec θα βασιστεί σε αντίστοιχο οδηγό του Aneesh Joshi [12]. Επίσης θα χρησιμοποιηθεί η βιβλιοθήκη numpy [13], η οποία προσφέρει λειτουργικότητα για επιστημονικούς υπολογισμούς.

Όπως φαίνεται στην Εικόνα 8, στις γραμμές 1 και 2 γίνεται η εισαγωγή των βιβλιοθηκών ενώ στις γραμμές 4 - 10 δηλώνεται η πρώτη βοηθητική μέθοδος που θα χρησιμοποιηθεί. Η μέθοδος ονομάζεται create_dictionary και αποσκοπεί στη δημιουργία ενός συνόλου μοναδικών λέξεων από ένα κείμενο.

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

Εικόνα 8: Υλοποίηση word2vec 1

Στη γραμμή 6 ορίζεται στη μεταβλητή words ένας κενός πίνακας, στον οποίο στις γραμμές 7 και 8 τοποθετούνται όλες οι λέξεις που υπάρχουν στο κείμενο. Στη γραμμή 7, μετατρέπονται όλες οι λέξεις του κειμένου σε πεζές, αντικαθίστανται οι τελείες με κενούς χαρακτήρες και στη συνέχεια με τη μέθοδο split χωρίζονται οι λέξεις. Στη γραμμή 9, με την κλήση της μεθόδου set, δημιουργείται ένα σύνολο από τις λέξεις του κειμένου, δηλαδή η κάθε λέξη εμφανίζεται μόνο μία φορά.

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

Εικόνα 9: Υλοποίηση word2vec 2

Στη συνέχεια, στην Εικόνα 9, παρουσιάζονται οι άλλες δύο βοηθητικές μέθοδοι της υλοποίησης. Στις γραμμές 12 - 17 δηλώνεται η μέθοδος get_sentences, η οποία σπάει το συνολικό κείμενο σε μία λίστα από προτάσεις. Στη γραμμή 15 το κείμενο χωρίζεται στο χαρακτήρα '.', ο οποίος χωρίζει και κάθε πρόταση από τις υπόλοιπες. Η γραμμή 16, για κάθε πρόταση, προσθέτει στον πίνακα sentences ένα νέο πίνακα με τις λέξεις τις κάθε πρότασης σε πεζούς χαρακτήρες και τελικά η γραμμή 17 επιστέφει αυτό τον πίνακα.

Η μέθοδος create_vector που ορίζεται στις γραμμές 19 - 23 αφορά τη μετατροπή ενός αριθμού δείκτη σε διάνυσμα. Στη γραμμή 21 με τη μέθοδο np.zeros(dictionary_size) δημιουργείται ένα μηδενικό διάνυσμα με μέγεθος το μήκος του λεξικού. Για παράδειγμα, για μήκος 5 θα δημιουργηθεί το διάνυσμα [0, 0, 0, 0, 0]. Στη γραμμή 22 μετατρέπεται ο δείκτης της εισόδου στο διάνυσμα σε 1. Οπότε για είσοδο τον δείκτη 3 το διάνυσμα που θα επιστραφεί θα είναι το [0, 0, 0, 1, 0].

Στην Εικόνα 10, ξεκινά η υλοποίηση του αλγορίθμου. Στη γραμμή 26 ορίζεται ένα κείμενο είσοδος για τον αλγόριθμο. Στις γραμμές 28 και 29 ορίζονται δύο λεξικά τα οποία θα χρησιμοποιηθούν για την απεικόνιση της κάθε λέξης με έναν αριθμό και το αντίστροφο. Στη γραμμή 31 δημιουργείται ένα λεξικό από το σύνολο του κειμένου, το μέγεθος του οποίου αποθηκεύεται στη μεταβλητή vocabulary_size στη γραμμή 33. Στις γραμμές 37 και 38 για κάθε λέξη στον πίνακα - λεξικό γίνεται μία καταχώρηση στα δύο άλλα λεξικά.

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

Εικόνα 10: Υλοποίηση word2vec 3

Με το πέρας της επανάληψης, αν για παράδειγμα μία λέξη βρίσκεται στη θέση 3 στο λεξικό, με την εντολή integerForWord[λέξη] επιστρέφεται ο αριθμός 3 ενώ αντίθετα με την wordForInteger[3] επιστρέφεται η λέξη. Στη γραμμή 40 με την κλήση της get_sentences ορίζονται στη μεταβλητή sentences οι προτάσεις του κειμένου.

Ο επόμενος κώδικας αφορά τη δημιουργία ζευγαριών από τις λέξεις που εμφανίζονται σε κάθε πρόταση. Όπως φαίνεται στην Εικόνα 11, στη γραμμή 42 ορίζεται το μέγεθος του παραθύρου αναζήτησης στον αριθμό 2. Στις γραμμές 46 - 52 δημιουργούνται ζευγάρια λέξεων με βάση το μέγεθος παραθύρου. Για παράδειγμα, για μέγεθος παραθύρου 2, για κάθε λέξη δημιουργείται από ένα ζευγάρι με τη ίδια τη λέξη και τις 4 διπλανές.

```
42  WINDOW_SIZE = 2
43
44  # Create pairs for each word with WINDOW_SIZE in mind
45  pairs = []
46  for sentence in sentences:
47      for j, word in enumerate(sentence):
48          leftBoundary = max(j - WINDOW_SIZE, 0)
49          rightBoundary = min(j + WINDOW_SIZE, len(sentence)) + 1
50          for nearby in sentence[leftBoundary : rightBoundary]:
51              if nearby != word:
52                  pairs.append([word, nearby])
```

```
05 = {list} <class 'list'>: ['goes', 'a']
06 = {list} <class 'list'>: ['goes', 'boy']
07 = {list} <class 'list'>: ['goes', 'to']
08 = {list} <class 'list'>: ['goes', 'school']
```

Εικόνα 11: Υλοποίηση word2vec 4

Θα δημιουργηθούν δηλαδή 4 ζευγάρια. Όπως φαίνεται στο κάτω μέρος της εικόνας, για τη λέξη goes, χρησιμοποιώντας παράθυρο μεγέθους 2, οι λέξεις είναι “a boy goes to school”. Οπότε δημιουργούνται 4 ζευγάρια με πρώτη λέξη τη goes και δεύτερη μία από τις υπόλοιπες για κάθε ζευγάρι. Αυτός είναι ο τρόπος που αποτυπώνεται η πληροφορία των λέξεων που βρίσκονται κοντά στη λέξη goes.

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

Εικόνα 12: Υλοποίηση word2vec 5

Στην Εικόνα 12 γίνεται η προετοιμασία των δεδομένων που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου. Στις γραμμές 54 και 55 δηλώνονται δύο πίνακες και στη συνέχεια, στις γραμμές 59 και 60 αποθηκεύονται σε αυτούς τα διανύσματα για κάθε ζευγάρι λέξεων. Συγκεκριμένα για κάθε λέξη με το integerForWord εντοπίζεται ο αριθμός της στο λεξικό και με το create_vector δημιουργείται ένα διάνυσμα για αυτόν τον αριθμό. Στον πίνακα input καταχωρούνται οι πρώτες λέξεις των ζευγαριών δηλαδή οι λέξεις για τις οποίες γίνεται η αναζήτηση των γειτονικών λέξεων. Στον πίνακα output αποθηκεύονται οι δεύτερες λέξεις, οι οποίες αποτελούν τις αναμενόμενες γειτονικές λέξεις. Τελικά στις γραμμές 63 και 64 οι πίνακες μετατρέπονται σε μορφή numpy και αποθηκεύονται στις μεταβλητές x_train και y_train. Αυτές οι μεταβλητές θα είναι και οι είσοδοι στο tensorflow για την εκπαίδευση του μοντέλου.

Στην Εικόνα 13, γίνεται η προετοιμασία για την εκτέλεση του tensorflow. Δύο βασικές τύποι τιμών στο tensorflow είναι τα placeholder και τα Variable. Ο τύπος placeholder αφορά την κατοχύρωση του σημείου στο οποίο θα γίνεται η εισαγωγή των δεδομένων εκπαίδευσης από εξωτερική πηγή. Στο παράδειγμα, στις γραμμές 68 και 69 δημιουργούνται δύο placeholders, ένα για τα δεδομένα εισόδου, τη λέξη για την οποία ψάχνουμε γειτονικές λέξεις και ένα για τα δεδομένα εξόδου, μία γειτονική λέξη για την προηγούμενη. Ο τύπος Variable ορίζει μεταβλητή της οποίας την τιμή θα ορίσει το tensorflow και θα μεταβάλλεται κατά της εκπαίδευση για την καλύτερη απεικόνιση των δεδομένων.

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

Εικόνα 13: Υλοποίηση word2vec 6

Στη συνέχεια, ορίζεται η συνάρτηση απεικόνισης του μοντέλου το οποίο χρησιμοποιεί 5 διαστάσεις για το διάνυσμα της κάθε λέξης που συναντά στο κείμενο. Για το word2vec αυτή είναι: “x * W1 + b1”. Το W1 είναι ο πίνακας που περιέχει τα διανύσματα για κάθε λέξη, μία λέξη ανά σειρά. Το x όντας το διάνυσμα για μία λέξη, λειτουργεί ως πίνακας αναζήτησης για τη λέξη αυτή. Το b1 αφορά σταθερά. Η συνάρτηση ορίζεται στη μεταβλητή representation. Ομοίως στις γραμμές 78 - 80 ορίζεται η συνάρτηση πρόβλεψης για μία λέξη. Συγκεκριμένα, εμπεριέχει κλήση στη συνάρτηση softmax, η οποία αναφέρεται στη βιβλιογραφία [14]. Ο υπολογισμός περιλαμβάνει τον πολλαπλασιασμό της συνάρτησης αναπαράστασης με τον πίνακα W2 και τη προσθήκη μιας σταθεράς b2. Δεν θα γίνει περαιτέρω αναφορά στην λειτουργία της συνάρτησης softmax καθώς δεν αφορά τη συγκεκριμένη έρευνα.

Στην Εικόνα 14, παρουσιάζεται η εκτέλεση του tensorflow για την εκπαίδευση του μοντέλου. Οι σειρές 83 - 85 αφορούν εντολές αρχικοποίησης του tensorflow. Στη γραμμή 88 ορίζεται η συνάρτηση απώλειας του μοντέλου. Στη γραμμή 91 ορίζεται το κάθε βήμα που θα εκτελείται κατά την εκπαίδευση. Σκοπός είναι η ελαχιστοποίηση της συνάρτησης απώλειας έτσι ώστε το μοντέλο να αντιπροσωπεύει όσο το δυνατό καλύτερα τα δεδομένα. Στη γραμμή 93 ορίζονται 1000 επαναλήψεις για την εκπαίδευση του μοντέλου και στη γραμμή 97 γίνεται η τελική εκπαίδευση του μοντέλου για κάθε επανάληψη με εισόδους τους πίνακες x_train και y_train που αναφέρθηκαν νωρίτερα. Τελικά, με το πέρας της εκπαίδευσης, αποθηκεύονται τα τελικά διανύσματα του μοντέλου στη μεταβλητή vectors, μετά από την προσθήκη του πίνακα b1 στον πίνκα W1.

```
83  session = tf.Session()
84  init = tf.global_variables_initializer()
85  session.run(init)
86
87  # Loss function
88  loss_function = tf.reduce_mean(-tf.reduce_sum(y * tf.log(prediction), reduction_indices=[1]))
89
90  # Minimize the loss function in each step
91  train_step = tf.train.GradientDescentOptimizer(0.1).minimize(loss_function)
92
93  iterations = 1000
94
95  # train with the train data on the placeholders
96  for k in range(iterations):
97      session.run(train_step, feed_dict={x: x_train, y: y_train})
98      print('Iteration: ', k, ': loss is : ', session.run(loss_function, feed_dict={x: x_train, y: y_train}))
99
100 # get the vectors after training
101 vectors = session.run(W1 + b1)
```

Εικόνα 14: Υλοποίηση word2vec 7

Στην Εικόνα 15, παρουσιάζεται μία μέθοδος που χρησιμοποιήθηκε για τη δοκιμή του μοντέλου. Ο κώδικας στις γραμμές 105 - 114 αφορά την υλοποίηση του υπολογισμού της απόστασης δύο διανυσμάτων με σκοπό τον εντοπισμό του κοντινότερου διανύσματος, για το δοθέν διάνυσμα από ένα πίνακα διανυσμάτων.

```
103 # simple method to find nearest with euclidean distance
104 def find_closest(index, vectors):
105     min_dist = 10000 # to act like positive infinity
106     min_index = -1
107     query_vector = vectors[index]
108     for v, vector in enumerate(vectors):
109         if not np.array_equal(vector, query_vector):
110             distance = np.sqrt(np.sum((vector-query_vector)**2))
111             if distance < min_dist:
112                 min_dist = distance
113                 min_index = v
114     return min_index
115
116
117 print('Output -> ', wordForInteger[find_closest(integerForWord['boy'], vectors)])
Output ->  girl
```

Εικόνα 15: Υλοποίηση word2vec 8

Τελικά, στη γραμμή 117 καλείται η μέθοδος με είσοδο τη λέξη boy και επιστρέφει ως κοντινότερο διάνυσμα, αυτό της λέξης girl, σύμφωνα πάντα με το κείμενο που δόθηκε ως είσοδος κατά τη δημιουργία του μοντέλου.

2.4.3 Αξιολόγηση μοντέλων word2vec

2.4.3.1 Η ανάγκη για αξιολόγηση

Ένα σημαντικό ζήτημα με τη δημιουργία μοντέλων για επεξεργασία φυσικής γλώσσας είναι η αξιολόγηση τους. Καθώς τα μοντέλα επεξεργάζονται τεράστια μεγέθη δεδομένων και δημιουργούν διανύσματα πολύ μεγάλων διαστάσεων, καθίσταται πολύ δύσκολη έως αδύνατη η ανθρώπινη αξιολόγηση τους. Υπάρχει ανάγκη για τεκμηριωμένες διαδικασίες, οι οποίες θα πρέπει να είναι σε θέση να αξιολογούν την αποτελεσματικότητα ενός μοντέλου και να την εκφέρουν με αριθμητικό τρόπο ώστε να είναι δυνατή η σύγκριση μοντέλων μεταξύ τους. Έχουν προταθεί διάφορες μέθοδοι αξιολόγησης και οι κυριότερες θα μελετηθούν στη συνέχεια.

Οι μέθοδοι αξιολόγησης χωρίζονται σε δύο κατηγορίες, τις μεθόδους εσωτερικής αξιολόγησης και τις μεθόδους εξωτερικής αξιολόγησης [15]. Οι μέθοδοι εσωτερικής αξιολόγησης βασίζονται στον έλεγχο του ίδιου του μοντέλου από μία συλλογή ερωτημάτων και έτοιμων αποτελεσμάτων που αφορούν συντακτικές και σημασιολογικές σχέσεις ανάμεσα στις λέξεις. Το μοντέλο κάνει προβλέψεις για τα ερωτήματα που δέχεται και τα αποτελέσματα των προβλέψεων συγκρίνονται με τα αναμενόμενα, δίνοντας μία απόλυτη αριθμητική μέτρηση για την αποτελεσματικότητα του μοντέλου. Οι μέθοδοι εξωτερικής αξιολόγησης χρησιμοποιούν το μοντέλο για την εκτέλεση εφαρμογών βασισμένων στην επεξεργασία φυσικής γλώσσας και η αξιολόγηση του γίνεται με γνώμονα την αποτελεσματικότητα του στις εφαρμογές αυτές. Αξίζει να σημειωθεί πως διαφορετικά μοντέλα μπορεί να αξιολογούνται διαφορετικά με διαφορετικές μεθόδους αξιολόγησης. Για παράδειγμα, ένα μοντέλο μπορεί να πετυχαίνει πολύ υψηλές επιδόσεις σε κάποια εξωτερική μέθοδο αξιολόγησης αλλά ταυτόχρονα κακές επιδόσεις σε κάποια εσωτερική μέθοδο. Η αξιολόγηση θα πρέπει να εκτελείται εν γνώση των παραπάνω και να βασίζεται σε μεθόδους που έχουν νόημα για το σκοπό που δημιουργήθηκε το μοντέλο.

2.4.3.2 Μέθοδοι εσωτερικής αξιολόγησης

Στις μεθόδους εσωτερικής αξιολόγησης μπορούν να χρησιμοποιηθούν απόλυτα ή συγκριτικά μέτρα αξιολόγησης. Τα απόλυτα μέτρα αφορούν τη δοκιμή του μοντέλου με έτοιμα στατικά δεδομένα. Τα συγκριτικά μέτρα δέχονται τη συμμετοχή εξωτερικών παρατηρητών στην αξιολόγηση του μοντέλου, όπως για παράδειγμα τη συμπλήρωση ερωτηματολογίων, όπου ζητείται η επιλογή της αναμενόμενης εξόδου του μοντέλου, για κάποια είσοδο. Οι κατηγορίες ελέγχων που μπορούν να χρησιμοποιηθούν για εσωτερική αξιολόγηση συμπεριλαμβάνουν τη σχετικότητα, την αναλογία, την εκλεκτική προτίμηση και τη συνοχή [16].

Η σχετικότητα (Relatedness) αναφέρεται στο βαθμό συγγένειας δύο λέξεων. Τα μοντέλα word2vec μπορούν να εκφέρουν με αριθμητική τιμή τη σχετικότητα ανάμεσα σε δύο λέξεις με την τιμή 0 να δηλώνει μηδενική συγγένεια και την τιμή 1 απόλυτη συγγένεια. Για παράδειγμα, η σχετικότητα μεταξύ των λέξεων *αγόρι* και *κορίτσι* θα είχε μία τιμή κοντά στο 1, ενώ η σχετικότητα μεταξύ των λέξεων *αγόρι* και *πέτρα* μία τιμή κοντά στο 0. Για να επιτευχθεί η αξιολόγηση, χρησιμοποιείται μία έτοιμη λίστα από τέτοια παραδείγματα με τις αντίστοιχες συσχετίσεις και μετράται η απόκλιση του μοντέλου για κάθε ένα από αυτά. Με το συνδυασμό των αποτελεσμάτων, προκύπτει μία συνολική εικόνα για την επίδοση του μοντέλου.

Η αναλογία (Analogy) αντιπροσωπεύει την δυνατότητα του μοντέλου να επιλύει προβλήματα σχέσεων λέξεων. Το μοντέλο εκτελώντας αλγεβρικούς υπολογισμούς στα διανύσματα των λέξεων, έχει τη δυνατότητα, με δεδομένη μία σχέση λέξης Α με μία λέξη Β, να προβλέψει μία λέξη Γ η οποία να σχετίζεται με ανάλογο τρόπο με μία λέξη Δ. Ένα παράδειγμα τέτοιας αναλογίας είναι η ακόλουθη. "Η λέξη *κοντά* σχετίζεται με τη λέξη *μακριά* όπως η λέξη *μικρός* σχετίζεται με τη λέξη ***μεγάλος***". Εδώ η αξιολόγηση αφορά τα απόλυτα ποσοστά επιτυχίας του μοντέλου για κάθε ερώτημα. Η επιλογή των ερωτημάτων πρέπει να πραγματοποιηθεί με προσοχή καθώς υπάρχουν λέξεις που μπορούν χρησιμοποιηθούν με διαφορετικές σημασιολογικές έννοιες, γεγονός που μπορεί να επηρεάσει τις προβλέψεις του μοντέλου.

Η εκλεκτική προτίμηση (Selectional preference) αφορά την τάση που έχουν κάποιες λέξεις να εμφανίζονται κοντά με κάποιο συγκεκριμένο συντακτικό τρόπο. Μερικά ουσιαστικά μπορεί να εμφανίζονται συχνότερα ως υποκείμενα ενός ρήματος και λιγότερο συχνά ως αντικείμενα αυτού. Ένα παράδειγμα είναι οι λέξεις *άνθρωπος* και

διαβάζω. Το αναμενόμενο είναι η λέξη άνθρωπος, να χαρακτηριστεί ως υποκείμενο για τη λέξη διαβάζω και όχι ως αντικείμενο. Αυτό συμβαίνει γιατί ένας άνθρωπος είναι πιθανότερο να διαβάζει ένα αντικείμενο (πχ ένα βιβλίο) παρά ένα άλλο υποκείμενο να διαβάζει έναν άνθρωπο. Με τον έλεγχο τέτοιων παραδειγμάτων μπορεί να μετρηθεί με απόλυτο αριθμητικό τρόπο το ποσοστό επιτυχίας του μοντέλου και να αξιολογηθεί αντίστοιχα.

Η συνοχή (Coherence) προϋποθέτει πως λέξεις που βρίσκονται σημασιολογικά κοντά, πρέπει να βρίσκονται και σε γειτονικές θέσεις στο διανυσματικό χώρο. Ένα καλό μοντέλο πρέπει να διατηρεί συνοχή στις γειτονικές λέξεις. Για παράδειγμα, με είσοδο ενός αριθμού λέξεων, όπου όλες εκτός από μία αφορούν ονομασίες χρωμάτων, η άσχετη λέξη πρέπει να μπορεί να εντοπιστεί εύκολα από το μοντέλο. Με αυτό τον τρόπο μπορεί να γίνει και η αξιολόγηση της συνοχής ενός μοντέλου.

Στην περίπτωση των απόλυτων μέτρων αξιολόγησης, η λίστα ερωτημάτων πρέπει να πληρεί κάποια κριτήρια για να είναι ποιοτική. Οι λέξεις που περιέχει πρέπει να εμφανίζονται με συγκεκριμένη συχνότητα στην εκάστοτε γλώσσα, αν για παράδειγμα, στόχος είναι η αξιολόγηση ενός μοντέλου στις επιδόσεις του σε σπάνιες λέξεις, είναι λογική η επιλογή λέξεων μικρής συχνότητας. Επίσης οι λέξεις πρέπει να ανήκουν σε συγκεκριμένα μέρη του λόγου. Η επιλογή λέξεων που είναι άρθρα, είναι αναμενόμενο να μην εκφέρει σωστά αποτελέσματα. Τέλος, οι λέξεις πρέπει να είναι σημασιολογικά συγκεκριμένες και όχι αφηρημένες.

Προσοχή απαιτείται, κατά τη συλλογική αξιολόγηση του μοντέλου με τη συσσωμάτωση αποτελεσμάτων διαφορετικών μεθόδων. Οι μέθοδοι εσωτερικής αξιολόγησης δεν παράγουν συγκρίσιμα μεταξύ τους αποτελέσματα. Μία επιτυχία 70% σε έναν έλεγχο αναλογιών δεν ισούται με μία επιτυχία 70% σε έναν έλεγχο σχετικότητας. Έτσι, με γνώμονα τις ανάγκες και τη χρήση του μοντέλου, πρέπει να τεθούν βάρη στις μεθόδους αξιολόγησης για να ληφθεί μία συνολικότερη εικόνα για το μοντέλο.

2.4.3.3 Μέθοδοι εξωτερικής αξιολόγησης

Οι μέθοδοι εξωτερικής αξιολόγησης ελέγχουν την απόδοση του μοντέλου σε κάποια συγκεκριμένη εφαρμογή επεξεργασίας φυσικής γλώσσας. Τέτοιες εφαρμογές

περιλαμβάνουν την ανάλυση συναισθημάτων, την ανάλυση εξαρτήσεων, την αναγνώριση οντοτήτων και την αντιστοίχηση λέξεων.

Η ανάλυση συναισθημάτων (Sentiment Analysis) αφορά τον εντοπισμό του χαρακτήρα μιας πρότασης. Λαμβάνοντας υπόψη τις θετικές και τις αρνητικές λέξεις που βρίσκονται στην πρόταση, πρέπει να είναι δυνατός ο χαρακτηρισμός της πρότασης ως θετική, αρνητική ή αδιάφορη. Με την παροχή προτάσεων και αναμενόμενων χαρακτηρισμών, το μοντέλο μπορεί να αξιολογηθεί σε μία εφαρμογή ανάλυσης συναισθημάτων, ως προς το ποσοστό επιτυχίας του σε αυτές.

Η ανάλυση εξαρτήσεων (Dependancy Parsing) αναφέρεται στον καθορισμό εξαρτήσεων λέξεων σε μία πρόταση. Για παράδειγμα, μία τέτοια εφαρμογή πρέπει να είναι σε θέση να εντοπίζει το υποκείμενο και το αντικείμενο ενός ρήματος. Η υλοποίηση της ανάλυσης εξαρτήσεων περιλαμβάνει συνήθως μία δενδρική δομή, στην οποία αποτυπώνονται οι εξαρτήσεις της πρότασης. Ένα μοντέλο που χρησιμοποιείται σε μία τέτοια εφαρμογή, μπορεί να αξιολογηθεί ως προς το ποσοστό επιτυχίας του στη συγκεκριμένη εργασία.

Η αναγνώριση οντοτήτων (Entity Recognition) υποδηλώνει μία εφαρμογή η οποία εντοπίζει οντότητες σε μία πρόταση. Προαπαιτείται καθορισμός της δομής της πρότασης, δηλαδή ορισμός των οντοτήτων που είναι πιθανόν αυτή να περιέχει και στη συνέχεια εντοπισμό αυτών σε άλλες προτάσεις. Η αναγνώριση οντοτήτων βρίσκει εφαρμογή στην κατανόηση κειμένου και στους ψηφιακούς βοηθούς, καθώς συνεισφέρει στην αποσαφήνιση των εντολών ενός χρήστη. Για παράδειγμα, μπορεί να οριστεί ότι μία πρόταση περιέχει έναν τομέα μιας εφαρμογής (πχ φώτα σε μία εφαρμογή ελέγχου έξυπνου σπιτιού), μία ενέργεια (πχ άνοιγμα ή κλείσιμο) και μία τοποθεσία (πχ κουζίνα). Η εφαρμογή πρέπει να είναι σε θέση στην πρόταση “κλείσε τα φώτα στο μπάνιο” να εντοπίσει ως τομέα τη λέξη *φώτα*, ως ενέργεια τη λέξη *κλείσε* και ως τοποθεσία τη λέξη *μπάνιο*. Ορίζοντας τέτοια παραδείγματα μπορεί να οριστεί το ποσοστό επιτυχίας του μοντέλου, το οποίο συνιστά αξιολόγηση του.

Η αντιστοίχηση λέξεων (Coreference Resolution) αφορά τη σωστή σύνδεση λέξεων με άλλες λέξεις σε μία πρόταση. Για παράδειγμα, στην πρόταση “Ο Κώστας έδειξε στο Νίκο τον κήπο του”, πρέπει να εντοπιστεί ότι το άρθρο “του” αναφέρεται στον Κώστα και όχι στο Νίκο. Με την χρήση ενός μοντέλου σε μία τέτοια εφαρμογή, μπορεί αυτό να αξιολογηθεί ως προς τα ποσοστά επιτυχίας του.

Είναι σημαντικό να γίνει αντιληπτό πως οι μέθοδοι εξωτερικής αξιολόγησης, αξιολογούν ένα μοντέλο σε πολύ συγκεκριμένες συνθήκες και δεν πρέπει να χρησιμοποιούνται ως γενικό κριτήριο αξιολόγησης για το μοντέλο. Είναι δόκιμο, να χρησιμοποιηθεί ως βασικό κριτήριο η αναγνώριση οντοτήτων, για ένα μοντέλο που αποσκοπεί σε αυτή τη λειτουργία, όχι όμως και σε ένα μοντέλο που θα χρησιμοποιείται για ανάλυση συναισθημάτων, καθώς τα αποτελέσματα του ίδιου μοντέλου σε αυτά τα δύο διαφορετικά είδη εργασιών μπορεί να έχουν μεγάλη απόκλιση. Τελικά, σημαντικό ρόλο παίζει η σωστή επιλογή μεθόδων αξιολόγησης για ένα μοντέλο και η σωστή αξιοποίηση των αποτελεσμάτων αυτών.

3 Δεδομένα για εκπαίδευση νευρωνικών δικτύων

3.1 Πηγές δεδομένων για επεξεργασία φυσικής γλώσσας

Ο βασικότερος παράγοντας που ορίζει την απόδοση ενός συστήματος επεξεργασίας φυσικής γλώσσας, είναι το μέγεθος και η ποιότητα των δεδομένων που θα χρησιμοποιηθούν για την εκπαίδευση αυτού. Παρά το γεγονός της συνεχής βελτιστοποίησης των αλγορίθμων που αναλαμβάνουν τη δημιουργία μοντέλων φυσικής γλώσσας, η χρήση περισσότερων ποιοτικών δεδομένων, θα βελτιώσει την απόδοση του συστήματος [17].

Υπάρχει πληθώρα διαθέσιμων πηγών δεδομένων στο διαδίκτυο για επεξεργασία φυσικής γλώσσας [18]. Οι περισσότερες από αυτές αφορούν συλλογές λογοτεχνικού περιεχομένου, δεδομένα από εφημερίδες, εγκυκλοπαίδειες όπως και περιεχόμενο από το διαδίκτυο. Οι περισσότερες πηγές παρέχουν δεδομένα για την αγγλική γλώσσα, με κάποιες από αυτές να παρέχουν περιεχόμενο και σε άλλες γλώσσες αλλά όχι στην ποσότητα της αγγλικής. Δεδομένου ότι για τη συγκεκριμένη έρευνα είναι αναγκαία η δημιουργία μοντέλων για την αγγλική και την ελληνική γλώσσα, οι επιλογές είναι αρκετά περιορισμένες.

Για να είναι λογική η σύγκριση των μοντέλων των δύο γλωσσών, τα δεδομένα θα πρέπει να είναι ανάλογης ποιότητας και ποσότητας. Δεδομένης της κατάστασης για την ελληνική γλώσσα η καλύτερη επιλογή είναι να γίνει χρήση δεδομένων από τη Wikipedia. Η Wikipedia παρέχει μεγάλη ποσότητα άρθρων σε οποιαδήποτε γλώσσα [19]. Για τα αγγλικά αυτή τη στιγμή, είναι διαθέσιμα πάνω από 5.000.000 άρθρα ενώ για τα ελληνικά σχεδόν 150.000 άρθρα. Είναι εμφανής η μεγάλη απόκλιση των δύο γλωσσών σε μέγεθος περιεχομένου. Παρόλα αυτά, ακόμη και τα 150.000 άρθρα είναι αρκετά για τις ανάγκες αυτής της έρευνας. Για να είναι συγκρίσιμα τα αποτελέσματα θα γίνει επιλογή μέρους του συνόλου των άρθρων για την αγγλική γλώσσα και όχι συνολική χρήση αυτών. Η επιλογή θα γίνει έτσι ώστε, ο αριθμών των συνολικών λέξεων της αγγλικής γλώσσας να προσεγγίζει τον αριθμό λέξεων της ελληνικής.

3.2 Στιγμιότυπα της Wikipedia

Η Wikipedia προσφέρει τη δυνατότητα λήψης αντιγράφων της βάσης δεδομένων της από διάφορες στιγμές στο χρόνο [20]. Η λήψη των αντιγράφων μπορεί να πραγματοποιηθεί μέσα από την ιστοσελίδα της Wikimedia [21], η οποία αποτελεί υπερσύνολο της Wikipedia και άλλων σχετικών προσπαθειών διανομής ελεύθερης γνώσης, όπως για παράδειγμα το WikiNews και το WikiBooks.

Είναι δυνατή η λήψη διαφόρων στοιχείων της Wikipedia για κάθε γλώσσα. Κάποια από τα σημαντικότερα είναι το σύνολο των άρθρων, σύνολα περιλήψεων, λίστα με όλους τους τίτλους, συνδέσεις μεταξύ άρθρων, αρχείο μεταβολών για όλα τα άρθρα καθώς και άλλα πολλά στοιχεία τα οποία μπορούν να φανούν χρήσιμα σε συγκεκριμένου τύπου εργασίες. Στην Εικόνα 16 παρουσιάζεται η σελίδα λήψης του ελληνικού αντιγράφου [22] και συγκεκριμένα το σύνολο των άρθρων αυτού.

Εικόνα 16: Σελίδα λήψης ελληνικού αντιγράφου Wikipedia

Μετά την απόκτηση των αναγκαίων δεδομένων από τη Wikipedia πρέπει να πραγματοποιηθεί η αναγκαία επεξεργασία πάνω σε αυτά ώστε να είναι έτοιμα για χρήση με το word2vec. Για τις ανάγκες της συγκεκριμένης έρευνας επιλέχθηκε το σύνολο των άρθρων για την ελληνική γλώσσα και ένα υποσύνολο αυτού για την αγγλική. Στη συνέχεια θα αναλυθεί η διαδικασία και τα εργαλεία που μπορούν να χρησιμοποιηθούν για την ανάλυση των δεδομένων αυτών.

3.3 Επεξεργασία των δεδομένων

Ο τρόπος που επιλέχθηκε για τη λήψη των δεδομένων οδηγεί σε ένα τεράστιο έγγραφο xml όπου βρίσκονται αποτυπωμένα όλα τα άρθρα της Wikipedia. Στην Εικόνα 17 φαίνεται ο τρόπος που καταγράφεται το κάθε άρθρο.

```
<page>
   <title>Λευκωσία</title>
   <ns>0</ns>
   <id>32</id>
   <revision>
      <id>6991029</id>
      <parentid>6973762</parentid>
      <timestamp>2018-04-13T09:00:51Z</timestamp>
      <contributor>
         <ip>xxx.xxx.xxx.xxx</ip>
      </contributor>
      <comment>Η μόνη διχοτομημένη πρωτεύουσα στην Ευρώπη, όχι στον κόσμο.</comment>
      <model>wikitext</model>
      <format>text/x-wiki</format>
      <text xml:space="preserve">...</text>
      <sha1>i64yvjs7mhvr589m6y405gwbu7xird1</sha1>
   </revision>
</page>
```

Εικόνα 17: Το άρθρο για τη Λευκωσία στο αντίγραφο της Wikipedia

Για κάθε άρθρο συμπεριλαμβάνονται πολλές πληροφορίες όπως για παράδειγμα ο τίτλος, ο αριθμός αναθεώρησης, η χρονική στιγμή δημοσίευσης, το άτομο που έκανε τη δημοσίευση, μία σύνοψη (digest) με τον αλγόριθμο SHA1 όπως επίσης και το κείμενο του άρθρου. Το χρήσιμο μέρος για τη δημιουργία του μοντέλου word2vec είναι το κείμενο του άρθρου.

Να σημειωθεί πως το κείμενο του κάθε άρθρου περιέχει πολλά σημεία τα οποία δεν πρέπει να υπάρχουν κατά τη δημιουργία και εκπαίδευση του μοντέλου, όπως για παράδειγμα σύμβολα και αρκετά σημεία σήμανσης που υπάρχουν μέσα σε αυτό, για τον ορισμό της μορφοποίησης και των περιεχομένων του εκάστοτε άρθρου. Απαιτείται λοιπόν μία προ επεξεργασία των δεδομένων πριν αυτά χρησιμοποιηθούν από το μοντέλο. Μία λύση είναι η εξαρχής δημιουργία ενός αναλυτή (parser), ο οποίος θα αναλάβει αυτή την επεξεργασία. Υπάρχει όμως ήδη μία γνωστή βιβλιοθήκη στην γλώσσα Python η οποία προσφέρει πολλά εργαλεία, χρήσιμα στην επεξεργασία φυσικής γλώσσας. Αυτή η βιβλιοθήκη ονομάζεται gensim [23] και θα χρησιμοποιηθεί σε αυτή την έρευνα για την προ επεξεργασία των κειμένων πριν από τη δημιουργία του μοντέλου όπως επίσης και για την αποδοτική υλοποίηση της πάνω στον αλγόριθμο word2vec, η οποία θα χρησιμοποιηθεί για τη δημιουργία του μοντέλου και για την αξιολόγηση του.

Η βιβλιοθήκη gensim περιέχει μία λίστα πακέτων για τη δημιουργία συλλογής δεδομένων και ένα από αυτά είναι το WikiCorpus, το οποία αναλαμβάνει την επεξεργασία δεδομένων από τη Wikipedia. Δέχεται ως όρισμα ένα αρχείο που περιέχει άρθρα της Wikipedia και ανάλογα με τις επιλογές που διαθέτει, μπορεί να επιστρέψει μόνο το αξιοποιήσιμο κείμενο από τα δεδομένα. Στην Εικόνα 18 παρουσιάζεται η υλοποίηση που έγινε στην Python για την προ επεξεργασία των δεδομένων.

```
1   from gensim.corpora import WikiCorpus
2
3   if __name__ == '__main__':
4       inputFileName = 'elwiki-20180101-pages-articles.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, 'w')
11
12      print("Process initiated")
13      for text in wiki.get_texts():
14          numberOfArticles += 1
15          output.write(' '.join(text).encode('utf-8') + '\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 18: Χρήση του πακέτου WikiCorpus για την εξαγωγή των κειμένων

Για τη χρήση του πακέτου προαπαιτείται η εισαγωγή του στο περιβάλλον της Python όπως φαίνεται στη γραμμή 1. Στη συνέχεια δηλώνονται τα αρχεία εισόδου και

εξόδου στις γραμμές 4 και 5. Το αρχείο εισόδου είναι αυτό που λήφθηκε από τη Wikipedia και το αρχείο εξόδου είναι το αρχείο στο οποίο θα αποθηκευτεί το τελικό κείμενο. Στη γραμμή 8 καλείται το πακέτο WikiCorpus το οποία αναλύει το σύνολο του κειμένου.

Αφού ολοκληρωθεί η ανάλυση, στη γραμμή 10 ανοίγει για εγγραφή το αρχείο εξόδου κειμένου. Στη γραμμή 13 με τη μέθοδο *get_texts()* υπάρχει πρόσβαση στο συνολικό επεξεργασμένο κείμενο σε επίπεδο προτάσεων. Για κάθε πρόταση, στη γραμμή 15 πραγματοποιείται εγγραφή αυτής στο αρχείο εξόδου. Στο τέλος το αρχείο κλείνει στη γραμμή 18. Κατά τη διάρκεια της εκτέλεσης παρουσιάζονται στο χρήστη και τα ανάλογα μηνύματα κατάστασης.

Στο τέλος της επεξεργασίας, έχει δημιουργηθεί το τελικό αρχείο δεδομένων, το οποίο δεν περιέχει σύμβολα και σε κάθε γραμμή του περιέχει μία πρόταση. Αυτή η διαδικασία πραγματοποιείται και για τις δύο γλώσσες. Τα δύο τελικά αρχεία θα χρησιμοποιηθούν στη συνέχεια, για τη δημιουργία και την εκπαίδευση των μοντέλων με τον αλγόριθμο word2vec στην Python.

4 Μοντέλα word2vec

4.1 Δημιουργία μοντέλου με τη βιβλιοθήκη gensim

4.1.1 Υλοποίηση σε Python

Για την υλοποίηση του αλγορίθμου word2vec και τη δημιουργία των μοντέλων για τις ανάγκες της έρευνας θα χρησιμοποιηθεί και πάλι η βιβλιοθήκη gensim. Η υλοποίηση της gensim για το word2vec [24] είναι αποδοτική και προσφέρει μία πιο αφαιρετική πρόσβαση στον αλγόριθμο word2vec και στις μεθόδους αξιοποίησης του.

Για τη δημιουργία ενός νέου μοντέλου αρκεί η χρήση του πακέτου Word2Vec από το gensim.models. Στην Εικόνα 19 παρουσιάζεται ο κώδικας που χρησιμοποιείται για την δημιουργία των μοντέλων της έρευνας.

```
9   from gensim.models import Word2Vec
10  from gensim.models.word2vec import LineSentence
11
12  # train the word2vec model from wikipedia data
13  if __name__ == '__main__':
14      program = os.path.basename(sys.argv[0])
15      logger = logging.getLogger(program)
16
17      logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s')
18      logging.root.setLevel(level=logging.INFO)
19      logger.info("running %s" % ' '.join(sys.argv))
20
21      # check and process input arguments
22      inp = "../wikipedia/parsed/wikipedia_el.text"
23      outp = "gr/wiki.el.skip.model"
24
25      model = Word2Vec(LineSentence(inp), sg=1, size=300, iter=1, window=5,
26                       min_count=10, workers=multiprocessing.cpu_count())
27
28      # trim unneeded model memory = use (much) less RAM
29      model.init_sims(replace=True)
30
31      model.save(outp)
```

Εικόνα 19: Εκπαίδευση ενός μοντέλου word2vec με τη βιβλιοθήκη gensim

Στη γραμμή 9 πραγματοποιείται η εισαγωγή του πακέτου Word2Vec ενώ στη γραμμή 10, η εισαγωγή του πακέτου LineSentence. Το πακέτο LineSentence θα χρησιμοποιηθεί για το διαχωρισμό των προτάσεων από το αρχείο που δημιουργήθηκε

από τη Wikipedia στο προηγούμενο κεφάλαιο. Στο αρχείο, είχε αποθηκευτεί η κάθε πρόταση, σε μία ξεχωριστή γραμμή και τώρα γίνεται η αντίστροφη διαδικασία. Δημιουργείται δηλαδή μία λίστα από προτάσεις διαβάζοντας το κείμενο ανά γραμμή. Αυτός είναι ο τρόπος που γίνεται η είσοδος το δεδομένων στο Word2Vec όπως θα παρουσιαστεί στη συνέχεια.

Στις γραμμές 22 και 23 ορίζονται τα αρχεία εισόδου και εξόδου. Το αρχείο εισόδου είναι αυτό που δημιουργήθηκε από τη Wikepedia ενώ το αρχείο εξόδου είναι το αρχείο στο οποίο θα αποθηκευτεί το μοντέλο αφού δημιουργηθεί. Στη συνέχεια καλείται το Word2Vec. Το πρώτο όρισμα είναι η λίστα με τις προτάσεις που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου και παράγονται από το LineSentence. Το όρισμα sg αφορά την αρχιτεκτονική που θα χρησιμοποιηθεί για τη δημιουργία του μοντέλου. Με τον αριθμό 0 επιλέγεται η αρχιτεκτονική CBOW ενώ με τον αριθμό 1 η αρχιτεκτονική Skip-gram. Στη συνέχεια το όρισμα size αφορά τον αριθμό των διαστάσεων που θα χρησιμοποιηθούν για τη δημιουργία του μοντέλου. Το όρισμα iter ορίζει τον αριθμό των επαναλήψεων που θα πραγματοποιηθούν κατά την εκπαίδευση του μοντέλου. Για παράδειγμα, με τον αριθμό 3 θα εκπαιδευτεί το μοντέλο πάνω στο ίδιο σύνολο κειμένων τρεις φορές. Το όρισμα window αφορά το μέγεθος παραθύρου, δηλαδή τον αριθμό των λέξεων που θα λαμβάνονται υπόψη πριν και μετά από κάθε λέξη. Με το όρισμα min_count είναι δυνατός ο περιορισμός των λέξεων, οι οποίες θα χρησιμοποιηθούν από το κείμενο. Για παράδειγμα, με τον αριθμό 10, όσες λέξεις εμφανίζονται λιγότερες από δέκα φορές, θα αφαιρεθούν από το κείμενο και δεν θα επηρεάσουν το μοντέλο. Επιλέγοντας μεγάλο αριθμό βελτιώνεται η απόδοση στις συχνές λέξεις ενώ με μικρό αριθμό στις σπανιότερες λέξεις. Τέλος, το όρισμα workers δηλώνει των αριθμό των νημάτων που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου. Να σημειωθεί ότι υπάρχουν πολλά ακόμα ορίσματα που μπορούν να χρησιμοποιηθούν με αυτά που αναλύθηκαν να είναι τα πιο σημαντικά.

Η γραμμή 29 αφορά βελτιστοποίηση στη χρήση της μνήμης ενώ στη γραμμή 31 αποθηκεύεται τελικά το μοντέλο στο αρχείο εξόδου. Να σημειωθεί πως οι επιλογές που έγιναν για τα ορίσματα είναι προσεγγιστικές και στο κεφάλαιο 5, κατά την αξιολόγηση, θα δημιουργηθούν πολλά μοντέλα με διαφορετικές ρυθμίσεις και θα μελετηθεί η επίδραση αυτών των ρυθμίσεων στην απόδοση τους.

Στη συνέχεια θα παρουσιαστεί η εκτέλεση του κώδικα και θα αναλυθούν τα δύο στάδια που περιλαμβάνει η εκτέλεση του. Στο πρώτο στάδιο γίνεται μία προ επεξεργασία

και στη συνέχεια, στο δεύτερο στάδιο η εκπαίδευση του μοντέλου και η αποθήκευση του για μελλοντική χρήση.

4.1.2 Εκτέλεση και δημιουργία των μοντέλων

Κατά την εκτέλεση του κώδικα ξεκινά το πρώτο στάδιο, το οποίο είναι η προ επεξεργασία του κειμένου. Λαμβάνοντας υπόψη την επιλογή του χρήστη για τον ελάχιστο αριθμό εμφανίσεων για κάθε λέξη, απορρίπτονται από το κείμενο όλες οι λέξεις που παρουσιάζουν μικρότερο αριθμό εμφανίσεων.

Αφού αναγνωσθεί όλο το κείμενο και μετρηθεί ο αριθμός εμφανίσεων για κάθε λέξη, διατηρούνται μόνο οι αναγκαίες λέξεις και από αυτές δημιουργείται το λεξικό που θα χρησιμοποιηθεί για την εκπαίδευση.

```
INFO : collecting all words and their counts
INFO : PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
INFO : PROGRESS: at sentence #10000, processed 8462971 words, keeping 363984 word types
INFO : PROGRESS: at sentence #20000, processed 14222356 words, keeping 504379 word types
INFO : PROGRESS: at sentence #30000, processed 19140639 words, keeping 602981 word types
INFO : PROGRESS: at sentence #40000, processed 23783774 words, keeping 691734 word types
INFO : PROGRESS: at sentence #50000, processed 28044615 words, keeping 763699 word types
INFO : PROGRESS: at sentence #60000, processed 32830420 words, keeping 840411 word types
INFO : PROGRESS: at sentence #70000, processed 37072737 words, keeping 909529 word types
INFO : PROGRESS: at sentence #80000, processed 41571283 words, keeping 975058 word types
INFO : PROGRESS: at sentence #90000, processed 45657605 words, keeping 1040376 word types
INFO : PROGRESS: at sentence #100000, processed 50092316 words, keeping 1107220 word types
INFO : PROGRESS: at sentence #110000, processed 54415081 words, keeping 1166561 word types
INFO : PROGRESS: at sentence #120000, processed 58664876 words, keeping 1224756 word types
INFO : PROGRESS: at sentence #130000, processed 62149025 words, keeping 1290656 word types
INFO : collected 1294954 word types from a corpus of 62487962 raw words and 130932 sentences
INFO : Loading a fresh vocabulary
INFO : min_count=10 retains 203879 unique words (15% of original 1294954, drops 1091075)
INFO : min_count=10 leaves 60224479 word corpus (96% of original 62487962, drops 2263483)

INFO : sample=0.001 downsamples 31 most-common words
INFO : downsampling leaves estimated 48694979 word corpus (80.9% of prior 60224479)
INFO : estimated required memory for 203879 words and 300 dimensions: 591249100 bytes
```

Εικόνα 20: Πρώτο στάδιο εκτέλεσης της δημιουργίας μοντέλου

Στην Εικόνα 20, παρουσιάζεται ένα στιγμιότυπο από το πρώτο στάδιο εκτέλεσης του αλγορίθμου. Το συνολικό κείμενο περιλαμβάνει γύρω στα 62 εκατομμύρια λέξεις με 1.2 εκατομμύρια από αυτές να είναι μοναδικές, πριν την επεξεργασία. Με τη διατήρηση των λέξεων με ελάχιστη συχνότητα εμφάνισης 10, περιορίζονται σε πολύ μεγάλο βαθμό

οι μοναδικές λέξεις, οι οποίες θα αποτελέσουν το λεξικό για την εκπαίδευση του μοντέλου. Όπως φαίνεται στην εικόνα, μόνο το 15% των λέξεων αυτών εμφανίζονται πάνω από δέκα φορές. Με αυτή τη μείωση, μειώνεται δραματικά το μέγεθος που θα χρειαστεί το μοντέλο όπως θα αναλυθεί στη συνέχεια και βελτιώνεται η επίδοση σε αυτό το 15% των λέξεων. Εντύπωση δημιουργεί και το γεγονός ότι αυτές οι λέξεις αποτελούν το 96% του κειμένου. Μπορεί δηλαδή να απορρίπτεται ένα μεγάλο μέρος των μοναδικών λέξεων, αυτό όμως συμβαίνει με τη διατήρηση σχεδόν του συνόλου του κειμένου. Αξίζει να σημειωθεί πως σε αυτό το 85% των λέξεων που απορρίπτονται, οι περισσότερες από αυτές μπορεί να είναι όροι, ακρωνύμια, λέξεις που δεν ανήκουν στη γλώσσα του κειμένου και γενικότερα πληροφορία η οποία δεν είναι απαραίτητη για τη σωστή λειτουργία του μοντέλου στην εκάστοτε γλώσσα.

Στη συνέχεια εφαρμόζεται περαιτέρω μείωση του κειμένου με τη διαγραφή λέξεων με πολύ μεγάλη συχνότητα εμφάνισης. Αυτή η συχνότητα ορίζεται στην παράμετρο sample, η οποία παραλείπεται από την Εικόνα 19 καθώς συστήνεται η χρήση της συχνότητας 0.001 για την πλειοψηφία των περιπτώσεων. Αυτός ο αριθμός σημαίνει ότι η κάθε λέξη δεν πρέπει να αποτελεί περισσότερο από το ένα χιλιοστό του συνολικού κειμένου. Στη συγκεκριμένη εκτέλεση, αυτές οι λέξεις είναι 31 και με την αφαίρεση τους μειώνεται το κείμενο κατά 20%. Λέξεις με τόσο μεγάλη συχνότητα εμφάνισης δεν παρέχουν αξιοποιήσιμη πληροφορία για το μοντέλο που θα δημιουργηθεί καθώς θα εμφανίζονται κοντά σε πολλές λέξεις οι οποίες μπορεί να μην σχετίζονται μεταξύ τους. Για παράδειγμα η λέξη ‘και’ μπορεί να εμφανίζεται πολλές φορές κοντά σε οποιαδήποτε λέξη χωρίς αυτό να δηλώνει κάποια σχέση σε οποιαδήποτε επίπεδο ανάμεσα στις λέξεις.

Με το πέρας λοιπόν του πρώτου σταδίου το σύνολο λέξεων βρίσκεται γύρω στα 48 εκατομμύρια και το λεξικό μοναδικών λέξεων περιλαμβάνει 203.879 λέξεις. Για κάθε μία από αυτές τις λέξεις θα δημιουργηθεί και ένα διάνυσμα. Με δεδομένο τον αριθμό των διαστάσεων, 300 στη συγκεκριμένη περίπτωση και το μέγεθος του λεξικού, μπορεί να υπολογιστεί με ακρίβεια το μέγεθος του τελικού μοντέλου.

```
INFO : training model with 4 workers on 203879 vocabulary and 300 features,
                        using sg=1 hs=0 sample=0.001 negative=5 window=5
INFO : PROGRESS: at 0.13% examples, 206345 words/s, in_qsize 8, out_qsize 0
INFO : PROGRESS: at 0.34% examples, 221868 words/s, in_qsize 7, out_qsize 0
INFO : PROGRESS: at 0.57% examples, 223769 words/s, in_qsize 5, out_qsize 2
INFO : PROGRESS: at 1.00% examples, 224406 words/s, in_qsize 7, out_qsize 0

INFO : PROGRESS: at 97.52% examples, 229508 words/s, in_qsize 7, out_qsize 0
INFO : PROGRESS: at 98.71% examples, 229631 words/s, in_qsize 7, out_qsize 0
INFO : PROGRESS: at 99.38% examples, 229714 words/s, in_qsize 8, out_qsize 0
INFO : training on 62487962 raw words (48693011 effective words) took 211.9s,
                                                    229817 effective words/s
INFO : saved gr/wiki.el.skip.model
```

Εικόνα 21: Δεύτερο στάδιο εκπαίδευσης του μοντέλου

Στην Εικόνα 21 παρουσιάζεται στιγμιότυπο από την ολοκλήρωση της εκπαίδευσης του μοντέλου. Η διαδικασία διήρκεσε 211 δευτερόλεπτα με μέσο όρο 229 χιλιάδες λέξεις το δευτερόλεπτο. Να σημειωθεί πως ορίζοντας μεγαλύτερο αριθμό επαναλήψεων ο χρόνος εκτέλεσης θα πολλαπλασιαστεί ανάλογα. Για παράδειγμα, εκπαίδευση τριών εποχών θα απαιτούσε τριπλάσιο χρόνο. Στη συγκεκριμένη περίπτωση, η εκπαίδευση διαρκεί μία εποχή ενώ κατά την αξιολόγηση όλα τα μοντέλα θα εκπαιδευτούν σε μεταβλητό αριθμό εποχών για να γίνει σύγκριση των αποτελεσμάτων.

4.2 Λειτουργίες ενός μοντέλου word2vec

4.2.1 Ομοιότητα δύο λέξεων

Η ομοιότητα μεταξύ δύο λέξεων μπορεί να εκφραστεί ως ένας αριθμός ανάμεσα στο μηδέν και τη μονάδα. Η μονάδα ορίζει την απόλυτη ομοιότητα των δύο λέξεων, ενώ το μηδέν, καμία ομοιότητα. Στο word2vec η ομοιότητα δύο λέξεων μπορεί να εντοπιστεί με τη μέθοδο similarity. Στην Εικόνα 22 παρουσιάζεται η χρήση της μεθόδου. Στη γραμμή 5 πραγματοποιείται η φόρτωση του μοντέλου που δημιουργήθηκε πριν στη μεταβλητή model. Στη συνέχεια ορίζονται στις γραμμές 7 και 8, οι λέξεις που θα συγκριθούν. Στη γραμμή 10 καλείται η μέθοδος similarity με ορίσματα τις δύο λέξεις, η οποία θα επιστρέψει έναν αριθμό κινητής υποδιαστολής, ο οποίος ορίζει και την ομοιότητα ανάμεσα στις δύο λέξεις. Τελικά, στη γραμμή 12 εκτυπώνεται το σχετικό μήνυμα στο χρήστη. Ο τρόπος που εξάγεται αυτή η πληροφορία βασίζεται στο δομικό

τρόπο λειτουργίας του ίδιου του μοντέλου. Έτσι, για να είναι δύο λέξεις όμοιες, πρέπει αυτές να εμφανίζονται κοντά σε κοινές λέξεις και αυτή την ομοιότητα υπολογίζει και αυτή η μέθοδος.

```
1   # -*- coding: utf8 -*-
2   from __future__ import unicode_literals
3   import gensim
4
5   model = gensim.models.Word2Vec.load("gr/wiki.el.skip.model")
6
7   word1 = 'αυτοκίνητο'
8   word2 = 'μηχανή'
9
10  similarity = model.wv.similarity(word1, word2)
11
12  print('Output -> Similarity between ' + word1 + ' and ' + word2 + ':', similarity)
Output -> Similarity between αυτοκίνητο and μηχανή: 0.453160354516
```

Εικόνα 22: Ομοιότητα δύο λέξεων

4.2.2 Εντοπισμός της πιο όμοιας λέξης

Μία χρήση της μονάδας ομοιότητας δύο λέξεων είναι ο εντοπισμός της πιο όμοιας λέξης για μία λέξη. Μία τέτοια λέξη θα μπορούσε να θεωρηθεί συνώνυμη με την προηγούμενη υπό κάποιες προϋποθέσεις. Η πιο όμοια λέξη για κάποια λέξη μπορεί να εντοπιστεί με τη μέθοδο most_similar όπως φαίνεται και στην Εικόνα 23. Όπως και πριν στη γραμμή 5 φορτώνεται το μοντέλο στη μεταβλητή model και ορίζεται στη γραμμή 7, η λέξη η οποία θα χρησιμοποιηθεί. Στη γραμμή 9 καλείται η μέθοδος most_similar η οποία δέχεται τη λέξη και επιστρέφει ένα πίνακα. Ο πίνακας, όπως φαίνεται και στην έξοδο της μεθόδου περιέχει με φθίνουσα σειρά τις λέξεις με τη μεγαλύτερη ομοιότητα για τη ζητούμενη λέξη.

Στο παράδειγμα που παρουσιάζεται, η ζητούμενη λέξη είναι η λέξη άντρας και η λέξη με τη μεγαλύτερη ομοιότητα είναι η λέξη άνδρας. Αυτό συμβαίνει καθώς η λέξη γράφεται και με τους δύο τρόπους και είναι λογικό οι δύο λέξεις να εμφανίζεται κοντά σε κοινές λέξεις. Οι λέξεις γέρος και ηλικιωμένος είναι και οι δύο λέξεις που σχετίζονται όντως με τη λέξη άντρας. Είναι συνηθισμένο οι λέξεις που εμφανίζονται με τη κλήση αυτής της μεθόδου να μην είναι οι αναμενόμενες ή να μην εμφανίζονται με την αναμενόμενη σειρά, καθώς η απόδοση του μοντέλου εξαρτάται από τα δεδομένα που

δέχτηκε ως είσοδο. Δεδομένου λοιπόν, ότι η γνώση του μοντέλου είναι περιορισμένη σε σχέση με τη γνώση που μπορεί να έχει ένας άνθρωπος είναι λογική και πιθανή διαφοροποίηση στα αναμενόμενα αποτελέσματα.

```
# -*- coding: utf8 -*-
from __future__ import unicode_literals
import gensim

model = gensim.models.Word2Vec.load("gr/wiki.el.skip.model")

word1 = 'άντρας'

array = model.wv.most_similar([word1])

print('Output -> Most similar words to ' + word1 + ':')
for tupl in array:
    print(tupl)
```

```
Output -> Most similar words to άντρας:
('άνδρας', 0.8374150991439819)
('γέρος', 0.7707920074462891)
('ηλικιωμένος', 0.7652866840362549)
```

Εικόνα 23: Εντοπισμός της πιο όμοιας λέξης

4.2.3 Αναλογίες λέξεων

Η εύρεση αναλογιών ανάμεσα σε λέξεις είναι μία λειτουργία που παρουσιάζει ιδιαίτερο ενδιαφέρον. Ο τρόπος με τον οποίο επιτυγχάνεται, περιλαμβάνει τον υπολογισμό αλγεβρικών πράξεων ανάμεσα στα διανύσματα των λέξεων. Η μέθοδος που χρησιμοποιείται για την παραγωγή των αναλογιών είναι και πάλι η most_similar καθώς βασίζεται στην ίδια λογική με πριν. Με δεδομένες τρεις λέξεις και την εκτέλεση του υπολογισμού Α + Β - Γ στα διανύσματα τους παράγεται ένα νέο διάνυσμα. Σκοπός της μεθόδου είναι στην ουσία, να βρει τη λέξη της οποίας το διάνυσμα βρίσκεται πιο κοντά σε αυτό. Στην Εικόνα 24 παρουσιάζεται ένα παράδειγμα.

```
1   # -*- coding: utf8 -*-
2   from __future__ import unicode_literals
3   import gensim
4
5   model = gensim.models.Word2Vec.load("gr/wiki.el.skip.model")
6
7   word1 = 'γυναίκα'
8   word2 = 'άντρας'
9   word3 = 'κορίτσι'
10
11  array = model.wv.most_similar(positive=[word2, word3], negative=[word1])
12
13  print('Output -> Equation ' + word2 + ' + ' + word3 + ' - ' + word1 + ':')
14  for tupl in array:
15      print(tupl)
Output -> Equation άντρας + κορίτσι - γυναίκα:
('αγόρι', 0.7301124334335327)
('μπαμπάς', 0.6684125661849976)
('φιλί', 0.6431673765182495)
```

Εικόνα 24: Αναλογίες λέξεων

Μετά τη φόρτωση του μοντέλου, ορίζονται οι τρεις λέξεις στις γραμμές 7, 8 και 9. Σκοπός είναι η παραγωγή της αναλογίας: "Η λέξη γυναίκα σχετίζεται με τη λέξη άντρας όπως η λέξη κορίτσι με τη λέξη …". Στη σειρά 11 γίνεται κλήση της μεθόδου most_similar όπου αυτή τη φορά δίνονται δύο ορίσματα. Το πρώτο είναι μία λίστα με τις θετικές λέξεις και το δεύτερο μία λίστα με τις αρνητικές. Οι θετικές είναι οι λέξεις άντρας και κορίτσι και αρνητική, η λέξη γυναίκα. Ο αλγεβρικός υπολογισμός που θα εκτελεστεί για την επίλυση θα είναι: "άντρας + κορίτσι - γυναίκα" και η πιο σχετική λέξη που εντοπίζεται είναι η λέξη αγόρι, η οποία θα ήταν και η απάντηση ενός ανθρώπου. Γενικά, τα μοντέλα word2vec λειτουργούν πολύ καλά στην παραγωγή τέτοιων αναλογιών και αυτό οφείλεται στη λογική που αυτά ορίζουν τις σχέσεις μεταξύ των λέξεων.

4.2.4 Αποκλεισμός της λιγότερο όμοιας λέξης

Τα μοντέλα word2vec προσφέρουν τη δυνατότητα, μέσα από τη σύγκριση των διανυσμάτων, εντοπισμού της λιγότερο όμοια λέξης μέσα από μία λίστα λέξεων. Αυτό είναι εφικτό με τη κλήση της μεθόδου doesnt_match. Η τελευταία, με δεδομένη μία

λίστα λέξεων επιστρέφει τη λέξη η οποία φέρεται να μην ταιριάζει με τις υπόλοιπες, όπως φαίνεται και στην Εικόνα 25.

```
1   # -*- coding: utf8 -*-
2   from __future__ import unicode_literals
3   import gensim
4
5   model = gensim.models.Word2Vec.load("gr/wiki.el.skip.model")
6
7   word1 = 'αυτοκίνητο'
8   word2 = 'μηχανή'
9   word3 = 'τιμόνι'
10  word4 = 'αράχνη'
11  word5 = 'ταχύτητα'
12
13  doesnt_match = model.wv.doesnt_match([word1, word2, word3, word4, word5])
14
15  print('Output -> Doesnt match between ' + word1
16        + ', ' + word2 + ', ' + word3 + ', '
17        + word4 + ', ' + word5 + ': ' + doesnt_match)
Output -> Doesnt match between αυτοκίνητο, μηχανή, τιμόνι, αράχνη, ταχύτητα: αράχνη
```

Εικόνα 25: Αποκλεισμός της λιγότερο όμοιας λέξης

Στις γραμμές 7, 8, 9, 10 και 11 δηλώνονται οι λέξεις που θα χρησιμοποιηθούν. Δεν είναι απαραίτητο να είναι πέντε, αρκεί να μην πάρα πολλές ή πολύ λίγες βάση κοινής λογικής. Στη γραμμή 13 στη μεταβλητή doesnt_match επιστρέφεται το αποτέλεσμα από τη κλήση της μεθόδου doesnt_match. Όπως φαίνεται στην έξοδο της εκτέλεσης, η λιγότερο σχετική λέξη στις λέξεις αυτοκίνητο, μηχανή, τιμόνι, αράχνη και ταχύτητα είναι η λέξη αράχνη, όπως και θα ήταν αναμενόμενο. Η απόδοση του μοντέλου σε αυτή τη λειτουργία εξαρτάται από την επιλογή και τον αριθμό των λέξεων. Για να παραχθεί αποτέλεσμα ανάλογο με αυτό που θα σκεφτόταν ένας άνθρωπος, πρέπει να υπάρχει μία λέξη η οποία να είναι αρκετά διαφορετική από τις υπόλοιπες. Για παράδειγμα, η λέξη αράχνη ίσως και να μην είναι η ξεκάθαρη επιλογή. Η λέξη ταχύτητα διαφέρει και αυτή, καθώς όλες οι υπόλοιπες λέξεις αφορούν υλικά σώματα ενώ αυτή ορίζει κάτι άυλο. Παρόλα αυτά, στη συγκεκριμένη περίπτωση, το μοντέλο εντόπισε την αναμενόμενη λέξη.

5 Διεπαφή παρουσίασης του μοντέλου

Για την παρουσίαση του μοντέλου word2vec που δημιουργήθηκε θα δημιουργηθεί μία εφαρμογή διαδικτύου, μέσω της οποίας, εξωτερικοί χρήστες θα έχουν τη δυνατότητα να εκτελούν ερωτήματα στο μοντέλο, πάνω στις λειτουργίες που προσφέρει. Η υποδομή που θα δημιουργηθεί αποτελείται από τρία επίπεδα. Το πρώτο επίπεδο είναι η υλοποίηση ενός προγράμματος διακομιστή σε Python, του οποίου η λειτουργία θα είναι η πρόσβαση στις λειτουργίες του μοντέλου μέσω αιτημάτων πρωτοκόλλου http. Το http [25] είναι ένα πρωτόκολλο επικοινωνίας που χρησιμοποιείται για τη μεταφορά δεδομένων. Χρησιμοποιείται από το σύνολο των ιστοσελίδων στο διαδίκτυο καθώς και σε άλλες εφαρμογές. Το δεύτερο επίπεδο της εφαρμογής θα περιλαμβάνει μία ιστοσελίδα γραμμένη με τη δημοφιλή βιβλιοθήκη της Javascript, React.js [26]. Τελικά, το τρίτο επίπεδο θα αποτελείται από μία απλή εφαρμογή διακομιστή η οποία θα εκτελείται από το περιβάλλον εκτέλεσης Node.js [27].

Η εφαρμογή στη Node.js θα παρέχει διαδικτυακή πρόσβαση στην ιστοσελίδα και στο πρόγραμμα διακομιστή του μοντέλου. Η ιστοσελίδα θα επικοινωνεί μέσω της εφαρμογής στη Node.js για να εκτελεί τις λειτουργίες του μοντέλου. Στη συνέχεια, θα παρουσιαστούν σε συνοπτικό βαθμό η υλοποίηση των παραπάνω.

5.1 Εφαρμογή πρόσβασης στο μοντέλο

Όπως αναφέρθηκε και νωρίτερα, για να είναι δυνατή η πρόσβαση στο μοντέλο από το διαδίκτυο είναι αναγκαία η δημιουργία εφαρμογής που θα εξυπηρετεί αυτό το σκοπό. Θα χρησιμοποιηθεί το πακέτο HTTPServer στην Python για να δημιουργηθεί ένας διακομιστής, ο οποίος θα απαντά σε ερωτήματα στο τοπικό δίκτυο με το πρωτόκολλο http.

Στην Εικόνα 26 παρακάτω παρουσιάζεται ένας σκελετός υλοποίησης αυτού του διακομιστή. Στη γραμμή 1 δηλώνεται η κωδικοποίηση UTF-8 καθώς θα γίνει επεξεργασία κειμένου στην ελληνική γλώσσα. Η γραμμή 2 εξυπηρετεί παρόμοιο σκοπό. Στη γραμμή 7 φορτώνεται το μοντέλο word2vec στη μεταβλητή model, καθώς θα χρησιμοποιηθεί στη συνέχεια. Στη γραμμή 9 δηλώνεται η κλάση RequestHandler η

οποία είναι υποκλάση της BaseHTTPRequestHandler και θα αναλάβει τη διεκπεραίωση των http αιτημάτων που θα δέχεται η εφαρμογή.

Τα αιτήματα του πρωτοκόλλου http που καλείται να υλοποιήσει η εφαρμογή είναι το OPTIONS, το οποίο αφορά τις διαθέσιμες επιλογές που προσφέρει, το GET, το οποίο αφορά λήψη δεδομένων, το POST το οποίο αφορά αποστολή δεδομένων, το PUT, το οποίο αφορά τοποθέτηση νέων δεδομένων και το DELETE το οποίο αφορά διαγραφή δεδομένων.

```
1   # -*- coding: utf8 -*-
2   from __future__ import unicode_literals
3   import json
4   import gensim
5   from BaseHTTPServer import HTTPServer, BaseHTTPRequestHandler
6
7   model = gensim.models.Word2Vec.load("gr/wiki.el.skip.model")
8
9   class RequestHandler(BaseHTTPRequestHandler):
10
11      def do_OPTIONS(self):
12
13      def do_GET(self):
14
15      def do_POST(self):
16
17      def do_PUT(self):
18
19      def do_DELETE(self):
20
21
22  if __name__ == "__main__":
23      port = 4748
24      print('Listening on port', port)
25      server = HTTPServer(('', port), RequestHandler)
26      server.serve_forever()
```

Εικόνα 26: Σκελετός εφαρμογής διακομιστή σε Python

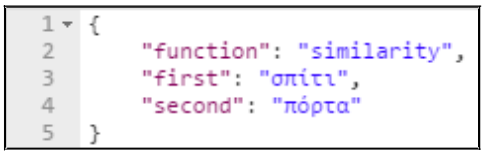
Στη συνέχεια, στη γραμμή 23 δηλώνεται η πόρτα που θα ακούει ο διακομιστής για συνδέσεις. Τελικά, στη γραμμή 24 δημιουργείται ένας HTTPServer με ορίσματα την πόρτα ακρόασης και το χειριστή αιτημάτων που δημιουργήθηκε νωρίτερα και εκτελείται με τη μέθοδο serve_forever του HTTPServer.

Για τη συγκεκριμένη εφαρμογή, η επικοινωνία θα γίνεται μέσω αιτημάτων POST καθώς η εφαρμογή θα δέχεται δεδομένα και θα απαντά με τα αποτελέσματα από την εφαρμογή του μοντέλου σε αυτά. Στη συνέχεια θα παρουσιαστεί η δομή της υλοποίησης για το χειρισμό των αιτημάτων POST.

```
29      def do_POST(self):
30          # Extract and print the contents of the POST
31          length = int(self.headers['Content-Length'])
32          data = json.loads(self.rfile.read(length).decode('utf-8'))
33          func = data[u'function']
34          print(data)
35          if func == "analogies":
36              array = model.wv.most_similar_cosmul(
37                  positive=[data[u'second'], data[u'third']], negative=[data[u'first']])
38              self.send_response(200)
39              self.send_header(b'Content-type', b'text/plain')
40              self.end_headers()
41              self.wfile.write(array[0][0].encode("UTF-8"))
42          elif func == "most_similar":
43              array = model.wv.most_similar(positive=[data[u'first']])
44              self.send_response(200)
45              self.send_header(b'Content-type', b'text/plain')
46              self.end_headers()
47              for tup in array:
48                  self.wfile.write(tup[0].encode("UTF-8"))
49                  self.wfile.write(" ")
50          elif func == "similarity":
51              single = model.wv.similarity(data[u'first'], data[u'second'])
52              self.send_response(200)
53              self.send_header(b'Content-type', b'text/plain')
54              self.end_headers()
55              self.wfile.write(single)
56          elif func == "doesnt_match":
57              single = model.wv.doesnt_match(data[u'first'].split())
58              self.send_response(200)
59              self.send_header(b'Content-type', b'text/plain')
60              self.end_headers()
61              self.wfile.write(single.encode("UTF-8"))
62          else:
63              self.send_response(200)
64              self.send_header(b'Content-type', b'text/plain')
65              self.end_headers()
66              self.wfile.write("UNKNOWN FUNCTION")
```

Εικόνα 27: Χειρισμός αιτημάτων POST

Στην Εικόνα 27, φαίνεται η κλήση στις διάφορες λειτουργίες του μοντέλου μέσω των αιτημάτων που δέχεται η εφαρμογή. Στη γραμμή 31 και 32 λαμβάνεται η πληροφορία από το εκάστοτε αίτημα. Η δομή ακολουθεί τη μορφή JSON [28] που είναι συνηθισμένος τύπος αρχείου για μετάδοση δεδομένων στο διαδίκτυο. Στην Εικόνα 28 φαίνεται ένα απλό αρχείο JSON. Στη μεταβλητή data αποθηκεύεται το σύνολο της πληροφορίας και αυτή θα χρησιμοποιηθεί στη συνέχεια.



Εικόνα 28: Αρχείο JSON

Στη γραμμή 33 γίνεται ανάθεση του πεδίου function στη μεταβλητή func. Το πεδίο αυτό ορίζει τη λειτουργία που θα εκτελεστεί. Οι πιθανές επιλογές είναι τέσσερις και θα ελεγχθούν στη συνέχεια. Από τη γραμμή 35 μέχρι το τέλος του αρχείου, βρίσκεται μία δομή επιλογής που υλοποιεί κάθε μία από αυτές τις επιλογές.

Η πρώτη επιλογή είναι η λειτουργία analogies. Η λειτουργία αφορά την παραγωγή αναλογιών λέξεων όπως αυτή αναλύθηκε στο κεφάλαιο 4.2.3. Χρησιμοποιούνται τα πεδία first, second και third από τη data που ορίζουν τις τρεις λέξεις που θα χρησιμοποιηθούν για τη λειτουργία και καλείται η μέθοδος most_similar_cosmul του μοντέλου, η οποία είναι ανάλογη της μεθόδου most_similar. Στη συνέχεια αποστέλλεται η απάντηση στο αίτημα με τις επόμενες τέσσερις γραμμές. Αποστέλλεται μόνο η πρώτη λέξη που εντοπίστηκε από το μοντέλο.

Η δεύτερη επιλογή είναι η λειτουργία most_similar. Όπως και στο κεφάλαιο 4.2.2, αναλαμβάνει την εύρεση των πιο όμοιων λέξεων για τη λέξη που δίδεται. Καλείται η μέθοδος most_similar του μοντέλου με όρισμα αυτή τη λέξη και η απάντηση περιλαμβάνει αυτή τη φορά, όλες τις λέξεις που εντοπίζει το μοντέλο.

Η τρίτη επιλογή similarity αναφέρεται στην ομοιότητα λέξεων. Όπως παρουσιάζεται στο κεφάλαιο 4.2.1, στη γραμμή 51 εκτελείται η μέθοδος similarity του μοντέλου με ορίσματα τις δύο λέξεις που παρέχονται από το χρήστη. Στη συνέχεια επιστρέφεται ο αριθμός που ορίζει την ομοιότητα των λέξεων.

Τελικά η τέταρτη επιλογή είναι η doesnt_match. Αναφορά γίνεται στο κεφάλαιο 4.2.4. Εδώ στο πεδίο first περιλαμβάνονται όλες οι λέξεις σε ένα αλφαριθμητικό χωρισμένες με κενούς χαρακτήρες. Με τη μέθοδο split στη γραμμή 57, οι λέξεις διασπώνται σε ένα πίνακα ο οποίος αποτελεί το όρισμα της μεθόδου doesnt_match του μοντέλου. Όπως και πριν αποστέλλεται τελικά το αποτέλεσμα στο χρήστη.

Αυτά ήταν τα βασικά σημεία του πρώτου επιπέδου της εφαρμογής διαδικτύου. Στη συνέχεια θα γίνει αναφορά στην ανάπτυξη της εφαρμογής διακομιστή στο περιβάλλον της Node.js που θα παρέχει πρόσβαση στην εφαρμογή που περιγράφηκε.

5.2 Εφαρμογή διακομιστή διαδικτύου

Η εφαρμογή που θα αναλαμβάνει την διαδικτυακή παρουσία της εφαρμογής θα αναπτυχθεί στη γλώσσα Javascript και θα εκτελείται στο περιβάλλον της Node.js που είναι περιβάλλον εκτέλεσης κώδικα Javascript. Η εφαρμογή είναι μικρή σε μέγεθος καθώς με τη χρήση έτοιμων πακέτων, πληθώρα των οποίων είναι διαθέσιμα, είναι εξαιρετικά εύκολη η δημιουργία τέτοιων εφαρμογών. Δεν θα γίνει εκτενής αναφορά σε λεπτομέρειες πάνω στη Javascript και στη Node.js καθώς κάτι τέτοιο ξεφεύγει από τις ανάγκες της διπλωματικής.

```
var http = require("http");
var express = require('express');
var app = express();
var bodyParser = require('body-parser');
var urlencodedParser = bodyParser.urlencoded({ extended: true });
var request = require('request');
var cors = require('cors');

app.use(cors());
app.use(express.static('demonstration/build'));

app.post('/checkRelations', urlencodedParser, function (req, res){
    let func = req.body.function;
    let json = null;
    switch (func) {
        case "relations": json = { function: func, first: req.body.first,
                                   second: req.body.second, third: req.body.third }; break;
        case "most_similar": json = { function: func, first: req.body.first }; break;
        case "similarity": json = { function: func, first: req.body.first,
                                   second: req.body.second }; break;
        case "doesnt_match": json = { function: func, first: req.body.first }; break;
        default: break;
    }
    if(json) {
        request.post('http://localhost:4748', { json: json }, function (error, response, body) {
            if (!error && response.statusCode == 200) {
                res.send({result: body});
                }
        });
    } else {
        res.send("ERROR");
    }
});


// Running Server Details.
var server = app.listen(80, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at %s:%s Port", host, port)
});
```

Εικόνα 29: Εφαρμογή διακομιστή διαδικτύου

Στην Εικόνα 29 παρουσιάζεται το σύνολο της διαδικτυακής εφαρμογής στη Node.js. Στις πρώτες επτά γραμμές γίνεται η εισαγωγή των απαραίτητων εξωτερικών

πακέτων. Το πακέτο http αφορά πρόσβαση στο πρωτόκολλο http. Το Express [29] είναι ένα πολύ δημοφιλές πακέτο το οποίο παρέχει το σκελετό για τη δημιουργία εφαρμογών διαδικτύου. Το πακέτο body-parser [30] χρησιμοποιείται για την πρόσβαση στα πεδία που αποστέλλονται με τη μέθοδο POST του πρωτοκόλλου http. Το πακέτο request [31] προσφέρει μία αφαιρετική προσέγγιση στη δημιουργία αιτημάτων http. Τελικά, το πακέτο cors [32] προσφέρει έτοιμη υλοποίηση για υποστήριξη αιτημάτων από ιστοσελίδες που δεν έχουν ίδιο όνομα τομέα με αυτόν που τρέχει η εφαρμογή.

Στη γραμμή 3, ορίζεται στη μεταβλητή app ο σκελετός που δημιουργείται από το πακέτο express. Στη γραμμή 10 ορίζεται με τη μέθοδο express.static πρόσβαση στο φάκελο demonstration/build όπου θα βρίσκεται η ιστοσελίδα. Με αυτόν τον τρόπο η εφαρμογή θα σερβίρει την ιστοσελίδα στους χρήστες. Οι γραμμές 12 - 33 περιλαμβάνουν τον χειρισμό POST αιτημάτων στην υποδιεύθυνση /checkRelations. Η λειτουργία που προσφέρει είναι η δημιουργία του αρχείου JSON, στις γραμμές 13 - 23, που θα χρησιμοποιήσει η εφαρμογή του κεφαλαίου 5.1 και η επικοινωνία με αυτή στη γραμμή 25. Στη γραμμή 27 επιστρέφει το αποτέλεσμα της άλλης εφαρμογής στο χρήστη. Τελικά, οι γραμμές 37 - 41 αφορούν την δημιουργία και εκτέλεση του εξυπηρετητή στην πόρτα 80.

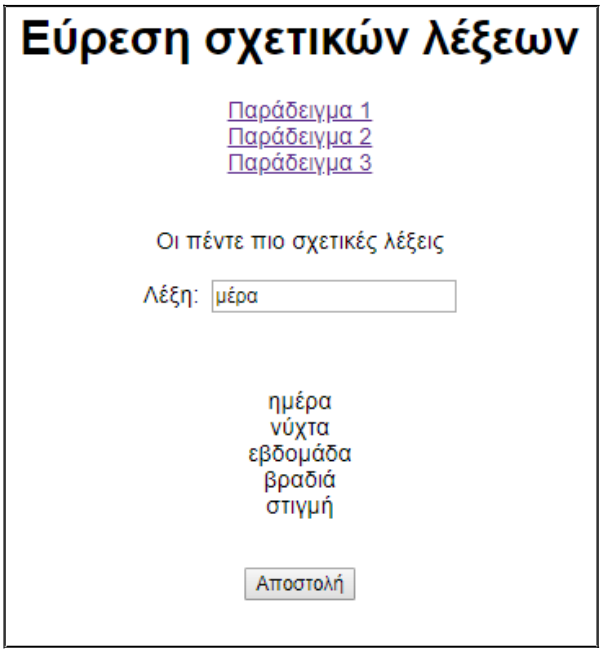
5.3 Ιστοσελίδα

Η ιστοσελίδα έχει αναπτυχθεί με τη βιβλιοθήκη React.js που προσφέρει τη δυνατότητα εύκολης δημιουργίας αποδοτικών εφαρμογών διαδικτύου. Ο κώδικας που απαρτίζει την ιστοσελίδα είναι πολύ μεγάλος σε όγκο για να αναλυθεί σε αυτό το κεφάλαιο καθώς δεν αποτελεί αντικείμενο της διπλωματικής. Από πλευράς κώδικα, θα αναλυθούν μόνο τα σημεία που αφορούν αποστολή δεδομένων στον διακομιστή καθώς αυτά αποτελούν τη σύνδεση με τα άλλα επίπεδα του συστήματος. Ο υπόλοιπος κώδικας είναι διαθέσιμος στη περιοχή της διπλωματικής στην ιστοσελίδα github.com [33], στη διαδρομή /core/web/demonstration. Επίσης θα παρουσιαστεί η ιστοσελίδα και οι λειτουργίες που παρέχει στους χρήστες.

5.3.1 Παρουσίαση

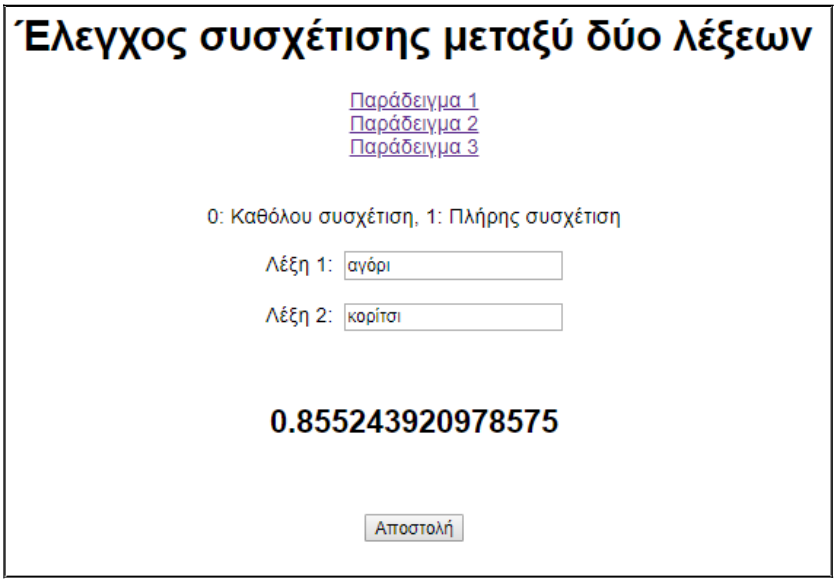
Η ιστοσελίδα, Εικόνα 30, βρίσκεται στη διεύθυνση deep.devasync.net και είναι ελεύθερα προσβάσιμη από εξωτερικούς χρήστες. Παρέχει πρόσβαση στις τέσσερις λειτουργίες που αναλύθηκαν το κεφάλαιο 4.2 για την ελληνική γλώσσα.

Εικόνα 30: Ιστοσελίδα παρουσίασης μοντέλου



Εικόνα 31: Εύρεση σχετικών λέξεων

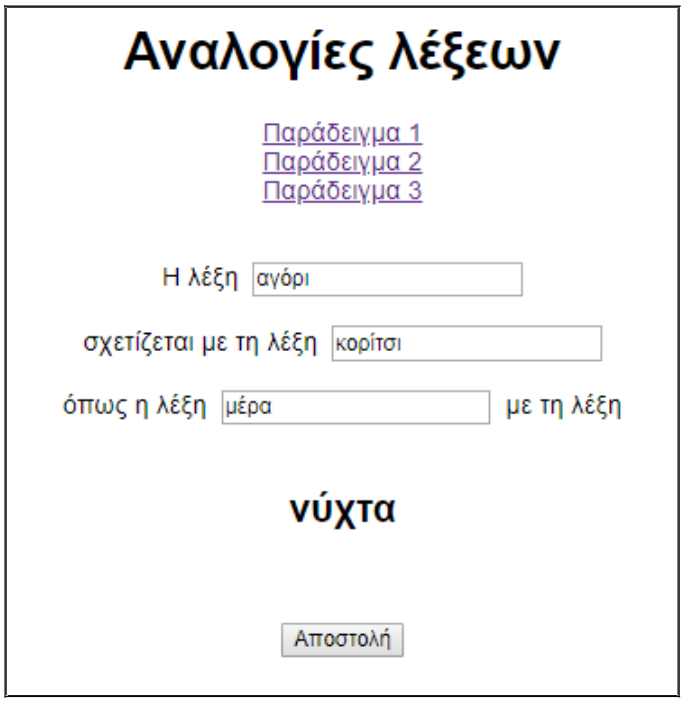
Για κάθε λειτουργία ο χρήστης μπορεί να επιλέξει να εκτελέσει ένα από τα τρία παρεχόμενα παραδείγματα ή να δώσει δική του είσοδο. Στην Εικόνα 31 φαίνεται η λειτουργία της εύρεσης σχετικών λέξεων όπου ο χρήστης εισάγει μία λέξη και η εφαρμογή επιστρέφει τις πέντε πιο σχετικές λέξεις με αυτή, ενώ στην Εικόνα 32, ο χρήστης μπορεί να δει το βαθμό συσχέτισης δύο λέξεων.



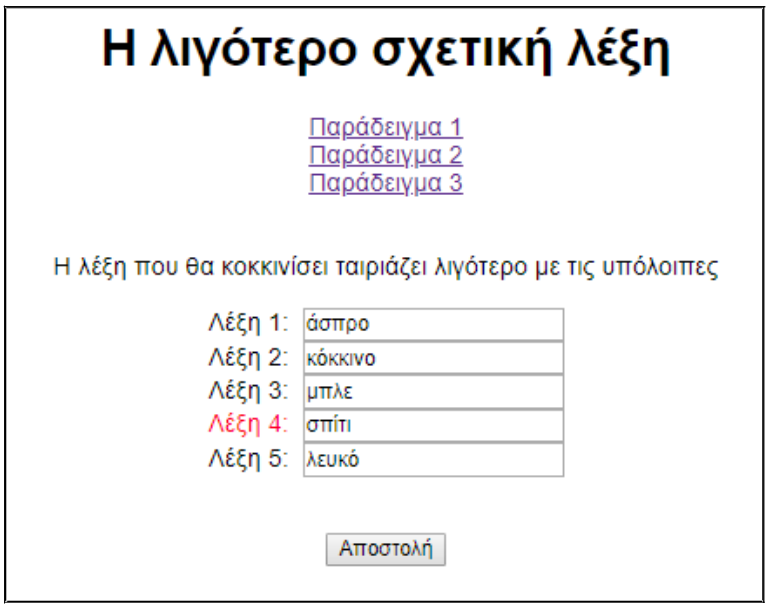
Εικόνα 32: Έλεγχος συσχέτισης μεταξύ δύο λέξεων

Να σημειωθεί πως η λέξη συσχέτιση χρησιμοποιείται αντί αυτής της ομοιότητας διατηρώντας την ίδια σημασία.

Στην Εικόνα 33 παρουσιάζεται η λειτουργία της παραγωγής αναλογιών ανάμεσα σε λέξεις. Ο χρήστης μπορεί να συμπληρώσει τις κενές λέξεις που υπάρχουν στο κείμενο και θα υπολογιστεί η αλγεβρική πράξη "ΛΕΞΗ2+ΛΕΞΗ3-ΛΕΞΗ1", δηλαδή στο παράδειγμα: "κορίτσι + μέρα - αγόρι" και το αποτέλεσμα είναι η λέξη νύχτα, γεγονός που παρουσιάζει ενδιαφέρον.



Εικόνα 33: Αναλογίες λέξεων



Εικόνα 34: Η λιγότερο σχετική λέξη

Η τελευταία λειτουργία που αναφέρθηκε είναι ο εντοπισμός της λιγότερο σχετικής λέξης από ένα σύνολο λέξεων και αυτή φαίνεται στην Εικόνα 34. Ο χρήστης εισάγει πέντε λέξεις, στη συγκεκριμένη περίπτωση, και η λέξη που σχετίζεται λιγότερο

με τις υπόλοιπες χρωματίζεται κόκκινη. Στο παράδειγμα, οι τέσσερις από τις πέντε λέξεις ορίζουν χρώματα και το μοντέλο μπορεί να ξεχωρίσει επιτυχώς την πέμπτη λέξη.

5.3.2 Επικοινωνία ιστοσελίδας με το διακομιστή

Όπως αναφέρθηκε νωρίτερα θα παρουσιαστεί ο τρόπος επικοινωνίας της ιστοσελίδας με το διακομιστή που είναι η εφαρμογή που αναλύθηκε στο κεφάλαιο 5.2.

Όπως αναφέρθηκε νωρίτερα ο διακομιστής αναμένει αιτήματα http τύπου POST με τις πληροφορίες που απαιτούνται. Η ιστοσελίδα αναλαμβάνει να δημιουργεί και να αποστέλλει αυτά τα αιτήματα όταν ο χρήστης πατάει το κουμπί Αποστολή σε κάθε λειτουργία. Θα παρουσιαστεί ο κώδικας που αφορά την ομοιότητα δύο λέξεων. Δεν θα γίνει αναφορά σε άλλη λειτουργία, καθώς ο τρόπος αποστολής των αιτημάτων είναι πανομοιότυπος.

```
54      handleSubmit(e) {
55          let word1 = this.state.word1;
56          let word2 = this.state.word2;
57          if (word1 === '' || word2 === '') {
58              alert("Συμπληρώστε όλες τις λέξεις");
59          } else {
60              $.ajax({
61                  type: "POST",
62                  url: "https://deep.devasync.net/checkRelations",
63                  data: {
64                      function: "similarity",
65                      first: word1,
66                      second: word2
67                  }
68              }).done((res) => {
69                      this.setState({word3: res.result});
70                  });
71          }
72          e.preventDefault();
73      }
```

Εικόνα 35: Αποστολή αιτημάτων από την ιστοσελίδα

Ο κώδικας που εκτελείται όταν πατηθεί το κουμπί Αποστολή, στη λειτουργία που αφορά την εύρεση ομοιότητας μεταξύ δύο λέξεων, φαίνεται στην Εικόνα 35. Στις

γραμμές 55 και 56 ορίζονται στις μεταβλητές word1 και word2 οι δύο λέξεις. Στη συνέχεια, αφού ελεγχθεί στη γραμμή 57 πως έχουν δοθεί λέξεις, στη γραμμή 60 δημιουργείται το αίτημα που θα αποσταλεί στο διακομιστή. Για τη διεκπεραίωση του αιτήματος χρησιμοποιείται η μέθοδος ajax της βιβλιοθήκης jQuery.

Η βιβλιοθήκη jQuery [34] είναι μία βιβλιοθήκη της Javascript η οποία χρησιμοποιείται από την πλειοψηφία των ιστοσελίδων στο διαδίκτυο. Η μέθοδος ajax αναφέρεται στον γενικότερο όρο Ajax [35] που ορίζει ένα σύνολο τεχνικών ανάπτυξης ιστοσελίδων, οι οποίες μπορούν να χρησιμοποιηθούν για την εκτέλεση αιτημάτων από μία ιστοσελίδα, ασύγχρονα, χωρίς δηλαδή την αναστολή της εκτέλεσης του κώδικα ενώ περιμένουν αποτέλεσμα.

Στη γραμμή 61 ορίζεται ο τύπος του αιτήματος ως POST, στη γραμμή 62 η διεύθυνση του διακομιστή και στη γραμμή 63 τα δεδομένα που θα αποσταλούν. Στη συγκεκριμένη περίπτωση, τα δεδομένα περιλαμβάνουν τρία πεδία. Το πρώτο πεδίο, function, είναι η λειτουργία που θα κληθεί και τα άλλα δύο αφορούν τις λέξεις που θα συγκριθούν. Τελικά, στη γραμμή 69, η απάντηση του διακομιστή εμφανίζεται στη σελίδα.

6 Αξιολόγηση

6.1 Διαδικασία

Η αξιολόγηση είναι μία απαραίτητη διαδικασία για τον έλεγχο της απόδοσης των μοντέλων. Προσφέρει ανατροφοδότηση στο χρήστη, για τις επιλογές του πάνω στην εκπαίδευση του μοντέλου. Για παράδειγμα, με την εκτέλεση πειραμάτων και με τη χρήση των κατάλληλων κριτηρίων αξιολόγησης, μπορεί ο χρήστης να δει ότι για μια συγκεκριμένη γλώσσα και κάποια δεδομένα, η αρχιτεκτονική Skip-gram παρέχει καλύτερη απόδοση στο μοντέλο από την αρχιτεκτονική CBOW. Στη συνέχεια η διαδικασία της αξιολόγησης θα εφαρμοστεί στα μοντέλα των δύο γλωσσών και θα αναλυθούν τα αποτελέσματα και η απόδοση τους.

Η αξιολόγηση που θα εφαρμοστεί θα είναι εσωτερική αξιολόγηση καθώς τα μοντέλα δεν θα χρησιμοποιηθούν για κάποιο συγκεκριμένο σκοπό. Η μέθοδος που θα χρησιμοποιηθεί είναι αυτή της αναλογίας καθώς είναι η πιο αντιπροσωπευτική για τα μοντέλα word2vec και προσφέρει ασφαλέστερα συμπεράσματα από τις υπόλοιπες μεθόδους.

Όπως θα αναλυθεί και στη συνέχεια, θα δημιουργηθούν κριτήρια αξιολόγησης για την ελληνική και την αγγλική γλώσσα για τη μέθοδο της αναλογίας. Θα γίνει υλοποίηση της διαδικασίας με τη βιβλιοθήκη gensim και θα παρουσιαστεί η λειτουργία της και τα αποτελέσματα που προσφέρει. Στη συνέχεια, για κάθε γλώσσα θα δημιουργηθούν πολλαπλά μοντέλα, με διαφορετικές επιλογές για την εκπαίδευση τους. Αυτά τα μοντέλα θα αξιολογηθούν έτσι ώστε να βρεθούν αυτά που αποδίδουν καλύτερα. Τελικά, θα γίνει σύγκριση της απόδοσης στις δύο γλώσσες.

6.2 Κριτήρια αξιολόγησης

Για τη διαδικασία της αξιολόγησης, δημιουργήθηκαν δύο αρχεία κειμένου, τα οποία περιέχουν έτοιμα ερωτήματα στον τομέα των αναλογιών. Αυτά τα αρχεία θα δοθούν ως είσοδο κατά την εκτέλεση της αξιολόγησης. Κατά τη δημιουργία των αρχείων

χρησιμοποιήθηκε πληροφορία από το αντίστοιχο αρχείο που προσφέρει η Google στην αρχική υλοποίηση της για το word2vec.

Το αρχείο ερωτημάτων που προσφέρει η Google περιέχει δεδομένα για 14 κατηγορίες ερωτημάτων, τα οποία προσεγγίζουν τα 20.000. Καλύπτουν ευρεία γκάμα περιπτώσεων όπως θέματα γραμματικής, αντιστοίχηση χωρών με πρωτεύουσες, εθνικότητες και άλλα.

Συγκεκριμένα οι κατηγορίες είναι οι εξής:

1. capital-common-countries

   Αφορά τις πρωτεύουσες των πιο γνωστών χωρών.
2. capital-world

   Περιέχει πολύ μεγαλύτερο αριθμό χωρών - πρωτευουσών από πριν.
3. currency

   Συνδυασμοί χωρών με τα νομίσματα τους.
4. city-in-state

   Περιέχει ερωτήματα για πόλεις και τις πολιτείες που ανήκουν αυτές, στις Η.Π.Α.
5. family

   Περιέχει ερωτήματα σχετικά με οικογενειακές σχέσεις.
6. gram1-adjective-to-adverb

   Γραμματική, συνδυασμοί επιθέτων - επιρρημάτων.
7. gram2-opposite

   Γραμματική, συνδυασμοί αντιθέτων.
8. gram3-comparative

   Γραμματική, συνδυασμοί θετικού - συγκριτικού βαθμού.
9. gram4-superlative

   Γραμματική, συνδυασμοί θετικού - υπερθετικού βαθμού.
10. gram5-present-participle

    Γραμματική, συνδυασμοί ρημάτων - μετοχών.
11. gram6-nationality-adjective

    Γραμματική, συνδυασμοί χωρών - εθνικοτήτων.
12. gram7-past-tense

    Γραμματική, παρελθοντικός χρόνος.
13. gram8-plural

    Γραμματική, συνδυασμοί ουσιαστικών - πληθυντικού τους.

14. gram9-plural-verbs

Γραμματική, συνδυασμοί ρημάτων - πληθυντικού τους.

Για τις ανάγκες της αξιολόγησης των μοντέλων χρησιμοποιήθηκαν δεδομένα από τις κατηγορίες 1, 5, 11 και 13. Τα ερωτήματα που συγκεντρώθηκαν μεταφράστηκαν μετά από επιμέλεια στην ελληνική γλώσσα. Η επιλογή των συγκεκριμένων κατηγοριών έγινε έτσι ώστε να μην δημιουργηθούν σημαντικές διαφοροποιήσεις στην αξιολόγηση των δύο γλωσσών. Για παράδειγμα, δεν χρησιμοποιήθηκε πληροφορία που αφορά επίθετα, καθώς στην αγγλική γλώσσα ένα επίθετο εμφανίζεται με μόνο ένα τρόπο (πχ beautiful) ενώ στην ελληνική γλώσσα το ίδιο επίθετο μπορεί να εμφανιστεί με πολλούς (πχ όμορφος, όμορφη, όμορφο, όμορφα). Τα ερωτήματα είναι αντίστοιχα στις δύο γλώσσες, με πολύ λίγες εξαιρέσεις. Το σύνολο των ερωτημάτων προσεγγίζουν τα 2000 και στις δύο γλώσσες.

6.3 Υλοποίηση

Η υλοποίηση της αξιολόγησης γίνεται με τη βιβλιοθήκη gensim και τη μέθοδο accuracy, που προσφέρει για αυτή την εργασία. Ο κώδικας παρουσιάζεται στην Εικόνα 36 και θα αναλυθεί στη συνέχεια.

Όπως φαίνεται και στην εικόνα, στη γραμμή 5 φορτώνει το μοντέλο στη μεταβλητή model και στη γραμμή 6 φορτώνει το αρχείο των ερωτημάτων στη μεταβλητή questions. Στη γραμμή 8 εκτελείται η αξιολόγηση. Το αρχείο των ερωτημάτων περιλαμβάνει δεδομένα της μορφής: “Αθήνα Ελλάδα Βερολίνο Γερμανία”. Οι τρεις πρώτες λέξεις αποτελούν το ερώτημα που θα εκτελεστεί και η τελευταία λέξη την αναμενόμενη απάντηση. Κατά τη διαδικασία της αξιολόγησης εκτελούνται λειτουργίες αναλογιών για τις τρεις λέξεις και το αποτέλεσμα τους, συγκρίνεται με την τελευταία λέξη. Στη συνέχεια αποθηκεύονται τα ερωτήματα που λύθηκαν επιτυχώς σε μία λίστα και αυτά που δεν κατάφεραν να επιλυθούν σε μία άλλη. Αυτή η διαδικασία εκτελείται με την κλήση της μεθόδου accuracy και το αποτέλεσμα αποθηκεύεται στη μεταβλητή result.

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

```
Output -> Success rate: 42% | Section: capital-countries, Tests: 72
          Success rate: 29% | Section: country-nationality, Tests: 38
          Success rate: 36% | Section: family, Tests: 240
          Success rate: 20% | Section: plural, Tests: 504
          Success rate: 26% | Section: total, Tests: 854
```

Εικόνα 36: Υλοποίηση της αξιολόγησης για το ελληνικό μοντέλο

Οι γραμμές 10 - 18 αφορούν τον υπολογισμό των ποσοστών επιτυχίας στις διάφορες κατηγορίες και την εκτύπωση αυτών στην οθόνη. Η μεταβλητή result είναι ένας πίνακας με ένα στοιχείο για κάθε κατηγορία και ένα στοιχείο για το σύνολο των ερωτημάτων. Το κάθε στοιχείο περιλαμβάνει ένα πεδίο section, ένα πεδίο correct και ένα πεδίο incorrect. Το πεδίο section είναι το όνομα της αντίστοιχης κατηγορίας των ερωτημάτων. Το πεδίο correct περιλαμβάνει μία λίστα με όλα τα ερωτήματα που επιλύθηκαν επιτυχώς ενώ το πεδίο incorrect αυτά που δεν επιλύθηκαν. Στη γραμμή 16 υπολογίζεται το ποσοστό επιτυχίας του μοντέλου ανά κατηγορία και στη γραμμή 17, τυπώνεται αυτό στην οθόνη.

Να σημειωθεί πως τα αποτελέσματα αφορούν μόνο τα ερωτήματα των οποίων όλες οι λέξεις εντοπίστηκαν στο λεξικό του μοντέλου. Καθώς τα μοντέλα εκπαιδεύονται σε δεδομένα σχετικά μικρού μεγέθους, είναι λογικό πολλές από τις λέξεις να μην έχουν εμφανιστεί αρκετές φορές ώστε να υπάρχουν για αυτές διανύσματα. Αυτά τα ερωτήματα αγνοούνται. Όπως φαίνεται στην έξοδο του προγράμματος, τα ερωτήματα που

εκτελέστηκαν ήταν 854 από τα περίπου 2000 διαθέσιμα. Παρόλα αυτά, ο αριθμός είναι αρκετά μεγάλος για να είναι αξιοποιήσιμα τα αποτελέσματα. Συγκριτικά, το αγγλικό μοντέλο κατάφερε να εκτελέσει 1706 από τα περίπου 2000 ερωτήματα, αριθμό διπλάσιο από το ελληνικό. Αυτό το γεγονός λογικά οφείλεται στον αριθμό των ξεχωριστών λέξεων της κάθε γλώσσας όπως και στη συχνότητα εμφάνισης κάποιων λέξεων σε αυτές.

Το ποσοστό επιτυχίας του ελληνικού μοντέλου στις τέσσερις κατηγορίες βρίσκεται ανάμεσα στο 20-42% με τον συνολικό μέσο όρο να βρίσκεται στο 26%. Όπως θα παρουσιαστεί στο επόμενο κεφάλαιο που θα δημιουργηθούν πολλαπλά μοντέλα, τα αποτελέσματα βελτιώνονται σε σημαντικό βαθμό. Παρόλα αυτά, πολύ σημαντικό ρόλο παίζει η ποσότητα των δεδομένων εκπαίδευσης και αυτή είναι που περιορίζει την απόδοση του μοντέλου.

```
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 37: Αποτελέσματα αξιολόγησης του μοντέλου της Google

Για παράδειγμα, όπως φαίνεται και την Εικόνα 37, το μοντέλο που προσφέρει η Google στην ιστοσελίδα του word2vec (GoogleNews-vectors-negative300.bin.gz), το οποίο περιέχει 100 δισεκατομμύρια λέξεις, πετυχαίνει ποσοστά επιτυχίας 77% στο σύνολο των σχεδόν 20.000 ερωτημάτων της, με κάλυψη λέξεων που πλησιάζει το 100%.

6.4 Εκτέλεση πειραμάτων

Στη συνέχεια, θα μελετηθεί η επίδοση των μοντέλων μεταβάλλοντας τις παραμέτρους εκπαίδευσης τους. Θα δημιουργηθούν μοντέλα και με τις δύο αρχιτεκτονικές, CBOW και Skip-gram, με μέγεθος διανυσμάτων 100, 200 και 300 και με παράθυρο αναζήτησης 5, 7, 9 και 11. Ο κώδικας στην Εικόνα 38 αναλαμβάνει τη δημιουργία μοντέλων με συνδυασμό αυτών των τιμών παραμέτρων και την αξιολόγηση αυτών.

```
24      questions = "wordnet/finalGreek.txt"
25      inp = "wikipedia_el.text"
26      lines = LineSentence(inp)
27      skipOrCBOW = ['CBOW', 'SKIP']
28
29      results = open("resultsGreek.txt", 'w')
30
31      for skip in range(0, 2):
32          for dim in range(100, 301, 100):
33              for win in range(5, 12, 2):
34                  results.write("wiki.el." + str(dim) +
35                                skipOrCBOW[skip] + str(win) + "\n")
36
37                  model = Word2Vec(lines, sg=skip, size=dim,
38                                   iter=3, window=win, min_count=5, workers=3)
39
40                  # trim unneeded model memory = use (much) less RAM
41                  model.init_sims(replace=True)
42                  # model.save(output)
43
44                  result = model.wv.accuracy(questions)
45
46                  for el in result:
47                      corrects = len(el['correct'])
48                      wrongs = len(el['incorrect'])
49                      section = el['section']
50                      ratio = 'Not available'
51                      if corrects + wrongs > 0:
52                          ratio = str(corrects * 100 / (corrects + wrongs)) + '%'
53                          results.write("Success rate: " + str(ratio) +
54                                        " | Section: " + section + '\n')
55                  results.write('\n')
56
57      results.close()
```

Εικόνα 38: Εκτέλεση πειραμάτων

Στις σειρές 24 και 25 ορίζονται τα ονόματα των αρχείων ερωτημάτων και δεδομένων για τη δημιουργία των μοντέλων. Στη γραμμή 26, καλείται η μέθοδος

LineSentence και στη μεταβλητή lines τοποθετούνται οι γραμμές του συνόλου του κειμένου. Η γραμμή 27 αφορά την ονοματολογία κατά την αποθήκευση των αποτελεσμάτων. Στη γραμμή 29 ανοίγει το αρχείο στο οποίο θα αποθηκευθούν τα αποτελέσματα.

Στις γραμμές 31, 32 και 33 βρίσκονται τρεις δομές επιλογής, οι οποίες αναλαμβάνουν τη μεταβολή των παραμέτρων που χρησιμοποιούνται. Στη γραμμή 37 δημιουργείται ένα μοντέλο με τις τρέχουσες παραμέτρους. Οι γραμμές 44 - 55, όπως και στην Εικόνα 36, αναλαμβάνουν την αξιολόγηση του μοντέλου, με τη διαφορά, ότι αυτή τη φορά τα αποτελέσματα αποθηκεύονται στο αρχείο εξόδου και όχι στην οθόνη.

Με την εκτέλεση του κώδικα, θα δημιουργηθεί ένα μοντέλο για κάθε διαφορετικό συνδυασμό παραμέτρων και αυτό στη συνέχεια θα αξιολογηθεί. Σε αυτή τη φάση θα δημιουργηθούν 24 διαφορετικά μοντέλα για την κάθε γλώσσα. Με την άμεση εκτέλεση της αξιολόγησης για κάθε μοντέλο, αποφεύγεται η αποθήκευση αυτών.

```
wiki.el.100CBOW7
Success rate: 29% | Section: capital-countries
Success rate: 31% | Section: country-nationality
Success rate: 33% | Section: family
Success rate: 18% | Section: plural
Success rate: 24% | Section: total

wiki.el.100CBOW9
Success rate: 30% | Section: capital-countries
Success rate: 28% | Section: country-nationality
Success rate: 31% | Section: family
Success rate: 19% | Section: plural
Success rate: 24% | Section: total
```

Εικόνα 39: Αποτελέσματα αξιολόγησης

Για κάθε μοντέλο που αξιολογείται, δημιουργείται και μία εγγραφή στο αρχείο αποτελεσμάτων, τμήμα του οποίου φαίνεται στην Εικόνα 39. Στην ονοματολογία του μοντέλου, για παράδειγμα για το πρώτο μοντέλο, το wiki.el δηλώνει ελληνικό μοντέλο, το 100 των αριθμό των διαστάσεων, το CBOW την αρχιτεκτονική που χρησιμοποιήθηκε και ο αριθμός 7, το παράθυρο αναζήτησης. Στη συνέχεια, για κάθε μοντέλο, καταγράφεται η απόδοση αυτού σε κάθε κατηγορία ξεχωριστά και στο σύνολο αυτών. Για την παρουσίαση των αποτελεσμάτων των δύο γλωσσών, στη συνέχεια, θα

χρησιμοποιηθεί μόνο η απόδοση στο σύνολο των κατηγορίων. Τα αποτελέσματα σε κάθε κατηγορία ξεχωριστά, βρίσκονται στην ιστοσελίδα GitHub μαζί με τον υπόλοιπο κώδικα.

6.5 Αποτελέσματα

6.5.1 Ελληνικό μοντέλο

Τα αποτελέσματα θα παρουσιαστούν σε δύο πίνακες, ένα για την αρχιτεκτονική CBOW με μεταβολή των υπόλοιπων παραμέτρων και ομοίως για την αρχιτεκτονική Skip-gram.

CBOW		WINDOW			
		5	7	9	11
DIMENSIONS	100	23%	24%	24%	25%
	200	23%	24%	26%	24%
	300	22%	24%	25%	25%

Εικόνα 40: Αποτελέσματα CBOW για το ελληνικό μοντέλο

Στην Εικόνα 40 παρουσιάζονται τα αποτελέσματα με την αρχιτεκτονική CBOW. Τα ποσοστά κυμαίνονται στο εύρος 22% - 26%. Όπως φαίνεται, δεν υπάρχει κάποια σημαντική μεταβολή στην επίδοση του μοντέλου με αυτή την αρχιτεκτονική. Γενικά παρατηρείται αύξηση της επίδοσης, αν και μικρή, με την αύξηση του μεγέθους του παραθύρου αναζήτησης. Εξαίρεση αποτελεί μόνο το μοντέλο με τις 200 διαστάσεις και παράθυρο μεγέθους 9, το οποίο παρουσιάζει και τη μεγαλύτερη επίδοση.

Στη συνέχεια, στην Εικόνα 41 παρουσιάζονται τα αποτελέσματα για την αρχιτεκτονική Skip-gram τα οποία παρουσιάζουν περισσότερο ενδιαφέρον. Εδώ τα ποσοστά επιτυχίας είναι αρκετά μεγαλύτερα σε σχέση με πριν. Πιο σημαντικό όμως, είναι το γεγονός ότι παρατηρείται σημαντικότερη μεταβολή στα αποτελέσματα με την μεταβολή των παραμέτρων στο μοντέλο. Συγκεκριμένα, παρατηρείται αύξηση, αρκετά μεγαλύτερη από πριν, με την αύξηση των διαστάσεων και του παραθύρου αναζήτησης.

Και πάλι υπάρχει ένα μοντέλο το οποίο αποτελεί εξαίρεση, αλλά αυτή τη φορά υπάρχει ξεκάθαρο πρότυπο στη σύνδεση των παραμέτρων με τα αποτελέσματα.

Skip-gram		WINDOW			
		5	7	9	11
DIMENSIONS	100	31%	33%	36%	37%
	200	34%	36%	38%	39%
	300	34%	38%	37%	40%

Εικόνα 41: Αποτελέσματα Skip-gram για το ελληνικό μοντέλο

Καθώς υπάρχει συνεχής άνοδος στην επίδοση του μοντέλου με την αύξηση των διαστάσεων και του παραθύρου αναζήτησης, θα γίνουν μερικές ακόμα δοκιμές για να διευκρινιστεί το σημείο που θα σταματήσει αυτή η βελτίωση.

Skip-gram		WINDOW	
		13	15
DIMENSIONS	350	38%	40%
	400	40%	40%

Εικόνα 42: Επιπλέον δοκιμές για το ελληνικό μοντέλο

Οι επιπλέον δοκιμές που πραγματοποιήθηκαν αφορούν διαστάσεις 350 και 400 και παράθυρα αναζήτησης μεγέθους 13 και 15. Όπως φαίνεται στην Εικόνα 42, δεν παρουσιάζεται περαιτέρω αύξηση στην επίδοση του μοντέλου. Οπότε, σύμφωνα με τα αποτελέσματα της αξιολόγησης, το καλύτερο μοντέλο για την ελληνική γλώσσα είναι αυτό που χρησιμοποιεί 300 διαστάσεις, την αρχιτεκτονική Skip-gram και παράθυρο αναζήτησης μεγέθους 11. Επίσης, φαίνεται να υπάρχει σαφές πλεονέκτημα με τη χρήση της αρχιτεκτονικής Skip-gram αντί για την CBOW για την ελληνική γλώσσα, το οποίο αποτελεί αρκετά σημαντική παρατήρηση.

Για όλα τα πειράματα η εκπαίδευση των μοντέλων έγινε σε τρεις εποχές. Στη συγκεκριμένη κατηγορία προβλήματος παίζει πολύ μεγαλύτερο ρόλο η ποιότητα και η ποσότητα των δεδομένων και όχι ο αριθμός επαναλήψεων εκμάθησης πάνω σε αυτά. Παρόλα αυτά, για την ελληνική γλώσσα, θα εκτελεστεί ένα πείραμα για να ελεγχθεί κατά

πόσο μπορεί να αυξηθεί η επίδοση του καλύτερου μοντέλου που εντοπίστηκε με την αύξηση των εποχών εκπαίδευσης.

Skip-gram		WINDOW = 11
		DIMENSIONS = 300
ITERATIONS	3	40%
	4	41%
	6	42%
	8	42%
	10	42%
	12	43%

Εικόνα 43: Εκπαίδευση του μοντέλου με μεταβλητό αριθμό επαναλήψεων

Για την εκτέλεση αυτού του πειράματος θεωρείται σημείο αναφοράς η απόδοση του μοντέλου με 3 επαναλήψεις που είναι 40%. Θα γίνει εκπαίδευση του μοντέλου σε 4, 6, 8, 10 και 12 επαναλήψεις. Όπως φαίνεται στην Εικόνα 43, παρουσιάζεται αύξηση στην επίδοση του μοντέλου με την αύξηση των επαναλήψεων. Βέβαια, η μεταβολή της επίδοσης είναι πολύ μικρότερη από τη μεταβολή των επαναλήψεων. Για τη μετάβαση από τις 3 επαναλήψεις στις 6, παρουσιάζεται αύξηση της επίδοσης κατά 2% με κόστος το διπλάσιο χρόνο εκτέλεσης για την εκπαίδευση. Δεν γίνεται αναφορά σε συγκεκριμένους χρόνους καθώς εξαρτώνται από τις υπολογιστικές υποδομές που θα χρησιμοποιηθούν. Στη συνέχεια, για να επιτευχθεί 1% περαιτέρω βελτίωση απαιτούνται 12 επαναλήψεις. Απαιτείται για άλλη μία φορά διπλασιασμός του χρόνου εκπαίδευσης για 1% βελτίωση, αυτή τη φορά.

Skip-gram		WINDOW = 11
		DIMENSIONS = 300
ITERATIONS	40	45%
	100	45%

Εικόνα 44: Εκπαίδευση του μοντέλου με μεγάλο αριθμό επαναλήψεων

Στην Εικόνα 44 φαίνονται τα αποτελέσματα μετά από εκπαίδευση σε 40 και 100 επαναλήψεις. Η μετάβαση από τις 12 στις 40 επαναλήψεις οδήγησε σε βελτίωση 2% στην απόδοση του μοντέλου. Η εκπαίδευση σε 100 επαναλήψεις όμως δεν πρόσθεσε περαιτέρω βελτίωση. Πρέπει να ληφθεί υπόψη ότι το κάθε μοντέλο έχει ένα όριο στην απόδοση του το οποίο εξαρτάται από τα δεδομένα στα οποία εκπαιδεύεται. Δεν είναι δυνατό να υπάρχει συνεχόμενη βελτίωση μόνο με την αύξηση στον αριθμό επαναλήψεων. Πάντως με την αύξηση του αριθμού επαναλήψεων από τις 3 που είναι προτεινόμενες, πραγματοποιήθηκε βελτίωση 5% στην απόδοση του μοντέλου.

Όπως είναι φανερό ο αριθμός των επαναλήψεων δεν επηρεάζει σε μεγάλο βαθμό την επίδοση του μοντέλου. Σημαντικότερο ρόλο παίζουν, η επιλογή των κατάλληλων παραμέτρων όπως και η ποιότητα και ποσότητα των δεδομένων εκπαίδευσης. Παρόλα αυτά αν υπάρχουν διαθέσιμοι υπολογιστικοί πόροι μπορεί να επιτευχθεί μία μικρή περαιτέρω βελτίωση στην επίδοση του μοντέλου.

6.5.2 Αγγλικό μοντέλο

Όπως και στην ελληνική γλώσσα τα αποτελέσματα θα παρουσιαστούν για τις δύο αρχιτεκτονικές ξεχωριστά. Στην Εικόνα 45 βρίσκονται τα αποτελέσματα για την αρχιτεκτονική CBOW.

CBOW		WINDOW			
		5	7	9	11
DIMENSIONS	100	51%	53%	54%	51%
	200	53%	54%	55%	55%
	300	53%	56%	56%	57%

Εικόνα 45: Αποτελέσματα CBOW για το αγγλικό μοντέλο

Τα ποσοστά φαίνονται πολύ υψηλότερα στο αγγλικό μοντέλο. Παρατηρείται αύξηση της επίδοσης με την αύξηση των διαστάσεων και του μεγέθους παραθύρου αναζήτησης, με μία και πάλι εξαίρεση. Γενικά δεν παρατηρείται μεγάλη μεταβολή στα αποτελέσματα με την επιλογή διαφορετικών παραμέτρων, παρόλα αυτά είναι μεγαλύτερη από αυτή που εμφανίστηκε στο ελληνικό μοντέλο με την αρχιτεκτονική CBOW.

Στη συνέχεια, στην Εικόνα 46 βρίσκονται τα αποτελέσματα με τη χρήση της αρχιτεκτονικής Skip-gram.

Skip-gram		WINDOW			
		5	7	9	11
DIMENSIONS	100	50%	50%	49%	50%
	200	53%	55%	56%	57%
	300	53%	54%	56%	57%

Εικόνα 46: Αποτελέσματα Skip-gram για το αγγλικό μοντέλο

Και σε αυτή την περίπτωση παρατηρείται αύξηση της επίδοσης με την αύξηση των διαστάσεων και του μεγέθους παραθύρου αναζήτησης, με μία εξαίρεση για ακόμη μία φορά. Τα επίπεδα μεταβολής της επίδοσης βρίσκονται σε παρόμοιο επίπεδο με την προηγούμενη αρχιτεκτονική. Οπότε, για την αγγλική γλώσσα δεν φαίνεται να υπάρχει μία υπερέχουσα αρχιτεκτονική. Ως καλύτερο μοντέλο για την αγγλική γλώσσα θα επιλεχθεί το μοντέλο που δημιουργήθηκε με την αρχιτεκτονική CBOW, με αριθμό διαστάσεων 300 και μέγεθος παραθύρου αναζήτησης 11. Για το αγγλικό μοντέλο δεν θα γίνουν πειράματα στον αριθμό επαναλήψεων εκπαίδευσης καθώς η επιρροή τους σχετίζεται με τη λειτουργία του αλγορίθμου word2vec και όχι με μία συγκεκριμένη γλώσσα. Ανάλογη μικρή αύξηση θα παρουσίαζε και το αγγλικό μοντέλο με αύξηση των επαναλήψεων.

6.6 Σύγκριση των δύο γλωσσών

Με το πέρας της αξιολόγησης θα συγκριθούν τα αποτελέσματα των δύο γλωσσών και θα γίνουν παρατηρήσεις πάνω σε αυτά. Ακολουθώντας ως απόλυτο μέτρο σύγκρισης το μέγιστο ποσοστό επιτυχίας που κατάφερε η κάθε γλώσσα, τα αγγλικά φαίνεται να υπερισχύουν. Είναι πολύ σημαντικό όμως, να τονισθεί πως η σύγκριση μεταξύ αυτών των δύο αποτελεσμάτων ίσως και να μην είναι δόκιμη. Υπάρχουν δύο παράγοντες που καθιστούν τη σύγκριση μη δόκιμη και θα αναλυθούν στη συνέχεια.

Ο πρώτος παράγοντας έχει να κάνει με την επιλογή των δεδομένων που χρησιμοποιήθηκαν για την εκπαίδευση των μοντέλων. Στα ελληνικά χρησιμοποιήθηκε το σύνολο της ελληνικής wikipedia ενώ στα αγγλικά ένα μικρό μέρος της αγγλικής. Δεν

είναι δεδομένο πως τα κείμενα της μίας γλώσσας είναι ανάλογης ποιότητας με της άλλης, οπότε αυτό το γεγονός μπορεί να δημιουργήσει διαφοροποιήσεις στην αποτελεσματικότητα των μοντέλων. Επίσης, καθώς οι δύο γλώσσες έχουν τεράστιες διαφορές μεταξύ τους, ίσως η επιλογή παρόμοιου αριθμού δεδομένων να είναι λάθος. Τα ελληνικά, όντας πιο πολύπλοκη γλώσσα, ίσως και να απαιτεί και μεγαλύτερο όγκο δεδομένων σε σχέση με τα αγγλικά για να αποδώσει ανάλογα.

Ο δεύτερος παράγοντας έχει να κάνει με τα κριτήρια αξιολόγησης. Εδώ για να υπάρξει αντιστοιχία μεταξύ των κριτηρίων των δύο γλωσσών, πραγματοποιήθηκε μετάφραση των κριτηρίων από την αγγλική γλώσσα στην ελληνική. Το γεγονός ότι η μετάφραση και η επιλογή των ερωτημάτων έγινε με επιμέλεια δεν είναι αρκετό για να θεωρηθούν αυτά απολύτως αντίστοιχα. Μπορεί για παράδειγμα, στην αγγλική γλώσσα κάποια ερωτήματα να θεωρούνται πιο ‘εύκολα’ από ότι τα αντίστοιχα στην ελληνική. Όπως αναφέρθηκε και νωρίτερα, τα επίθετα στην ελληνική γλώσσα εμφανίζονται σε πολλές μορφές καθώς μπορεί να αναφέρονται σε διαφορετικό πρόσωπο ή να χρησιμοποιούνται σε διαφορετική πτώση, σε αντίθεση με την αγγλική.

Παρόλα αυτά, ο βασικός σκοπός αυτής της έρευνας δεν είναι η ανάδειξη μίας από τις δύο γλώσσες ως καλύτερη, αλλά η παρουσίαση της διαδικασίας δημιουργίας μοντέλων και της αξιολόγησης τους. Κατά τη διαδικασία της αξιολόγησης παρατηρήθηκε βελτίωση της απόδοσης και στις δύο γλώσσες με την αύξηση των διαστάσεων και του παραθύρου αναζήτησης μέχρι ένα σημείο. Επίσης, η αρχιτεκτονική Skip-gram φέρεται να είναι καλύτερη επιλογή σε σχέση με την CBOW για την ελληνική γλώσσα ενώ στα αγγλικά δεν φαίνεται να υπάρχει σημαντική διαφορά.

Τελικά, παρόλο που δεν είναι δυνατό να αποδειχθεί πως τα μοντέλα της αγγλικής γλώσσας είναι καλύτερα από αυτά της ελληνικής, το συμπέρασμα είναι ότι η δημιουργία μοντέλων σε αυτή είναι αρκετά ευκολότερη. Όπως φάνηκε τα μοντέλα στα αγγλικά έχουν καλύτερη απόδοση με παρόμοιο αριθμό δεδομένων και αφού υπάρχουν τάξης μεγέθους περισσότερα δεδομένα διαθέσιμα στο διαδίκτυο για την αγγλική γλώσσα, η δημιουργία αποδοτικότερων μοντέλων είναι ευκολότερη για αυτή.

7 Επίλογος

7.1 Σύνοψη και συμπεράσματα

Η επεξεργασία φυσικής γλώσσας είναι μία τεχνολογία που έχει δει μεγάλη άνθιση τα τελευταία χρόνια. Οι εφαρμογές της στη σημερινή κοινωνία είναι πολλές και συνεχώς αυξανόμενες. Για να είναι εφικτές αυτές οι εφαρμογές και όσες άλλες εμφανιστούν στο μέλλον είναι απαραίτητη η όσο το δυνατό πιο αποτελεσματική χρήση της.

Για την υλοποίηση της επεξεργασίας φυσικής γλώσσας δημιουργούνται μοντέλα γλωσσών μέσα στα οποία είναι αποθηκευμένη η πληροφορία για το σύνολο της γλώσσας. Με τη χρήση διαδικασιών εκμάθησης, μέσα από νευρωνικά δίκτυα, αυτά τα μοντέλα εκπαιδεύονται για να επιτύχουν το σκοπό τους. Η απόδοση αυτών των μοντέλων βασίζεται στον όγκο και την ποιότητα των δεδομένων που θα χρησιμοποιηθούν στην εκμάθηση όπως επίσης και στον αλγόριθμο που θα την αναλάβει. Βασικό φράγμα για τη δημιουργία αποδοτικών μοντέλων στο παρελθόν ήταν η έλλειψη υπολογιστικών πόρων.

Με την αξιοποίηση των νευρωνικών δικτύων έχουν δημιουργηθεί πολλές νέες κατηγορίες εφαρμογών όπως είναι η αναγνώριση προτύπων, η αναγνώριση ομιλίας και η επεξεργασία φυσικής γλώσσας. Σε όλες αυτές τις κατηγορίες δεν ήταν εφικτή η δημιουργία εφαρμογών με στοιχειώδη αποτελέσματα στο παρελθόν. Αυτή η έρευνα επικεντρώθηκε στην επεξεργασία φυσικής γλώσσας.

Η επεξεργασία φυσικής γλώσσας φέρει πολλές δυσκολίες βάση της φύση της. Στη φυσική γλώσσα σχεδόν το σύνολο της αναγκαίας πληροφορίας για την επικοινωνία παραλείπεται με αποτέλεσμα την πολύ δύσκολη κατανόηση της από έναν ηλεκτρονικό υπολογιστή. Η προσέγγιση που έχει ακολουθηθεί είναι η δημιουργία και εκπαίδευση μοντέλων γλωσσών με τον τρόπο που θα μάθαινε μία γλώσσα και ένας άνθρωπος.

Στα μοντέλα αυτά δίδεται τεράστιος όγκος κειμένων και αυτά καλούνται να δημιουργήσουν διανύσματα για την απεικόνιση των λέξεων που συναντούν. Ο καθορισμός των διανυσμάτων γίνεται με τέτοιο τρόπο ώστε λέξεις που σχετίζονται μεταξύ τους να βρίσκονται κοντά και στο διανυσματικό χώρο. Έχουν προταθεί πολλοί

αλγόριθμοι για τη δημιουργία αυτών των μοντέλο με τον πιο δημοφιλή να είναι ο word2vec.

Ο word2vec είναι ένας αλγόριθμος δημιουργίας διανυσμάτων λέξεων ο οποίος προτάθηκε από τη google. Δέχεται ως είσοδο ένα σύνολο κειμένων και δημιουργεί διανύσματα για κάθε λέξη που συναντά σε αυτά. Προσφέρει κάποιες λειτουργίες όπως είναι η εύρεση ομοιότητας ανάμεσα σε δύο λέξεις, ο εντοπισμός της λέξης που βρίσκεται πιο κοντά σε μία δεδομένη λέξη, η δημιουργία αναλογιών και ο εντοπισμός λέξεων μη σχετικών μέσα σε ένα σύνολο άλλων λέξεων. Η πιο σημαντική λειτουργία είναι η δημιουργία αναλογιών καθώς βοηθείται σε μεγάλο βαθμό από τον τρόπο αναπαράστασης των λέξεων και προσφέρει ενδιαφέροντα αποτελέσματα. Με τον ορισμό δύο λέξεων που σχετίζονται με κάποιο τρόπο και μιας τρίτης, δύναται μέσω αλγεβρικών υπολογισμών να εντοπίζει μία τέταρτη λέξη, η οποία σχετίζεται αντίστοιχα με την τρίτη.

Στη συγκεκριμένη έρευνα δημιουργήθηκαν μοντέλα για την ελληνική και την αγγλική γλώσσα με τον αλγόριθμο word2vec. Τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση των μοντέλων προήλθαν από τη wikipedia. Παρουσιάστηκε η διαδικασία που ακολουθείται από την αναζήτηση των δεδομένων μέχρι τη δημιουργία των μοντέλων και την αξιοποίηση αυτών. Επίσης, δημιουργήθηκε μία εφαρμογή διαδικτύου, αποτελούμενη από τρεις άλλες εφαρμογές, η οποία αναλαμβάνει να δώσει πρόσβαση στις λειτουργίες του ελληνικού μοντέλου σε εξωτερικούς χρήστες.

Σημαντικό μέρος της διαδικασίας δημιουργίας μοντέλων για επεξεργασία φυσικής γλώσσας είναι και η αξιολόγηση αυτών. Χωρίς την αξιολόγηση δεν μπορεί να μετρηθεί απόλυτα η επίδοση ενός μοντέλου. Υπάρχουν δύο κατηγορίες αξιολόγησης, η εσωτερική και η εξωτερική. Με την εσωτερική αξιολόγηση το μοντέλο ελέγχεται με συγκεκριμένα ερωτήματα για να κριθεί αν είναι σωστή η απεικόνιση των λέξεων στο διανυσματικό χώρο. Στην εξωτερική, το μοντέλο χρησιμοποιείται για την επίτευξη κάποιας συγκεκριμένης εργασίας και μετράται η επίδοση του σε αυτή. Διαφορετικά μοντέλα μπορεί να παρουσιάζουν αποκλίσεις στα ποσοστά επιτυχίας τους σε διαφορετικές τεχνικές αξιολόγησης ανάλογα με τη χρήση για την οποία προορίζονται.

Στη συγκεκριμένη έρευνα χρησιμοποιήθηκε εσωτερική αξιολόγηση με τη μέθοδο των αναλογιών για την εντοπισμό των καταλληλότερων παραμέτρων κατά την εκπαίδευση των μοντέλων στην ελληνική και την αγγλική γλώσσα για τα συγκεκριμένα δεδομένα. Δημιουργήθηκαν πολλαπλά μοντέλα με διαφορετικές παραμέτρους και αυτά αξιολογήθηκαν ξεχωριστά. Για την ελληνική γλώσσα έγιναν και πειράματα για το βαθμό

εξάρτησης της επίδοσης του μοντέλου σε σχέση με τον αριθμό επαναλήψεων που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου πάνω στα ίδια δεδομένα. Τελικά, έγινε μία υποτυπώδης σύγκριση στα αποτελέσματα των δύο γλωσσών και αναλύθηκαν οι παράγοντες που χρήζουν προσοχής σε τέτοιες συγκρίσεις.

7.2 Όρια και περιορισμοί της έρευνας

Η έρευνα περιορίζεται από το γεγονός ότι αφορά την ελληνική γλώσσα. Καθώς η επεξεργασία φυσικής γλώσσας εξαρτάται σε μεγάλο βαθμό από την ποσότητα των δεδομένων που θα χρησιμοποιηθούν κατά την εκπαίδευση, στην ελληνική τα δεδομένα αυτά είναι πολύ περιορισμένα. Ο όγκος των δεδομένων που απαιτείται σύμφωνα με τη βιβλιογραφία για να φανούν ικανοποιητικά αποτελέσματα είναι τάξεις μεγεθών μεγαλύτερα από αυτά που μπορούν να βρεθούν ελεύθερα για την ελληνική γλώσσα. Παρόλα αυτά, σκοπός της έρευνας είναι η παρουσίαση των εργαλείων και της διαδικασίας που ακολουθείται για τη δημιουργία και την αξιοποίηση των μοντέλων και όχι η δημιουργία μοντέλων που να μπορούν να χρησιμοποιηθούν σε εμπορικές εφαρμογές.

Ένας άλλος περιορισμός αφορά τη διαδικασία που ακολουθήθηκε για την αξιολόγηση των μοντέλων όσον αφορά τη σύγκριση των αποτελεσμάτων τους. Δεδομένου ότι τα ελληνικά και τα αγγλικά είναι δύο πολύ διαφορετικές γλώσσες, η απόλυτη σύγκριση των ποσοστών επιτυχίας των μοντέλων σε συγκρίσιμα ερωτήματα δεν είναι δόκιμη. Και εδώ όμως σκοπός είναι η παρουσίαση της διαδικασίας αξιολόγησης και ανάλυσης των αποτελεσμάτων και όχι ο εντοπισμός της καλύτερης γλώσσας.

7.3 Μελλοντικές επεκτάσεις

Η έρευνα μπορεί να επεκταθεί στο μέλλον σε τρεις τομείς. Ο ένας τομέας είναι η χρήση αντίστοιχης προσέγγισης για τη μελέτη της απόδοσης άλλων γλωσσών στην επεξεργασία φυσικής γλώσσας και κάτω από συγκεκριμένες συνθήκες η σύγκριση όμοιων γλωσσών. Ο δεύτερος τομέας είναι η συλλογή μεγαλύτερου όγκου δεδομένων για την ελληνική γλώσσα με σκοπό τη δημιουργία μοντέλων που να προσεγγίζουν την

απόδοση των αγγλικών. Ο τρίτος τομέας αφορά την χρήση διαφορετικών αλγορίθμων δημιουργίας διανυσμάτων και τη μελέτη των επιδόσεων των μοντέλων για κάθε ένα από αυτούς.

Βιβλιογραφία

[1] Machine learning - Wikipedia. 2018. Machine learning - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Machine_learning. [Accessed 28 April 2018].

[2] LeCun Yann, Bengio Yoshua, Hinton Geoffrey, 2015. Deep learning. Nature, Volume
521, Pages 436-444.

[3] Artificial neural network - Wikipedia. 2018. Artificial neural network - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed 28 April 2018].

[4] A. K. Jain, Jianchang Mao and K. M. Mohiuddin, 1996. Artificial neural networks: a tutorial. in Computer, Volume 29, Issue 3, Pages 31-44.

[5] Deep learning - Wikipedia. 2018. Deep learning - Wikipedia. [ONLINE] Available at:
https://en.wikipedia.org/wiki/Deep_learning. [Accessed 28 April 2018].

[6] TensorFlow. 2018. TensorFlow. [ONLINE] Available at:
https://www.tensorflow.org/. [Accessed 24 May 2018].

[7] Natural-language processing - Wikipedia. 2018. Natural-language processing - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Natural-language_processing. [Accessed 28 April 2018].

[8] Mikolov, T., Chen, K., Corrado, G. and Dean, J. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[9] Analytics Vidhya. 2018. Intuitive Understanding of Word Embeddings: Count Vectors to Word2Vec. [ONLINE] Available at: https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/. [Accessed 25 June 2018].

[10] Baroni, M., Dinu, G. and Kruszewski, G., 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 238-247).

[11] Word2vec - Wikipedia. 2018. Word2vec - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Word2vec. [Accessed 28 April 2018].

[12] Towards Data Science. 2018. Learn Word2Vec by implementing it in tensorflow – Towards Data Science. [ONLINE] Available at: https://towardsdatascience.com/learn-word2vec-by-implementing-it-in-tensorflow-45641adaf2ac. [Accessed 25 May 2018].

[13] NumPy — NumPy. 2018. NumPy — NumPy. [ONLINE] Available at: http://www.numpy.org/. [Accessed 24 May 2018].

[14] Softmax function - Wikipedia. 2018. Softmax function - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Softmax_function. [Accessed 25 May 2018].

[15] Schnabel, T., Labutov, I., Mimno, D. and Joachims, T. 2015. Evaluation methods for unsupervised word embeddings. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. Pages 298-307.

[16] Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal, 2016. The role of context types and dimensionality in learning word embeddings. HLT-NAACL, Pages 1030–1040.

[17] Michele Banko, Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. Proceedings of the 39th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Stroudsburg, PA, USA, 26-33.

[18] List of text corpora - Wikipedia. 2018. List of text corpora - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/List_of_text_corpora. [Accessed 29 April 2018].

[19] List of Wikipedias - Wikipedia. 2018. List of Wikipedias - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/List_of_Wikipedias. [Accessed 29 April 2018].

[20] Wikipedia:Database download - Wikipedia. 2018. Wikipedia:Database download - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Wikipedia:Database_download. [Accessed 29 April 2018].

[21] Wikimedia Downloads. 2018. Wikimedia Downloads. [ONLINE] Available at: https://dumps.wikimedia.org/. [Accessed 29 April 2018].

[22] elwiki dump progress on 20180420. 2018. elwiki dump progress on 20180420. [ONLINE] Available at: https://dumps.wikimedia.org/elwiki/20180420/. [Accessed 29 April 2018].

[23] gensim: Topic modelling for humans. 2018. gensim: Topic modelling for humans. [ONLINE] Available at: https://radimrehurek.com/gensim/. [Accessed 29 April 2018].

[24] gensim: models.word2vec – Deep learning with word2vec. 2018. gensim: models.word2vec – Deep learning with word2vec. [ONLINE] Available at: https://radimrehurek.com/gensim/models/word2vec.html. [Accessed 20 May 2018].

[25] Hypertext Transfer Protocol - Wikipedia. 2018. Hypertext Transfer Protocol - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Accessed 20 May 2018].

[26] React (JavaScript library) - Wikipedia. 2018. React (JavaScript library) - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/React_(JavaScript_library). [Accessed 20 May 2018].

[27] Node.js - Wikipedia. 2018. Node.js - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Node.js. [Accessed 20 May 2018].

[28] JSON - Wikipedia. 2018. JSON - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/JSON. [Accessed 20 May 2018].

[29] Express - Node.js web application framework. 2018. Express - Node.js web application framework. [ONLINE] Available at: https://expressjs.com/. [Accessed 20 May 2018].

[30] npm. 2018. body-parser - npm. [ONLINE] Available at: https://www.npmjs.com/package/body-parser. [Accessed 20 May 2018].

[31] npm. 2018. request - npm. [ONLINE] Available at: https://www.npmjs.com/package/request. [Accessed 20 May 2018].

[32] npm. 2018. cors - npm. [ONLINE] Available at: https://www.npmjs.com/package/cors. [Accessed 20 May 2018].

[33] GitHub. 2018. GitHub - GiannisMP/DeepLearning. [ONLINE] Available at: https://github.com/GiannisMP/DeepLearning. [Accessed 20 May 2018].

[34] jQuery - Wikipedia. 2018. jQuery - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/JQuery. [Accessed 20 May 2018].

[35] Ajax (programming) - Wikipedia. 2018. Ajax (programming) - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Ajax_(programming). [Accessed 20 May 2018].